Universidad del País Vasco
Euskal Herriko Unibertsitatea

ZIENTZIA ETA TEKNOLOGIA FAKULTATEA
FACULTAD DE CIENCIA Y TECNOLOGÍA

Trabajo Fin de Grado

Grado en Ingeniería electrónica

Predicción de la estructura secundaria de las proteínas mediante redes neuronales.

Autor/a:

Ane Miren Gutiérrez Muto

Director/a:

Javier Echanobe

Leioa, 27 de junio de 2017

ÍNDICE DE CONTENIDO.

Anexos.

Predicción de la estructura secundaria de las proteínas mediante redes neuronales.

Ane Miren Gutiérrez.

Objetivo y desarrollo.

En el mundo real, no es siempre posible ser precisos en la búsqueda de respuestas o soluciones a los problemas. En muchas situaciones, los sistemas naturales tienden a hacer cosas que éstos creen que son verdad, más que cosas que son forzosamente ciertas. Los fundamentos de definir métodos estrictos y normas fallan si la convicción subyacente lo hace. Pero esto nos da un punto de vista fascinante de un mundo que está conducido por la convicción, la imprecisión y las aproximaciones que se realizan cada día en numerosas ocasiones a la hora de tomar decisiones.

Hoy en día, muchos de los problemas que se encuentran dentro del ámbito de la biología y la bioquímica, tales como secuenciación, predicción, etc., son resueltos mediante métodos computacionales. Las proteínas adquieren una estructura que reúne las propiedades de disposición en el espacio provenientes de su secuencia de aminoácidos y de las características físicas de su entorno, pudiendo resultar muy compleja. Pero dicha estructura es la responsable de generar determinadas funciones que son esenciales para los seres vivos, como por ejemplo la reserva energética, el transporte y las funciones tanto defensiva como hormonal. La estructura de las proteínas se puede estudiar desde cuatro niveles de complejidad, pero este trabajo se centra en la estructura secundaria de las proteínas, la cual se basa en la ordenación espacial de los aminoácidos.

De esta forma, el objetivo principal se basa en predecir la estructura secundaria de las proteínas mediante redes neuronales, buscando el mayor porcentaje de reconocimiento posible. Esto se realiza a partir de tres codificaciones que servirán como muestras de entrada para dichas redes, de forma que puedan conocerse con gran precisión las interacciones que se forman y así poder deducir sus funciones a nivel biológico. Estas propiedades y funciones varían según las propiedades moleculares y las preferencias de los aminoácidos para formar diferentes tipos de estructuras secundarias. Por tanto, si un esquema de codificación permite que las técnicas de redes neuronales predigan efectivamente la probabilidad de sustitución de aminoácidos que ocurren en la naturaleza, se esperará que estos mismos esquemas sean capaces de actuar también mediante distintas tareas y de una forma similar para así obtener la función de la proteína, su estructura, composición, etc.

En este trabajo se han creado tres codificaciones diferentes, mediante el protocolo de ventana deslizante, y habiendo utilizado cada una de éstas como muestra de entrada de la red neuronal utilizada. Estas tres representaciones vienen definidas como: ortogonal, estadística y mediante grafos dirigidos. En la primera de ellas, se hace uso de vectores binarios, cuyas componentes vienen formadas por 0 y 1. Por otro lado, en la representación estadística se hace uso de información estadística tanto de la proteína al completo como de la sección que se está analizando, dando información mucho más específica sobre ella. Y

finalmente, en la representación a partir de grafos dirigidos, entra en juego la codificación genética de cada aminoácido, que viene dada por los cuatro nucleótidos que forman el RNA. Una vez creadas las matrices anteriores, que servirán como muestras de entrada, se decidirá la topología de la red neuronal, que en este caso vendrá dada por una red multicapa *backpropagation* de 3 capas (2 capas ocultas y 1 de salida), cuya función de entrenamiento vendrá dada por la secante y utilizará como función de transferencia para las capas ocultas la tangente hiperbólica sigmoidea.

Para concluir, se han obtenido los resultados de reconocimiento de la red neuronal tras emplear las tres codificaciones matriciales como muestras de entrada, y posteriormente, en un análisis más exhaustivo, se ha aplicado un barrido neuronal al resultado más favorable para la búsqueda de la red neuronal óptima. De este modo, se obtiene que la red que presenta un mayor porcentaje de precisión y un menor error en la predicción de la estructura secundaria es aquella que viene caracterizada por la representación matricial estadística, con una longitud de ventana de 11 aminoácidos y 15000 iteraciones aproximadamente. Además, se concluye que el mayor porcentaje de reconocimiento y el mejor comportamiento de red se obtienen con 25 neuronas por capa oculta.

El trabajo está organizado como sigue: en el primer capítulo se realiza una introducción en el ámbito de la bioquímica para conocer el origen de las proteínas a partir del ADN y la secuenciación de aminoácidos. Además, se destacarán las diferentes estructuras secundarias de las proteínas ya que el objetivo del trabajo se fundamenta en la predicción de éstas. A continuación, en el capítulo 2 se introducirán las técnicas de *Soft Computing*, mencionando las más empleadas actualmente y haciendo especial hincapié en las Redes Neuronales, las cuales serán utilizadas para la resolución del trabajo.
En el capítulo 3 comienza el desarrollo del procedimiento seguido: en primer lugar se realiza un análisis de la base de datos y herramienta de programación utilizadas. Posteriormente se comienza con la manipulación de datos y la formación de las muestras tanto de entrada como de salida. Como se podrá comprobar ulteriormente, se han hecho uso de tres esquemas de codificación para la obtención de datos (esquema ortogonal, estadístico y mediante grafos dirigidos). Para concluir con el capítulo, se desarrollan las decisiones tomadas respecto a la topología de la red neuronal, y así en el capítulo 4 analizar y discutir los resultados obtenidos en las diferentes redes neuronales. Además se realiza una clasificación de los resultados más significativos con intención de mejorar dichos resultados. Finalmente, se efectúan una serie de conclusiones de cierre en las que se incorporan futuras líneas de seguimiento para la máxima optimización del problema.

CAPÍTULO 1.

ADN, PROTEÍNAS Y AMINOÁCIDOS.

El genoma humano es el código genético de los seres humanos, es decir, es el conjunto de genes humanos. La secuenciación de nuestro genoma es realmente un hito en la historia de la humanidad, que empieza a ser realidad a partir de los años sesenta mediante una serie de descubrimientos que dieron origen a lo que se ha dado a llamar "*la nueva genética*" [1]. Esta secuencia contiene una cantidad enorme de información, parte de la cual podemos ahora extraer e interpretar, pero con otra gran parte que sólo estamos empezando a comprender. La secuencia genómica, además de tener implicaciones en la comprensión de la salud y enfermedades humanas, es fuente de un profundo conocimiento en otros aspectos de la biología y cultura humanas.

El ADN y el RNA son polímeros lineales largos, llamados ácidos nucleótidos, que contienen información de tal forma que puede ser transmitida de una generación a la siguiente. Estas macromoléculas están formadas por un gran número de nucleótidos unidos (combinaciones de los cuatro nucleótidos {A, T, C, G} en el ADN y {A, U, C, G} en el RNA), cada uno de ellos compuesto por un azúcar, un grupo fosfato y una base.
La estructura del ADN reveló cómo se almacena la información en la secuencia de bases contenida en la hebra. En la replicación del ADN cada una de las hebras sirve de molde para la otra. La función más importante del ADN consiste en codificar las secuencias de las proteínas. Al igual que el ADN, las proteínas son polímeros lineales. Los genes de todas las células y de muchos virus están hechos de ADN, pero sin embargo, algunos virus utilizan el RNA como material genético (Véase *Figura 1*).

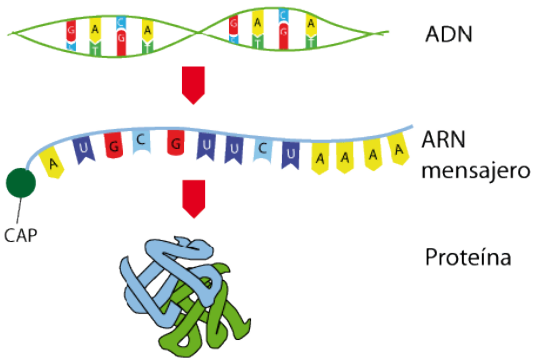


Figura 1: *Flujo de información genética (expresión genética) en células normales.*

Las proteínas se diferencian del ADN en dos aspectos importantes. Primero, las proteínas se construyen a partir de 20 monómeros, llamados *aminoácidos*, en vez de sólo los cuatro nucleótidos que configuran el ADN. La complejidad química aportada por esa variedad de monómeros capacita a las proteínas para

realizar una amplia gama de funciones. Y segundo, las proteínas se pliegan espontáneamente en elaboradas estructuras tridimensionales determinadas únicamente por su secuencia de aminoácidos [2].

Las proteínas se sintetizan en forma de una secuencia de aminoácidos unidos en una estructura poliamídica (polipéptido) lineal, aunque al realizar sus funciones adoptan complejas formas tridimensionales [3]. En las proteínas se encuentras habitualmente 20 tipos de cadenas laterales que varían en función del tamaño, forma, carga, capacidad de formar puentes de hidrógeno, carácter hidrofóbico y reactividad química. De hecho, todas las proteínas de todas las especies –bacterias, árqueas y eucariotas- se construyen con los mismos 20 aminoácidos con unas pocas excepciones. La extraordinaria variedad de funciones realizadas por las proteínas es el resultado de la diversidad y versatilidad de estos 20 sillares de su construcción. De hecho, algunos aminoácidos pueden encontrarse en forma libre o combinada, pero no en las proteínas. Estos 20 aminoácidos proteicos consisten en un átomo de carbono al cual se encuentran unidos un grupo de aminos primario, una cadena lateral (grupo R), un átomo de hidrógeno H y un carboxilo, tal y como se muestra en la *Figura 2*. Existe una gran cantidad de aminoácidos en la naturaleza, pero sólo 20 de ellos son proteinogénicos [4]. Éstos vienen clasificados como: Serina (S), Treonina (T), Cisteína (C), Asparagina (N), Glutamina (Q), Tirosina (Y), Glicina (G), Alanina (A), Valina (V), Leucina (L), Isoleucina (I), Metionina (M), Prolina (P), Fenilalanina (F), Triptófano (W), Ácido Aspártico (D), Ácido Glutámico (E), Lisina (K), Arginina (R) e Histidina (H), cuyas siglas serán utilizadas a lo largo del trabajo.

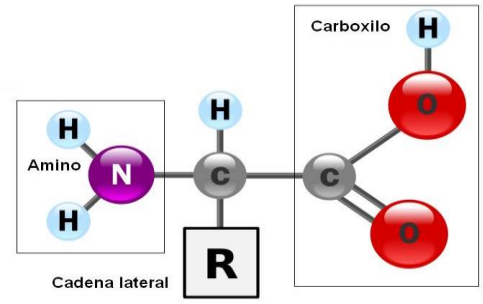


Figura 2: *Estructura atómica general de los aminoácidos.*

Las proteínas son los principales polímeros estructurales y funcionales de los seres vivos, y tienen una amplia gama de actividades como la catálisis de las reacciones metabólicas y el transporte de las vitaminas, minerales, oxígeno y alimentos. En los microorganismos, las plantas y los animales se encuentran aproximadamente 300 aminoácidos, pero de éstos sólo 20 son codificados por el ADN para formar las proteínas [3]. Como se expondrá a continuación las proteínas constan de cuatro estructuras, siendo éstas dependientes de las múltiples configuraciones que sus aminoácidos puedan adoptar en el espacio.

1.1. Composición y estructura de las proteínas.

Las proteínas son las macromoléculas más versátiles de los seres vivos y desempeñan funciones cruciales en prácticamente todos los procesos biológicos. Funcionan como catalizadores, transportan y almacenan otras moléculas como el oxígeno, proporcionan apoyo mecánico y protección inmunológica, generan movimiento, trasmiten impulsos nerviosos y controlan el crecimiento.
Varias propiedades clave permiten a las proteínas participar en una gama de funciones tan amplia y variada. En primer lugar, cabe destacar que las proteínas se pliegan espontáneamente en estructuras tridimensionales que vienen determinadas por la secuencia de aminoácidos del polímero proteico. La función de una proteína depende directamente de su estructura tridimensional. Es decir, las proteínas son la transición desde el mundo unidimensional lineal de las secuencias hasta el mundo tridimensional de las moléculas capaces de realizar diferentes actividades. Por otro lado, las proteínas pueden interaccionar entre sí y con otras macromoléculas biológicas para formar asociaciones complejas. Estas asociaciones pueden ser sinérgicas, de forma que se generan capacidades que no existían en los componentes proteicos individuales. Estas asociaciones incluyen maquinarias macromoleculares que llevan a cabo la replicación precisa del ADN, la transmisión de señales dentro de las células y muchos otros procesos esenciales [2].

Como se ha mencionado anteriormente, en las proteínas se pueden distinguir cuatro niveles de estructuración: estructura primaria, secundaria, terciaria y cuaternaria.
La **estructura primaria** de la proteína es la secuencia lineal de sus aminoácidos, formando un enlace de tipo amida (péptido). Las unidades de aminoácido de una cadena peptídica se denominan *residuos aminoácidos* (de esta forma una cadena peptídica formada por tres residuos aminoácidos se denomina tripéptido (véase *Figura 3a*). Por otro lado, la **estructura secundaria** de una proteína se refiere a la estructura local de la cadena polipeptídica. Esta estructura depende de las interacciones de los puentes de hidrógeno entre los residuos y el esqueleto peptídico, tal y como se muestra en la *Figura 3b*. Así mismo, la conformación tridimensional, plegada y biológicamente activa de una proteína se conoce como su **estructura terciaria**, visible e la *Figura 3c*. Esta estructura se debe al plegamiento de la cadena peptídica y refleja la forma global de la molécula. Por último, la **estructura cuaternaria** hace referencia a un complejo o ensamblaje de dos o más cadenas peptídicas separadas que se mantienen juntas a través de las interacciones existentes entre las mismas (véase *Figura 3d*). Muchas proteínas que constan de múltiples subunidades están formadas por diferentes clases de subunidades funcionales [3].

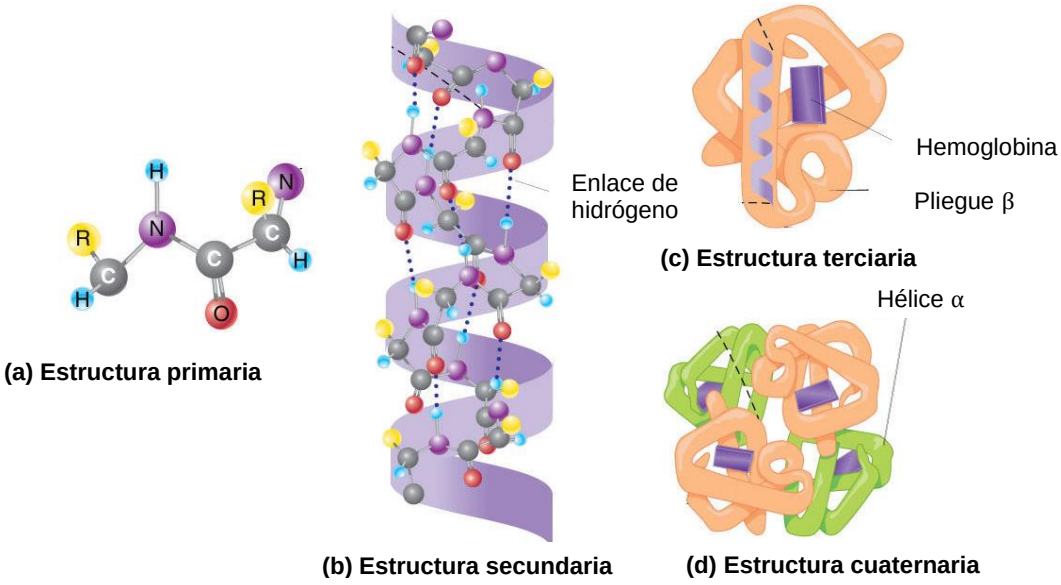


Figura 3: *Estructuras de las proteínas: (a) primaria, (b) secundaria, (c) terciaria y (d) cuaternaria.*

1.2. Estructura secundaria de la proteína.

En 1951 Linus Pauling y Robert Corey propusieron dos estructuras periódicas llamadas la *hélice α* y la *hoja plegada β*. Posteriormente se identificaron otras estructuras como el *giro β* y el *bucle omega (Ω)*. Aunque no son periódicas, estas estructuras habituales como los giros o los bucles están bien definidas y contribuyen junto con las hélices α y las hojas β en la formación de la estructura proteica final [5].

Al evaluar las potenciales estructuras, Pauling y Corey consideraron qué conformaciones peptídicas estaban permitidas estéricamente y cuáles aprovechaban mejor la capacidad de los grupos NH y CO de esqueleto para formar puentes de hidrógeno. La primera de las estructuras que propusieron, la hélice α (la cual a lo largo de los programas se va denotar mediante la letra **H**), es una estructura en forma de cilindro. Un esqueleto helicoidal muy firmemente plegado forma la parte interior del cilindro, mientras que las cadenas laterales de los residuos aminoácidos se extienden hacia fuera en una distribución helicoidal (en casi todas las proteínas naturales, la hélice se enrolla hacia la derecha (véase *Figura 4*). La hélice α se estabiliza por puentes de hidrógeno entre los grupos NH y CO de la cadena principal [2], [3], [5].

Figura 4: *Visualización de la estructura de la proteína 3OGH mediante el software gráfico Chimera.*

Pauling y Corey descubrieron otro motivo estructural periódico al que llamaron *hoja plegada β*, la cual a lo largo de los trabajos vendrá denotada por la letra **E**. En este caso, si los puentes de hidrógeno mencionados anteriormente se forman entre enlaces peptídicos de cadenas diferentes, éstas se disponen en sentido paralelo o antiparalelo entre sí y forman la hoja plegada β [3]. La hoja β es muy diferente de las hélices α cilíndricas. Está formada por dos o más cadenas polipeptídicas, llamadas *hebras β*. Las *hebras β* están casi completamente extendidas en vez de estar enrolladas y empaquetadas como en la hélice α. Hay toda una gama de estructuras extendidas permitidas estéricamente (véase *Figura 5*) [2].

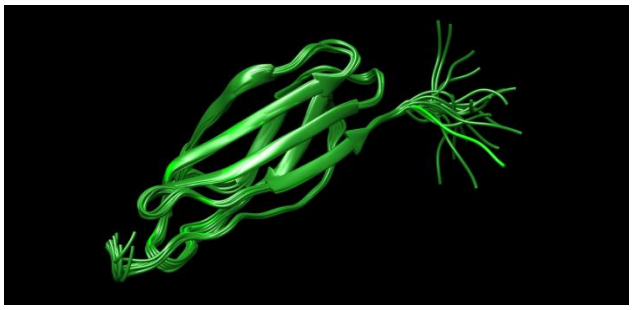


Figura 5: *Visualización de la estructura de la proteína 5LKN mediante el software gráfico Chimera.*

La mayor parte de las proteínas tienen formas compactas y globulares, que requieren cambios en la dirección de sus cadenas polipeptídicas. Muchos de estos cambios se acompañan por un elemento estructural común llamado el *giro inverso*. Por otro lado, se encuentran las estructuras denominadas como *bucles*, que a su vez vendrán denotadas por un **guión** (-). Contrariamente a las hélices α y las hebras β, los bucles no tienen estructuras periódicas. Sin embargo, las estructuras tipo bucle son a menudo rígidas y

bien definidas. Los giros y bucles se encuentran invariablemente en la superficie de las proteínas, y por ello participan en interacciones entre las proteínas y otras moléculas [2].

CAPÍTULO 2.

INTELIGENCIA COMPUTACIONAL O *SOFT COMPUTING*.

En el presente capítulo se mencionarán las bases de la inteligencia computacional. Además, se analizarán los diferentes algoritmos utilizados dentro de dichas técnicas, haciendo especial énfasis en las redes neuronales ya que son el algoritmo utilizado para el problema planteado en el trabajo.

2.1. Introducción.

La incertidumbre no existe solo en sistemas naturales con los que estamos familiarizados, sino que también en la mayor parte de sistemas creados por el ser humano que podamos encontrarnos. De hecho, muchos de los sistemas que hoy en día se pueden valorar como "maravillosas creaciones del ser humano" utilizan principios de *Soft Computing*.

Soft Computing es una aproximación emergente a la computación, la cual simultáneamente remarca la habilidad de la mente del ser humano de razonar y aprender en un entorno lleno de incertidumbre e imprecisión. Se trata de una serie de técnicas alternativas para el abordaje de problemas con un cierto grado de complejidad. Fue un término inventado para describir el uso simbiótico de las múltiples disciplinas computacionales emergentes. Los procesos naturales han sido siempre la inspiración detrás del diseño de las técnicas de *Soft Computing*. Éstas tratan de imitar la funcionalidad dada por procesos naturales, especialmente el cerebro humano. Además generan diversos conjuntos de soluciones que pueden ser optimizadas para así devolver la solución óptima [6].
Se consideran parte del término global de *Soft Computing* la lógica difusa (*Fuzzy Logic*), el razonamiento probabilístico (*Probabilistic Reasoning*), las redes neuronales (*Neural Networks*) y los algoritmos genéticos (*Genetic Algorithms*). El denominador común de estas tecnologías es la divergencia con respecto a las clásicas aproximaciones de razonamiento y modelizado, las cuales sueles estar basadas en lógica booleana, modelos analíticos y búsqueda determinista. En formulaciones de problemas idealizados, los sistemas que deben ser modelados o controlados vienen descritos por información completa y precisa. En estos casos, los sistemas de razonamiento formal (así como la demostración automática de teoremas, ATP) pueden ser utilizados para conectar valores binarios verdaderos a sentencias que describen el estado o comportamiento del sistema físico [7].

A lo largo del trabajo se hará uso únicamente de redes neuronales, ya que se trata de la mejor herramienta para lograr predecir la estructura secundaria de las proteínas. Aun así, a continuación vendrán descritas de forma breve y sencilla algunas de las herramientas mencionadas anteriormente.

2.2. Sistemas Fuzzy.

La lógica difusa es una lógica multivaluada que permite representar matemáticamente la incertidumbre y la vaguedad, proporcionando herramientas formales para su tratamiento. Ésta nos permite traducir e incorporar conocimientos cualitativos y empíricos acerca del problema para resolverlo mediante sistemas de razonamiento capaces de desarrollar una búsqueda de patrones aproximados e interpolación.
Los sistemas de lógica difusa son implementados por un conjunto de normas llamadas *reglas difusas*, que son grupos definidos. El procedimiento general a la hora de trabajar con estos sistemas consiste en el modelaje de entrada. Básicamente, cualquier problema del mundo puede resolverse como "dado un conjunto de variables de entrada, obtener un valor adecuado de variables de salida". La lógica difusa permite establecer este mapeo de una forma adecuada, atendiendo criterios de significado y no de precisión [6], [8].

2.3. Algoritmos genéticos.

Los algoritmos genéticos son una familia de modelos computacionales inspirados por la evolución y su base genético-molecular. Estos algoritmos codifican una posible y probable solución a un problema específico mediante una base de datos de cromosomas simples, y aplican operadores recombinacionales para preservar la información crítica. Los cromosomas evolucionan a través de interacciones, también denominadas como generaciones. En cada generación, los cromosomas son evaluados usando alguna medida de aptitud (vienen seleccionados). Las siguientes generaciones son creadas aplicando los operadores genéticos repetidamente a partir de los respectivos operadores (véase *Figura 6*).

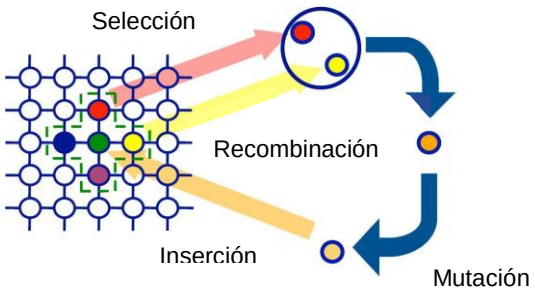


Figura 6: *Esquema en el que se basan los algoritmos genéticos para problemas de poblaciones.*

En el sentido más estricto, los algoritmos genéticos hacen referencia al modelo estudiado y desarrollado por John Holland. Aun así, hoy en día se denomina también como algoritmo genético a todo modelo basado en poblaciones que haga uso de operadores de selección y recombinación, para generar nuevas muestras en un espacio limitado [9].

2.4. Redes neuronales.

Se cree que el cerebro humano está compuesto por millones de pequeñas unidades procesadoras, llamadas neuronas, que trabajan simultáneamente de forma ordenada a través de conexiones sinápticas. Este conjunto, denominado red neuronal, es capaz de llevar a cabo funciones de alta sofisticación a partir de células de gras simplicidad. Una neurona biológica está constituida por un cuerpo denominado *soma*, entradas denotadas como *dendritas*, y una salida denominada *axón* (véase *Figura 7*).

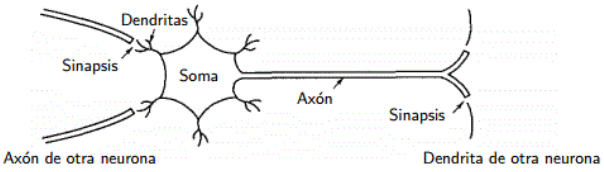


Figura 7: *Modelo simplificado de neurona biológica.*

Cada neurona individual toma las entradas de un conjunto de neuronas que luego procesa y se suministra a la salida a otro conjunto de neuronas, para un posterior procesamiento. El cerebro humano es una red compleja de neuronas en el cual las conexiones se rompen y se forman continuamente. En 1943, McCulloch y Pitts [10] desarrollaron un modelo matemático que emula el comportamiento del sistema nervioso fisiológico, tal y como se muestra en la *Figura 8*. La red neuronal, mediante su simulación biológica, es una nueva arquitectura de computación y de algoritmización relativa a la computación convencional. Permite mediante el uso de simples operaciones computacionales resolver problemas complejos, problemas no lineales o problemas estocásticos.

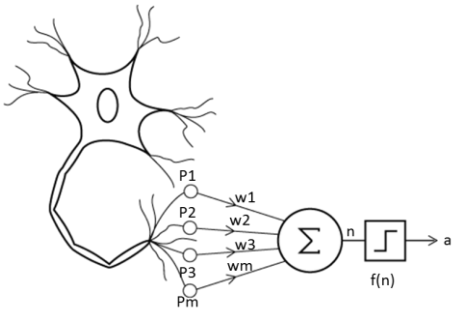


Figura 8: *Modelo de neuronas artificiales obtenido a partir del modelo de neuronas biológicas.*

Las Redes Neuronales comenzaron en los años 60 como algoritmos capaces de entrenar elementos adaptables. Fueron principalmente desarrolladas por Rosenbaltt [11] y Widrow y Hoff [12], y son una tentativa de imitar la funcionalidad del cerebro humano. Se trata de estructuras computacionales que pueden ser entrenadas para descubrir patrones a partir de ejemplos. El sistema comprende los datos mediante la repetición del proceso de aprendizaje para un cierto número de iteraciones. Mediante el uso de una colección de entrenamiento que muestra la relación entre los valores de entrada y de salida, junto

con un algoritmo de retropropagación, las redes neuronales muestran un algoritmo de aprendizaje supervisado que cumple la optimización local [6].

Siguiendo el esquema mostrado en la *Figura 8*, las entradas (p_i) son ponderadas por unos pesos (w_i) y sumadas en el cuerpo, donde se puede añadir un *offset* o *bias* (θ) adicional. Al resultado de la suma ponderada se le aplica una función de activación, dando lugar a una salida de la neurona que se rige por la siguiente ecuación:

$$y = f\left(\sum_{i=1}^{m} p_i w_i + \theta\right)$$

Ecuación 1: *Ecuación que caracteriza las neuronas en el modelo desarrollado por McCulloch y Pitts.*

Una red neuronal multicapa está compuesta con una red de procesamiento de unidades o neuronas. Ésta consiste en una capa de entrada, una de salida y al menos una capa oculta, y el número de unidades en cada capa oculta (también conocidas como neuronas) depende del problema que se esté estudiando.

Estas capas se encuentran interconectadas tal y como se muestra en la *Figura 9* [4]. Cada unidad en la capa de entrada suministra señal a cada unidad de la primera capa oculta. La salida, la cual es calculada mediante una función de transformación, pasa a las unidades de la siguiente capa oculta. De esta forma, las neuronas conectadas forman una red, haciendo que cada neurona represente la suma de los pesos de su entrada, utilizando la suma resultante como argumento de una función de activación no lineal. Originalmente, las funciones de activación venían dadas por funciones umbral bruscas (o del tipo Heavyside), que evolucionaron a funciones lineales de saturación segmentadas, sigmoides y Gaussianas. Para una topología de interconexión dada, las redes neuronales son entrenadas de forma que su vector de pesos minimice la función cuadrática de error [7]. Típicamente, estas redes están divididas en *redes feedforward* y *redes recurrentes*. En la primera de éstas, la información fluye desde la entrada hasta las salidas, sin existir realimentación entre las salidas de las neuronas y las capas anteriores. Sin embargo, la red recurrente puede llegar a contener conexiones realimentadas, de forma que un nodo puede estar unido a los de la capa anterior, e incluso a sí mismo [13].

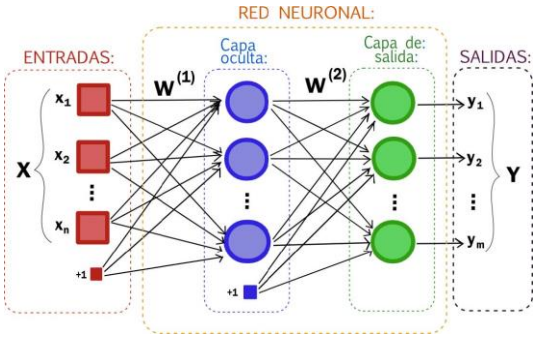


Figura 9: *Arquitectura de una red neuronal multicapa.*

A lo largo del presente trabajo se considerará el aprendizaje de las redes neuronales dentro del contexto de *Soft Computing* explicado anteriormente. Por lo tanto, se limitará el contenido al aprendizaje estructural y paramétrico, siendo éstos los equivalentes a los sistemas de identificación y los parámetros de estimación de la teórica clásica de sistemas. En redes neuronales, el aprendizaje estructural se refiere a la síntesis de la topología de red (por ejemplo, el número de capas ocultas y nudos de la estructura), mientras que el conocimiento paramétrico implica la determinación de los vectores de peso que son asociados a cada enlace en una determinada topología de red [7].

2.4.1. Multilayer Backpropagation Neural Network.

En la redes neuronales multicapa cada una de las capas ocultas de la red juega un rol diferente. Esta característica califica estas redes como muy potentes.
Las redes neuronales más utilizadas son las *feedforward* con el algoritmo de aprendizaje *backpropagation*. Este algoritmo es un proceso iterativo en el que en cada paso del mismo se utiliza el gradiente del error entre la salida deseada y la salida calculada para calcular los valores de los pesos. Es la generalización del aprendizaje propuesto por Widrow y Hoff [12] para redes multicapa y funciones de transferencia diferenciables no lineales. Las muestras de entrada y salida se utilizan para entrenar la red de forma que ésta pueda aproximar una función [14].

Dentro de la búsqueda de los resultados más significativos en las pruebas realizadas con redes neuronales multicapa, se ha observado que la función `trainoss` es aquella con la que se obtienen mejores resultados. Se trata de una variante del algoritmo *backpropagation* que sirve para acelerar el proceso. En ésta, los valores de los pesos y umbrales se van actualizando en base al método de la secante. Una de las características de este algoritmo es que no almacena la matriz H (salida de la capa oculta) al completo, si no que asume que para cada iteración la matriz H previa es la matriz identidad. Esto tiene la ventaja de que la nueva búsqueda de dirección puede ser calculada sin tener que realizar la inversa de la matriz [15].

En las pruebas realizadas, se han utilizado todas las proteínas de la base de datos como muestras tanto de entrada como de salida. Esto se debe a que MATLAB, mediante la herramienta de redes neuronales `nntool`, realiza una división automática de las muestras, utilizando un 70% de éstas para el entrenamiento, el 15% para la validación, y el otro 15% para el test. De esta forma, durante el entrenamiento, los pesos van variando mientras el error decrece, y una vez se encuentra el mínimo de error de validación (el cual es indicado en los *Anexos VI y VII*), los pesos correspondientes a dicha iteración servirán para la simulación en la que obtendremos los resultados predichos por la red neuronal.
Por otro lado, los parámetros de red fijados durante las pruebas son los siguientes:

- red multicapa de tres capas (dos capas ocultas y la de salida).
- función de entrenamiento `trainoss`.
- 19 neuronas por capa oculta.

- `tansig` como función de transferencia entre capas ocultas.
- Error cuadrático medio normalizado (`MSE`) como función de actuación de red.

Posteriormente, se han realizado análisis más detallados sobre el resultado más valioso, pudiendo especificar con más exactitud los parámetros anteriores. Asimismo, esté análisis será detallado en el capítulo de *Resultados y discusión.*

2.4.2. Extreme Learning Machine.

Las máquinas de aprendizaje extremo o ELM resultan muy atrayentes debido a la superación de algunos problemas que presentan otras técnicas, como por ejemplo, su lenta velocidad de aprendizaje. Además, otra de las ventajas que presenta este tipo de red neuronal es que no tiene mínimos locales, por lo que no da lugar a resultados erróneos. Estas máquinas trabajan principalmente con redes monocapa de tipo *feedforward*, también conocidas con SLFN (*Single-hidden Layer Feedforward Network*) [16].
Una vez que los pesos de entrada y los umbrales de la capa oculta son elegidos de forma aleatoria, esta red puede considerarse como un sistema lineal donde los pesos de la salida se pueden determinar analíticamente mediante la inversa generalizada de una matriz. Esta asignación aleatoria se puede hacer si las funciones de activación en la capa oculta son infinitamente diferenciables, siendo la más popular de éstas la función sigmoide. En este trabajo se han realizado pruebas tanto con la función de transferencia *Logarítmica Sigmoidal* (`logsig`) y la función *Tangente Hiperbólica* (`tansig`, véase la *Figura 10*).

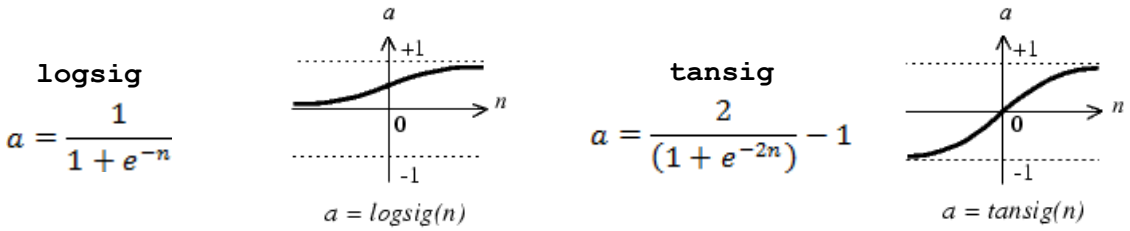


Figura 10: *Expresión matemática y representación gráfica de las funciones de transferencia más comunes en las SLFN.*

A continuación se procederá con el desarrollo matemático que rige las ELM y su interpretación en MATLAB. Para un número arbitrario *N* de muestras $(\mathbf{x_i}, \mathbf{t_i})$ (entrada y salida de la red respectivamente), donde $\mathbf{x_i} = [x_{i1}, x_{i2},\ldots, x_{in}]^T \in \mathbf{R}^n$ y $\mathbf{t_i} = [t_{i1}, t_{i2},\ldots, t_{im}]^T \in \mathbf{R}^m$, la red neuronal ELM con $\tilde{N}$ neuronas en la capa oculta y una función de activación *g(x)* (de las ya mencionadas anteriormente) viene matemáticamente modelizada como:

$$\sum_{i=1}^{\tilde{N}} \beta_i g(\boldsymbol{w}_i \cdot \boldsymbol{x}_i + b_i) = \boldsymbol{o}_j$$

Ecuación 2: *Modelo matemático utilizado para las SLFN.*

En esta ecuación, $j=1,\ldots,N$ recorre todas las muestras y $\mathbf{w_i} = [w_{i1}, w_{i2},\ldots, w_{in}]^T$ es el vector de pesos que conecta la *i*-ésima neurona de la capa oculta con cada una de las entradas, $\boldsymbol{\beta_i} = [\beta_{i1}, \beta_{i2},\ldots, \beta_{im}]^T$ es el vector de pesos que conecta la *i*-ésima neurona de la capa oculta con las salidas, mientras que b_i es el valor umbral de la *i*-ésima neurona de la capa oculta. De esta forma, en caso de que el error sea nulo, las redes ELM con $\tilde{N}$ neuronas en la capa oculta se aproximan como sigue:

$$\sum_{i=1}^{\tilde{N}} \beta_i g(\boldsymbol{w}_i \cdot \boldsymbol{x}_j + b_i) = \boldsymbol{t}_j$$

Ecuación 3: *Modelo matemático de las redes SLFN en caso de no existir error.*

Asimismo, las *N* ecuaciones descritas anteriormente puedes ser representadas de forma compacta mediante la siguiente ecuación lineal:

$$\boldsymbol{H}\beta = \boldsymbol{T}$$

Ecuación 4: *Expresión compacta de la ecuación que rige la SLFN.*

Cada una de las matrices anteriores viene definida como:

$$\boldsymbol{H} = \begin{pmatrix} g(\boldsymbol{w}_1 \cdot \boldsymbol{x}_1 + b_1) & \cdots & g(\boldsymbol{w}_{\tilde{N}} \cdot \boldsymbol{x}_1 + b_{\tilde{N}}) \\ \vdots & \ddots & \vdots \\ g(\boldsymbol{w}_1 \cdot \boldsymbol{x}_N + b_1) & \cdots & g(\boldsymbol{w}_{\tilde{N}} \cdot \boldsymbol{x}_N + b_{\tilde{N}}) \end{pmatrix}$$

Ecuación 5: *Definición matemática de la matriz* ***H****, matriz de salida de la capa oculta de la red.*

$$\beta = \begin{pmatrix} \beta_1^T \\ \vdots \\ \beta_{\tilde{N}}^T \end{pmatrix}_{\tilde{N}xm}$$

Ecuación 6: *Matriz de pesos que conecta la capa oculta con la salida.*

$$\boldsymbol{T} = \begin{pmatrix} \boldsymbol{t}_1^T \\ \vdots \\ \boldsymbol{t}_N^T \end{pmatrix}_{Nxm}$$

Ecuación 7: *Matriz de salida de la red.*

Tal y como se presenta en Huan et. al [17], **H** viene denotada como la matriz de salida de la capa oculta dentro de la red, por lo que la *i*-ésima columna de **H** hará referencia a la salida de la *i*-ésima neurona de la capa oculta con respecto a las entradas $\mathbf{x}_1, \mathbf{x}_2,\ldots, \mathbf{x}_N$.

Una vez se obtiene la ecuación, se continúa con el entrenamiento y prueba de la red. De modo que, para ello, se tendrán dos tipos de muestras, por un lado las de entrenamiento, y por otro lado las de ensayo. En primer lugar, con las muestras de entrenamiento (tanto entradas como salidas) se obtendrán los pesos que conectan la capa oculta con la salida mediante la siguiente ecuación:

$$\beta = \boldsymbol{H}^{\dagger}\boldsymbol{T}$$

Ecuación 8: *Expresión matemática para la obtención de los pesos que conectan la matriz de salida de la capa oculta con las salidas de la red SLFN.*

La inversa de la matriz **H** viene calculada mediante la inversa generalizada de Moore-Penrose a través del comando `pinv`. A continuación, se vuelve a calcular **H** para las muestras de ensayo (vendrá representada por $\mathbf{H}^{*}$), y a partir de la *Ecuación 4* y la matriz β (calculada anteriormente a partir de las muestras de entrenamiento) se obtendrán las salidas predichas $\mathbf{T}^{*}$ por la red neuronal:

$$\boldsymbol{H}^{*}\beta = \boldsymbol{T}^{*}$$

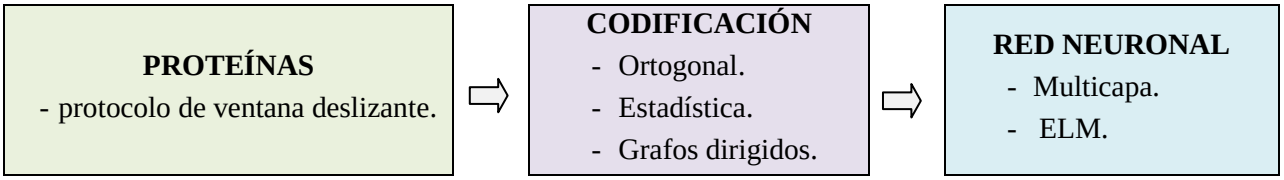
Ecuación 9: *Expresión matemática para la obtención de las salidas predichas por la red SLFN una vez conocidos los pesos β.*

Los resultados obtenidos mediante la predicción de la red ELM vendrán comparados con los resultados originales (***T*** de las muestras de ensayo), obteniendo así un porcentaje de reconocimiento. Como se puede observar en el *Anexo I*, el programa realizado para la obtención de la predicción de la estructura secundaria de las proteínas de la base de datos RS126 sigue el esquema analizado anteriormente, para el cual el número de muestras de entrenamiento y de prueba es el mismo, esto es, el conjunto completo de proteínas. Más adelante, en la sección número cuatro se expondrán los resultados obtenidos junto con un análisis de los mismos.

CAPÍTULO 3.

MODELIZACIÓN.

En este capítulo se disponen los datos con los que se han realizado las pruebas de predicción de la estructura secundaria de las proteínas. Además, se explican tanto los tres modelos de codificación implantados en el trabajo (ortogonal, estadístico y mediante grafos dirigidos, que representarán las muestras de entrada) como las muestras de salida, en los que los datos han sido procesados de diferente forma. El esquema que vendrá seguido será el siguiente:



3.1. Base de datos.

En este estudio se ha hecho uso de la base de datos de proteínas más conocida en el ámbito, la RS126. Este conjunto de datos consta de 126 secuencias no homólogas, en las que la similitud de cada par de secuencias no supera el 25% cuando la longitud de éstas alcanza más de los 80 residuos de aminoácidos. Asimismo, la RS126 contiene aproximadamente 24000 residuos, con un promedio de 185 residuos de aminoácidos por secuencia [18]. Además en la base de datos se encuentra la estructura secundaria de cada uno de los aminoácidos que forman las proteínas, siendo éstas de forma **E, H** y **– (guión)**, tal y como se mencionó anteriormente. A lo largo del trabajo, se realizarán los diferentes modelos teniendo en cuenta el análisis de las secuencias de proteínas mediante PHD (*Profile neuronal Network systems from HeiDelberg)*, ya que éste ha mostrado los mejores resultados en los estudios analizados en la bibliografía [19].

3.2. Herramienta de programación.

Para la realización de la red neuronal capaz de lograr los objetivos descritos anteriormente, se ha hecho uso de MATLAB, un lenguaje de programación interactivo que se está utilizando cada vez más en aquellas áreas científicas en las que es necesario llevar a cabo cálculos numéricos de gran envergadura.
MATLAB es un potente lenguaje diseñado para la computación técnica. El nombre MATLAB proviene de *Matrix Laboratory*, dado que el tipo básico que gestiona es una matriz. MATLAB puede ser utilizado en computación matemática, modelado y simulación, análisis y procesamiento de datos, visualización y

representación de gráficos, así como para el desarrollo de algoritmos [20]. Al ser un lenguaje interactivo, el usuario no necesita realizar el programa de compilación requerido por la mayoría de lenguajes de programación. Además, MATLAB, tiene la ventaja de poseer un gran número de rutinas internas que permiten realizar las operaciones y cálculos diversos [21].

3.3. Organización y procesado de datos.

La base de datos RS126 proporciona cada una de las proteínas en diferentes archivos de texto. Cada uno de éstos contiene información específica de la proteína, así como diversos tipos de secuencias de aminoácidos, y múltiple información acerca de la estructura secundaria de cada una de ellas predicha a partir de diferentes métodos (*DSC*[1], *ZPRED*[2], *MUL*[3], *PRED*[4], *PHD*[5]). Para la conversión de estos datos, se ha creado un programa mediante el software MATLAB que ha ido leyendo cada uno de los archivos y ha seleccionado la información definida por el usuario. De esta forma, tal y como se muestra en el *Anexo II* se crean dos matrices de caracteres (*MATp* y *MATs*) que contienen todas las proteínas y sus estructuras secundarias respectivamente. De esta forma, una vez se obtienen las matrices con cada una de las proteínas y estructuras secundarias con las que se va a trabajar, y una vez conocida su disposición, es mucho más sencillo su posterior tratamiento.

A continuación se analizan los tres esquemas de codificación creados para las diversas pruebas realizadas mediante redes neuronales. Cada una de las representaciones matriciales contiene información particular de la proteína analizada, pudiendo llegar a proporcionar información muy específica de ésta. Además, en cada uno de los tres esquemas se utilizará el protocolo de ventana (o secuencia) deslizante, ya que nos permite transmitir múltiples paquetes de información. Asimismo, el tamaño de la ventana será un número impar, ya que lo que se busca es predecir la estructura secundaria determinada por el aminoácido central. En este trabajo se han creado matrices a partir de las siguientes longitudes de ventana: *k=3, k=7, k=11, k=15* y *k=19.*

3.4. Codificación de los datos.

Las muestras de entrada para la obtención de resultados vienen dadas por tres matrices, conteniendo cada una de éstas información particular acerca de la proteína (cada una de las codificaciones en MATLAB viene dada en el *Anexo III*). De esta forma, cada matriz contendrá en filas y columnas los valores de entrada de cada red neuronal.

---

[1] DSC: obtención de la estructura secundaria a partir de discriminación lineal.
[2] ZPRED: obtención de la estructura secundaria mediante la predicción del número de conservación de los pesos.
[3] *MUL*: obtención de la estructura secundaria mediante combinación de secuencias simples.
[4] *PRED*: obtención de la estructura secundaria a partir de la tendencia de adherencia del hidrógeno.
[5] *PHD*: obtención de la estructura secundaria a partir del perfil de red de HeiDelberg.

3.4.1. Codificación ortogonal.

El esquema de codificación ortogonal, sugerido por Holley y Karpus [22], usa los dígitos binarios 0 y 1 para representar un aminoácido. La codificación de datos propone la conversión de aminoácidos, los cuales vienen representados por un simple código de una única letra, a un equivalente numérico.

Para una determinada fila en la matriz, la representación de un aminoácido viene dada por 1, mientras que el resto de entradas viene marcado por 0. Es decir, teniendo los aminoácido clasificados en el siguiente orden, [A, C, D, E, F, G, H, I, K, L, M, N, P, Q, R, S, T, V, W, Y], y refiriéndonos al aminoácido D, el vector correspondiente a éste vendría dado por V_D = [0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]. Además, a lo largo del trabajo se utilizará el protocolo de ventana (o secuencia) deslizante. De esta forma, los datos de entrada están subdivididos por ventanas correderas etiquetadas. Estas etiquetas ayudan en la arquitectura del diseño de la red neuronal, donde el número de unidades o neuronas de la capa de entrada es creado en función del tamaño de ventana.

El esquema mencionado anteriormente proporciona los datos de entrada en formato matricial tal y como se explica a continuación. Mientras la ventana se desliza por cada uno de los aminoácidos, y en cada uno de éstos deslizamientos, se va a crear una matriz de paso, cuyo número de filas corresponderá a la longitud de la ventana, y el número de columnas corresponderá a 20, el número de aminoácidos totales (véase *Figura 11*). Nótese que cada una de las filas está formada por los vectores ortogonales mencionados anteriormente, en los que cada uno de los aminoácidos ocupa una posición invariante en todo el proceso.

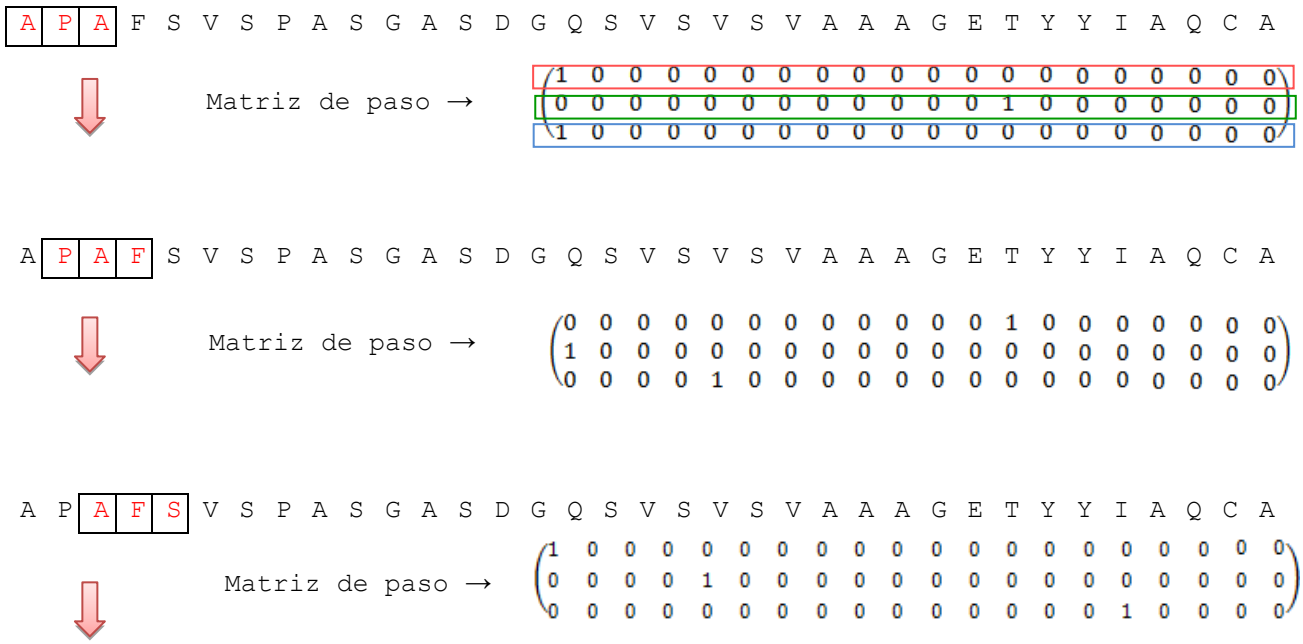


Figura 11: *Esquematización del protocolo de ventana deslizante utilizado para la obtención de las matrices de paso en función de la longitud de ventana para un ejemplo de longitud 3.*

Posteriormente, cada una de estas matrices de paso, se irá colocando en forma de fila de la matriz final, tal y como se muestra en la *Figura 12*. Por tanto, la matriz final tendrá como filas el número de deslizamientos realizados a lo largo de las proteínas de la base de datos, y como columna el número correspondiente a *k x* 20, es decir, el producto entre la longitud de la ventana y los 20 aminoácidos.

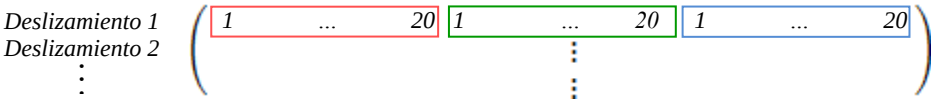


Figura 12: *Esquematización de la formación de la matriz final mediante la representación ortogonal a partir de las matrices de paso creadas en cada deslizamiento.*

3.4.2. Codificación estadística.

Para el esquema de codificación estadístico, se va a realizar un proceso similar al modelo presentado anteriormente, pero para no crear confusiones en la notación con la estructura secundaria de tipo bucle (ya que viene designada como un guión en la base de datos), se hará uso de la letra **C** en lugar de dicho símbolo (del inglés, *coil*). En esta ocasión, la matriz de paso creada para cada uno de los deslizamientos de la ventana no vendrá codificada mediante los dígitos binarios 0 y 1. Asimismo, para la representación de cada uno de los aminoácidos, el vector V_i correspondiente a uno de los veinte aminoácidos vendrá multiplicado por la siguiente magnitud:

$$\left(\frac{N_{ix}}{min\{N_{Hx}, N_{Ex}, N_{Cx}\}}\right) \cdot \left(\frac{N_{ix}}{N_i}\right)$$

Ecuación 10: *Expresión por la que vendrá multiplicado cada uno de los vectores ortogonales para la formación de la matriz de paso.*

Esta magnitud representa la información estadística de los aminoácidos en función de su estructura secundaria. La ventana que vendrá analizada, y por ende el conjunto de aminoácidos contenidos en la estructura, vendrá representada por *x*, mientras que la referencia a la estructura secundaria de un único aminoácido dentro de dicha ventana vendrá dado por $i \in \{H, E, C\}$.
Asimismo, $\mathbf{N_H}$, $\mathbf{N_E}$ y $\mathbf{N_C}$ denotan, respectivamente, el número de aminoácidos que tienen H, E y C como estructura secundaria a lo largo de la proteína completa. Por otro lado, $\mathbf{N_{Hx}}$, $\mathbf{N_{Ex}}$ y $\mathbf{N_{Cx}}$ representan el número de aminoácidos que contienen H, E y C respectivamente como estructura secundaria dentro de la ventana que se esté analizando (véase *Figura 13*).

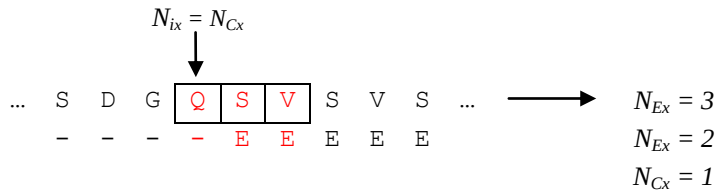


Figura 13: *Esquema de la obtención de los parámetros estadísticos dentro de la sección analizada.*

Cabe destacar que cada uno de los parámetros mencionados anteriormente, durante su ejecución en MATLAB, es inicializado a 1 debido a que cabe la posibilidad que una de las estructuras secundarias no se encuentre en la proteína estudiada y por tanto la división por 0 resultaría error.

Siguiendo el proceso anterior y teniendo en mente el ejemplo expuesto en las *Figuras 11 y 12*, se puede observar que este modelo copia el desarrollo del método ortogonal, solo que en vez de tratar con datos binarios (0 y 1) se obtendrán matrices de paso de la misma forma y dimensión pero con números decimales obtenidos a partir de la *ecuación 10* en lugar de unos. Con cada una de las matrices de paso se formará la matriz final correspondiente a la representación estadística, en la que el número de filas vendrá dado por el número total de deslizamientos de ventana en todas las proteínas de la base de datos, y el número de columnas vendrá dado por *k x 20*, siendo *k* el tamaño de la ventana.

De esta forma, la matriz final correspondiente al modelo estadístico vendrá formada por dicha información adicional, en la que se combinan dos tipos de información. Por un lado la probabilidad de que el residuo de un aminoácido aparezca en esas tres estructuras, y por otro lado la información de que cada residuo de aminoácido aparezca en la misma estructura [23].

3.4.3. Codificación mediante grafos dirigidos.

Este nuevo modelo propone una codificación raramente usada pero que también ha sido considerada en este trabajo. La premisa que se utiliza para este método es que las funciones biológicas tienden a preservarse a través de las proteínas que se encuentran relacionadas en organismos interconectados. Como ya se ha mencionado anteriormente, el flujo de información genética transcribe del ADN al ARN para traducirse en proteínas. El código es prácticamente el mismo para ambos polímeros pero el RNA difiere del ADN en dos aspectos. En primer lugar las unidades de azúcar son diferentes, y además una de las cuatro bases principales del RNA es el uracilo (U) en lugar de la timina (T). Por tanto, se propone un nuevo esquema de codificación acorde al codón genético que está formado por bloques de aminoácidos. Cada codón es una combinación de tres nucleótidos seleccionados de forma natural a partir del set {A, U, G y C} [24].

Un codón es un triplete de nucleótidos, y en el código genético, cada aminoácido está codificado por uno o varios codones. Por ejemplo, en el caso de la Valina, esta viene derivada por los codones del set {GUU,

GUC, GUA, GUG}. A partir de aquí se realiza el grafo y se obtiene la matriz correspondiente, tal y como se muestra en la *Figura 14*.

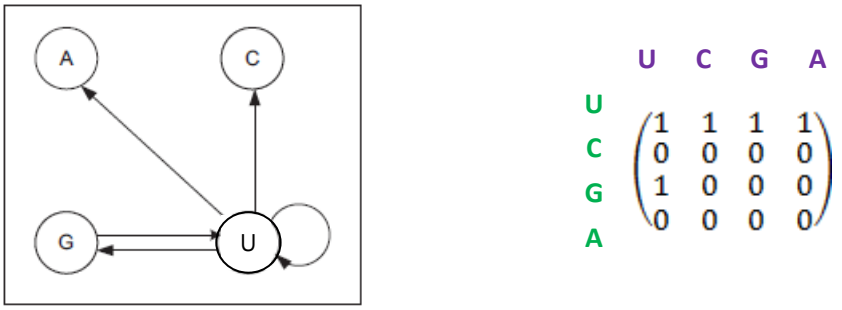


Figura 14: *Representación del grafo y matriz correspondiente a los codones que forman el aminoácido de la Valina, basado en los cuatro nucleótidos {A, U, G, C}.*

Como se puede observar en el ejemplo anterior, dado que se trata de grafos dirigidos, las aristas que unen los nucleótidos siguiendo la codificación de codones tendrán un sentido definido. De esta forma los nucleótidos marcados en verde muestran los puntos de partida de las aristas y los nucleótidos marcados en morado el final de éstas. Por tanto si se tiene U → G, la matriz marcará como 1 la posición que tenga U como fila y G como columna.

Se realizan los grafos y matrices correspondientes a cada aminoácido, y posteriormente éstos vienen se convierten en vectores de 16 parámetros de ceros y unos (véase *Anexo IV*). Una vez obtenido el vector correspondiente a cada uno de los aminoácidos, se continúa con la formación de las matrices de paso. Al igual que en los dos modelos analizados anteriormente, ortogonal y estadístico, la realización de las matrices de paso se va a realizar de la misma forma, mediante el protocolo de ventana deslizante. La única diferencia es que para este modelo el vector que representa cada aminoácido no es un vector ortogonal o estadístico de *1 x 20*, si no que será un vector de *1 x 16* que contiene información de codificación genética. Asimismo, las matrices de paso creadas serán de dimensión *k x 16*, siendo *k* el tamaño de ventana, y formarán una matriz final cuyo número de filas vendrá dado por los deslizamientos totales realizados en todas y cada una de las proteínas de la base de datos y el número de columnas vendrá dado por la cantidad de parámetros de cada matriz de paso, esto es, *k x 16*.

3.5. Muestras de salida.

Al contrario que en las representaciones matriciales de entrada propuestas anteriormente, para las muestras de salida y su representación matricial bastará con un único método, con el que se obtendrán diferentes matrices en función de la longitud de ventana que se elija.

Para ello, se seguirá básicamente el modelo ortogonal. Pero dado que el objetivo es el de predecir la estructura secundaria del aminoácido central de la ventana, esta vez, en lugar de codificar cada una de las estructuras de la ventana, se codificará únicamente el central.
Asimismo, y tal y como se muestra en la *Figura 15*, la matriz final se va creando directamente mediante la detección de la estructura secundaria central de la ventana. Como se puede observar en el *Anexo V*, al leer cada deslizamiento, en la matriz final se coloca un 1 en la posición que coincide con la estructura secundaria, siendo la primera columna la estructura **E** (correspondiente a las hojas β), la columna central la **H** (correspondiente a las hélices α), y la columna final el **guión** (correspondiente a los bucles).

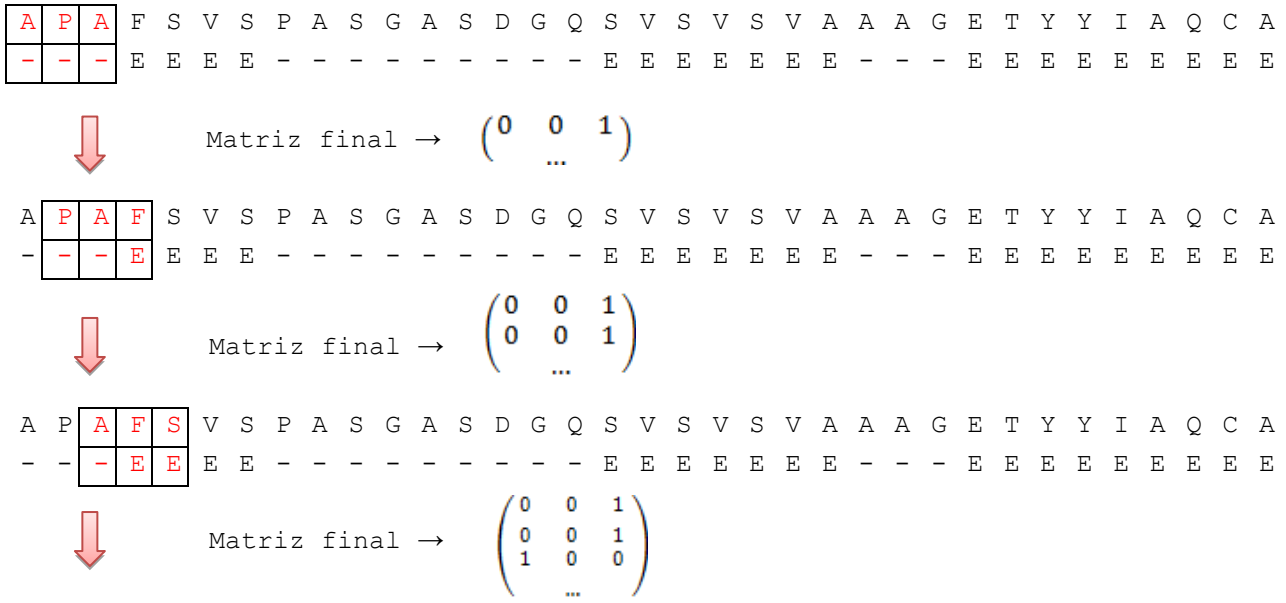


Figura 15: *Esquematización del protocolo de ventana deslizante utilizado para la obtención de la matriz de salida en función de la longitud de ventana para un ejemplo de longitud 3.*

Finalmente se obtiene una matriz de salida que tendrá como número de filas la cantidad total de deslizamientos realizados a los largo de todas las proteínas y estructuras, y tres columnas, correspondientes a cada una de las estructuras secundarias posible (E, H o -). Por tanto, en el caso de las matrices de salida, es únicamente el número de filas el que variará dependiendo de la longitud de la ventana deslizante, mientras que el número de columnas se mantendrá fijo.

3.6. Experimentación.

Continuando con el trabajo y teniendo en mente en todo momento el objetivo principal del mismo, se han realizado dos tipos de estudios a partir de las matrices formadas y ya estudiadas anteriormente. En primer lugar se propone un estudio a partir de ELM (*Extreme Learning Machine*), ya que se trata de un algoritmo sencillo y muy veloz a pesar de la cantidad de datos obtenidos (nótese que las matrices finales constan de millares de datos cada una). Posteriormente, y tras la visualización de los resultados, se pasa a un estudio de redes neuronales multicapa mediante la extensión de MATLAB a partir de la herramienta `nntool`, por el cual se han obtenido resultados muy satisfactorios.

En los ensayos realizados se han utilizado las muestras de entrada de diversas formas. Por un lado, para el entrenamiento de la red ELM se ha hecho uso de la mitad de las proteínas para el entrenamiento de la red y la otra mitad para su simulación. Sin embargo, en el caso de las redes multicapa, se ha hecho uso de todas las proteínas como muestras de entrada, ya que como se ha mencionado anteriormente, ésta divide y trabaja con las muestras de la siguiente forma: el 70% las usa como entrenamiento, el 15% para validación, y el otro 15% para simulación.
El análisis pormenorizado de los experimentos mencionados anteriormente sr muestra con más detalle en el capítulo siguiente.

CAPÍTULO 4.

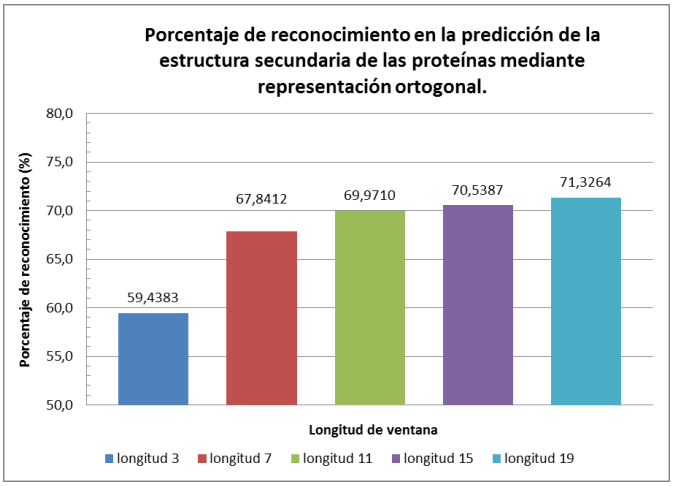
RESULTADOS Y DISCUSIÓN.

Tras la realización de las pruebas experimentales, se obtienen los resultados definitivos. Por una parte, se logran los valores del porcentaje de reconocimiento en la predicción de la estructura secundaria de las proteínas (objetivo principal del trabajo), y de igual se analizará el comportamiento de la red durante el aprendizaje.

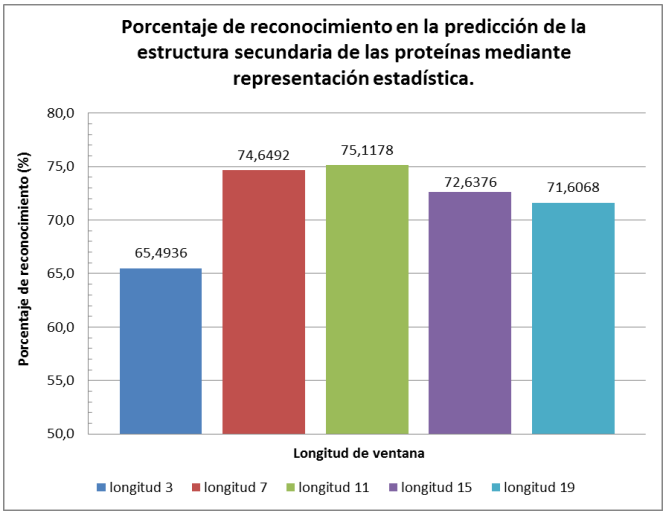
4.1. Porcentajes de reconocimiento para la predicción de la estructura secundaria de las proteínas.

En primer lugar, cabe destacar que se toman como punto de referencia los resultados obtenidos por los diferentes estudios analizados en la bibliografía, en los que en el mejor de los casos se obtiene un 71% de precisión en la predicción de la estructura secundaria de las proteínas [4], [19].
Siguiendo un exhaustivo análisis de cada una de estas metodologías, se ha concluido que la predicción a partir de las máquinas de aprendizaje extremo (ELM) no es válida, ya que los porcentajes obtenidos no son para nada significativos. Sin embargo, tras realizar las pruebas correspondientes a las redes neuronales multicapa, se han obtenido resultados que igualan e incluso mejoran los analizados en la bibliografía. Tal y como se puede observar en el *Anexo VI*, gracias a la implementación de diferentes representaciones matriciales, y por tanto la adición de información más explícita acerca de la proteína, se han obtenido resultados que superan hasta en 5 puntos de porcentaje los resultados previstos.

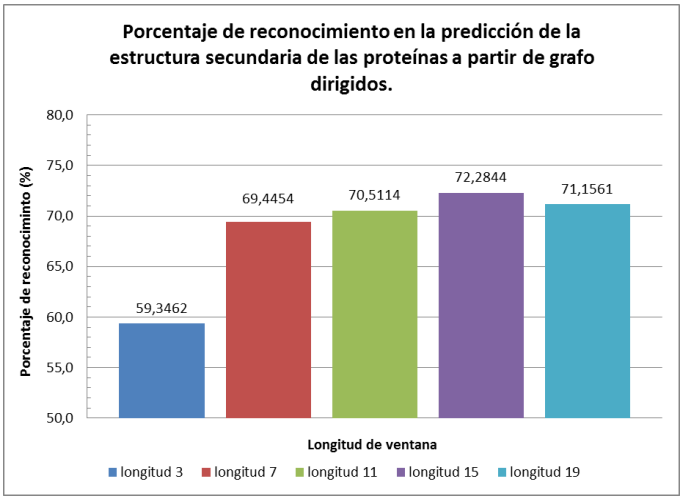
En la primera consecución de resultados se realiza un barrido en iteraciones para cada una de las longitudes de ventana mencionadas en el trabajo para una estimación inicial del resultado que se va a obtener. Como se puede ver en las tablas de resultados (véase *Anexo VI*), para ciertos casos, los resultados resultan ser muy disparen en función de las iteraciones seleccionadas, por lo que se deduce que el número de iteraciones que realiza la red es un parámetro fundamental para conseguir la red óptima. En las gráficas pertenecientes a la *Figura 16* se puede observar cómo varían los porcentajes medios obtenidos en el barrido de iteraciones en función de la longitud de ventana y del esquema de codificación seguido.



(a)



(b)

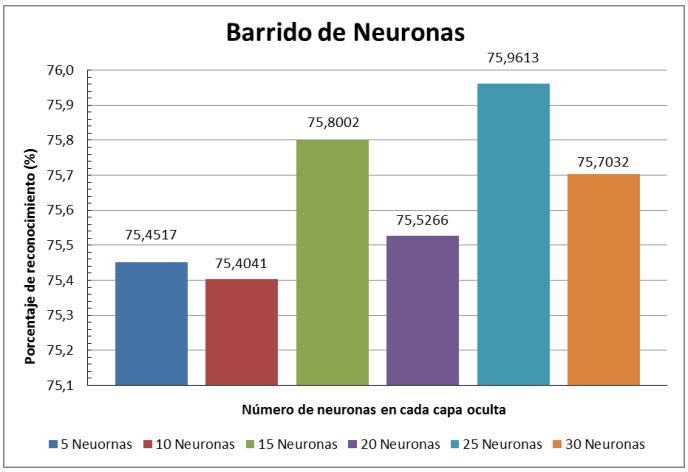


(c)

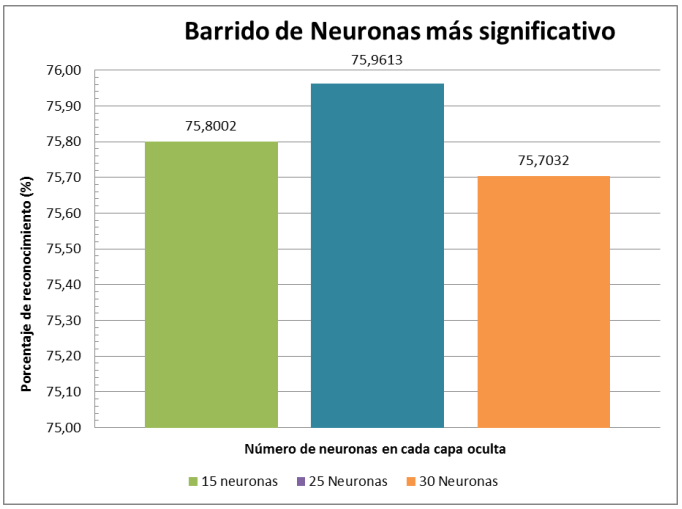
Figura 16: *Tablas de porcentaje de reconocimiento media en la predicción de la estructura secundaria de la proteína en función de la longitud de ventana para (a) representación ortogonal, (b) representación estadística y (c) a partir de grafos dirigidos.*

Cabe destacar que en las gráficas anteriores se han seleccionado los datos más significativos para cada grupo de valores. Estos es, en el caso de haber algún dato muy dispar al resto (aquellos señalados en rojo en las tablas de resultados del *Anexo VI*), éste es suprimido a la hora de realizar el promedio, de forma que dicho valor sea lo más próximo a la media real posible.

Una vez obtenida esta primera etapa de resultados, se pasa a profundizar aún más en el mejor resultado logrado, es decir, una red neuronal de tres capas con 19 neuronas por capa oculta que tiene como entrada la representación matricial estadística formada por un protocolo de deslizamiento de longitud de ventana 11. Teniendo fijos estos parámetros a excepción del número de neuronas por capa oculta, se va a realizar un barrido neuronal en un rango de 5 a 30 (en intervalos de 5) manteniendo un número de 15000 iteraciones, ya que como se puede observar en el *Anexo VII*, con este número de interacciones se obtienen los mejores resultados (véase *Figura 17*).



(a)



(b)

Figura 17: *Tablas de porcentaje de reconocimiento media en la predicción de la estructura secundaria de la proteína a partir de un barrido de neuronas en intervalos de 5, para una representación matricial estadística de las muestras. (a) Resultados generales obtenidos y (b) valores más significativos.*

Finalmente, se puede concluir que la red neuronal óptima para un porcentaje de reconocimiento de la estructura secundaria de las proteínas de aproximadamente 76%, es una red neuronal multicapa *backpropagation* de 3 capas ocultas, con 25 neuronas en cada capa, y muestras de entrada que contengan información estadística acerca de cada una de las proteínas.

4.2. Análisis de entrenamiento.

En esta sección se analizarán los resultados obtenidos desde el punto de vista del error de red. La referencia empleada para que el error en los algoritmos de entrenamiento supervisado converja rápidamente hacia un mínimo es el MSE (*Mean Squared Error*), error cuadrático medio normalizado. Se trata de una búsqueda para obtener el valor mínimos posible de la suma de los errores cuadrados de las neuronas de salidas en cada muestra, donde se establece un mínimo del MSE en el cual se dejan de actualizar los pesos y se da por terminado el entrenamiento de una red neuronal.

El primer resultado inmediato que se extrae de los datos obtenidos en las pruebas experimentales (véase la *Figura 18*), es que el error presenta una función cóncava en dependencia con la longitud de ventana deslizante, obteniendo el mínimo en *k=11*. Por tanto, se puede establecer que el resultado óptimo vendrá dado por dicha longitud de secuencia, tal y como ya se había deducido de los resultados obtenidos a partir del porcentaje de reconocimiento. Además, cabe destacar que valor mínimo de error se alcanza también mediante la representación estadística de las muestras de entrada.

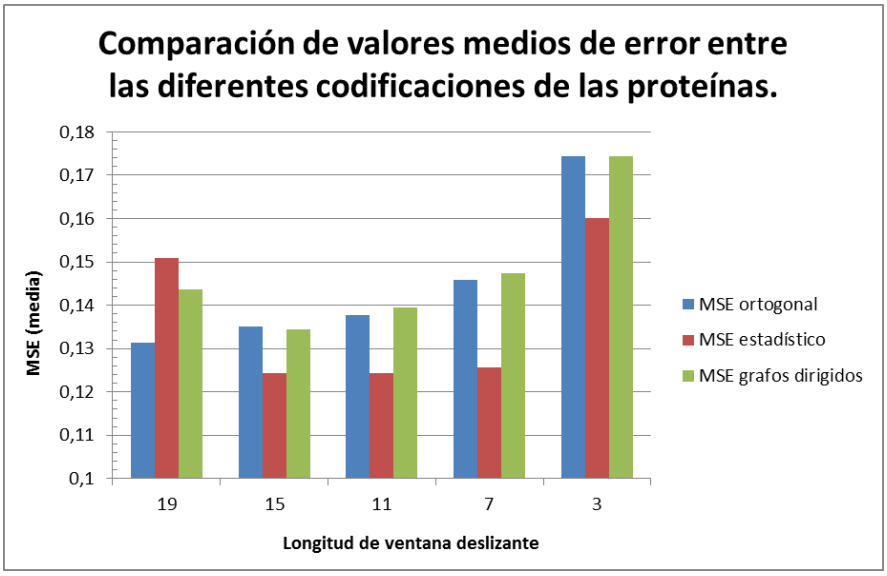


Figura 18: *Gráfica de comparación de errores cuadráticos medios normalizados para las diferentes codificaciones de las proteínas: ortogonal (azul), estadística (rojo) y a partir de grafos dirigidos (verde).*

Teniendo en cuenta los resultados obtenidos, se va a analizar el valor del error en función de las iteraciones, de forma que se tenga una idea del funcionamiento de red y de los valores exactos que utiliza. Tal y como se puede observar en la *Figura 19*, al realizar la representación gráfica del error a lo largo de las iteraciones para una longitud de ventana de 11, se ve que el valor mínimo del error de validación se obtiene en la iteración 14488. Por tanto, a la hora de realizar la simulación, son los pesos de dicha iteración los que quedan almacenados para su uso. Por tanto, el valor del error cuadrático para la red utilizada en la prueba de la *Figura 19* será de 0.1322 con un porcentaje de reconocimiento del 75%.

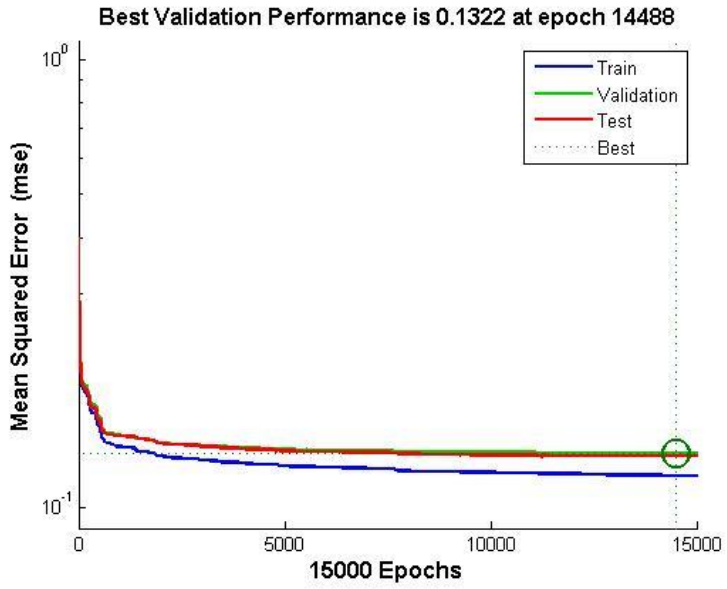
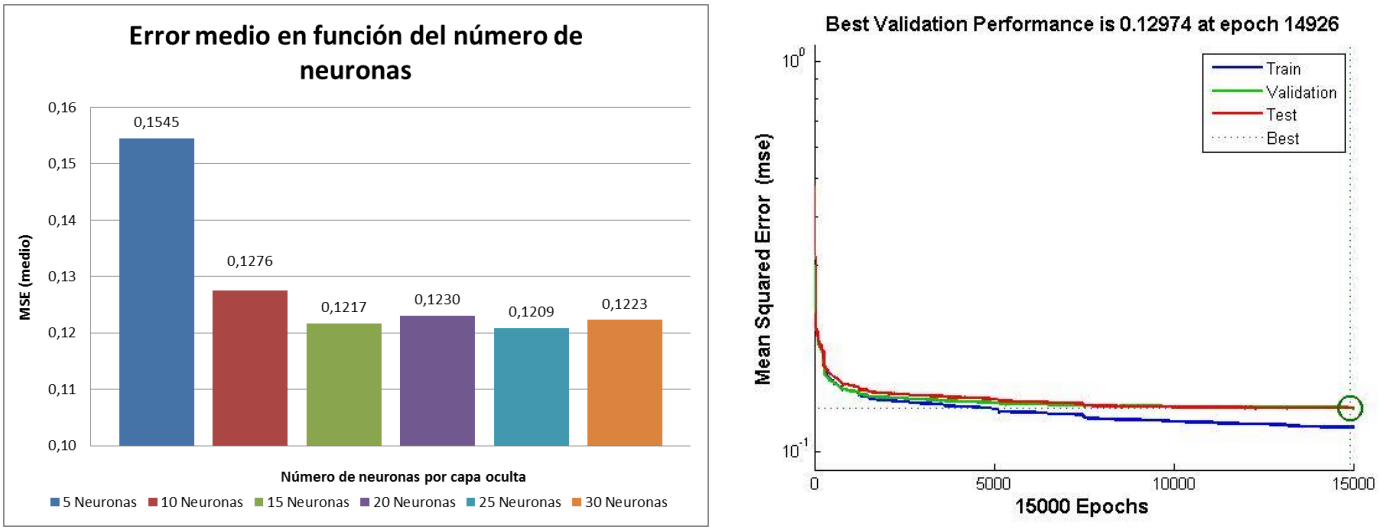


Figura 19: *Gráfica del comportamiento de red para k=11y 19 neuronas por capa oculta en función del número de iteraciones. Mediante un círculo verde viene destacado el error mínimo de validación.*

Por otro lado, analizando los valores de error obtenidos durante estudio del barrido neuronal (véase *Figura 20*), se puede observar que, al igual que en los resultados de reconocimiento, el error mínimo de red se obtiene para un número de 25 neuronas por capa oculta. Además, si se analiza la evolución de entrenamiento, se obtiene un error de red correspondiente a un porcentaje de reconocimiento de la estructura secundaria de las proteínas de poco más de un 76%, valor máximo alcanzado en el trabajo, y el cual supera a los encontrados en bibliografía.



(a) *(b)*

Figura 20: *Graficas de barrido neuronal: (a) comparación de los errores medios de red y (b) comportamiento de red para k=11 y 25 neuronas por capa oculta en función de las iteraciones.*

CAPÍTULO 5.

CONCLUSIONES.

El presente trabajo propone un sistema basado en redes neuronales para la predicción de la estructura secundaria de proteínas a partir de tres métodos de codificación diferentes: ortogonal, estadístico y mediante grafos dirigidos, utilizando RS126 como base de datos. Tras el exhaustivo desarrollo experimental, se ha llegado a que la red neuronal óptima para dicha predicción es una red multicapa (de 3 capas) *backpropagation* implementada por el algoritmo de entrenamiento de la secante. Además, estando dicha red formada por muestras de entrada obtenidas a partir de secuencias de longitud fija 11 y mediante la codificación de la información estadística de cada proteína dentro de la base de datos, se consigue superar en 5 puntos de porcentaje los valores de predicción de la estructura secundaria de las proteínas obtenidas en el resto de estudios hasta el momento.
Por otro lado, se ha realizado un análisis de los valores de error de la red utilizada, y cabe destacar que los resultados más significativos de error se han obtenido también a partir de las muestras de entrada que contienen información estadística acerca de la proteína y de la secuencia (ventana deslizante) analizada.

Para finalizar, se propone como futura línea de investigación la resolución de ciertas cuestiones y propuestas que han surgido a lo largo de la realización del trabajo. En primer lugar, y para profundizar más en la investigación, sería conveniente realizar un análisis más completo en lo que a la obtención de resultados se refiere. Por un lado, la adición de longitudes de ventana deslizante, así como $k=5$, $k=9$, $k=13$, $k=17$ y $k=21$. Y por otro lado, la realización de una mayor cantidad de pruebas (variando éstas en parámetros como las iteraciones, el número de capas ocultas, el número de neuronas por capa oculta, etc.) para así lograr valores medios representativos y poder realizar un estudio bioestadístico fiable de los resultados obtenidos.
Por último, se plantea la opción de manipular las muestras de entrada mediante la técnica de validación cruzada (de entre 5 y 7 iteraciones), tal y como emplean Rost y Sanders, y Zamani y Kremer [18], [19]. De esta forma, las muestras son divididas de tal forma que aquellas que se usan para el entrenamiento de la red, no participan en el ensayo de éstas, siendo la predicción final el global de dichas pruebas.

BIBLIOGRAFÍA.

[1] S. D. Bergel, «La Declaración Universal de la UNESCO sobre el genoma humano y los derechos humanos», *Alegatos,(49)*, pp. 217–232, 1998.
[2] J. M. Berg, J. L. Tymoczko, y L. Stryer, *Bioquímica, sexta edición*, 6.ª ed. Barcelona: Editorial Reverté, 2007.
[3] N. Taniguchi, «Capítulo 2: Aminoácidos y proteínas», en *Bioquimica Médica*, 2.ª ed., Elsevier, 2006, pp. 7-23.
[4] J. Dongardive y S. Abraham, «Reaching optimized parameter set: protein secondary structure prediction using neural network», *Neural Comput. Appl.*, ene. 2016.
[5] L. Pauling, R. B. Corey, y H. R. Branson, «The structure of proteins: two hydrogen-bonded helical configurations of the polypeptide chain», *Proc. Natl. Acad. Sci.*, vol. 37, n.º 4, pp. 205–211, 1951.
[6] A. Shukla, R. Tiwari, y R. Kala, *Real Life Applications of Soft Computing*, 1.ª ed. United States: CRC Press, Auerbach Book, 2010.
[7] P. P. Bonissone, «Soft computing: the convergence of emerging reasoning technologies», *Soft Comput.*, vol. 1, n.º 1, pp. 6–18, 1997.
[8] C. González Morcillo, «Lógica Difusa. Una introducción Práctica.», Universidad de Castilla-La Mancha.
[9] D. Whitley, «A genetic algorithm tutorial», *Stat. Comput.*, vol. 4, n.º 2, pp. 65–85, 1994.
[10] W. S. McCulloch y W. Pitts, «A logical calculus of the ideas immanent in nervous activity.», *Bull. Math. Biophys.*, vol. 5, n.º 4, pp. 115-133, 1943.
[11] F. Rosenblatt, «Two theorems of statistical separability in the perceptron», en *Mechanization of Tought Processes*, HM Stationary Office, London: United States Department of Commerce, 1959, pp. 421-456.
[12] B. Widrow y M. E. Hoff, «Adaptative switching circuits», presentado en IRE Western Electric Show and Convention Record, 1960, vol. Part 4, pp. 96-104.
[13] K. Ben y P. Van der Smagt, «An introduction to neural networks», *Univ. Amst.*, 1996.
[14] H. Demuth, M. Beale, y M. Hagan, *Neural network toolbox™ 6*. 2008.
[15] D. Upadhyay, «Classification of eeg signals under different mental tasks using wavelet transform and neural network with one step secant algorithm», *Int. J. Sci. Eng. Technol.*, vol. 2, n.º 4, pp. 256–259, 2013.
[16] F. R. Mercado, «EXTREME LEARNING MACHINE», presentado en TECHZONE, Bolivia, 2015.
[17] G.-B. Huang, Q.-Y. Zhu, y C.-K. Siew, «Extreme learning machine: Theory and applications», *Neurocomputing*, vol. 70, n.º 1-3, pp. 489-501, dic. 2006.
[18] M. Zamani y S. C. Kremer, «Protein secondary structure prediction using an evolutionary computation method and clustering», en *Computational Intelligence in Bioinformatics and Computational Biology (CIBCB), 2015 IEEE Conference on*, 2015, pp. 1–6.
[19] B. Rost y C. Sander, «Prediction of Protein Secondary Structure at Better than 70% Accuracy.», *J. Mol. Biol.*, vol. 232, pp. 584-599, 1993.
[20] A. Gilat, «Introduction», en *Matlab: An Introduction with Applications*, Second Edition., Hoboken (NJ), USA: John Wiley & Sons, Inc., 2005.

[21] A. R. Carrera Amuriza y M. Martínez Nebreda, «Capítulo 1: aspectos más elementales de MATLAB», en *Introducción a MATLAB y a la creación de interfaces gráficas*, Bilbao: Servicio Editorial de la Universidad del País Vasco, 2004.
[22] L. H. Holley y M. Karplus, «Protein secondary structure prediction with a neural network», *Proc. Natl. Acad. Sci.*, vol. 86, n.º 1, pp. 152–156, 1989.
[23] G. Wang, Y. Zhao, y D. Wang, «A protein secondary structure prediction framework based on the Extreme Learning Machine», *Neurocomputing*, vol. 72, n.º 1-3, pp. 262-268, dic. 2008.
[24] M. Zamani y S. C. Kremer, «Amino acid encoding schemes for machine learning methods», en *Bioinformatics and Biomedicine Workshops (BIBMW), 2011 IEEE International Conference on*, 2011, pp. 327–333.

ANEXOS.

- *Anexo I.*

Programa y función realizados en MATLAB para la predicción de la estructura secundaria de las proteínas mediante el método ELM analizado en el apartado 3.5.1.

```
% Cargamos las N muestras (x,t) de entrada y salida con las que se va a
% trabajar.
% En este caso se van a utilizar las mismas muestras tanto como para
% entrenamiento como para prueba.

load input % cargamos los datos de entrada.
load ort_out % cargamos los datos de salida.

% Definimos cuántas neuronas va a tener la capa oculta a partir de las
% dimensiones de la matriz de entrada (tanto del entrenamiento como de
% prueba).

[neuronas_train,segment_train] = size(input);
[neuronas_test,segment_test] = size(input);

for capa_oculta = 0:10:1000 % definimos
    %%% TRAIN %%%

    % Hacemos las operaciones de la capa oculta con las matrices.
    % Primero creamos la matriz H completa de ceros.
    mat_h = zeros(neuronas_train,capa_oculta);

    % Creamos los umbrales aleatorios para insertar en la función.
    biashd = randi([0,1],capa_oculta,1);

    % Creamos los pesos aleatorios para insertar en la función.
    whid = randi([0,1],capa_oculta,segment_train);

    % Vamos formando la matriz H con los valores correspondientes.
    mat_h = repmat(biashd',neuronas_train,1);
    mat_h = mat_h + input*whid';
    mat_h = logsig(mat_h);

    % Sacamos el vector de los pesos que conectan la matriz H con la
    % salida a partir de la función inversa de Moore-Penrose.
    v_train = pinv(mat_h)*ort_out;


    % Pasamos a obtener los datos de salida a partir de la matriz obtenida
    % anteriormente v_train y la H de las muestras de prueba.
```

```
    %%% TEST %%%

    mat_h = zeros(neuronas_test,capa_oculta);
    whid = randi([0,1],capa_oculta,segment_test);

    % Obtenemos la nueva H* de las muestras de prueba.
    mat_h = repmat(biashd',neuronas_test,1);
    mat_h = mat_h+input*whid';
    mat_h = logsig(mat_h);


    % Obtenemos las salidas predichas por la red SLFN.
    out = mat_h*v_train;

    % Ahora calculamos la precisión de la red comparando el resultado de
    % out obtenido con el original.

    %%% PRECISIÓN %%%
    prec = accuracy(out,ort_out);
    fprintf('Precisión: %4.2f; capa oculta: %d \n',prec, capa_oculta);

end
```

Desarrollo de la función `accuracy` utilizada para calcular la precisión de la red ELM:

```
% Esta función va a calcular el porcentaje de acierto de la red neuronal
% creada en otros programas.
%
function porcentaje = accuracy(pred, out)

% Obtenemos el valor y la posición del máximo tanto en la predicción como
% en los valores originales para posteriormente poder compararlas.
[value_pred,pos_pred] = max(pred');
[value_prot,pos_prot] = max(out');

% Se crea un vector que nos diga cuántas posiciones coinciden (valor
% 1), y cuantas posiciones son diferentes (valor cero).
coincidencias = (pos_pred==pos_prot);

% Se calcula el porcentaje de acierto de la predicción realizada por la
% red.
porcentaje = (sum(coincidencias)/length(coincidencias))*100;

end
```

- *Anexo II.*

Programa creado en MATLAB para la lectura de los archivos de texto proporcionados por la base de datos RS126 y su conversión a matrices de caracteres con las proteínas y sus estructuras secundarias correspondientes.

```
%
%
% Vamos a realizar una función que vaya leyendo las proteínas y su
% estructura de los archivos RS126

disp('Reading RS126 database');

% Especificamos el directorio en el que se encuentran todos los archivos de
% texto. Cada uno de ellos hace referencia a una de las 126 proteínas.
files = dir('G:\MatLab_ama\Prueba_2_ELM\rs126\*.txt');

% Inicializamos las matrices finales, en las que se guardarán (como
% caracteres) las proteínas y su estructura secundaria.
MATp = []; % guarda las proteínas.
MATs = []; % guarda las estructuras.

for filename= 1: length(files)
    Archivo = files(filename).name; % Leemos cada uno de los archivos.

    fid=fopen(Archivo, 'r'); % abrimos el texto correspondiente
    v_lectura=fscanf(fid,'%s'); % lo escaneamos como cadena de caracteres

    % Se crean dos vectores en los que guardaremos cada proteina y cada
    % estructura. Posteriormente añadiremos éstos a la matriz final, que
    % irá aumentando con cada archivo que se lea.
    proteina = [];
    estructura = [];
    for i = 1:(length(v_lectura)-3)

        % Se crea una cadena que tenga los caracteres que el usuario desea,
        % para así saber qué información debe guardarse como válida.
        conc = strcat(v_lectura(i),v_lectura(i+1),v_lectura(i+2),v_lectura(i+3));

        % Elegimos la proteína original.
        if strcmp(conc,'Orig')==1
            n = i+8; % Creamos 'n' para coger el inicio de la proteína.
            while strcmp(v_lectura(n),'c')==0 % Especificamos hasta donde
                % queremos que lea.
                % Vamos añadiendo al vector cada uno de los aminoácidos
                % mediante adición horizontal.
                proteina = [proteina v_lectura(n)];
                n = n+1;
            end
```

```
            % Ahora bien, para que la matriz final no de errores de tamaño
            % (dado que cada una de las proteínas tiene longitud diferente)
            % se rellena cada proteína con puntos para obtener una matriz
            % final de 126 x 1000
            for rellenar = length(proteina):1000
                proteina = [proteina '.'];
            end

            MATp = [MATp; proteina]; % Se añade cada una de las proteínas
            % de cada archivo mediante adición vertical.
        end

        % Elegimos la estructura secundaria obtenida a partir de PHD.
        if strcmp(conc,'phd:')==1
            m = i+5; % Creamos 'm' para coger el inicio de la estructura.
            while strcmp(v_lectura(m),'-')==1 | strcmp(v_lectura(m),'E')==1 | strcmp
(v_lectura(m),'H')==1
                % Vamos añadiendo al vector cada uno de las estructuras
                % mediante adición horizontal.
                estructura = [estructura v_lectura(m)];
                m = m+1;
            end
            % Realizamos el mismo relleno que con las proteínas por la
            % misma razón.
            for re = length(estructura):1000
                estructura = [estructura '.'];
            end

            MATs = [MATs; estructura]; % Se añade cada una de las estructuras
            % de cada archivo mediante adición vertical.
        end
    end
end

% Se guardan ambas matrices.
save MATp MATp
save MATs MATs
```

- *Anexo III.*

Programas realizados en MATLAB para la obtención de las matrices de entrada de las redes neuronales. Como se desarrolla en la sección *3.2.* se tienen tres representaciones matriciales: ortogonal, estadístico y a partir de grafos dirigidos. Primero se van a mostrar las funciones para la obtención de cada una de las matrices de paso, y posteriormente se presentará la función de conversión a la matriz final (matriz de entrada de la red neuronal).

- Representación matricial ortogonal:

```
%
% Se va a crear una función que convierta la matriz de caracteres que
% contiene todas las proteínas de la base de datos a una matriz que se va
% formando en función de la longitud de ventana elegida mediante la
% codificación ortogonal.
%
function out = prot2matrix(mprot)

% Tenemos los 20 aminoácidos en un vector.
aminoacidos =↙
['A','C','D','E','F','G','H','I','K','L','M','N','P','Q','R','S','T','V','W','Y'];

out=[]; % Inicializamos el vector de salida para crear la matriz de paso.
%
% Vamos leyendo cada proteína y metemos los vectores ortogonales en la
% matriz dependiendo del tamaño de la sección.
%
for i = 1:length(mprot(:,1)) % Vamos recorriendo las diferentes proteinas.

    proteina = mprot(i,:); % Definimos las proteinas en la variable 'proteina'.
    for j = 1:1001
        % La recorremos para crear las secciones.
        if strcmp(proteina(j+14),'.')==0
            % Aquí indicamos el tamaño que queremos que tenga la seccion.
            %%% En este caso k=15.
            seccion = proteina(j:j+14);
            for k = 1:length(seccion)
                % Inicializamos los vectores ortogonales a cero.
                vsec = zeros(1,20);
                % Hacemos que vaya comparando la proteína seleccionada con
                % aquellas del vector de aminoácidos. En el caso de que
                % coincidan marcará dicha posición con un 1.
                for amino =1:length(aminoacidos)
                    if strcmp(seccion(k),aminoacidos(amino))==1
                        vsec(1,amino) = 1;
                    end
                end
```

```
                % Vamos formando la matriz de paso mediante adición
                % vertical. El número de columnas será el correspondiente a
                % la longitud de ventana deslizante.
                %%% En este caso k=15.
                out = [out;vsec];
            end
        else
            break
        end
    end
end

end
```

- Representación matricial estadística:

```
%
% Se va a crear una función que convierta la matriz de caracteres que
% contiene todas las proteínas de la base de datos a una matriz que se va
% formando en función de la longitud de ventana elegida mediante la
% codificación estadística.
%
% En esta función es necesario pasar como parámetro de entrada también la
% matriz de estructuras ya que lleva información estadística sobre éstas.
%
function [outp,outs] = ps2matrix_std(mprot,mstr)
%
% Tenemos los 20 aminoácidos en un vector
aminoacidos =↙
['A','C','D','E','F','G','H','I','K','L','M','N','P','Q','R','S','T','V','W','Y'];

% Inicializamos los vectores de salida para crear la matriz de paso.
outp=[];
%
for i = 1:length(mprot(:,1)) % Vamos recorriendo las diferentes proteinas.
    % Definimos cada proteína con su correspondiente estructura en las
    % variables 'proteina' y 'estructura'.
    proteina = mprot(i,:);
    estructura = mstr(i,:);

    % Primero tenemos que leer la proteína entera y sacar los valores
    % correspondientes para la fnción estadística. Inicializamos todos a 1
    % para que no de errores de división por 0.
    nh = 1;
    ne = 1;
    nc = 1;
    % Vamos sumando cuantas estructuras de cada tipo hay en la proteína
    % completa.
```

```
for cont = 1:length(estructura)
    if strcmp(estructura(cont),'.')==0
        switch estructura(cont)
            case 'H'
                nh = nh+1;
            case 'E'
                ne = ne+1;
            case '-'
                nc = nc+1;
        end
    end
end
%
%
% A continuación vamos recorriendo la proteína para ir formando las
% secciones. En este caso realizamos la prueba con longitud de ventana
% deslizante k=15.
for j = 1:1001
    if strcmp(proteina(j+14),'.')==0
        % Aquí indicamos el tamaño que queremos que tenga la seccion
        %%%
        sec_p = proteina(j:j+14);
        sec_s = estructura(j:j+14);

        %%%
        % Ahora sacamos los valores de la sección de la proteina, que
        % son los correspondientes a la información estadíctica de la
        % sección. En el trabajo viene denominado como 'x'
        %
        % Inicializamos los valores a 1 para que no den error de
        % división por 0.
        nhx = 1;
        nex = 1;
        ncx = 1;
        % Obtenemos la información estadísticade las estructuras
        % secundarias dentro de la sección con la que se está
        % trabajando.
        for c = 1:length(sec_s)
            switch sec_s(c)
                case 'H'
                    nhx = nhx+1;
                case 'E'
                    nex = nex+1;
                case '-'
                    ncx = ncx+1;
            end
        end
```

```
            %
            % Escribimos los valores totales en un vector para luego elegir
            % el minimo.
            nx = [nhx,nex,ncx];
            %
            for k = 1:length(sec_p)
                % Inicializamos el vector estadístico a cero.
                vsec = zeros(1,20);
                % Se va recorriendo la sección y dependiendo del aminoácido
                % que sea se coloca en dicha posición el valor obtenido
                % mediante el parámetro:
                %       (Nix/min{Nhx,Nex,Ncx})·(Nix/Ni)
                %
                for amino =1:length(aminoacidos)
                    if strcmp(sec_p(k),aminoacidos(amino))==1
                        switch sec_s(k)
                            case 'H'
                                vsec(1,amino) = (nhx/min(nx))*(nhx/nh);
                            case 'E'
                                vsec(1,amino) = (nex/min(nx))*(nex/ne);
                            case '-'
                                vsec(1,amino) = (ncx/min(nx))*(ncx/nc);
                        end
                    end
                end
                % Se van añadiendo los vectores mediante adición vertical
                % para obtener la matriz de paso.
                outp = [outp;vsec];
            end
        else
            break
        end

        end
    end

    end
```

- Representación matricial a partir de grafos dirigidos:

```
%
% Se va a crear una función que convierta la matriz de caracteres que
% contiene todas las proteínas de la base de datos a una matriz que se va
% formando en función de la longitud de ventana elegida mediante la
% codificación a partir de grafos dirigidos.
%
function [outp,outs] = ps2matrix_graph(mprot)
%
% Inicializamos la matriz de salida (matriz de paso).
outp=[];
```

```
%
% Vamos recorriendo las diferentes proteinas.
for i = 1:length(mprot(:,1))

    % Definimos cada proteína en la variable 'proteina'.
    proteina = mprot(i,:);

    % Recorremos la proteína para crear ls secciones. En este caso se hará
    % con una longitud de ventana deslizante de k=15.
    for j = 1:1001
        if strcmp(proteina(j+14),'.')==0
            %
            % Creamos cada sección.
            sec_p = proteina(j:j+14);

            % Inicializamos el vector de salida.
            vsec = [];

            % Recorre cad aminoácido de la sección.
            for k = 1:length(sec_p)
                % Carga el vector correspondiente al aminoácido
                % seleccionado. Los vectores creados a partir de los grafos
                % dirigidos han sido guardados individualmente con el
                % nombre de cada aminoácido en el mismo directorio que la
                % función.
                amino=load(sec_p(k));
                vsec = [vsec; eval(strcat('amino.',sec_p(k)))];
                amino = '';
                %
                % Se va formando la matriz de paso mediante la adición
                % vertical de los vectores de grafos.
                outp = [outp;vsec];
                vsec = [];
            end
        else
            break
        end
    end
end

end
```

- Conversión de la matriz de paso a la matriz final:

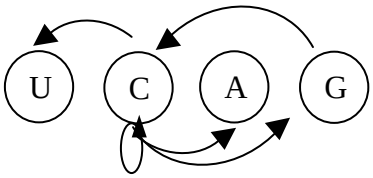
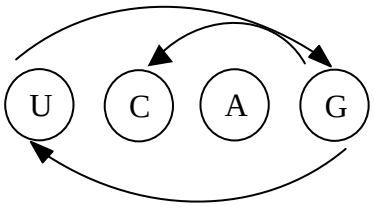
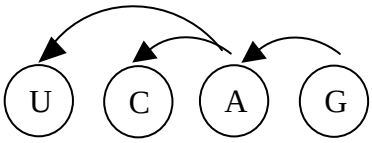
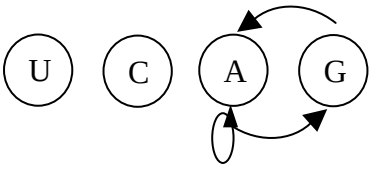
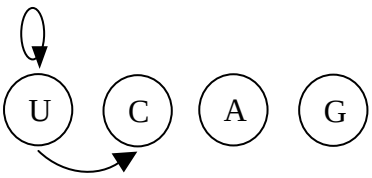
```
%
% Se va a crear una función que convierta las matrices de paso en la matriz
% final que servirá como entrada para las redes neuronales. Esta conversión
% depende de la codificación y longitud de ventana (k) que se elija.
%
```

```
function out = conversion(mat_pass) % Matriz de paso

% Definimos el número de filas y columnas totales.
[filas,columnas] = size(mat_pass);

% Vamos a recorrer la matriz por columnas, por lo que hacemos la
% traspuesta.
mat_pass = mat_pass';

% Inicializamos la matriz final.
out = [];
dat_in = [];

% Utilizamos 'n' como contador. Vamos a ir recorriendo la matriz por
% columnas, y cuando llegue al valor requerido se formará un vector con los
% valores recorridos para insertar éste en la matriz final.
%
% En el caso de la codificación ortogonal y estadística, habrá que coger en
% grupos de k*20 valores (debido a los 20 aminoácidos).
% Esto es, en el caso de una longitud de ventana k=15, se formará un vector
% de 300 (=15*20) valores. La matriz final tendrá unas dimensiones de
% m x (k*20).
%
% En el caso de la codificación mediante grafos dirigidos, habrá que coger
% grupos de k*16 (debido a que el tamaño de los vectores que caracterizan
% cada uno de los aminoácidos en de 16).
% Esto es, en el caso de una longitud de ventana k=15, se formará un vector
% de 240 (=15*16) valores. La matriz final tendrá unas dimensiones de
% m x (k*16).
%
n=1;
i=1;
while i<=(filas*columnas)
    % Aquí se especifica qué tipo de matriz se quiere convertir. Se define
    % la variable 'num_val' (número de valores) que vendrá dado por k*20 o
    % k*16 en función de la codificación y la longitud de ventana.
    num_val = 300;
    %
    if i==num_val*n
        % Se va creando el vector que se insertará en la matriz final
        % mediante adición horizontal.
        dat_in = [dat_in mat_pass(i)];

        % Se añade el vector a la matriz final mediante adición vertical.
        out = [out; dat_in];
        dat_in = [];
        n = n+1; % Avanza el contador.
    else
        % Se va creando el vector que se insertará en la matriz final
```

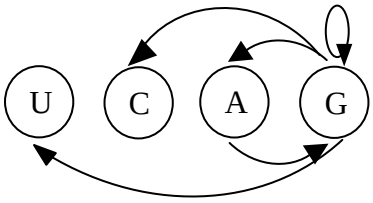
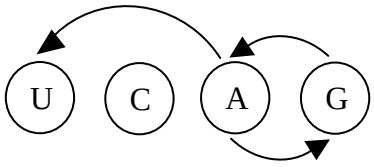
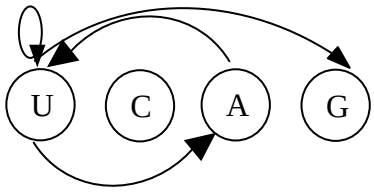
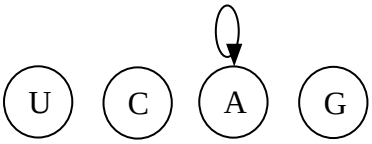
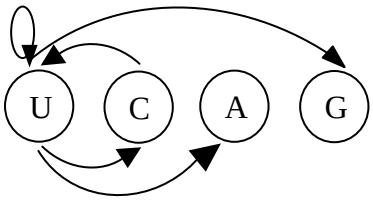
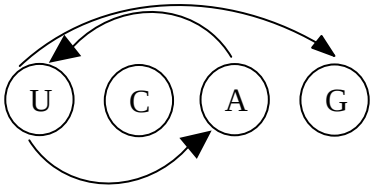
```
            % mediante adición horizontal.
            dat_in = [dat_in mat_pass(i)];
        end
        i = i+1; % Avaza en bloques.
    end
    end
```

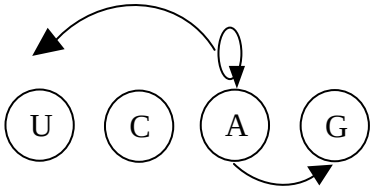
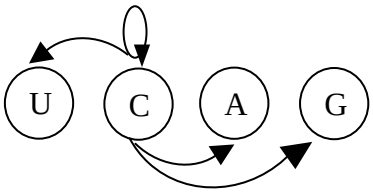
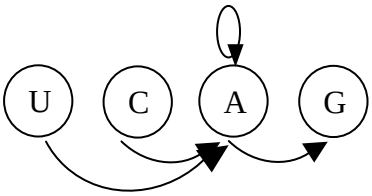
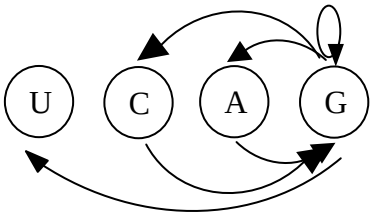
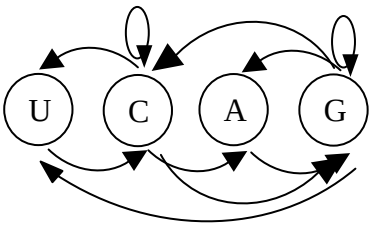
- *Anexo IV.*

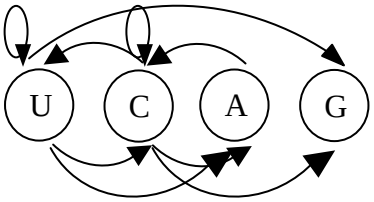
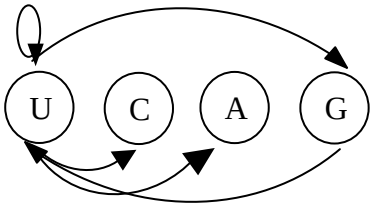
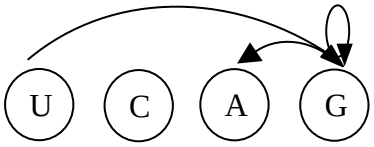
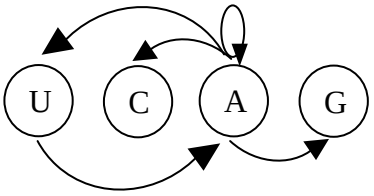
Codificación de los 20 aminoácidos proteicos mediante codones genéticos.

AA[6]	**Codones asociados**	**Grafo**	**Matriz asociada**	**Vector asociado**
Alanina	GCU GCC GCA GCG		U C G A U 0 0 0 0 C 1 1 1 1 G 0 1 0 0 A 0 0 0 0	[0000111101000000]
Cisteína	UGU UGC		U C G A U 0 0 1 0 C 0 0 0 0 G 1 1 0 0 A 0 0 0 0	[0010000011000000]
Ácido Aspártico	GAU GAC		U C G A U 0 0 0 0 C 0 0 0 0 G 0 0 0 1 A 1 1 0 0	[0000000000011100]
Ácido Glutámico	GAA GAG		U C G A U 0 0 0 0 C 0 0 0 0 G 0 0 0 1 A 0 0 1 1	[0000000000010011]
Fenilalanina	UUU UUC		U C G A U 1 1 0 0 C 0 0 0 0 G 0 0 0 0 A 0 0 0 0	[1100000000000000]

[6] Siglas de aminoácido.

Glicina	GGU GGC GGA GGG AGA AGG		U C G A U 0 0 0 0 C 0 0 0 0 G 1 1 1 1 A 0 0 1 0	[0000000011110010]
Histidina	GAU GAG		U C G A U 0 0 0 0 C 0 0 0 0 G 0 0 0 1 A 1 0 1 0	[0000000000011010]
Isoleucina	AUU AUG AUA		U C G A U 1 0 1 1 C 0 0 0 0 G 0 0 0 0 A 1 0 0 0	[1011000000001000]
Lisina	AAA		U C G A U 0 0 0 0 C 0 0 0 0 G 0 0 0 0 A 0 0 0 1	[0000000000000001]
Leucina	UUA UUG CUU CUC CUG		U C G A U 1 1 1 1 C 1 0 0 0 G 0 0 0 0 A 0 0 0 0	[1111100000000000]
Metionina	AUA AUG		U C G A U 0 0 1 1 C 0 0 0 0 G 0 0 0 0 A 1 0 0 0	[0011000000001000]

Asparagina	AAU AAG AAA		U C G A U 0 0 0 0 C 0 0 0 0 G 0 1 0 0 A 1 0 1 1	[0000000000001011]
Prolina	CCU CCC CCA CCG		U C G A U 0 0 0 0 C 1 1 1 1 G 0 0 0 0 A 0 0 0 0	[0000111100000000]
Glutamina	UAA UAG CAA CAG		U C G A U 0 0 0 1 C 0 0 0 1 G 0 0 0 0 A 0 0 1 1	[0001000100000011]
Arginina	CGU VGC CGA CGG AGA AGG		U C G A U 0 0 0 0 C 0 0 1 0 G 1 1 1 1 A 0 0 1 0	[0000001011110010]
Serina	UCU UCC UCA UCG AGU AGC AGA AGG		U C G A U 0 1 0 0 C 1 1 1 1 G 1 1 1 1 A 0 0 1 0	[0100111111110010]

Treonina	CUU CUC CUA CUG ACU ACC ACA ACG			[1111111100000100]
Valina	GUU GUC GUA GUG			[1111000010000000]
Triptófano	UGA UGG			[0010000000110000]
Tirosina	UAC UAU UAA UAG			[0001000000001111]

	U	C	G	A
U	1	1	1	1
C	1	1	1	1
G	0	0	0	0
A	0	1	0	0

	U	C	G	A
U	1	1	1	1
C	0	0	0	0
G	1	0	0	0
A	0	0	0	0

	U	C	G	A
U	0	0	1	0
C	0	0	0	0
G	0	0	1	1
A	0	0	0	0

	U	C	G	A
U	0	0	0	1
C	0	0	0	0
G	0	0	0	0
A	1	1	1	1

- *Anexo V.*

Programa creado en MATLAB para la creación de la matriz de salida de la estructura secundaria en función de la longitud de ventana de deslizamiento.

```
% Se crea un programa en el que a partir de la matriz de caracteres de la
% estructura secundaria se cree la matriz de salida en función de la
% longitud de ventana deslizante.
%
% Primero se carga la matriz de estructura secundaria de caracteres.
load mat_est

% Inicializamos la matriz en la que obtendremos la salida.
outs=[];

% Vamos recorriendo las diferentes estructuras (tener en cuenta que la
% estructura secundaria de cada proteína viene dada por filas.
for i = 1:length(mat_est(:,1))

    % Separamos cada proteína. Definimos cada una de la estructura
    % secundaria en la variable 'estructura'.
    estructura = mat_est(i,:);

    for j = 1:1001 % Se recorre la matriz para crear las secciones.

        % Se ha mencionado anteriormente que las matrices se rellenan para
        % que no de fallo de error. Por eso se recorren hasta que inicien
        % los puntos.

        % Se va a realizar el programa para longitud de ventana deslizante
        % k=3.
        if strcmp(estructura(j+2),'.')==0

            % Definimos la sección con la estructura secundaria de tres
            % aminoácidos
            sec_s = estructura(j:j+2);

            % Se crea el vector en que se pondrá a 1 la posición que
            % coincida con la estructura del aminoácido central.
            vout = zeros(1,3);

            % Comprobamos qué estructura tiene el aminoácido central y lo
            % marcamos en el vector.
            switch sec_s(2)
                case 'E'
                    vout(1)=1; % E -> columna 1
                case 'H'
                    vout(2)=1; % H -> columna 2
                case '-'
                    vout(3)=1; % - -> columna 3
            end
```

```
            % Se va creando la matriz de salida mediante adición vertical
            % de los vectores de cada deslizamiento.
            outs=[outs;vout];
        else
            break
        end
    end
end

% Se guarda la matriz de salida para dicha longitud de ventana deslizante.
save outs
```

- *Anexo VI.*

Resultados obtenidos para los diferentes grupos de representaciones matriciales en función de la longitud de ventana y las iteraciones seleccionadas.

- Representación matricial ortogonal:

	Iteraciones	**Error**	**%**
Longitud de ventana 19	20000	0.1289	71.4882
	15000	0.1302	71.1609
	10000	0.1297	71.1609
	10000	0.1311	71.2368
	7000	0.1322	71.3696
	7000	0.1313	71.2795
	5000	0.1330	71.3601
	1000	0.1330	71.4882
	500	0.1334	71.3933

	Iteraciones	**Error**	**%**
Longitud de ventana 15	10000	0.1352	70.7406
	7000	0.1361	70.4999
	7000	0.1299	70.3285
	5000	0.1342	70.7409
	3000	0.1337	70.7501
	1000	0.1360	70.8613
	500	0.1361	70.6204
	100	0.1384	69.7679

	Iteraciones	**Error**	**%**
Longitud de ventana 11	7000	0.1377	69.7197
	5000	0.1374	69.8148
	3000	0.1379	70.0276
	3000	0.1402	70.6479
	2000	0.1384	70.6751
	2000	0.1352	69.7831
	1000	0.1368	69.9416
	500	0.1390	69.1583

	Iteraciones	Error	%
Longitud de ventana 7	10000	0.1435	67.2834
	7000	0.1486	67.1949
	3000	0.1475	68.4299
	3000	0.1423	68.8593
	3000	0.1438	68.5229
	3000	0.1476	67.3277
	1000	0.1483	67.2702

	Iteraciones	Error	%
Longitud de ventana 3	7000	0.1735	58.9695
	5000	0.1730	59.3851
	3000	0.1750	59.9740
	3000	0.1750	59.1903
	3000	0.1752	59.4414
	1000	0.1745	59.7835
	1000	0.1747	59.3245

- Representación matricial estadística:

	Iteraciones	Error	%
Longitud de ventana 19	20000	0.1481	70.4967
	15000	0.1410	72.3421
	15000	0.1402	72.8403
	10000	0.1433	70.9094
	7000	0.1476	71.8772
	7000	0.1503	71.4455
	5000	0.1499	71.3364
	1000	0.1665	66.3646
	500	0.1706	32.1566

	Iteraciones	Error	%
Longitud de ventana 15	15000	0.1211	72.7193
	10000	0.1226	73.7618
	10000	0.1203	73.1872
	7000	0.1233	73.2011

	7000	0.1210	71.9733
	7000	0.1285	72.2467
	5000	0.1296	71.9918
	5000	0.1276	72.0196

	Iteraciones	**Error**	**%**
Longitud de ventana 11	15000	0.1243	76.1443
	15000	0.1248	75.0418
	15000	0.1273	75.2479
	10000	0.1238	74.8313
	7000	0.1276	75.2615
	7000	0.1211	75.3384
	7000	0.1223	75.0623
	7000	0.1246	74.6412
	5000	0.1238	74.4918

	Iteraciones	**Error**	**%**
Longitud de ventana 7	15000	0.1280	74.6935
	15000	0.1280	75.4371
	15000	0.1287	75.5699
	10000	0.1251	74.8572
	7000	0.1229	74.1579
	7000	0.1217	73.9809
	7000	0.1253	73.8480

	Iteraciones	**Error**	**%**
Longitud de ventana 3	15000	0.1598	65.8108
	15000	0.1605	65.7155
	10000	0.1609	65.5163
	7000	0.1590	64.9318

- Representación matricial mediante grafos:

	Iteraciones	**Error**	**%**
Longitud de ventana 19	15000	0.1294	70.7292
	10000	0.1338	73.4855

	10000	0.1335	63.3854
	10000	0.1726	71.0565
	10000	0.1357	70.9474
	10000	0.1365	71.2131
	7000	0.1754	71.4550
	7000	0.1350	69.2063
	5000	0.1418	62.6690

	Iteraciones	**Error**	**%**
Longitud de ventana 15	10000	0.1369	70.5416
	10000	0.1310	72.5416
	10000	0.1411	69.4111
	10000	0.1311	71.3664
	7000	0.1348	73.4560
	7000	0.1312	73.5162

	Iteraciones	**Error**	**%**
Longitud de ventana 11	10000	0.1415	68.3298
	10000	0.1353	69.7741
	7000	0.1454	69.5568
	7000	0.1403	72.6128
	7000	0.1398	70.0140
	7000	0.1345	71.5217
	7000	0.1412	70.5166
	5000	0.1377	69.5839

	Iteraciones	**Error**	**%**
Longitud de ventana 7	10000	0.1475	73.9055
	10000	0.1501	65.8758
	7000	0.1423	68.0492
	7000	0.1541	66.9116
	7000	0.1471	67.7040
	7000	0.1425	70.6565

	Iteraciones	**Error**	**%**
Longitud de ventana 3	7000	0.1738	59.5150
	7000	0.1743	59.2812
	7000	0.1752	59.2423

- *Anexo VII:*

Resultados obtenidos para el barrido neuronal de 15000 iteraciones en la red multicapa con una longitud de ventana deslizante de tamaño 11 y muestras de entrada obtenidas a partir de la representación estadística de las proteínas.

	Error	**%**
Número de neuronas 5	0.1246	75.2841
	0.1247	75.2841
	0.1232	75.6689
	0.2105	48.4810
	0.1237	76.1624
	0.1251	75.5467
	0.2460	27.8399
	0.1336	65.1786
	0.1239	75.7640

	Error	**%**
Número de neuronas 10	0.1247	75.3158
	0.1228	75.5859
	0.1517	65.1016
	0.1259	74.7317
	0.1233	75.4969
	0.1208	75.9225
	0.1237	75.3747

	Error	**%**
Número de neuronas 15	0.1212	75.6961
	0.1218	75.7369
	0.1231	75.5965
	0.1219	76.0628
	0.1215	75.8500
	0.1214	75.8410
	0.1209	75.8184

	Error	**%**
Número de neuronas 20	0.1232	75.4969
	0.1237	75.5150
	0.1196	76.3345

	0.1251	74.8721
	0.1214	75.9044
	0.1248	75.1754
	0.1234	75.3882

	Error	**%**
	0.1198	76.2304
	0.1215	75.8591
Número de neuronas 25	0.1202	76.0674
	0.1230	75.4969
	0.1211	75.7097
	0.1199	76.3979
	0.1206	75.9678

	Error	**%**
	0.1224	75.3339
	0.1216	75.8772
Número de neuronas 30	0.1219	75.8546
	0.1247	75.2026
	0.1231	75.7142
	0.1226	75.7006
	0.1199	76.2394