Facultade de Informática
UNIVERSIDADE DA CORUÑA

TRABALLO FIN DE GRAO
GRAO EN ENXEÑARÍA INFORMÁTICA
MENCIÓN EN COMPUTACIÓN

Evolución de características de aminoácidos para la predicción de la estructura secundaria de proteínas

Estudante: Héctor Rivas Dorado
Dirección: José Santos Reyes

A Coruña, febreiro de 2022.

Curiosity is the engine of achievement

Agradecimientos

En primer lugar me gustaría agradecer al director de este proyecto, José Santos Reyes, ya que sin su ayuda no sería posible la realización de este trabajo. También agradecer a mis padres y a mi hermano que siempre han estado ahí para lo que necesitase. Por último, a los amigos que hice en la facultad, gracias por todos los buenos ratos.

Resumen

El objetivo de este trabajo es obtener automáticamente, mediante un algoritmo evolutivo, un conjunto de propiedades para codificar los aminoácidos en el ámbito de la predicción de estructuras secundarias de proteínas. Se pretende que estas propiedades reflejen las relaciones que existen entre los aminoácidos, además de servir como reducción de dimensionalidad en la entrada de clasificadores/predictores actuales. Para conseguir automáticamente estas propiedades se ha utilizado un algoritmo evolutivo, Evolución Diferencial, y dos algoritmos de clasificación: Minimum Distance Classifier y k-NN. Las propiedades obtenidas se han comparado con la codificación neutra y estándar que no presupone ninguna relación entre los aminoácidos. La comparación se ha realizado con un clasificador neuronal. Los resultados muestran que las propiedades artificiales tienen un rendimiento muy similar, incluso superior, y que podrían emplearse para reducir el número de entradas de clasificadores actuales.

Palabras clave:

- Selección automática de características
- Predicción de estructura secundaria de proteínas
- Evolución diferencial
- Redes de neuronas artificiales

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

Introducción

1.1 Introducción a las proteínas

Las proteínas son moléculas grandes y complejas que se encargan de una gran cantidad de funciones críticas en el cuerpo. Estas funciones son muy variadas y entre ellas encontramos:

- Enzimas: Actúan como catalizadores biológicos en reacciones químicas.
- Estructurales: Dan estructura y soporte a la célula.
- De defensa: Protegen al organismo contra invasiones externas.
- Reguladoras: Controlan actividades fisiológicas o celulares, entre ellas están las hormonas.
- Transporte: Enlazan y llevan sustancias de un lugar a otro.

Todas las proteínas tienen en común que están formadas por unas unidades más pequeñas llamadas aminoácidos. En la Figura 1.1 vemos la estructura general de un aminoácido: un átomo de carbono (C) unido a un grupo amino (NH_2), un grupo carboxilo (COOH), un átomo de hidrógeno (H) y una cadena lateral o radical (R). El radical es único entre cada aminoácido y es lo que los diferencia.

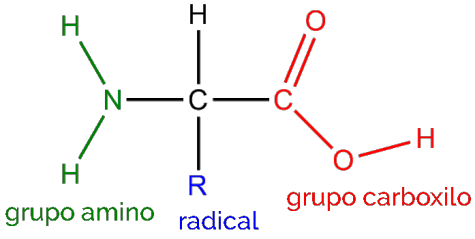

Figura 1.1: Estructura de un aminoácido

En la síntesis de proteínas, es decir, en su proceso de formación, los aminoácidos se van uniendo unos a otros mediante enlaces, llamados enlaces peptídicos, que vemos en la Figura 1.2. Se trata de enlaces covalentes entre el grupo carboxilo de un aminoácido y el grupo amino de otro. El resultado de esta reacción es una molécula de agua, ya que es una reacción de deshidratación y los dos aminoácidos unidos terminan formando un dipéptido.

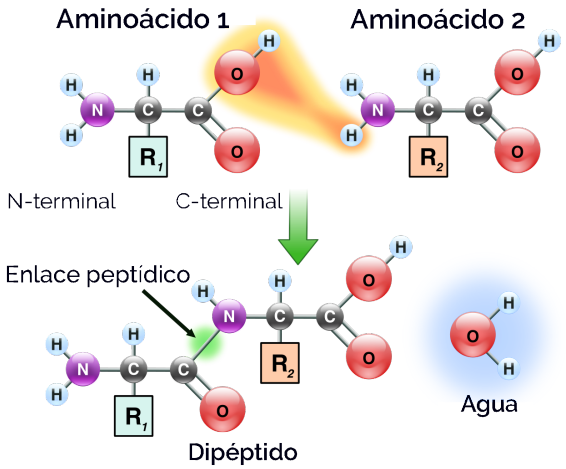

Figura 1.2: Enlace peptídico

Este proceso se repite con los aminoácidos que vayan a formar la proteína, uniéndose unos a otros, formando lo que se llama un polipéptido. Una proteína está formada por una o más de estas estructuras.

1.1.1 Clasificación de los aminoácidos

Los aminoácidos se distinguen entre sí por su radical y según los elementos que lo compongan tendrán distintas propiedades. Gracias a esto podemos clasificarlos, como muestra la Figura 1.3, en:

- **No Polares**: Los radicales que componen estos aminoácidos los hacen hidrófobos, así que cuando la proteína se pliegue tenderán a quedarse en el interior.
- **Polares**: La mayoría de radicales en este grupo tienen elementos que pueden realizar enlaces de puente de hidrógeno con agua y otras moléculas.
- **Ácidos**: Los elementos de su radical hacen que sean donadores de protones por lo que tienen carga negativa.
- **Básicos**: Al contrario que los ácidos, los aminoácidos básicos aceptan protones por lo que tienen carga positiva.

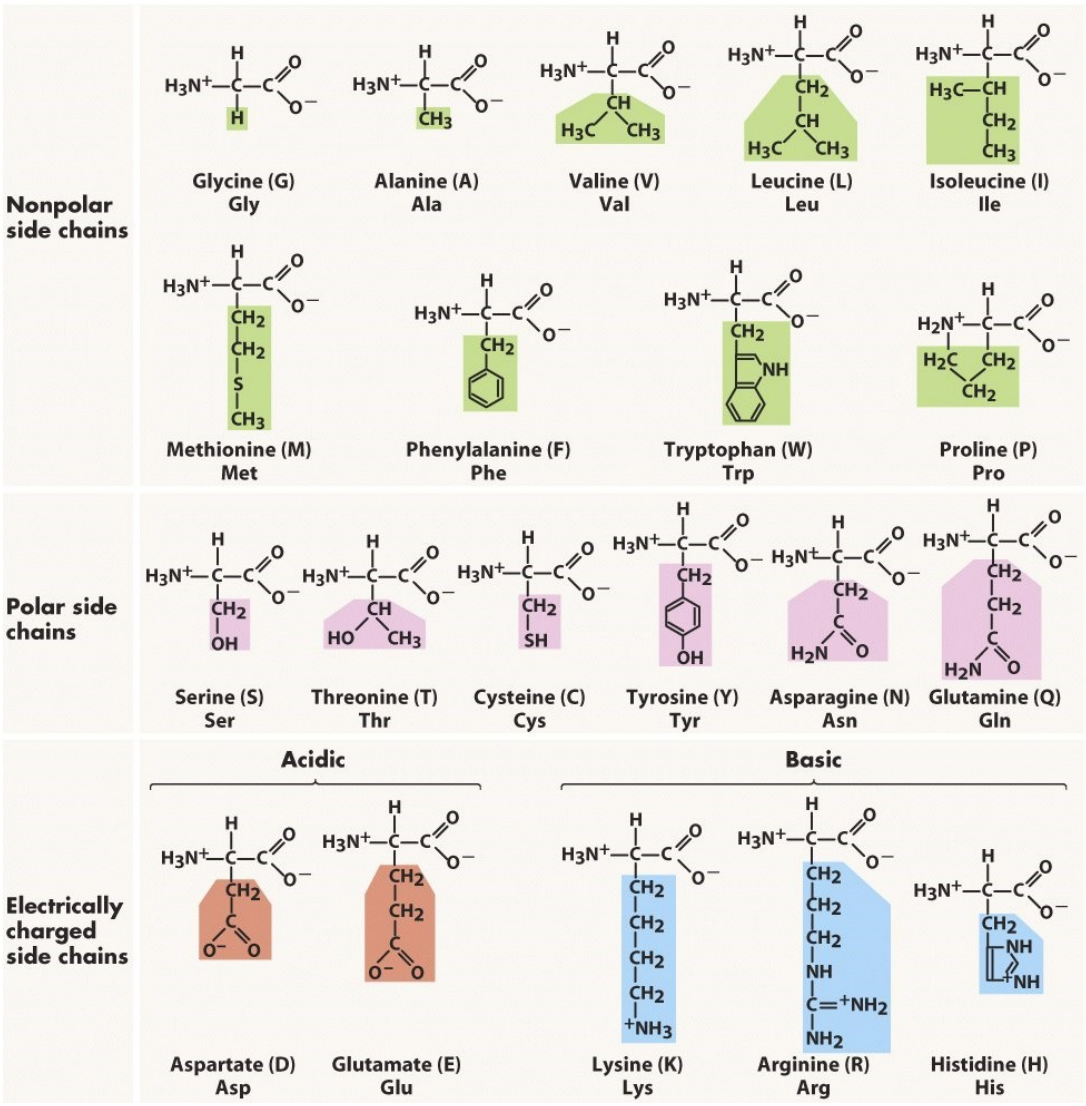

Figura 1.3: Clasificación de los 20 aminoácidos según su radical. Figura obtenida de [1]

1.2 Niveles de estructura de las proteínas

La complejidad de las proteínas hace que éstas manifiesten cuatro niveles distintos de organización estructural.

El nivel más simple de estructura de una proteína es la estructura primaria, que es solo la secuencia de aminoácidos que conforma la proteína, es decir, qué aminoácidos son y en qué orden están.

La estructura secundaria son las estructuras regulares que se forman de manera local a lo largo de la cadena, debido principalmente a la formación de puentes de hidrógeno entre aminoácidos cercanos. Aunque existen más tipos de estructuras, en la Figura 1.4 se aprecian las dos clases más comunes: la hélice alfa y la lámina plegada beta.

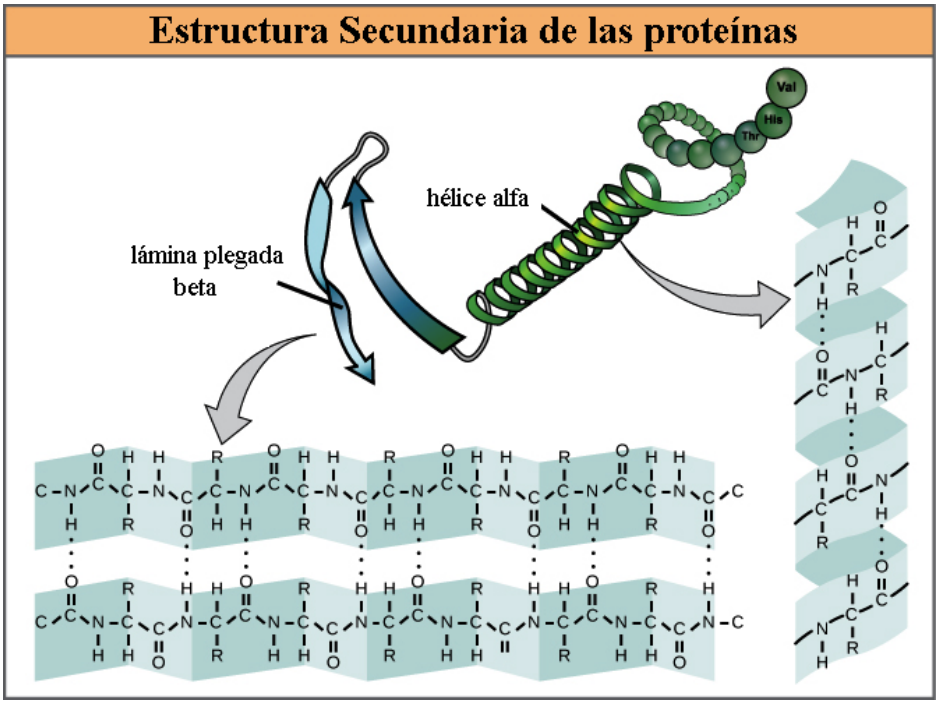

Figura 1.4: Estructura secundaria

La estructura terciaria es la apariencia que tiene la proteína en tres dimensiones y es lo que define qué función tiene. La forma que adopta se debe a las distintas interacciones entre los radicales de los aminoácidos que la conforman.

Las estructuras primaria, secundaria y terciaria se refieren a la forma de un solo polipéptido. Si la proteína se compone de varios entra en juego la estructura cuaternaria. Como ejemplo de la estructura cuaternaria de una proteína tenemos una molécula de hemoglobina en la Figura 1.5. Esta estructura se conforma debido a interacciones, en general, similares a las de la estructura terciaria.

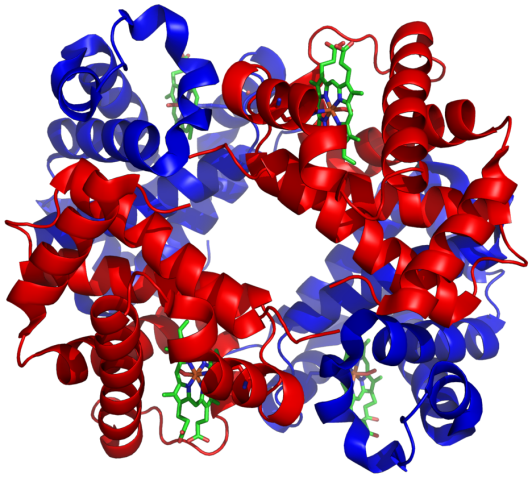

Figura 1.5: Molécula de hemoglobina

Para terminar, en la Figura 1.6 vemos un resumen de todos los niveles de estructura; de cómo de una serie de aminoácidos se forman estructuras locales que luego contribuyen a qué forma toma en tres dimensiones y, por tanto, a su función, a cómo se ensamblan proteínas de mayor tamaño.

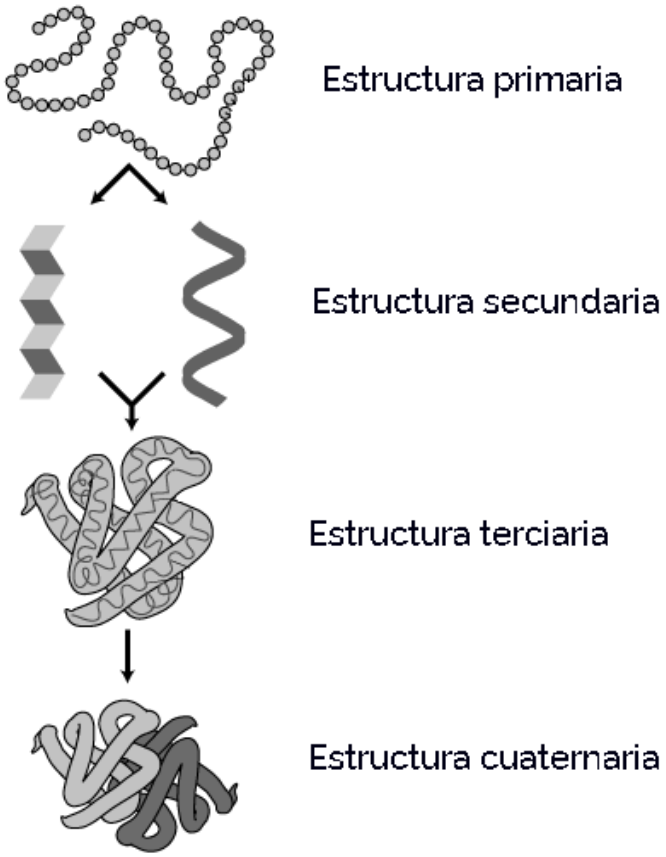

Figura 1.6: Resumen de los cuatro niveles de estructura de los aminoácidos

1.3 Predicción computacional de estructuras de proteínas

La predicción de la estructura de proteínas es uno de los problemas más importantes en bioinformática. Esto se debe a que la estructura de una proteína determina su función. La motivación de desarrollar sistemas de predicción es el hecho de que de los millones de secuencias que se conocen, alrededor de 219 millones según GenBank [2], solo se conoce la estructura de unas 178.000, que se pueden consultar en el Protein Data Bank [3]. Esto, sumado a que los métodos existentes (como cristalografía de rayos X y Resonancia Magnética Nuclear) para determinar estas estructuras son costosos, sugiere que la mejor alternativa es el uso de métodos de aprendizaje automático.

Existen varias técnicas para la predicción de estructura terciaria. Por ejemplo, la aproximación más difícil, *ab initio*, que consiste en determinar la forma tridimensional de una proteína utilizando sólo la estructura primaria. Este método asume que toda la información necesaria

para el plegamiento de la proteína está contenida en la secuencia de aminoácidos (dogma de Anfinsen) [4] y que la estructura nativa de una proteína posee el mínimo de energía libre de Gibbs.

Otros métodos se basan en comparar proteínas con estructura desconocida con otras cuya estructura ya se conoce. Entre ellos están el *threading* y la homología. El *threading* consiste en predecir la estructura de proteínas con estructura desconocida, comparándolas con proteínas cuya estructura ya se conoce. El proceso se compone de dos partes: diseñar una función que mida la similitud entre proteínas y comparar la proteína a predecir con las que ya se conoce. El primer paso consiste en diseñar una función que toma la secuencia desconocida y la conocida y devuelve un valor de como de similares son. El segundo paso consiste en comparar la proteína a predecir con las ya conocidas, usando la función de similitud previamente diseñada. El resultado de la predicción es la alineación estadísticamente más probable. La homología consiste en comparar la secuencia de aminoácidos que se quiere predecir con proteínas conocidas homólogas, es decir, que comparten parte de la secuencia. Se basa en el hecho de que proteínas que hayan diferido en su secuencia pero tengan aún similitud tendrán estructuras similares. La diferencia entre *threading* y homología es que *threading* usa tanto información de estructura como de la secuencia mientras que homología sólo utiliza información de la secuencia.

1.3.1 Predicción de la estructura secundaria

Sin embargo, este trabajo se centra en la predicción de estructuras secundarias. Se trata de un paso intermedio que intenta simplificar la predicción de la estructura completa usando sólo la cadena de aminoácidos. El conocimiento de la estructura secundaria simplifica la tarea de predicción de la estructura final terciaria, ya que las interacciones principales para formar la estructura final se generan entre los elementos de la estructura secundaria. Además, la estructura secundaria impone restricciones en los ángulos (denominados dihédricos) entre los átomos de la cadena principal (es decir, excluyendo las cadenas laterales o radicales), incorporando así restricciones en el enorme espacio de búsqueda de conformaciones tridimensionales.

El inicio de la predicción de estructuras secundarias precede a la primera estructura conocida de una proteína [5]. Fue en 1951 [6] cuando se propusieron las dos estructuras secundarias más comunes, hélices y laminas; esto fue demostrado experimentalmente después. Desde su descubrimiento se han usado una variedad de técnicas para determinarlas y construir conjuntos de datos para la predicción.

Una de las primeras técnicas de predicción de estructuras secundarias es el método de Chou-

Fasman [7]. Se trata de un método estadístico basado en la propensión de cada uno de los aminoácidos a formar los distintos tipos de estructuras secundarias, según lo observado en proteínas de estructura conocida. El método consiste en tres pasos. Por ejemplo, para identificar hélices alfa, primero se buscan las regiones de la secuencia que tengan 4 de 6 aminoácidos consecutivos con un valor de propensión a formar hélices mayor de 100. Estas secuencias se extienden en ambas direcciones hasta que cuatro aminoácidos consecutivos tengan una propensión a hélice menor de 100. Por último, se calcula el sumatorio de propensión a formar hélice y lámina de las regiones extendidas. Si la región es mayor de 5 aminoácidos y el sumatorio de hélice es mayor al de lámina la región se clasifica como hélice.

Más tarde se comenzaron a usar técnicas de aprendizaje automático para la predicción. Es el caso de Rost y Sander [8], que usaron una red de neuronas artificiales de dos capas para realizar la predicción de estructuras secundarias, consiguiendo un acierto de un 70.8%. Aparte de la información de la cadena de aminoácidos, usaron una técnica conocida como *Multiple Sequence Alignment* (MSA), que consiste en buscar secuencias conocidas similares a las que se está clasificando, y usar esa información adicional para mejorar la predicción. Está técnica también la usa el método de predicción de estructuras secundarias basado en redes de neuronas más conocido y usado, PSIPRED [9]. PSIPRED usa dos redes de neuronas para realizar la predicción. Primero, a la secuencia que se está prediciendo se le aplica el algoritmo PSI-BLAST [10], que es un algoritmo que sirve para encontrar proteínas similares a una secuencia dada. La salida del algoritmo es la entrada de la primera red, que contiene una capa oculta de 75 neuronas y una capa con tres salidas, que realiza una predicción inicial. La salida de esta red se usa como entrada de la segunda red que da la predicción final. Es una red con una capa oculta de 60 neuronas y una capa con 3 salidas. PSIPRED consigue llegar a un 81.6% de acierto de media.

Los porcentajes de acierto de los dos métodos citados hacen referencia a la métrica conocida como Q3, que es la medida estándar de calidad de los métodos de predicción de estructuras secundarias. Q3 hace referencia a la predicción de los tres tipos de estructuras secundarias que se detallan en DSSP [11]. DSSP es el método más estándar a la hora de establecer los elementos de la estructura secundaria. La categorización la realiza simplemente chequeando los ángulos (llamados dihédricos) entre los átomos principales de los aminoácidos (excluyendo la cadena lateral o radical). Generalmente se consideran tres de estas estructuras, que son las que se usarán en este trabajo: hélices, láminas y *coils*. Las dos métricas de calidad que se usarán en este trabajo, Q3 y el porcentaje de acierto individual de cada tipo de estructura secundaria, se explicarán en un apartado posterior.

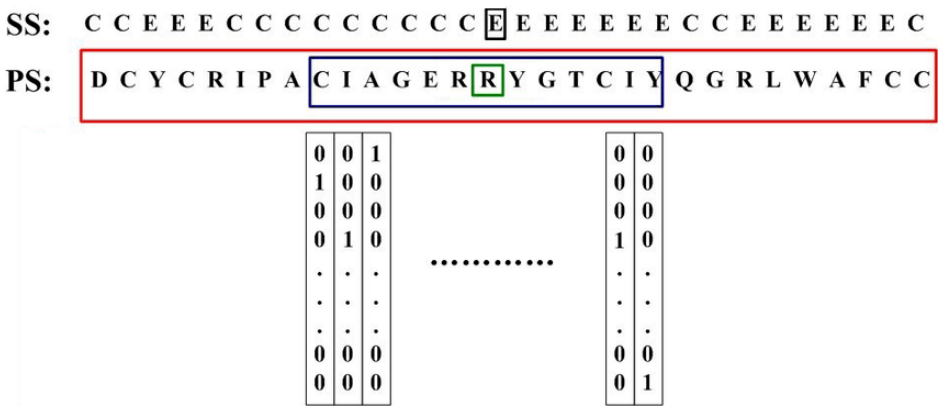

Figura 1.7: Método de predicción de la estructura secundaria

La Figura 1.7 muestra, en concreto, como trabaja un clasificador de estructuras secundarias. Estos usan la información de la secuencia de aminoácidos, en concreto predicen la estructura de un aminoácido central teniendo en cuenta una ventana de aminoácidos vecinos. En la figura, el aminoácido rodeado de verde en la secuencia es del que queremos predecir la estructura y lo rodeado de azul es la ventana que se tiene en cuenta. En este trabajo se usará una ventana de 15 aminoácidos, el mismo tamaño que usa PSIPRED.

Esta técnica se puede utilizar con cualquier método de predicción/clasificación de los elementos de la estructura secundaria. En estas técnicas los aminoácidos se suelen codificar con lo que se llama codificación neutra o binaria. Cada aminoácido se representa por una secuencia de unos y ceros. Esta codificación se muestra en la Figura 1.7, donde, por ejemplo, el aminoácido A se representa con un 1 en la primera posición y el resto a 0. El problema de esta codificación es que no expresa las relaciones existentes entre los aminoácidos, que pueden ser clave en la formación de la estructura de una proteína. Además de esto, como cada aminoácido se representa con 21 bits, la entrada de los clasificadores de estructuras secundarias que utilizan la codificación neutra es de alta dimensionalidad. Por ejemplo, con una ventana de 15 aminoácidos, el número de entradas sería 15*21, es decir, 315 valores. Se usan 21 valores, 20 son para representar cada uno de los aminoácidos y uno extra de relleno para los casos donde la ventana que se está teniendo en cuenta sobrepasa los límites de la secuencia.

El objetivo de este trabajo es obtener un conjunto de propiedades artificiales para substituir la codificación neutra y que, por un lado, plasmen las relaciones que existen entre aminoácidos y, por otro, sirvan como reducción de dimensionalidad de la entrada de los clasificadores.

Capítulo 2

Métodos Utilizados

En está sección se detallarán los métodos utilizados para obtener automáticamente las propiedades artificiales que representan a los aminoácidos, así como las técnicas utilizadas para comprobar su eficacia frente a la codificación neutra. Se comenzará comentando algunas propiedades físico-químicas de los diferentes aminoácidos que son generalmente usadas en el análisis de estos. Más tarde, se detallarán los métodos utilizados para obtener las propiedades artificiales, estos son: el algoritmo evolutivo Evolución Diferencial y los clasificadores k-nearest neighbors y Minimum Distance Classifier. A continuación, se explicará como se integran estos métodos para conseguir las propiedades artificiales. Seguidamente, se explicará el método usado para comparar el rendimiento entre codificación neutra y propiedades artificiales, las Redes de Neuronas Artificiales, así como la herramienta que se usó para implementarlas. Por último, se detallarán las métricas usadas para definir la calidad de los conjuntos de propiedades obtenidos.

2.1 Propiedades de aminoácidos

El objetivo de este trabajo es obtener automáticamente las propiedades (artificiales o abstractas) de los aminoácidos en un espacio de propiedades multidimensional y que faciliten la clasificación o predicción de los elementos de la estructura secundaria. Primero se describirán algunas propiedades físico-químicas en los aminoácidos, que se compararán en el capítulo de resultados con las propiedades obtenidas con el proceso evolutivo. Estas propiedades son: *hydropathy*, *polar requirement* y *molecular volume*. Se han elegido estas tres propiedades por ser comúnmente usadas, por ejemplo, en el estudio de la optimalidad del código genético (análisis que tiene en cuenta los cambios en las propiedades de un aminoácido ante cambios o mutaciones en las bases de los codones del código genético) [12][13].

Tanto *hydropathy* como *polar requirement* son medidas de hidrofobicidad del aminoácido.

Aminoácido	*Polar Requirement*	*Hydropathy*	*Molecular Volume*
Ala	7	1.8	31
Arg	9.1	-4.5	124
Asp	13	-3.5	54
Asn	10	-3.5	56
Cys	4.8	2.5	55
Glu	12.5	-3.5	83
Gin	8.6	-3.5	85
Gly	7.9	-0.4	3
His	8.4	-3.2	96
Ile	4.9	4.5	111
Leu	4.9	3.8	111
Lys	10.1	-3.9	119
Met	5.3	1.9	105
Phe	5	2.8	132
Pro	6.6	-1.6	32.5
Ser	7.5	-0.8	32
Thr	6.6	-0.7	61
Trp	5.2	-0.9	170
Tyr	5.4	-1.3	136
Val	5.6	4.2	84

Tabla 2.1: Propiedades de los aminoácidos

Hydropathy es un número que representa las características hidrofílicas o hidrófobas del radical del aminoácido. Cuanto más alto sea este índice más hidrófobo será y viceversa. Por otro lado, el *polar requirement* [14] es una propiedad que mide vagamente el número de moléculas de agua que puede enlazar firmemente cada aminoácido. Por último, el *molecular volume* de cada aminoácido es una propiedad que se obtiene del resultado de restar el volumen de residuo del aminoácido [15] al volumen constante del péptido.

En la Tabla 2.1 se muestran los valores de estas tres propiedades. Debido a que el algoritmo evolutivo que se va a usar codifica los valores de sus individuos con números reales en el rango [0,1], los valores de esta tabla se normalizarán a ese rango. De esta manera se podrán comparar con las propiedades artificiales resultado del proceso evolutivo para ver si existe alguna correlación entre ellas.

2.2 Evolución Diferencial

Evolución diferencial (ED) [16][17] es un algoritmo evolutivo útil para resolver problemas de optimización global. En este trabajo se usará para evolucionar las propiedades artificiales asociadas a cada aminoácido. ED ha sido escogido debido a que ha demostrado ser un algo-

ritmo robusto para numerosos problemas y porque los genotipos, es decir, los valores que codifican a una posible solución, son números reales, ideales para representar las propiedades artificiales de los aminoácidos.

Al tratarse de un algoritmo evolutivo, ED, mantiene una población donde cada individuo es una solución candidata al problema. Como otros algoritmos basados en población, ED crea nuevos individuos a partir de los ya existentes en la población. En concreto, lo que diferencia a ED es que los crea a partir de la diferencia escalada de dos individuos aleatorios de la población sumada a un tercero, también aleatorio. ED consiste en cuatro pasos: inicialización, mutación, recombinación y selección.

Algoritmo 1: Algoritmo estándar de Evolución Diferencial

```
1  Inicializar la población aleatoriamente;
2  repetir
3      para todo individuo x de la población hacer
4          Sean x1, x2 y x3 ϵ a la población obtenidos aleatoriamente {de modo que x1,
             x2, x3 y sean distintos entre sí} ;
5          Sea R ϵ 1, ..., n, obtenido aleatoriamente {n especifica la dimensión del espacio
             de búsqueda};
6          para i = 1 a n hacer
7              Escoger ri ϵ U(0,1) uniformemente del rango abierto (0,1);
8              si (i = R) ∨ (r_i < CR) entonces
9                  yi ← x1i + F(x2i − x3i);
10                 en otro caso
11                     yi = xi;
12                 fin
13             fin
14         fin
15         si f(y) ≤ f(x) entonces
16             El individuo candidato y creado reemplaza al x;
17         fin
18     fin
19 hasta que se cumpla condición de parada;
   Salida: z ϵ población \∀ t ϵ población, f(z) ≤ f(t)
```

En el primer paso se inicializa la población. Para todos los individuos de la población se aleatorizan todos sus valores en el rango [0,1].

Con la población inicializada empieza un proceso iterativo durante un número concreto de generaciones o con otra condición de parada. En cada generación se recorre la población entera y, para cada individuo, se realizan los pasos de mutación, recombinación y selección.

Primero la mutación. Para cada individuo se escogen tres vectores aleatorios de la población, vectores que no pueden coincidir (X1, X2 y X3 en el pseudo-código). Con los vectores escogidos se genera el vector donador, que es resultado de restar el segundo y tercer vector, multiplicado por el parámetro F, y sumar esto al primer vector (denominado vector base), como se muestra en la línea 9 del pseudo-código. Este paso de mutación se muestra también gráficamente en la figura 2.1. En esta última los puntos en negro son los vectores escogidos aleatoriamente y el punto gris el vector resultado, es decir, el vector donador.

ED depende solamente de tres parámetros: NP, el número de individuos de la población, CR, que es un valor en rango [0,1] y define la fracción de valores que se copian del vector donador, y F, que controla la velocidad a la que evoluciona la población, típicamente también el rango [0,1].

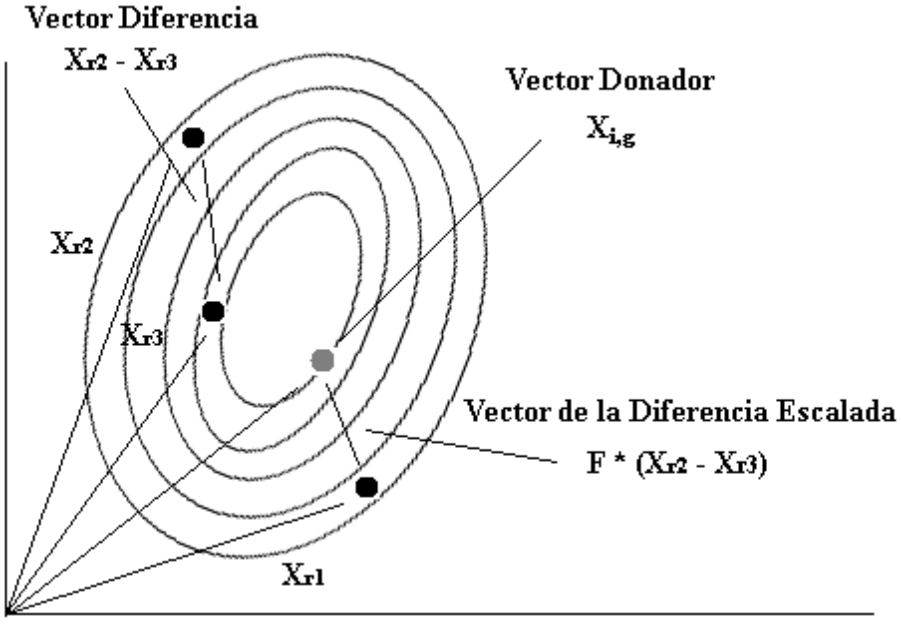

Figura 2.1: Paso de mutación en ED

El siguiente paso es el de recombinación. Para este paso se necesitan generar dos elementos: un vector de números aleatorios entre 0 y 1 del tamaño de un individuo de la población, que llamaremos V, y un número entero en el rango del tamaño del individuo de la población, que llamaremos R.

En este paso de recombinación obtendremos el vector denominado candidato (y en el pseudo-código), que es resultado de combinar (mediante cruce de los genotipos) el vector donador y el vector de la población actual. Recorremos el vector V y, para cada posición, si el valor es menor a CR, o la posición es igual a R, el valor de la posición del vector candidato será igual al de esa posición del vector donador. En otro caso será igual al de la posición del vector actual de la población (vector x en el pseudo-código). Por tanto, cuanto mayor es el valor de CR, se usa el valor del vector donador con mayor frecuencia. También es importante R, ya que asegura que se toma del vector donador al menos una posición en la generación del candidato de cada individuo x de la población.

El último paso es la selección, en la línea 15 del pseudo-código. Este paso consiste en decidir si el vector candidato es mejor que el individuo de la generación actual. Para esto se utiliza una función, llamada función objetivo, que evalúa la calidad de cada uno. Si el vector candidato es mejor que el actual, lo sustituye en la generación siguiente. Al acabar con todos los individuos se reemplaza la población y se pasa a la siguiente generación. Este proceso continúa hasta un número de generaciones concreto u otro criterio de parada.

2.3 k-nearest neighbors

k-nearest neighbors[18] (k-NN) es un algoritmo de clasificación y regresión supervisado y no paramétrico. Un método de aprendizaje automático supervisado es aquel que aprende a partir de ejemplos que ya tienen una etiqueta o un valor asociados a ellos. Un método paramétrico es aquel que resume la información que tiene en un conjunto de parámetros de tamaño fijo. k-NN es no paramétrico, ya que, en su entrenamiento no optimiza ningún parámetro, realiza clasificación o regresión usando los ejemplos de los que dispone. El único parámetro de k-NN es k, el número de vecinos a tener en cuenta en la clasificación y que se especifica antes del entrenamiento.

k-NN funciona de la siguiente manera. Partiendo de un conjunto de datos, el primer paso es el entrenamiento. En el caso de k-NN consiste en dividir estos datos en un conjunto de entrenamiento y un conjunto de test. En el conjunto de entrenamiento, los datos ya tienen su categoría correcta asociada (en un problema de clasificación), al contrario de los datos del conjunto de test. Para realizar una predicción sobre un elemento, que es un vector uni o multidimensional de valores numéricos, se calcula su distancia con todos los ejemplos del conjunto de entrenamiento. La distancia que se usa normalmente es la distancia euclídea, pero otras son válidas. Cuando todas las distancias están calculadas se ordenan y se cogen los k

primeros valores.

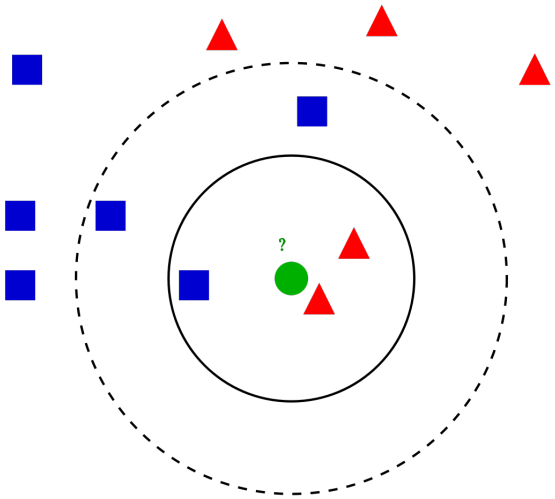

Figura 2.2: Ejemplo de k-NN con k=3

Con los k vecinos más cercanos el algoritmo devuelve el resultado. En problemas de clasificación el algoritmo devuelve la etiqueta de la moda si k > 1, o la etiqueta del más cercano si k = 1. En regresión, devuelve la media de los vecinos si k > 1 o el valor del más próximo si k = 1. En la Figura 2.2 se muestra un ejemplo del funcionamiento de k-NN; si queremos clasificar el punto verde con k = 3, la etiqueta resultado será el triángulo rojo, ya que de los 3 vecinos 2 pertenecen a esa clase.

Aunque k-NN es útil es muchos casos presenta algunos problemas. El primero es que, aunque respecto a otros algoritmos de clasificación el tiempo de entrenamiento es prácticamente nulo, el tiempo de ejecución puede ser problemático. Esto se debe a que, para cada elemento de test, se tiene que calcular la distancia con todos los elementos de entrenamiento, y esto en conjuntos grandes de datos es bastante costoso en tiempo de ejecución. Otro problema es la conocida como maldición de la dimensión. En el caso de k-NN funciona bien cuando cada elemento tiene pocas dimensiones y hay muchos ejemplos de entrenamiento, en este caso, los vecinos más cercanos estarán cerca en el espacio. Sin embargo, a medida que las dimensiones aumentan, la distancia entre ejemplos y vecinos aumenta también. Esto quiere decir que el número de ejemplos que se usan debería crecer de la misma forma para mantener su densidad en el espacio. Por lo tanto, antes de decidir si aplicar k-NN a un problema, se debe conocer cómo son los datos así cómo cuántos hay, para evitar caer en problemas de este tipo. Por último, es importante escoger un buen valor para k. El problema es que no existe un método concreto para determinar qué número de vecinos escoger, sino que es dependiente del problema a tratar. En el caso de este trabajo se harán pruebas con k = 3 y k = 15 vecinos, 2

valores usados en muchos ejemplos de clasificación con k-NN.

2.4 Minimum Distance Classifier

Minimum Distance Classifier (MDC) es otro algoritmo de clasificación supervisado y no paramétrico. Para clasificar un elemento usando MDC, primero, se determina un conjunto de entrenamiento. En este conjunto todos los elementos, que son vectores en un espacio multidimensional de características, tienen una etiqueta asociada a una de las n clases que se están intentando predecir o clasificar. Estos vectores se usan para crear n vectores nuevos, uno por clase. Estos elementos actúan como representantes de cada clase y son resultado de calcular la media de todos los vectores que pertenezcan a cada una de las clases. Esto hace que todo el conjunto de entrenamiento se condense en solamente un número de elementos igual al número de clases a predecir. Con los representantes calculados, para clasificar un nuevo dato, se calcula la distancia de este con cada representante y se devuelve la etiqueta del representante más cercano.

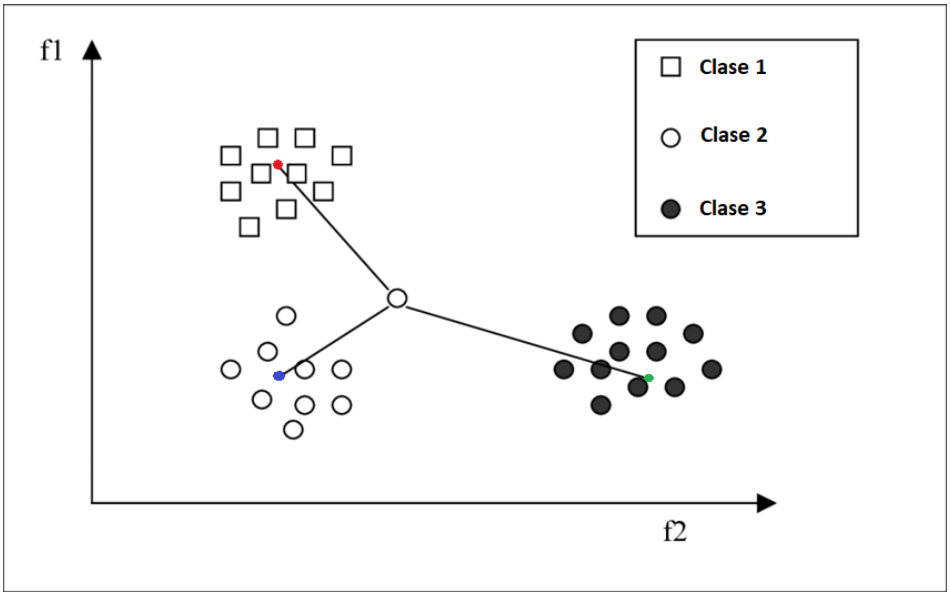

Figura 2.3: Funcionamiento de MDC

En la Figura 2.3 se muestra el funcionamiento de MDC. Los círculos de distinto color son los resultados de calcular los representantes de cada una de las tres clases. Para el círculo blanco del centro de la figura, que es el nuevo dato a clasificar, se calculará la distancia con cada representante y se le asignará la etiqueta más cercana. En este caso la etiqueta resultado será ”Clase 2”, ya que el representante más cercano es el punto azul.

Una ventaja que MDC presenta, es que mucho más eficiente, en cuanto a tiempo, que otros algoritmos. Mientras que k-NN, por ejemplo, tiene que calcular la distancia de todos los puntos de test con todos los de entrenamiento, MDC calcula solo un número de distancias igual al número de clases que haya.

2.5 Esquema general del proceso de obtención automática de las propiedades: definición de calidad de cada combinación de propiedades con los clasificadores k-NN y MDC

Esta sección detallará cómo los algoritmos ED, k-NN y MDC se combinan para determinar la calidad de los vectores de propiedades artificiales de los aminoácidos. En este trabajo, cada uno de los vectores de ED codifica un conjunto de propiedades para cada aminoácido, miesntras que los clasificadores k-NN y MDC determinan la función objetivo del algoritmo evolutivo para evaluar la calidad de cada vector de la población.

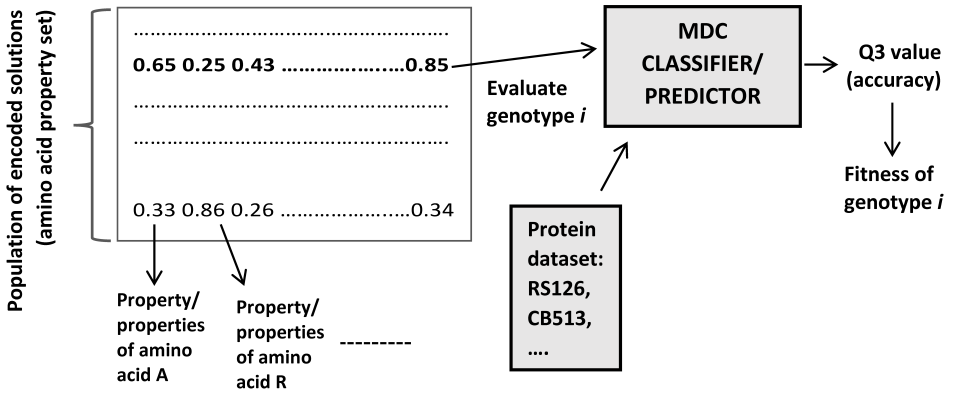

Figura 2.4: Esquema general de cómo se obtienen las propiedades artificiales

La Figura 2.4 muestra el proceso general que se sigue para determinar la calidad o *fitness* de cada uno los vectores de ED. La idea general es que, si la asociación de las propiedades artificiales asociadas a cada aminoácido (y codificadas en cada posición en el vector solución de ED) es adecuada, se facilitará la clasificación proporcionada por k-NN o MDC, generando una medida de clasificación alta. Si las propiedades no son adecuadas, la clasificación será más difícil y el clasificador proporcionará peores métricas de clasificación.

Como se puede observar el clasificador tiene dos entradas. La primera es cada uno de las soluciones candidatas de ED, con los valores para cada aminoácido. La segunda es el conjunto de datos, en concreto, como se detalló en el apartado 1.3.1, un conjunto de aminoácidos centrales rodeados cada uno con su ventana correspondiente. Teniendo esto en cuenta el pro-

ceso para determinar la calidad de un vector de ED es el siguiente: primero, los aminoácidos de la ventana se sustituyen por su valor correspondiente de las propiedades codificadas en el vector, después a este conjunto se le aplica el clasificador. Una vez que el clasificador se ha usado para determinar la clase (hélice, lámina o *coil*) de cada elemento central de cada ventana de aminoácidos, se calcula una medida de calidad de la predicción que se retorna al algoritmo evolutivo como la calidad o fitness de la solución (vector de propiedades). Esta medida de calidad es el Q3, ya mencionado en el apartado 1.3.1 y que se explicará posteriormente.

Por último, cabe destacar los conjuntos de datos que se usaron para el entrenamiento y el test. En el caso de MDC se usaron dos conjuntos de datos: el Rost-Sander 126 (RS126)[8] y el Cuff-Barton 513 (CB513) [19], conteniendo estos 126 y 513 proteínas no redundantes respectivamente. k-NN se consideró usando sólo el conjunto RS126 debido al gran tamaño del CB513. El RS126 y el CB513 proveen 24437 y 82003 asociaciones respectivamente, entre una ventana de aminoácidos y una estructura de un aminoácido central. Es decir, cada conjunto tiene ese número de ventanas, cada una con un aminoácido central y, por tanto, una estructura secundaria asociada.

2.6 La neurona y las redes de neuronas

Las redes de neuronas artificiales son sistemas computacionales inspirados en el funcionamiento del cerebro de los animales. Estas redes están formadas por una unidad básica, la neurona artificial, cuyo funcionamiento está inspirado en la neurona biológica.

Dado que uno de los métodos más extendidos para la predicción de estructuras secundarias son las redes de neuronas artificiales, en este trabajo se usarán para comparar la efectividad en la predicción entre codificar los aminoácidos con las propiedades artificiales y usar la codificación neutra.

2.6.1 La neurona artificial

Una neurona artificial es un modelo matemático basado conceptualmente en las neuronas biológicas. El primer modelo de neurona artificial se propuso en 1943 por Warren McCulloch y Walter Pitts [20]. A cada neurona le llegan una serie de entradas cada una con un peso asociado y produce una salida acorde a su comportamiento, que puede ser la entrada de otra neurona.

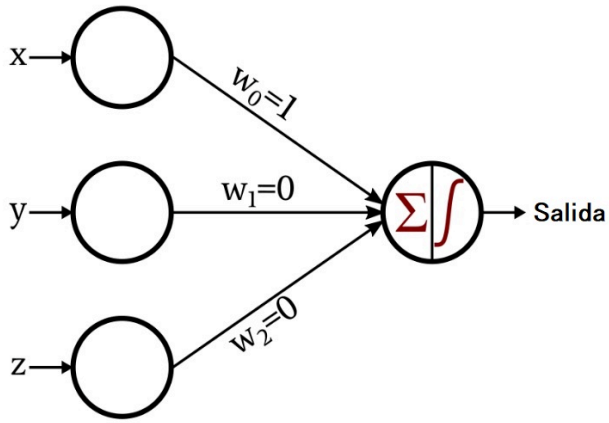

Figura 2.5: Modelo de neurona artificial

En la Figura 2.5 se muestra la estructura de una neurona artificial. Por un lado están las entradas, x, y y z, y sus correspondientes pesos w_0, w_1 y w_2. El funcionamiento de una neurona artificial cuenta con dos pasos. Primero se calcula el sumatorio de todas las entradas multiplicadas por sus respectivos pesos, y a esto se le suma un bias. El resultado de este sumatorio es la entrada del segundo paso del funcionamiento de una neurona artificial, la función de activación. Tomando el valor del sumatorio, esta función es la que decide si la neurona se activa o no. La elección de la función de activación es muy importante, ya que dependiendo de la que sea, cuando se entrene la red, esta será capaz de aprender las relaciones que le permita su función de activación. Por ejemplo, si se está tratando un problema de clasificación no lineal, es decir, que sus clases no son separables por una línea, la función de activación ha de ser no lineal, para que la red pueda aprender ese tipo de relación. Con una función lineal, la red no la podría aprender y, por lo tanto, sus resultados serían malos. Existen varios tipos de funciones de activación típicas, entre ellas se encuentran la función umbral, la sigmoide o también la lineal.

2.6.2 Redes de Neuronas Artificiales

Las neuronas descritas en el apartado anterior se usan como unidad básica de las redes de neuronas artificiales. Estas redes son sistemas computacionales basados en las redes de neuronas biológicas presentes en los cerebros de los animales.

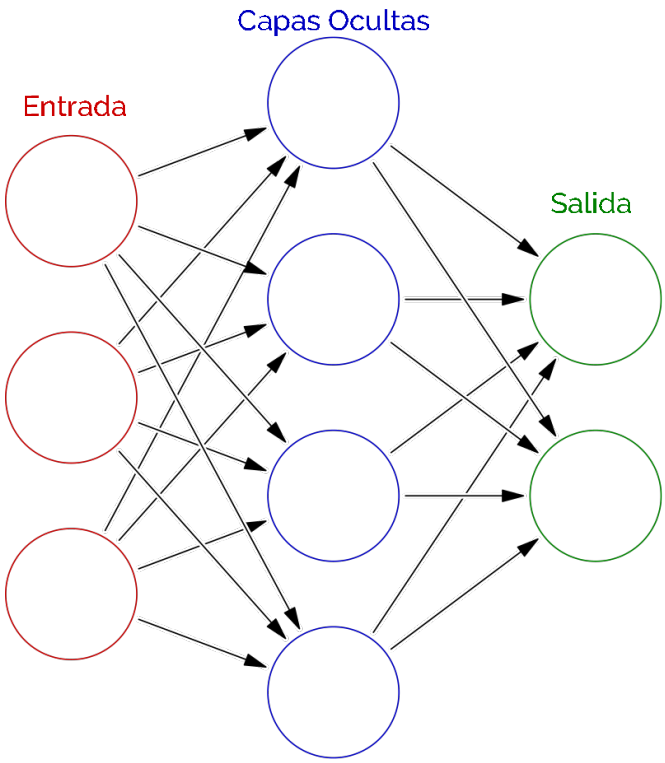

Figura 2.6: Topología general de una red de neuronas artificiales sin realimentación

La Figura 2.6 muestra la estructura típica de una red de neuronas artificiales. Cuenta con una serie de entradas, un número determinado de capas ocultas, cada una con sus correspondientes neuronas, y una capa de salida que ofrece el resultado. Las neuronas se pueden conectar como vemos en la figura, mandando la señal solo hacia la capa siguiente; estas redes se conocen como *feed-forward*. No obstante, también existen redes que mandan las salidas de sus neuronas a la misma capa o a capas anteriores, conocidas como redes recurrentes.

Por lo tanto, para construir una red de neuronas, los parámetros básicos son: cuantas entradas tiene la red, cuántas salidas, el número de capas ocultas, el número de neuronas en cada capa y su función de activación. Una vez se deciden todos los parámetros el funcionamiento de una red es relativamente sencillo. Las entradas pasan de capa a capa según las conexiones entre neuronas y su función de activación hasta llegar a la capa de salida donde la red proporciona el resultado.

Un paso clave del funcionamiento de las redes de neuronas artificiales es cómo aprenden. En el esquema de aprendizaje supervisado clásico y ampliamente utilizado, esto se realiza mediante un mecanismo conocido como *back-propagation*. Este algoritmo se usa para corregir el

error obtenido en el resultado que devuelve la red. Esto implica recalcular el peso asignado a cada conexión. Cuando se está entrenando la red de neuronas y ésta devuelve un resultado incorrecto, ya que conocemos la salida deseada, podemos calcular el error que se ha cometido. Este error se propaga hacia atrás (de ahí el nombre) a través de todas las capas para corregir los pesos asociados a cada entrada. Esto se basa en la idea de que cada nodo es responsable de una fracción del error, por lo que este error se divide acordemente y se usa para corregir las conexiones.

Teniendo en cuenta la topología y el funcionamiento de las redes de neuronas artificiales, existen muchos tipos de ellas, desde modelos con una sola capa oculta a los modelos de *deep learning*, con un gran número de neuronas por capa y un enorme número de capas ocultas. En este trabajo se usará una red *feed-forward* de 2 capas con 10 neuronas por capa para comparar el rendimiento de las propiedades artificiales obtenidas y la codificación neutra de los aminoácidos, tal como se explicó previamente en el apartado 1.3.1.

2.6.3 Librería Keras en Python

Python es un lenguaje de programación interpretado y una de las herramientas más extendidas en el campo del aprendizaje máquina. Esto se debe a que es un lenguaje simple y que ofrece una gran número de librerías para toda clase de problemas de aprendizaje automático. En concreto, para este trabajo, se utilizó la librería *Keras* [21] para la implementación de las redes de neuronas. *Keras* es una librería de código abierto que provee una interfaz en Python para redes de neuronas artificiales. Se caracteriza por permitir la creación de prototipos de todo tipo de redes de forma rápida mediante su sencilla interfaz.

```
model = tf.keras.Sequential()
#Capa de entrada
model.add(tf.keras.layers.Input(input_shape = (15 *p, )))
#Capas ocultas
model.add(tf.keras.layers.Dense(10, activation = "relu"))
model.add(tf.keras.layers.Dense(10, activation = "relu"))
#Capa de salida
model.add(tf.keras.layers.Dense(3, activation = "softmax"))

model.compile(optimizer = "adam", loss='mean_squared_error', metrics = ["accuracy"])
model.fit(X_train, y_train, epochs = 150)
```

Figura 2.7: Ejemplo de definición de una red de neuronal artificial en la librería Keras

La Figura 2.7 muestra cómo se define una red de neuronas *feed-forward* en Keras. Para crear una red, primero se instancia un objeto *Sequential*, donde se van a ir añadiendo las capas. Después, con el método *add* se van añadiendo las capas deseadas, con su número de neuronas y función de activación. En concreto, la red de la figura se corresponde a la usada en este trabajo. Se trata de una red con una capa de entrada, que tiene un tamaño de 15, el tamaño

de la ventana de aminoácidos, multiplicado por *p*, el número de propiedades por aminoácido. Dos capas ocultas de 10 neuronas cada una con la función de activación *relu* o rectificador, que es una función simple, rápida y que ofrece buenos resultados en muchos problemas de clasificación o predicción. Y una capa de salida con 3 neuronas, una por cada tipo de estructura secundaria. La función de activación escogida en este caso es *softmax*, que tiene buenos resultados en problemas de clasificación con más de dos clases. Se ha escogido como función de "pérdida" el *mean squared error* o MSE y como optimizador para *back-propagation*, *adam* [22], debido a que es sencillo, computacionalmente eficiente y muestra buenos resultados con conjuntos grandes de datos. Adam se diferencia del *back-propagation* clásico en que ajusta de forma distinta todos los pesos de la red, mientras que, el algoritmo clásico los ajusta todos de la misma manera. Como otros optimizadores, su principal parámetro es el *learning rate* o tasa de aprendizaje. Este valor simboliza lo rápido que la red neuronal aprende un problema, en este caso se ha usado 0.01, un valor muy común y que ha proporcionado un aprendizaje "suave" y continuo a lo largo de las iteraciones de aprendizaje.

Para evitar que, al entrenar la red neuronal, la selección aleatoria de conjuntos de entrenamiento y test cree resultados sesgados, y para observar qué tal generaliza la red a nuevos datos, se ha utilizado la técnica conocida como *k-fold cross-validation*, que está ya implementada en Keras.

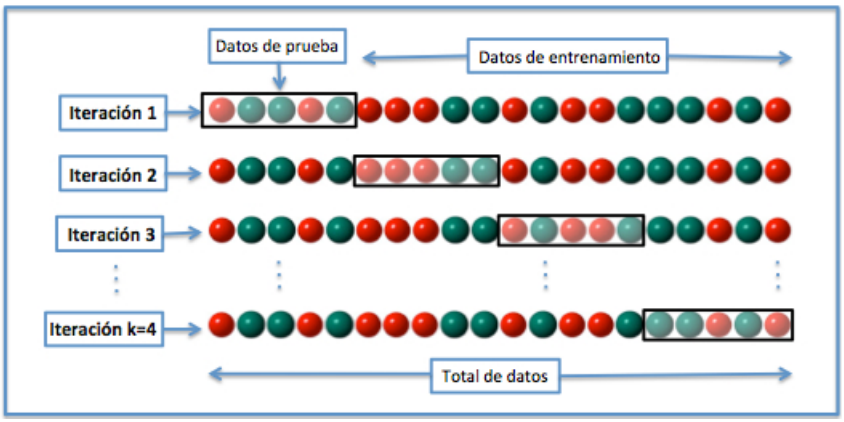

Figura 2.8: *k-fold cross-validation* con k = 4

La Figura 2.8 muestra el funcionamiento de *k-fold cross-validation*. Para entrenar las redes de neuronas, en este trabajo, se han dividido inicialmente los conjuntos de datos en entrenamiento y test, con una proporción de un 80%-20%. Posteriormente se ha aplicado un 10-*fold cross-validation* en el conjunto de entrenamiento, es decir, que en cada iteración del *cross-validation* se ha usado, del conjunto de entrenamiento, un 90% para entrenar y un 10% para validar. Luego, la red final se ha probado con el conjunto de test. Esta técnica se ha usado

también para estimar el número de ciclos de entrenamiento o *epochs* de la red de neuronas artificiales. Observando el valor de la función de "pérdida" en los conjuntos de validación y observando cuándo ésta se estanca, es decir, cuando el error ya no baja más. El valor que se obtuvo es de 150 ciclos.

2.7 Métricas

Por último, se explicarán las dos métricas usadas para medir la calidad de los resultados obtenidos.

Estructura	Hélice	Lámina	*Coil*
Hélice	2000	325	727
Lámina	546	803	532
Coil	932	419	2967

Tabla 2.2: Ejemplo de matriz de confusión en la predicción de estructuras secundarias

La Tabla 2.2 muestra un ejemplo de una matriz de confusión resultado de la predicción de estructura secundaria de proteínas. Una matriz de confusión es una herramienta que sirve para determinar el rendimiento de un algoritmo de clasificación, mostrando qué elementos se clasificaron correctamente y cuáles no. La primera métrica es el Q3, mencionada anteriormente en la sección 1.3. Se denomina de esta forma debido a que hace referencia a los 3 tipos de estructuras y es, básicamente, el porcentaje de acierto global de la predicción. Se calcula dividiendo el resultado de la suma de la diagonal de la matriz, es decir, el número de elementos clasificados correctamente entre la suma de todos los elementos de la matriz. Por tanto, es la medida clásica de *accuracy* en el problema de clasificación actual. Aunque las clases no están balanceadas en el número de ejemplos en las tres categorías a clasificar (hélices, láminas y *coils*) , tampoco están fuertemente descompensados en los conjuntos estándar de *benchmark*, por lo que es la medida más usual en predicción de estructura secundaria.

La otra métrica que se usará es el porcentaje de acierto individual de cada elemento. Por ejemplo, de todas las hélices que había, ¿cuántas se predijeron correctamente? Para el caso de hélices se calcularía dividiendo el número de hélices clasificadas correctamente, que es donde se cruzan la fila y columna de hélice entre la suma de todos los elementos de la fila de hélices, es decir, todos los elementos que eran hélices.

El Q3 se usará como métrica de calidad del algoritmo evolutivo, ED, para decidir qué individuo es mejor que otro. El Q3 y el porcentaje individual se usarán en las pruebas finales

con RNA para analizar los resultados de la comparación entre la codificación neutra y las propiedades artificiales.

Capítulo 3

Metodología de desarrollo

En esta sección se detallará, primero, la metodología escogida para realizar el proyecto. Después se explicarán brevemente las etapas de desarrollo que se llevaron a cabo así como los costes estimados del proyecto. Por último, se enumerarán las herramientas utilizadas.

3.1 Metodología

Este proyecto comenzó con un estudio exhaustivo del problema. La finalidad del estudio fue comprender las técnicas y algoritmos necesarios y, de esta manera, atacar el problema de forma concreta. Un conocimiento extenso del dominio también permite valorar mejor los resultados y tomar decisiones a partir de ellos. Una vez identificado el problema, la mejor solución resultó ser la elección de una metodología ágil para adaptarse mejor a los requisitos.

Estos requisitos incluyen la toma de decisiones a partir de los resultados obtenidos, el posible cambio de requisitos del problema o la minimización de los posibles riesgos del proyecto. Por todas estas razones, se ha elegido la metodología *Scrum*, ya que ofrece gran flexibilidad a la hora de adaptarlo a un proyecto. *Scrum* es una metodología ágil que divide las metas de un proyecto en distintas iteraciones, llamadas *sprints*, donde en cada *sprint* se pretende llevar a cabo una tarea concreta. Sin embargo, es cierto que no se ha usado la metodología *Scrum* estándar, dado que uno de sus aspectos principales es el trabajo en equipo y el reparto de tareas entre sus miembros, mientras que este proyecto ha sido desarrollado solo por el alumno junto al director. Identificándolos con los roles de *Scrum* serían, por un lado, el dueño del producto, en este caso el tutor, José Santos Reyes y, por otro, el desarrollador, asumido por el alumno.

3.2 Planificación del trabajo

En esta sección se describirán brevemente los distintos *sprints* en los que se han agrupado las tareas del proyecto, luego se estima el tiempo y coste del mismo.

3.2.1 *Sprints*

El proyecto se ha dividido en los siguientes *sprints*:

- ***Sprint* 0** - Estudio inicial: Aquí se estudiaron los algoritmos y conjuntos de datos iniciales del proyecto
- ***Sprint* 1** - Implementación del algoritmo ED
- ***Sprint* 2** - Implementación del clasificador k-NN
- ***Sprint* 3** - Pruebas con k-NN
- ***Sprint* 4** - Implementación de MDC
- ***Sprint* 5** - Pruebas con MDC
- ***Sprint* 6** - Comparar propiedades obtenidas con k-NN y MDC y compararlas con las propiedades físico-químicas
- ***Sprint* 7** - Implementar las RNA
- ***Sprint* 8** - Pruebas con RNA
- ***Sprint* 9** - Comparar resultados entre las propiedades artificiales y la codificación neutra usando RNA
- ***Sprint* 10** - Redacción de la memoria

3.2.2 Tiempo y coste

Excluyendo los largos tiempos de ejecución del algoritmo evolutivo, la estimación de tiempo de trabajo es de unas 530 h. Esto supone que el tiempo medio de cada sprint son unas 53 h. Si se estima que el sueldo medio a la hora de un programador es de 16 €, esto supone un coste de 8480 €.

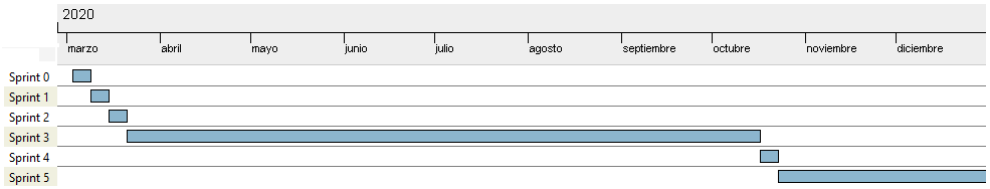

Figura 3.1: Diagrama de Gantt de los *sprints* correspondientes al año 2020

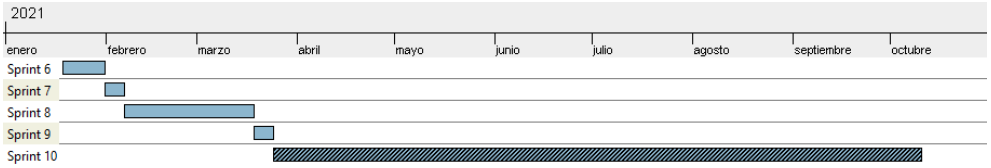

Figura 3.2: Diagrama de *Gantt* de los *sprints* correspondientes al año 2021

Las Figuras 3.1 y 3.2 muestran los diagramas de Gantt correspondientes a los *sprints* en los años 2020 y 2021.

3.2.3 Herramientas

- Bibliografía
- Python
- C
- Spyder (Entorno de desarrollo en Python)
- Ordenador (Windows 10 64 bits) (Intel i7-8750H 16 GB RAM)

Capítulo 4

Resultados

En este capítulo se presentarán los resultados obtenidos en el trabajo. La primera sección detalla los resultados obtenidos al usar usar MDC para definir la función objetivo del algoritmo evolutivo. En la segunda, se mostrarán los resultados de usar k-NN para definir la función objetivo. A su vez, cada una de estas secciones tendrá un apartado para cada número de propiedades evolucionadas por aminoácido, en concreto, 1, 2 y 3 propiedades por aminoácido. Para obtener los resultados que se muestran en estas dos secciones, se realizaron, en cada apartado, 10 ejecuciones independientes del algoritmo evolutivo, cada uno de 1500 generaciones. Se realizaron ejecuciones independientes para observar los distintos conjuntos de propiedades obtenidos y para asegurarse que el algoritmo convergía a un óptimo.

Los parámetros del algoritmo evolutivo fueron: NP = 100, CR = 0.9, F = 0.5. Estos valores se determinaron de modo experimental, y son los que proporcionan evoluciones sin que exista convergencia prematura en pocas generaciones. La última sección contiene los resultados de comparar el uso de las propiedades artificiales para codificar los aminoácidos contra la codificación neutra. Esta comparación se llevó a cabo mediante el uso de redes de neuronas artificiales.

4.1 Minimum Distance Classifier

En esta sección se comentarán los resultados obtenidos de evolucionar 1, 2 y 3 propiedades por aminoácido utilizando el clasificador MDC a la hora de asignar la calidad (función objetivo) a cada solución de la población genética del algoritmo evolutivo. Habrá una sección para cada número de propiedades y, dentro de cada una, los resultados de aplicar MDC, primero evolucionando con RS126 y después con CB513.

4.1.1 Evolución de una propiedad por aminoácido

En esta sección se presentan los resultados obtenidos usando el algoritmo MDC y evolucionando una propiedad por aminoácido con el algoritmo evolutivo.

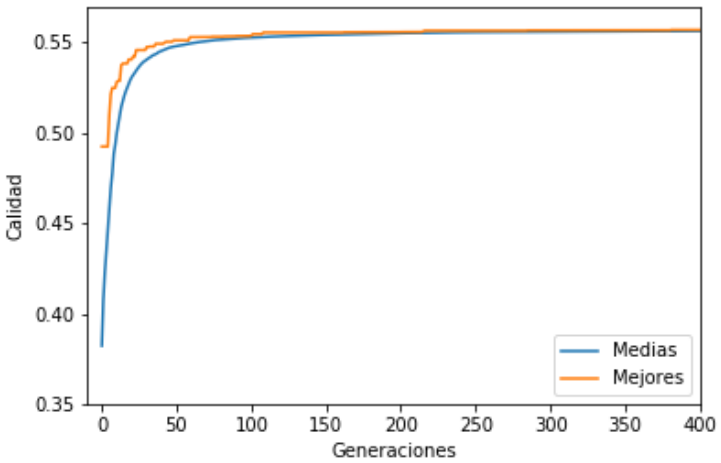

Figura 4.1: Calidad media y mejor de la población a lo largo de las generaciones, con 1 propiedad y RS126

La Figura 4.1 muestra la calidad media de la población y la calidad del mejor individuo a lo largo de las generaciones del proceso evolutivo en una ejecución concreta. Aunque el proceso evolutivo se ejecutó durante 1500 generaciones, la gráfica sólo muestra hasta la generación 400 para que se aprecie mejor la evolución. Como se ve, el algoritmo converge alrededor de la generación 200, y a partir de ahí se mantiene estable. La calidad del mejor individuo se estanca alrededor del 55.57%, en esta ejecución en particular (recordemos que es la medida de clasificación Q3) y sube muy poco durante el resto de generaciones, llegando a un máximo de 55.91%. Como muestra la figura, los valores de mejor y media de población llegan a ser prácticamente iguales. Esto se debe a la alta presión de selección en ED.

El tiempo de cómputo típico involucrado es el siguiente: cada proceso evolutivo independiente (1500 generaciones), con la población indicada y el conjunto RS126, en una plataforma con Ubuntu 18.04, processador Intel i7-8750H, y 16 Gbytes de memoria, es de 450 minutos.

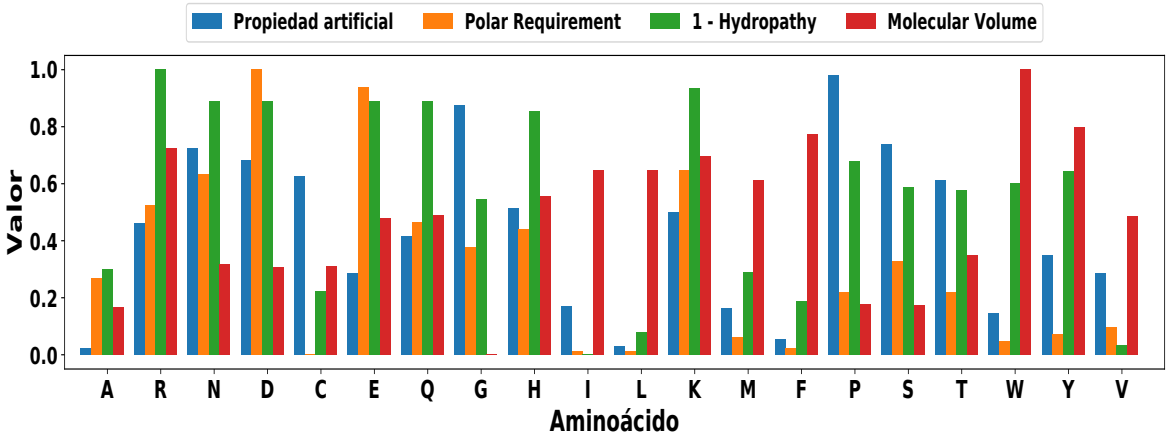

Figura 4.2: Propiedades obtenidas para cada aminoácido con MDC y RS126 en un espacio unidimensional de propiedades

La Figura 4.2 muestra una comparación de las propiedades obtenidas por cada aminoácido con las propiedades conocidas: *hydropathy*, *polar requirement* y *molecular volume*. En vez de comparar con la *hydropathy* se compara con la inversa (1 - *hydropathy*). Se compara con la inversa, ya que así se muestra mejor la correlación que existe entre las dos medidas de hidrofobicidad, que es alta en la mayoría de los aminoácidos. El eje X se corresponde con los 20 aminoácidos, usando la nomenclatura estándar de una letra en su denominación. Los resultados de esta figura corresponden a los valores de propiedades obtenidos en una de las ejecuciones independientes del algoritmo evolutivo, teniendo en cuenta que los valores evolucionados son muy similares en las diferentes ejecuciones de ED, con bajas desviaciones estándar.

Comparando las propiedades obtenidas con las propiedades físicas vemos algunas similitudes. Por ejemplo, el aminoácido Leucina (L) o el aminoácido Fenilalanina (F), muestran valores bajos similares entre la propiedad evolucionada y su *polar requirement*. También aminoácidos como la Asparagina (N) o la Lisina (K), tienen valores altos en la propiedad evolucionada y valores similares en *polar requirement*. Estas observaciones se pueden aplicar también a la *hydropathy* inversa, ya que la correlación entre éste y el *polar requirement* es fuerte en la mayoría de casos.

Por último, comentaremos los resultados de calcular el coeficiente de correlación de Pearson entre las propiedades evolucionadas y las conocidas. Este índice es una buena forma de comprobar si existe alguna relación entre dos conjuntos de propiedades. El índice toma valores entre -1 y 1: si es 0 es que no existe correlación; sin embargo, cuanto más grande, ya sea

al máximo positivo o al valor mínimo negativo, indica una relación entre los valores: cuanto más grande el valor más alta la correlación.

La correlación más fuerte se da con el *molecular volume*, con un valor de -0.64. Esto resulta extraño, ya que, en principio, no es una propiedad que tenga relevancia a la hora del plegamiento de una proteína. Los valores obtenidos (correlación de Pearson) para *hydropathy* y *polar requirement* tienen un valor medio-bajo, en concreto, -0.50 y 0.39 respectivamente.

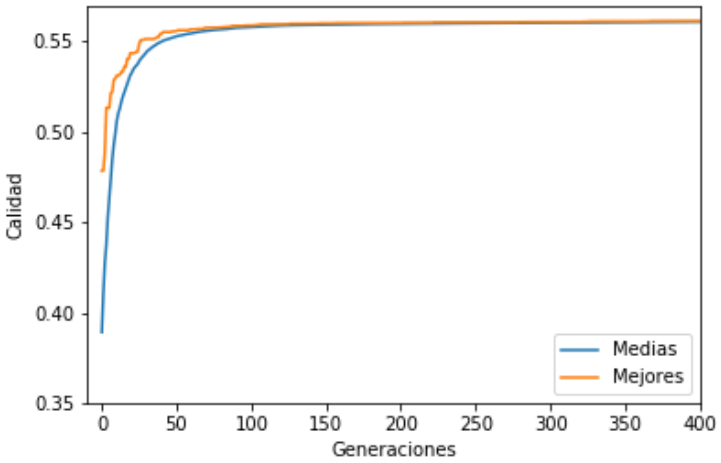

Figura 4.3: Calidad media y mejor de la población a lo largo de las generaciones, con 1 propiedad y CB513

La Figura 4.3 muestra la calidad media de los individuos de la población así como la calidad del mejor individuo en cada una de las generaciones. En este caso los resultados se obtuvieron usando el clasificador MDC con el conjunto CB513. Igual que con los resultados obtenidos con RS126, solo se muestra hasta la generación 400, ya que la calidad se estanca en ambos casos. En este caso, el mejor individuo alcanzó un porcentaje de acierto de un 56.25%, ligeramente mayor que usando RS126 para evolucionar las propiedades. La media de la calidad de los mejores individuos en las 10 ejecuciones independientes fue de un 56.19%.

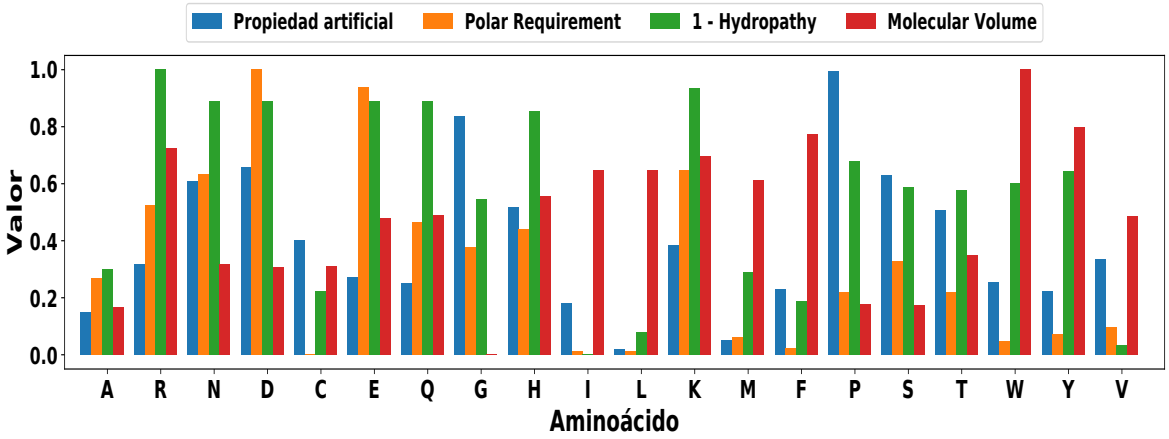

Figura 4.4: Propiedades obtenidas para cada aminoácido con MDC y CB513 en un espacio unidimensional de propiedades

La Figura 4.4 muestra una comparación entre los valores obtenidos para las propiedades artificiales y las propiedades conocidas usando el clasificador MDC con el conjunto CB513. Como en el caso anterior, se aprecian similitudes entre algunos valores de la propiedad evolucionada y el *polar requirement*. Por ejemplo, la Asparagina (N) o la Histidina (H), tienen valores medio/altos en ambas propiedades. Estas similitudes se dan también en aminoácidos de valores bajos de *polar requirement* como la Metionina (M) o la Tirosina (Y).

Las propiedades obtenidas entrenando con el conjunto CB513 y las propiedades conocidas presentan correlaciones similares a los resultados obtenidos con RS126. Los índices de correlación de Pearson son más pequeños en las propiedades que miden hidrofobicidad. Con la *hydropathy* tiene un valor de -0.41 par y con el *polar requirement* de 0.34. Al igual que entrenando con RS126, la mayor correlación con una propiedad se encuentra en el *molecular volume* con un valor de -0.66.

4.1.2 Evolución de dos propiedades por aminoácido

En esta sección se presentan los resultados obtenidos usando el algoritmo MDC y evolucionando dos propiedades por aminoácido, usando los conjuntos de datos RS126 y CB513.

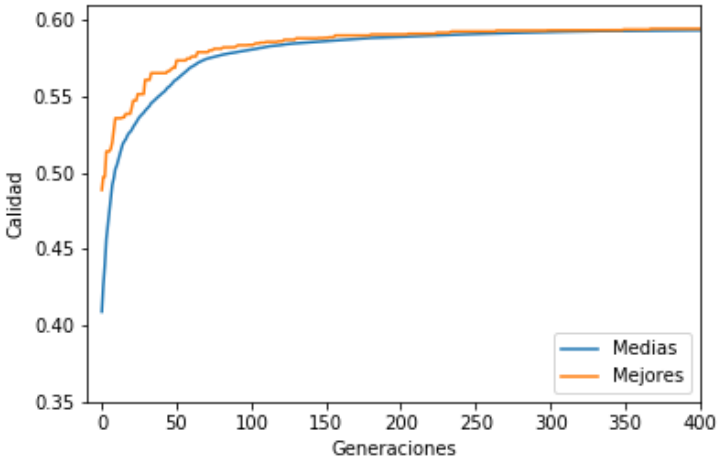

Figura 4.5: Calidad media y mejor de la población a lo largo de las generaciones, con 2 propiedades y RS126

La Figura 4.5 muestra el porcentaje de acierto medio de la población así como el mejor a lo largo de las generaciones cuando se utilizó el MDC con el conjunto RS126 y se evolucionaron 2 propiedades por aminoácido. En este caso, como pasó con una propiedad, los valores de ambas curvas convergen, aunque en este caso se necesitan más generaciones, lo cual es lógico al aumentar la dimensionalidad del problema. Sin embargo, aunque la tendencia es similar, la calidad del mejor individuo sube hasta un 59.79%. Esto supone una mejora de alrededor de un 4% con respecto a evolucionar una sola propiedad. La media de calidad de los mejores individuos de la población en este caso fue de un 59.75%.

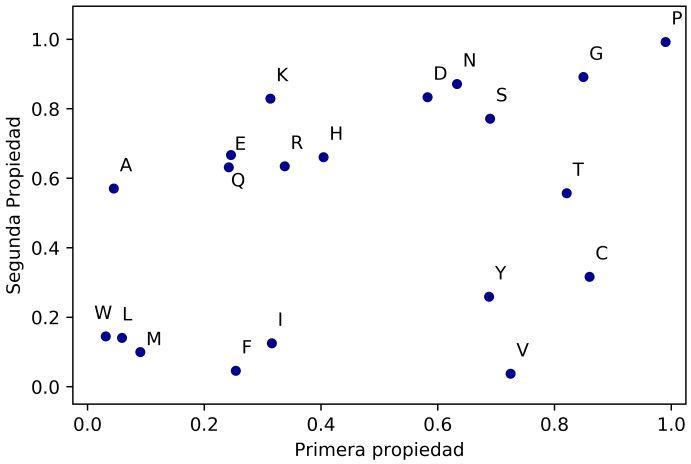

Figura 4.6: Propiedades evolucionadas en 2 dimensiones con MDC y RS126

La Figura 4.6 muestra en dos dimensiones las propiedades obtenidas usando el clasificador MDC con el conjunto RS126. En este caso, también se observan similitudes entre el *polar requirement* y las propiedades evolucionadas. En concreto, aminoácidos como la Metionina (M), la Leucina (L) o el Triptófano (W), que están en la parte inferior derecha de la figura, presentan valores bajos en ambas propiedades artificiales, que se corresponde con su *polar requirement*. Esto se cumple también, por ejemplo, con el Ácido aspártico (D), que tiene el valor más alto de *polar requirement* y un valor en su segunda propiedad también bastante alto. Aunque esto no se cumple en todos los casos, aminoácidos como la Serina (S) o la Treonina (T), tienen un *polar requirement* bajo, pero ambas propiedades artificiales son medianamente altas.

Si calculamos el coeficiente de correlación de Pearson con el segundo conjunto de propiedades obtenido, los resultados muestran una correlación de 0.75 para *hydropathy* y un 0.71 para *polar requirement*. Ambos índices muestran una correlación fuerte y podrían indicar que estas medidas de hidrofobicidad son importantes de cara a la formación de estructuras secundarias, como lo son para la formación de la estructura terciaria. En cuanto al *molecular volume*, la correlación fue algo más baja, de -0.61. Al contrario que con una propiedad, la correlación más alta es con las mediadas de hidrofobicidad y no con el *molecular volume*, esta es una tendencia que se mantiene cuando se evolucionan 2 y 3 propiedades por aminoácido.

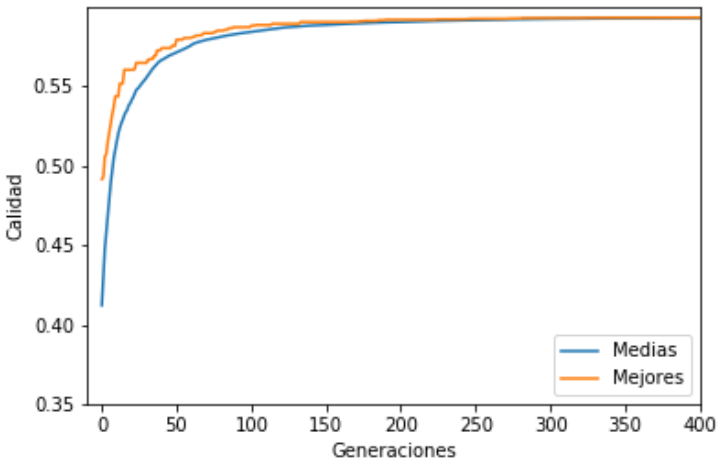

Figura 4.7: Calidad media y mejor de la población a lo largo de las generaciones, con 2 propiedades y CB513

La Figura 4.7 muestra la media de calidad de la población y del mejor individuo a lo largo de las generaciones, evolucionando dos propiedades por aminoácido y usando el clasificador MDC con el conjunto CB513. El porcentaje de acierto del mejor individuo en este caso fue de un 59.49%. Es un valor algo más pequeño, pero muy similar al obtenido entrenando con RS126. Como se puede observar, ahora evolucionando 2 propiedades y, en comparación a los resultados en la sección anterior, la calidad no se ve apenas afectada por el conjunto que se esté usando, ambos ofrecen porcentajes muy similares. La media de calidad de los mejores individuos para esta configuración fue de un 59.48%.

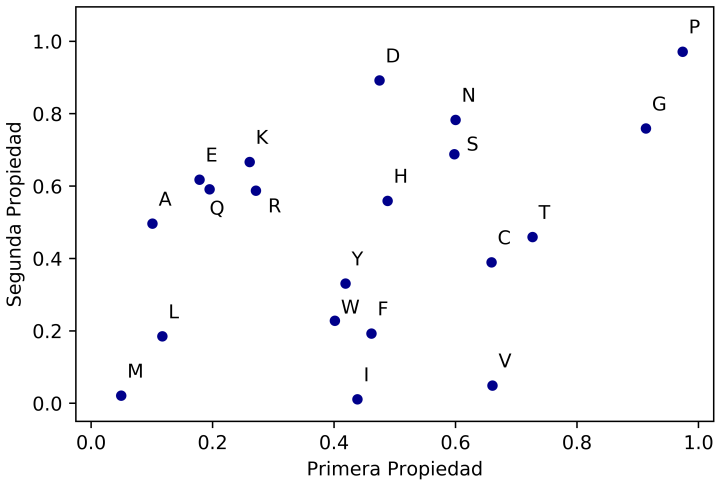

Figura 4.8: Propiedades evolucionadas en 2 dimensiones con MDC y CB513

La Figura 4.8 muestra los valores de las dos propiedades artificiales obtenidas usando el conjunto CB513 y usando el clasificador MDC. Si se observan los valores de la primera propiedad se ve que, como en casos anteriores, las propiedades artificiales toman valores similares a las propiedades conocidas. Esto es, por ejemplo, el caso de aminoácidos como la Asparagina (N) o el Ácido aspártico (D), que tienen valores próximos de primera propiedad evolucionada y *polar requirement*. Otra observación interesante es que la Prolina (P), tanto evolucionando con CB513 como con RS126, tiene valores extremos en ambas propiedades, que separan al aminoácido de la mayoría de aminoácidos de su grupo, los no polares, como muestra la clasificación de la sección 1.1.1. Además, si comparamos estos valores con los obtenidos entrenando con el conjunto RS126, se observa que varios aminoácidos toman valores muy similares y se agrupan en el espacio. Se observan en los grupos formados por los aminoácidos Ácido aspártico (D), Asparagina (N) y Serina (S), o el grupo de Ácido glutámico (E) y Glutamina (Q). Esto podría indicar las relaciones que existen entre ellos.

Calculando los índices de correlación de Pearson de las propiedades artificiales obtenidas usando el conjunto CB513, se ven valores similares a los resultados del conjunto RS126. El índice de correlación de Pearson en la segunda propiedad para la *hydropathy* es de -0.76 y para el *polar requirement* 0.69, ambos continúan siendo valores de correlación fuertes. Igual que en el caso anterior, el índice de correlación más pequeño con una propiedad física es con el *molecular volume*, con un valor de -0.59, que aun siendo el más pequeño es relativamente alto.

4.1.3 Evolución de tres propiedades por aminoácido

En esta sección se presentan los resultados obtenidos usando el algoritmo MDC y evolucionando tres propiedades por aminoácido con el algoritmo evolutivo.

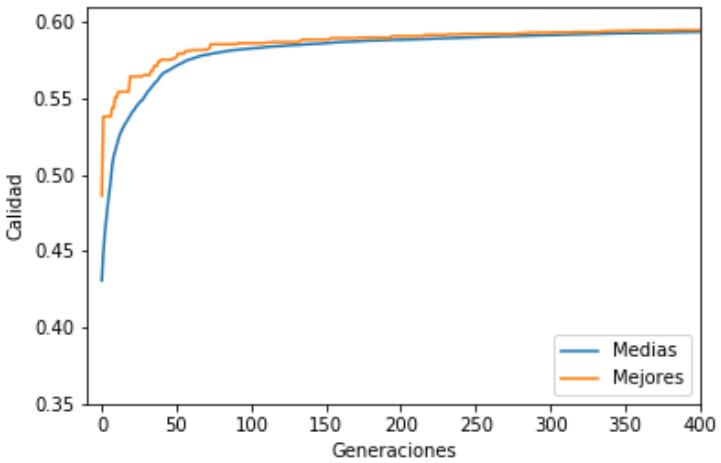

Figura 4.9: Calidad media y mejor de la población a lo largo de las generaciones, con 3 propiedades y RS126

La Figura 4.9 muestra la calidad media de la población y del mejor individuo de la población a lo largo de las generaciones cuanto se usa el clasificador MDC y el conjunto RS126 para evolucionar 3 propiedades por aminoácido. Ambas curvas siguen la misma tendencia que evolucionando 1 y 2 propiedades por aminoácido, se estanca y luego mejora muy poco hasta el final del proceso evolutivo. En este caso, el mejor individuo obtuvo una calidad de un 60.2%, ligeramente superior a los resultados obtenidos con 2 propiedades. Por otro lado, la media de acierto de los mejores individuos fue de un 60.06%. Debido a que la mejora de 2 a 3 propiedades en cuanto a porcentaje de acierto fue pequeña no se probaron a evolucionar más propiedades por aminoácido.

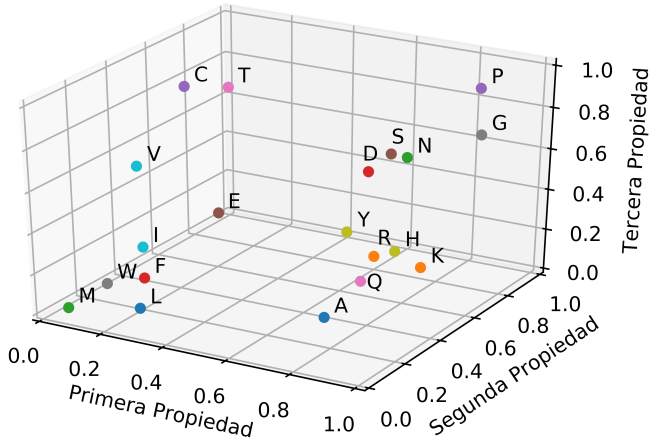

Figura 4.10: Propiedades evolucionadas en 3 dimensiones con MDC y RS126

La Figura 4.10 muestra, en un espacio tridimensional, los valores de las tres propiedades artificiales obtenidas usando el algoritmo MDC y entrenándolo con el conjunto RS126. Las observaciones son similares a las realizadas con dos propiedades. Por ejemplo, los aminoácidos Triptófano (W), Leucina (L) y Metionina (M), que estaban próximos con dos propiedades también lo están con tres propiedades. Lo mismo pasa con el grupo de aminoácidos de Ácido aspártico (D), Asparagina (N) y Serina (S), que vuelven a estar cercanos en el espacio de propiedades. Otra observación es que, como ocurrió en casos anteriores, el aminoácido P toma valores extremos en todas sus propiedades.

En cuanto al índice de correlación de Pearson, en este caso, los coeficientes fueron -0.54 para la *hydropathy*, 0.61 para el *polar requirement* y -0.52 para el *molecular volume* en la primera propiedad. Aun siendo valores más bajos que en resultados anteriores, siguen siendo mayores las correlaciones con las medidas de hidrofobicidad, en comparación con el *molecular volume*, como ocurría en el caso de evolucionar dos propiedades por aminoácido.

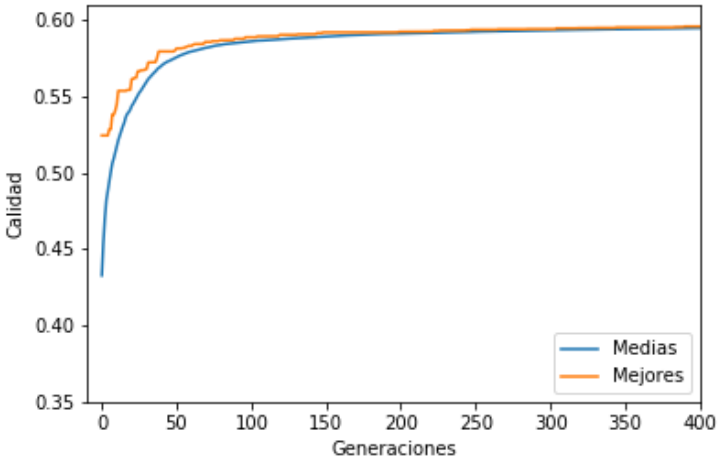

Figura 4.11: Calidad media y mejor de la población a lo largo de las generaciones, con 3 propiedades y CB513

La Figura 4.11 muestra los valores de calidad de la media de la población así como los mejores individuos a lo largo de las generaciones, usando el algoritmo MDC con el conjunto CB513. El porcentaje de acierto del mejor individuo con esta configuración fue de un 59.94%, más bajo que los resultados con el conjunto RS126. La media de los mejores individuos por lo tanto también baja, hasta un 59.78%.

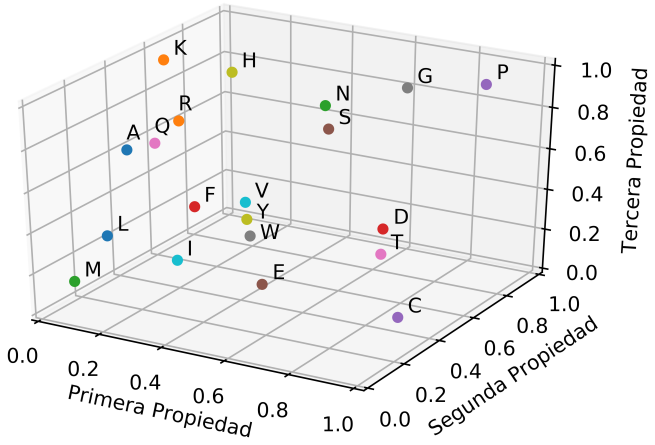

Figura 4.12: Propiedades evolucionadas en 3 dimensiones con MDC y CB513

La Figura 4.12 muestra los valores de las tres propiedades evolucionadas en el espacio de

propiedades. Se observa, por ejemplo, que los aminoácidos Asparagina (N) y Serina (S), siguen muy juntos en el espacio mientras que el Ácido aspártico (D) se ha desplazado. También se observa que aminoácidos como Metionina (M), Leucina (L) o Isoleucina (I), ocupan una región del espacio similar a las propiedades obtenidas con el conjunto RS126. Como muestran los resultados de todos los apartados, los valores obtenidos para las propiedades son muy similares entre sí.

Usando MDC con el conjunto CB513, se obtienen unos mejores valores en los índices de correlación. En concreto, -0.72 para la *hydropathy* y 0.64 para el *polar requirement* en la segunda propiedad. En este caso la correlación con el *molecular volume* bajó bastante, siendo sólo -0.29. De nuevo, con 2 y 3 tres propiedades, las medidas de hidrofobicidad están más correlacionadas con las propiedades físicas.

	RS126	CB513
1 Propiedad	55.91	56.25
2 Propiedades	59.79	59.49
3 Propiedades	60.2	59.94

Tabla 4.1: Resumen de calidad de los mejores individuos evolucionando 1, 2 y 3 propiedades con MDC

La Tabla 4.2 muestra un resumen de la calidad de los mejores individuos al evolucionar 1,2 y 3 propiedades por aminoácido usando el clasificador MDC y los conjunto RS126 y CB513. Como se puede observar los resultados son muy similares con ambos conjuntos. La mejor puntación se obtuvo evolucionando 3 propiedades por aminoácido y usando el conjunto RS126.

4.2 k-NN

En esta sección se comentarán los resultados obtenidos de evolucionar 1, 2 y 3 propiedades por aminoácido utilizando el algoritmo k-NN para determinar la función objetivo del algoritmo evolutivo. Habrá una sección para cada número de propiedades y, en cada una, los resultados de aplicar k-NN, primero con k = 15 y después con k = 3 vecinos.

4.2.1 Evolución de una propiedad por aminoácido

En esta sección se presentarán los resultados de evolucionar una propiedad por aminoácido usando el algoritmo k-NN.

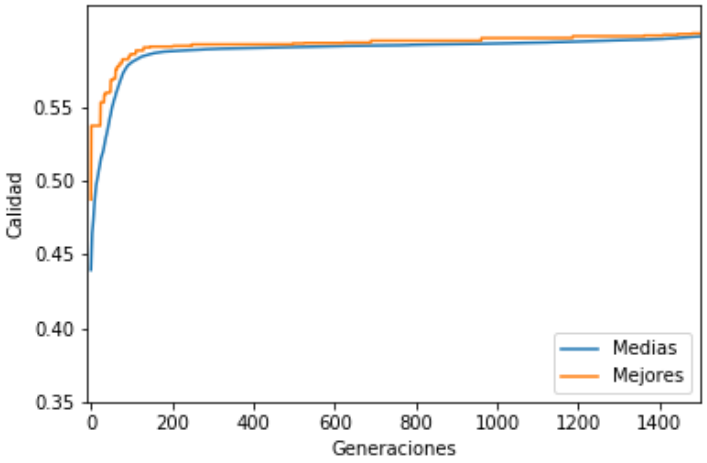

Figura 4.13: Calidad media y mejor de la población a lo largo de las generaciones, con 1 propiedad y k = 15

La Figura 4.13 muestra los valores del mejor individuo de la población y la media de calidad a lo largo de las generaciones utilizando el algoritmo k-NN con 15 vecinos. Al contrario que en el capítulo anterior usando MDC, estas figuras llegan hasta la última generación, la 1500. Esto se debe a que, cómo se puede observar, la curva para el mejor y la media no llegan a converger como ocurría usando MDC y se pueden distinguir correctamente. En este caso, el mejor individuo obtuvo una calidad de un 60%. Esto supone una mejora de alrededor de un 4% con respecto a evolucionar una sola propiedad con MDC. La media de la calidad de los mejores individuos fue de un 59.92%.

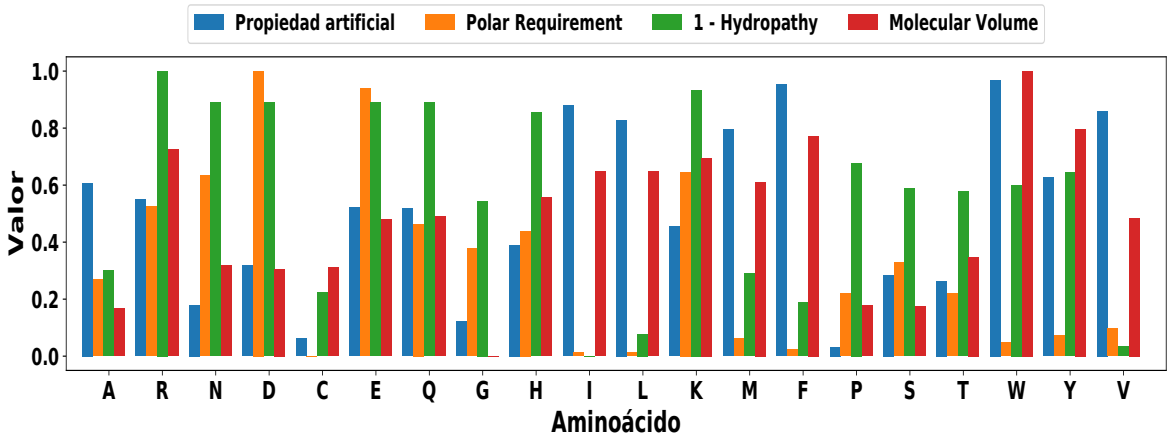

Figura 4.14: Propiedades obtenidas para cada aminoácido con k-NN y k = 15 en un espacio unidimensional de propiedades

La Figura 4.14 muestra una comparación con las propiedades artificiales obtenidas usando ED y el algoritmo k-NN con k = 15. Las observaciones que se pueden hacer son similares a las realizadas sobre los resultados obtenidos con MDC. Comparando los valores de las propiedades obtenidas, aminoácidos como Arginina (R), Serina (S), Treonina (T) o Glutamina (Q), todos muestran valores muy cercanos a sus valores de *polar requirement*. Aunque no se cumple en todos, como en los valores de Triptófano (W) o Ácido aspártico (D), que tiene el valor más alto de *polar requirement* y un valor de propiedad artificial muy distante, sí que muchos aminoácidos toman valores similares.

El coeficiente de correlación de Pearson muestra resultados similares que evolucionando una propiedad con MDC, en el sentido de que la mayor correlación se obtienen con el *molecular volume*, 0.76. Esta relación sólo se obtiene con 1 propiedad, evolucionando dos o tres siempre es mayor la correlación con las medidas de hidrofobicidad. En este caso, los índices fueron 0.47 con *hydropathy* y -0.43 con *polar requirement*.

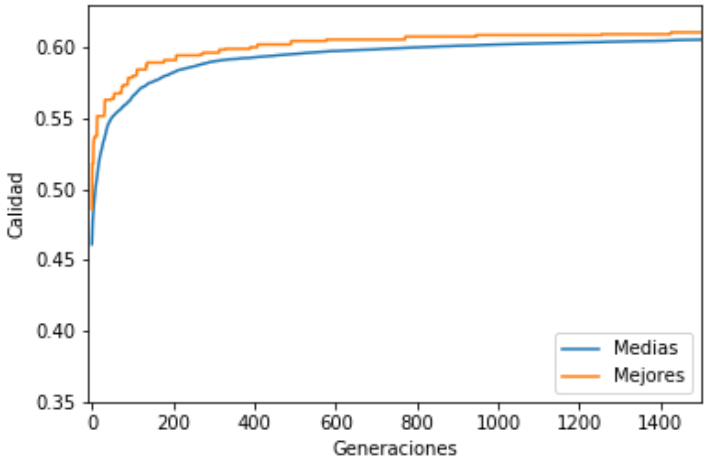

Figura 4.15: Calidad media y mejor de la población a lo largo de las generaciones, con 1 propiedad y k = 3

La calidad media de la población y del mejor individuo en cada generación obtenidos con el algoritmo k-NN con 3 vecinos y el conjunto RS126 se muestra en la Figura 4.15. El porcentaje de acierto del mejor individuo con esta configuración se sitúa en un 61%, ligeramente más alto que con 15 vecinos, pero aun así, siguen siendo valores muy cercanos. La media de la calidad de los mejores individuos en las 10 ejecuciones fue de un 60.95%. Ambos resultados de evolucionar una propiedad por aminoácido con k-NN son bastante superiores a su contraparte de MDC, mejorando el resultado en un 4-5%.

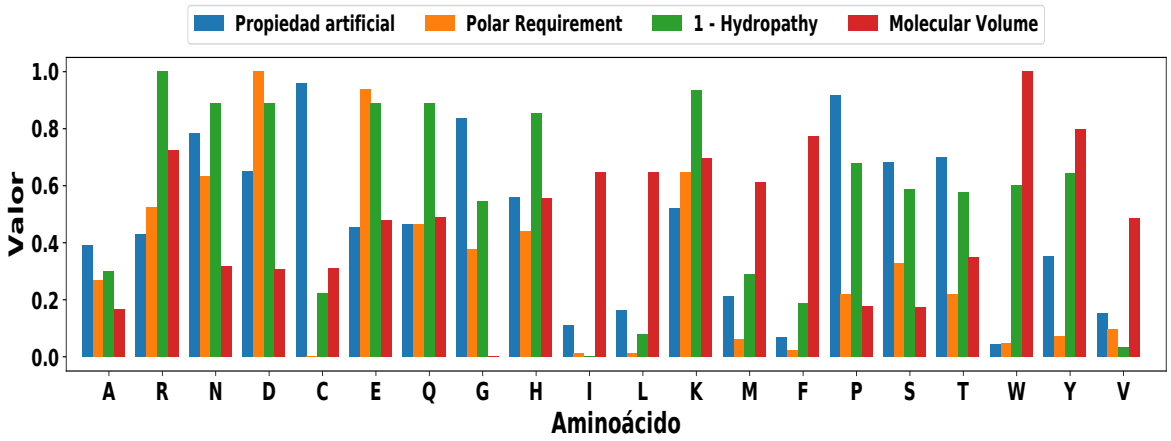

Figura 4.16: Propiedades obtenidas para cada aminoácido con k-NN y k = 3 en un espacio unidimensional de propiedades

La Figura 4.16 muestra la comparación de las propiedades obtenidas usando el clasificador k-NN con k = 3 y evolucionando una propiedad por aminoácido contra las propiedades conocidas. Como en los otros casos evolucionando una propiedad, existen coincidencias entre las propiedades evolucionadas y el *polar requirement* de los aminoácidos. Esto se observa, por ejemplo, en aminoácidos como Isoleucina (I), Leucina (L) o Fenilalanina (F), todos con valores muy bajos en ambas propiedades. Todos los resultados muestran claras similitudes en el valor de propiedades que podrían sugerir su importancia en la formación de las estructuras secundarias.

Igual que evolucionando una propiedad con 15 vecinos, la mayor correlación se corresponde con el *molecular volume*, con un valor de -0.76. *Hydropathy* y *polar requirement* tienen coeficiente de -0.45 y 0.41 respectivamente. Son valores de correlación muy similares a los obtenidos con k = 15.

4.2.2 Evolución de dos propiedades por aminoácido

En esta sección se presentarán los resultados de evolucionar dos propiedades por aminoácido usando el algoritmo k-NN.

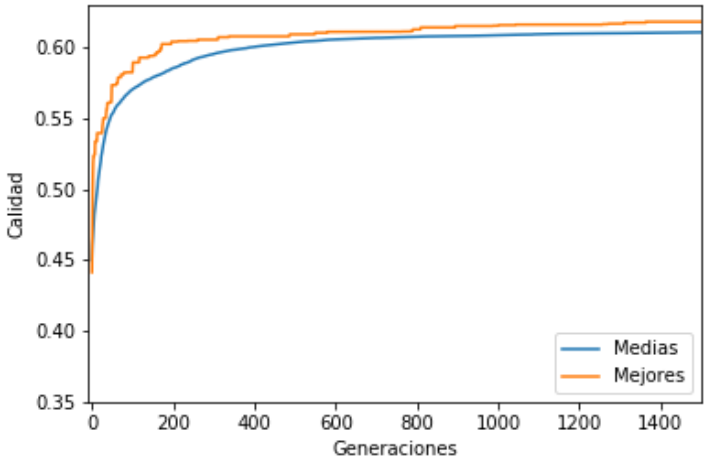

Figura 4.17: Calidad media y mejor de la población a lo largo de las generaciones, con 2 propiedades y k = 15

La Figura 4.17 muestra la calidad media de la población y del mejor individuo a lo largo de las generaciones cuando se aplica el clasificador k-NN con k = 15. Como se observa en la figura, la diferencia entre el mejor y la media es mucho más notable que en casos anteriores, tanto en k-NN como en MDC. Con k = 15, el mejor individuo obtuvo una calidad de un 61.76%. Esto supone una pequeña mejora en porcentaje de acierto con respecto a evolucionar una sola propiedad por aminoácido con k-NN. Aunque comparando con el cambio entre 1 y 2 propiedades usando el clasificador MDC, el salto es mucho más pequeño en este caso, siendo un 4% en MDC y un 1% en este. La media de porcentaje de acierto de los mejores individuos en las 10 ejecuciones fue de un 61.58%.

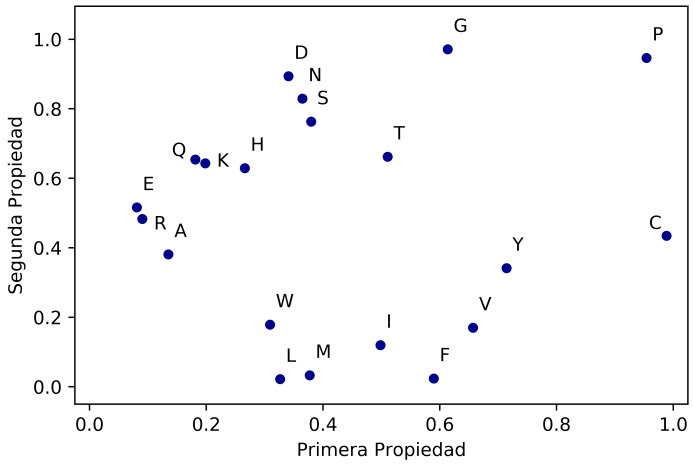

Figura 4.18: Propiedades evolucionadas en 2 dimensiones con k-NN y k = 15

En la Figura 4.18 se muestran las propiedades obtenidas con el algoritmo k-NN con k = 15 en un espacio de dos dimensiones. Como ocurre en apartados anteriores, las propiedades son similares a las ya obtenidas. Por ejemplo, en todos los casos el Ácido aspártico (D) y la Aparagina (N) están muy cerca entre sí, normalmente acompañados por la Serina (S), como es en el caso de esta figura. Otra constante es el valor alto en cuanto a propiedades evolucionadas de la Prolina (P). El aminoácido que suele estar más cerca de la Prolina (P) a pesar de sus valores extremos, es la Glicina (G), que tiene cierto sentido teniendo en cuenta que ambos pertenecen al mismo grupo de aminoácidos, los no polares. La mayoría de valores de las propiedades obtenidas son similares a los resultados de aplicar el algoritmo MDC.

El valor de los coeficientes de correlación con las tres propiedades físicas es muy similar en este caso, -0.69, 0.63 y -0.68, para *hydropathy*, *polar requirement* y *molecular volume* respectivamente. Aún que en este caso no predominan las correlaciones con medidas de hidrofobicidad, los tres índices indican correlaciones fuertes.

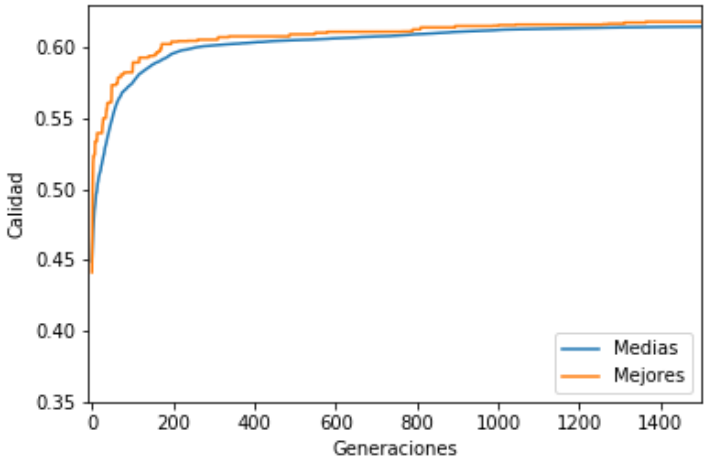

Figura 4.19: Calidad media y mejor de la población a lo largo de las generaciones, con 2 propiedades y k = 3

La Figura 4.19 muestra la calidad de los mejores individuos y la media de la población utilizando k-NN con k = 3 vecinos. En este caso, la calidad del mejor individuo coincide con el resultado de k = 15, un 61.76%. Como se observa en la figura, ambas curvas, como en todos los casos anteriores, siguen la misma tendencia. También coincide que, con respecto a evolucionar propiedades con k-NN, la diferencia entre el número de vecinos no supone una diferencia en la calidad obtenida. Con esta configuración la calidad media de los mejores individuos de todas las ejecuciones fue de un 61.6%.

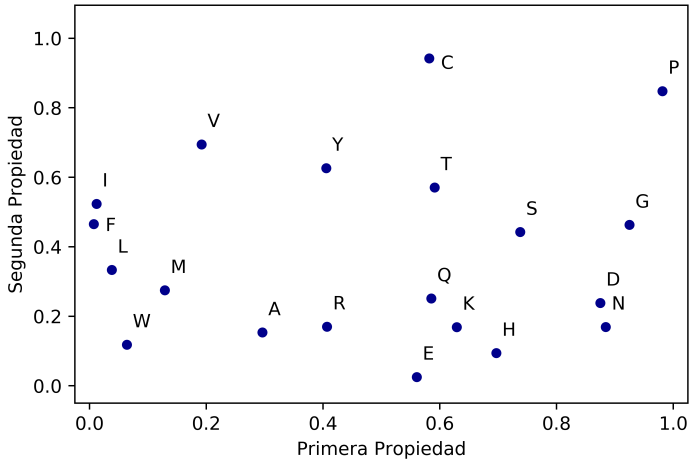

Figura 4.20: Propiedades evolucionadas en 2 dimensiones con k-NN y k = 3

La Figura 4.20 muestra los valores de las dos propiedades evolucionadas usando k-NN con k = 3. Existen muchas similitudes a los valores obtenidos con k = 15, incluyendo, valores similares en los aminoácidos D y N, y los valores extremos del aminoácido P. Los valores de las propiedades obtenidas en aminoácidos como W, L o M coinciden también con los resultados de apartados anteriores.

Igual que ocurría con k = 15, todos los valores de correlación son muy similares entre sí. En concreto, -0.66 para *hydropathy*, 0.62 para *polar requirement* y -0.69 con *molecular volume* en la primera propiedad.

4.2.3 Evolución de tres propiedades por aminoácido

En esta sección se presentan los resultados obtenidos del algoritmo evolutivo usando el algoritmo k-NN para determinar la función objetivo y evolucionando tres propiedades por aminoácido.

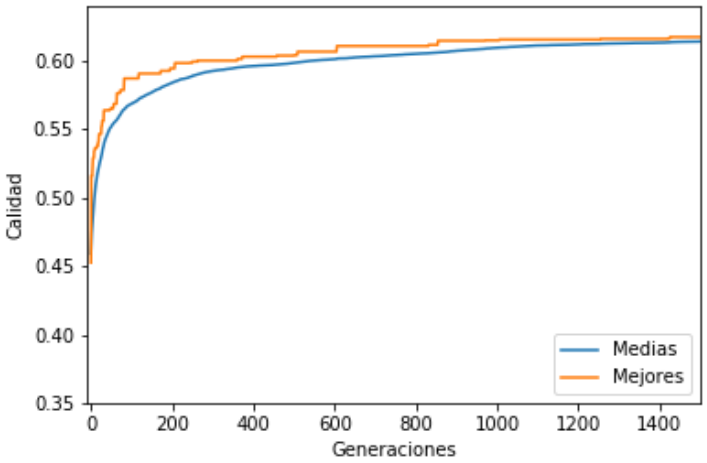

Figura 4.21: Calidad media y mejor de la población a lo largo de las generaciones, con 3 propiedades y k = 15

La Figura 4.21 muestra el porcentaje de acierto del mejor individuo y la media de la población a lo largo del proceso evolutivo. El valor del mejor porcentaje de acierto del mejor individuo de la población fue de un 61.68%, muy similar al de los resultados de evolucionar dos propiedades con k-NN. Al igual que con MDC, el paso de dos a tres propiedades no afecta prácticamente al porcentaje de acierto. La media del porcentaje de acierto de los mejores individuos en las 10 ejecuciones fue de un 61.74%.

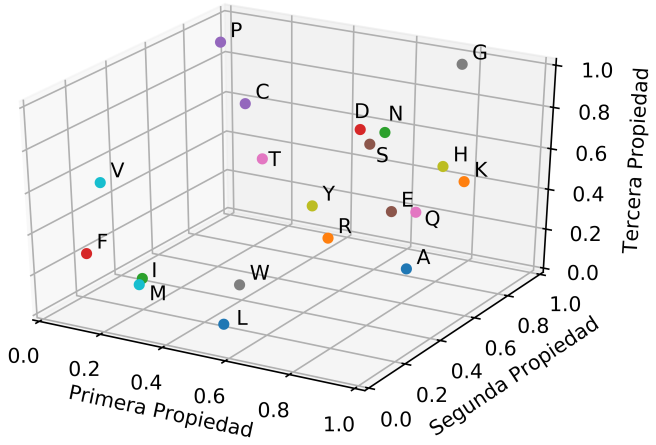

Figura 4.22: Propiedades evolucionadas en 3 dimensiones con k-NN y k = 15

En la Figura 4.22 se muestra el conjunto de tres propiedades obtenido usando k-NN con k = 15 en un espacio tridimensional. Los valores de las propiedades siguen siendo coherentes con las de apartados anteriores. Siempre se observa el grupo de Ácido aspártico (D), Asparagina (N) y Serina (S) muy juntos en el espacio, así como los aminoácidos P (Prolina) y Glicina (G) tomando valores extremos.

Con tres propiedades, los coeficientes de correlación vuelven a ser predominantes en las medidas de hidrofobicidad; con -0.75 en *hydropathy* y 0.75 en *polar requirement* en la segunda propiedad, ambas correlaciones bastante fuertes. La correlación con el *molecular volume*, aunque alta, es menor, -0.62.

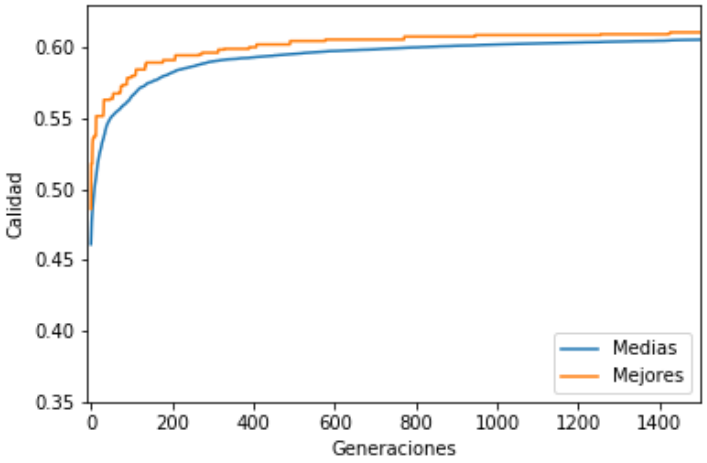

Figura 4.23: Calidad media y mejor de la población a lo largo de las generaciones, con 3 propiedades y k = 3

La Figura 4.23 muestra la calidad de la media de la población y del mejor individuo a lo largo de las 1500 generaciones con 3 vecinos y evolucionando 3 propiedades por aminoácido. La calidad del mejor individuo llega a un 61%, ligeramente menor que usando 15 vecinos. La media de la calidad de los individuos que obtuvieron una mejor puntuación fue de un 60.9%.

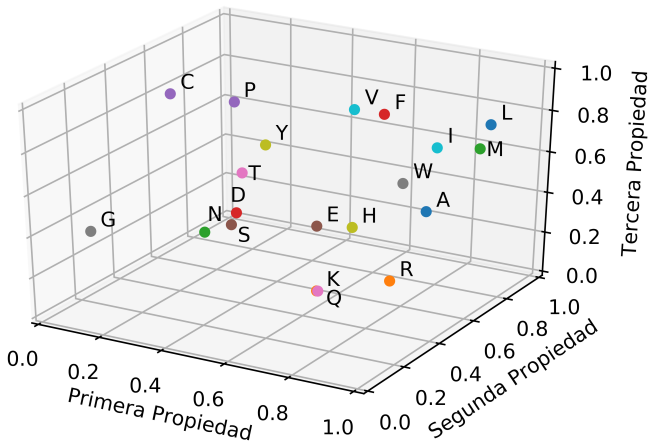

Figura 4.24: Propiedades evolucionadas en 3 dimensiones con k-NN y k = 3

En la Figura 4.24 vemos el conjunto de propiedades evolucionadas. Las observaciones que se pueden realizar son muy similares a las conseguidas con k = 15 en cuanto a corresponden-

cia entre valores de propiedades evolucionadas y *polar requirement*.

Con esta configuración, se obtuvo el conjunto de propiedades que presenta una mayor correlación con las medidas de hidrofobicidad. Teniendo un valor de 0.84 con *hydropathy* y -0.85 con *polar requirment*, valores que rozan el valor más fuerte de correlación, con un valor muy bajo de correlación con *molecular volume* de 0.26.

Todos los resultados obtenidos con dos o tres propiedades utilizando ambos algoritmos muestran que la mayor correlación existen entre una de las propiedades evolucionadas y la dos medidas de hidrofobicidad. Esto podría ser indicativo de la importancia de la hidrofobicidad también de cara a formar las distintas estructuras secundarias.

	K = 15	K = 3
1 Propiedad	60	60.95
2 Propiedades	61.76	61.76
3 Propiedades	61.68	61

Tabla 4.2: Resumen de calidad de los mejores individuos evolucionando 1,2 y 3 propiedades con MDC

La Tabla 4.2 muestra un resumen de la calidad de los mejores individuos al evolucionar 1,2 y 3 propiedades por aminoácido usando el clasificador k-NN con k= 15 y k = 3 vecinos. Como se puede ver el cambiar el número de vecinos no impacta de forma significativa el rendimiento. La mejor calidad se obtuvo con 2 propiedades tanto con 15 como 3 vecinos con un 61.76% de acierto.

4.3 Utilización de las propiedades evolucionadas con un clasificador neuronal

En esta sección se compararán los resultados, en cuanto a Q3, entre los conjuntos de propiedades artificiales detallados en el apartado anterior y la codificación neutra de los aminoácidos. También se incluirán algunas matrices de confusión, con el objetivo de comparar los porcentajes de acierto de cada estructura secundaria por separado en ambas codificaciones. En concreto, primero en cada apartado se mostrará el Q3 medio obtenido en el proceso de validación mencionado en el apartado 2.6.3 (una vez entrenada la red), tanto de la codificación neutra como de la artificial. Después, se expondrán el Q3 y los porcentajes individuales obtenidos en el conjunto de test y se compararán con los de la codificación neutra. Las redes de neuronas artificiales fueron entrenadas con los conjuntos RS126 y CB513.

4.3.1 1 propiedad por aminoácido

En esta sección se expondrán los resultados obtenidos cuando la RNA utiliza la codificación neutra en los aminoácidos y cuando utiliza una propiedad artificial evolucionada por aminoácido.

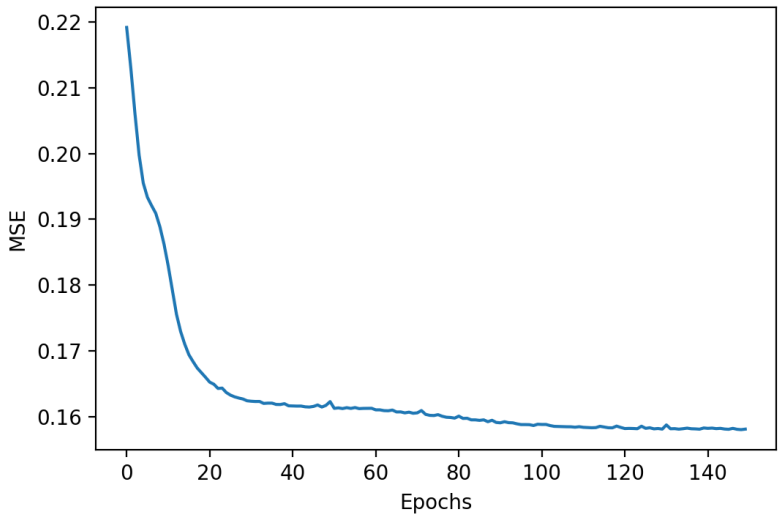

Figura 4.25: Evolución del MSE a lo largo de las iteraciones de entrenamiento (*epochs*)

La Figura 4.25 muestra un ejemplo de cómo evoluciona el MSE, la función de pérdida de la red de neuronas artificiales, frente a los *epochs*, los ciclos de entrenamiento de la red, en el proceso de validación. Como se mencionó en el apartado 2.6.3, se usó el MSE para determinar el número de *epochs*, en este caso, 150. Como se puede observar, el error baja rápidamente en los primeros ciclos, y sigue bajando hasta el final de la figura. A partir de ahí, el valor del error oscila sobre los mismos valores, sin bajar más.

	Neutra	MDC RS126	MDC CB513	k-NN K = 15	k-NN K = 3
RS126	60.08	58.95	58.31	59.42	58.60
CB513	65.51	58.91	59.03	59.93	59.37

Tabla 4.3: Comparación de Q3 entre codificación neutra y artificial con 1 propiedad

La Tabla 4.3 muestra los resultados de validación al entrenar una red de neuronas con ambos conjuntos de datos, usando las distintas codificaciones y una propiedad por aminoácido. En concreto, las filas son los conjuntos que se han usado para entrenar la red de neuronas artificiales y las columnas, la codificación neutra y los conjuntos de propiedades obtenidos con el algoritmo evolutivo y cada uno de los métodos de clasificación. Por ejemplo, donde se

cruza RS126 y MDC 513, es el Q3 de validación obtenido de entrenar la red de neuronas artificiales con el conjunto RS126 y codificando los aminoácidos con las propiedades obtenidas usando el clasificador MDC y el conjunto CB513. Como se puede observar todas las propiedades artificiales consiguen un Q3 muy similar, y están algo alejados de los resultados de la codificación neutra. Entrenando con el conjunto RS126, el mejor resultado de propiedades artificiales lo obtiene el conjunto obtenido usando el algoritmo k-NN con k = 15, con una media de un 59.42% de acierto frente a un 60.08% de la codificación neutra. Por otro lado, al entrenar con CB513, el mejor resultado de propiedades evolucionadas se vuelve a obtener con k-NN y k = 15, con un 59.93% de acierto. Sin embargo, se encuentra mucho más lejos del resultado con codificación neutra, con una media de porcentaje de acierto en validación de un 65.54%, alrededor de un 6% mayor.

Estructura	Hélice	Lámina	*Coil*
Hélice	927	143	427
Lámina	207	378	311
Coil	454	225	1314

(a) Codificación Neutra

Estructura	Hélice	Lámina	*Coil*
Hélice	821	155	521
Lámina	243	280	373
Coil	354	148	1491

(b) Codificación artificial

Tabla 4.4: Matrices de confusión con codificación neutra y 1 propiedad artificial y usando el conjunto RS126

La Tabla 4.4 contiene las matrices de confusión resultado de realizar la clasificación de estructuras secundarias usando una RNA entrenada con el conjunto RS126 en el conjunto de test. Por un lado, están los resultados obtenidos con la codificación neutra y, por otro, los obtenidos con el conjunto de propiedades, en este caso obtenidas con el algoritmo k-NN y 15 vecinos. En este caso, la codificación neutra consigue un Q3 de 59.71% mientras que la codificación artificial obtiene un 59.09%, algo inferior, pero ambos muy similares a sus puntuaciones de validación. En cuanto al porcentaje de estructuras individuales, la codificación neutra obtiene 61.92, 42.19 y 65.93 por ciento respectivamente para hélices, láminas y *coils*. Por otro lado, la codificación artificial consigue 54.84, 31.25 y 74.81 por ciento para las mismas estructuras. Los resultados muestran que ambas codificaciones son malas haciendo predicciones de láminas, mientras que tienen porcentajes próximos en las otras dos estructuras.

Estructura	Hélice	Lámina	*Coil*
Hélice	6816	925	2930
Lámina	982	3258	2501
Coil	1731	1305	9516

(a) Codificación neutra

Estructura	Hélice	Lámina	*Coil*
Hélice	6858	857	2956
Lámina	1878	2274	2589
Coil	2633	1017	8902

(b) Codificación artificial

Tabla 4.5: Matrices de confusión con codificación neutra y 1 propiedad artificial y usando el conjunto CB513

La Tabla 4.5 es una comparación de las matrices de confusión obtenidas con la codificación neutra y las propiedades evolucionadas con el clasificador k-NN y k = 15, cuando la red de neuronas se entrenó con el conjunto CB513. En este caso, se obtuvo un Q3 de 65.38 y 60.18 por ciento para codificación neutra y artificial respectivamente. En cuanto a los porcentajes de estructuras individuales, para hélices, se predijo correctamente con la codificación neutra un 63.87% del total y un 64.26% con las propiedades artificiales. Para láminas, un 48.33% con la codificación neutra, mientras que, para la artificial se desploma a un 33.73%. Por último, un 75.81% de *coils* se clasificaron correctamente usando la codificación binaria. Esta puntuación es mejor con las propiedades artificiales, consiguiendo un 70.92% de acierto. Por lo tanto, el porcentaje de hélices y *coils* es relativamente similar en ambas codificaciones, sin embargo, en cuanto a láminas la codificación neutra es muy superior a la artificial.

4.3.2 2 propiedades por aminoácido

En esta sección se detallan los resultados de comparar las propiedades artificiales obtenidas con el algoritmo evolutivo, ED, al evolucionar dos propiedades por aminoácido y la codificación binaria.

	Neutra	MDC RS126	MDC CB513	k-NN K = 15	k-NN K = 3
RS126	60.08	61.54	61.87	61.13	60.72
CB513	65.51	62.65	63.32	62.82	62.24

Tabla 4.6: Comparación de Q3 entre codificación neutra y artificial con 2 propiedades

La Tabla 4.6 muestra la media de porcentajes de acierto durante el periodo de validación, entrenando la red con los dos conjuntos de datos y las distintas codificaciones para los aminoácidos. Con respecto al uso de una propiedad, ambos porcentajes de propiedades artificiales suben: alrededor de un 2% en el entrenamiento con RS126 y un 3/4% con CB513. En este caso, la red de neuronas artificiales consiguió su mejor resultado entrenándola con el conjunto de datos CB513, y usando las propiedades obtenidas entrenando el algoritmo MDC con CB513.

Estas mejoras de porcentaje de acierto, en el caso de RS126 hace que superen la calidad obtenida con la codificación neutra y con CB513 lo acerca a sólo un 2% en el mejor de los casos.

Estructura	Hélice	Lámina	*Coil*
Hélice	927	143	427
Lámina	207	378	311
Coil	454	225	1314

(a) Codificación neutra

Estructura	Hélice	Lámina	*Coil*
Hélice	896	160	441
Lámina	212	372	312
Coil	365	180	1448

(b) Codificación artificial

Tabla 4.7: Matrices de confusión con codificación neutra y 2 propiedades artificiales RS126

La Tabla 4.7 muestra las matrices de confusión obtenidas en la predicción en el conjunto de test, usando, primero la codificación neutra y luego la artificial, obtenida con el algoritmo MDC y el conjunto RS126. El Q3 en concreto es de un 61.34% y un 61.92% para la codificación binaria y la artificial respectivamente. Como se dijo en el párrafo anterior, con 2 propiedades se mejora el resultado obtenido con la codificación neutra. En cuanto a porcentajes de predicción de las estructuras individuales, la codificación artificial predice correctamente un 59.85% de hélices, un 41.52% de láminas y un 72.62%. Esto supone una mejora general respecto a los porcentajes obtenidos con una propiedad.

Estructura	Hélice	Lámina	*Coil*
Hélice	6816	925	2930
Lámina	982	3258	2501
Coil	1731	1305	9516

(a) Codificación neutra

Estructura	Hélice	Lámina	*Coil*
Hélice	6739	1106	2826
Lámina	1136	3158	2447
Coil	2049	1308	9195

(b) Codificación artificial

Tabla 4.8: Matrices de confusión con codificación neutra y 2 propiedades artificiales CB513

La tabla 4.8 contiene las matrices de confusión de la predicción del conjunto de test mediante codificación neutra y las propiedades artificiales, obtenidas usando el conjunto CB513 con el clasificador MDC. En este caso, el Q3 de las propiedades artificiles es de un 63.30%, acercándose al porcentaje de la codificación neutra. En cuanto a los porcentajes individuales, la codificación artificial consigue un 59.62% de hélices clasificadas correctamente, un 46.69% de láminas y un 75.36% de *coils*. Con respecto al uso de una propiedad con CB513, el porcentaje de hélices clasificadas correctamente cae, pero el resto suben. Comparando con los datos entrenando con RS126, CB513 mejora en todas las estructuras individualmente menos en hélices.

4.3.3 3 propiedades por aminoácido

En esta sección se compararán los resultados obtenidos entre entrenar una RNA con la codificación neutra y codificar los aminoácidos mediante tres propiedades artificiales.

	Neutra	MDC RS126	MDC CB513	k-NN K = 15	k-NN K = 3
RS126	60.08	61.89	61.35	61.67	60.91
CB513	65.51	63.17	63.58	62.80	62.18

Tabla 4.9: Comparación de Q3 entre codificación neutra y artificial con 3 propiedades

La Tabla 4.9 muestra la media de Q3 en el proceso de validación en las pruebas realizadas con 3 propiedades por aminoácido y la codificación neutra. Como se puede observar, el porcentaje de acierto entrenando con RS126 sube ligeramente en el mejor caso, y sigue siendo mejor que la codificación neutra. En el caso de entrenar con CB513, los valores son mayores, pero aún así, similares a los obtenidos con 2 propiedades, aunque, igual que con RS126, el mejor Q3 sube ligeramente.

Estructura	Hélice	Lámina	*Coil*
Hélice	927	143	427
Lámina	207	378	311
Coil	454	225	1314

(a) Codificación neutra

Estructura	Hélice	Lámina	*Coil*
Hélice	794	140	563
Lámina	165	369	362
Coil	282	161	1550

(b) Codificación artificial

Tabla 4.10: Matrices de confusión con codificación neutra y 3 propiedades artificiales RS126

La tabla 4.10 contiene las matrices de confusión resultado de la predicción entrenando la RNA con el conjunto RS126 y las propiedades conseguidas usando MDC con RS126. Con esta configuración, los porcentajes individuales para las estructuras secundarias son de un 53.04% para hélices, un 41.18% para láminas y un 77.77% para *coils*. Este caso lo que mantiene un Q3 similar es el hecho de que subió el porcentaje de *coils* predichas correctamente, mientras que el porcentaje de las otras dos bajó con respecto a codificar los aminoácidos con 2 propiedades.

Estructura	Hélice	Lámina	*Coil*
Hélice	6816	925	2930
Lámina	982	3258	2501
Coil	1731	1305	9516

(a) Codificación neutra

Estructura	Hélice	Lámina	*Coil*
Hélice	7391	919	2361
Lámina	1631	2949	2161
Coil	2661	1367	8524

(b) Codificación artificial

Tabla 4.11: Matrices de confusión con codificación neutra y 3 propiedades artificiales CB513

La tabla 4.11 contiene las matrices de confusión de la predicción en el conjunto de test al entrenar la red de neuronas con el conjunto CB513. Las propiedades artificiales usadas para codificar los aminoácidos, en este caso, fueron obtenidas usando el clasificador MDC con el conjunto CB513. El Q3 es de 65.37 y de 62.96 por ciento, para codificación neutra y artificial respectivamente. Aunque el Q3 de la codificación neutra es ligeramente superior, son valores próximos. En cuanto a porcentajes de estructuras secundarias individuales, la codificación artificial consigue un 69.26% de acierto en hélices, un 43.74% en láminas y un 67.91% en *coils*. Todos estos son valores muy similares a los mencionados anteriormente, tanto para codificación neutra como para artificial. Todos los resultados muestran que la estructura con más porcentaje de acierto son los *coils*, seguidas de hélices y, por último, láminas.

Capítulo 5

Conclusiones

El objetivo de este trabajo era obtener un conjunto de propiedades artificiales para codificar los aminoácidos de cara a la predicción de estructuras secundarias de proteínas. El objetivo de estas propiedades artificiales era que intentasen plasmar automáticamente la relación que existen entre aminoácidos, que puede ser muy importante de cara a la conformación de la estructura de una proteína, frente a la codificación neutra que no representa estas relaciones. Además, las propiedades artificiales supondrían una reducción en dimensionalidad con respecto a los clasificadores actuales que usan la codificación neutra (sobre las que se añade información de alineamiento múltiple de secuencia). Parte de los resultados del proyecto están descritos en el artículo publicado en unos de los congresos de referencia en computación evolutiva (IEEE-CEC 2021, IEEE Congress on Evolutionary Computation) [23].

Las propiedades artificiales se obtuvieron combinando el algoritmo evolutivo, ED, con los clasificadores MDC y k-NN para establecer la función objetivo. Las propiedades obtenidas se compararon con tres propiedades conocidas de los aminoácidos: *hydropathy*, *polar requirement* y *molecular volume*. Los resultados mostraron que los conjuntos de propiedades obtenidos siempre tendían a tener valores similares para cada aminoácido, independientemente del benchmark usado (RS126, CB513) y de la propia estocasticidad de la ejecuciones del algoritmo evolutivo. Además de esto, al calcular el índice de correlación entre las propiedades artificiales y las propiedades conocidas, se vio que cuando se evolucionaban 2 y 3 propiedades por aminoácido, la correlación no sólo era mayor en las dos medidas de hidrofobicidad, *hydropathy* y *polar requirement*, sino que además su valor era bastante alto. Con esto se puede sacar en conclusión que estas son propiedades importantes de cara a la formación de estructuras secundarias como lo son a la formación de la estructura terciaria.

Una vez obtenidos los conjuntos de propiedades artificiales, estos se probaron con una RNA, uno de los métodos más extendidos y robustos para la predicción de estructuras secundarias.

Se usó también una RNA para comprobar los resultados de la codificación neutra y compararlos con las propiedades resultado del proceso evolutivo. Los resultados mostraron que, evolucionando 2 y 3 propiedades por aminoácido, los valores obtenidos con las propiedades artificiales y la codificación neutra eran muy similares, siendo 3 propiedades por aminoácido la configuración que se acercó más. Ambas configuraciones consiguieron un porcentaje de acierto de algo más de un 63%, aunque no es un valor muy alto, sí satisface uno de los propósitos del trabajo. Se puede concluir que, debido a que ambas codificaciones obtienen un porcentaje muy similar, las propiedades artificiales se pueden usar en vez de la codificación neutra y que sirvan para reducir la dimensionalidad de la entrada de los clasificadores de estructuras secundarias.

Por último, con estos resultados, el trabajo futuro se podría enfocar a varios frentes. Primero, se podrían intentar conseguir nuevos conjuntos de propiedades, ya sea mediante otro algoritmo evolutivo, nuevos clasificadores u otros conjuntos de proteínas que usar como referencia. De la misma forma, una vez se tienen las propiedades artificiales se podrían usar otros clasificadores como SVM para ver qué resultados ofrecen. Obviamente, también se podría probar el uso de estas nuevas propiedades en métodos de predicción actuales que utilicen la codificación neutra, para ver si ofrecen mejores resultados. También usar la información adicional que da el MSA mencionado en la sección 1.3, ya que claramente es el método que más ayuda a mejorar el acierto de los métodos de predicción de estructura secundaria de proteínas.

Bibliografía

[1] "Clasificación de los aminoácidos segun su radical." [En línea]. Disponible en: http://gmoid.blogspot.com/p/chemistry-of-amino-acids.html

[2] D. A. Benson, K. Clark, I. Karsch-Mizrachi, D. J. Lipman, J. Ostell, and E. W. Sayers, "GenBank," *Nucleic Acids Research*, vol. 43, no. D1, pp. D30–D35, 11 2014.

[3] "Protein Data Bank." [En línea]. Disponible en: https://www.rcsb.org/

[4] C. B. Anfinsen, "Principles that govern the folding of protein chains," *Science*, vol. 181, no. 4096, pp. 223–230, Jul 1973.

[5] J. C. Kendrew, G. Bodo, H. M. Dintzis, R. G. Parrish, H. Wyckoff, and D. C. Phillips, "A three-dimensional model of the myoglobin molecule obtained by x-ray analysis," *Nature*, vol. 181, no. 4610, pp. 662–666, 1958.

[6] L. Pauling, R. B. Corey, and H. R. Branson, "The structure of proteins: Two hydrogen-bonded helical configurations of the polypeptide chain," *Proceedings of the National Academy of Sciences*, vol. 37, no. 4, pp. 205–211, 1951.

[7] P. Y. Chou and G. D. Fasman, "Prediction of protein conformation," *Biochemistry*, vol. 13, no. 2, pp. 222–245, Jan. 1974.

[8] B. Rost and C. Sander, "Prediction of protein secondary structure at better than 70% accuracy," *Journal of Molecular Biology*, vol. 232, no. 2, pp. 584–599, Jul 1993.

[9] D. T. Jones, "Protein secondary structure prediction based on position-specific scoring matrices," *Journal of Molecular Biology*, vol. 292, no. 2, pp. 195–202, Sep 1999.

[10] "PSI-BLAST." [En línea]. Disponible en: https://blast.ncbi.nlm.nih.gov/

[11] W. Kabsch and C. Sander, "Dictionary of protein secondary structure: Pattern recognition of hydrogen-bonded and geometrical features," *Biopolymers*, vol. 22, no. 12, pp. 2577–2637, 1983.

[12] L. L. de Oliveira, P. S. de Oliveira, and R. Tinós, "A multiobjective approach to the genetic code adaptability problem," *BMC Bioinformatics*, vol. 16, no. 1, Feb 2015.

[13] J. Santos and A. Monteagudo, "Inclusion of the fitness sharing technique in an evolutionary algorithm to analyze the fitness landscape of the genetic code adaptability," *BMC Bioinformatics*, vol. 18:195, Mar 2017.

[14] C. R. Woese, D. H. Dugre, S. A. Dugre, M. Kondo, and W. C. Saxinger, "On the fundamental nature and evolution of the genetic code," *Cold Spring Harb Symp Quant Biol*, vol. 31, pp. 723–36, 1966.

[15] R. Grantham, "Amino acid difference formula to help explain protein evolution," *Science*, vol. 185, no. 4154, pp. 862–864, 1974.

[16] R. Storn and K. Price, *Journal of Global Optimization*, vol. 11, no. 4, pp. 241–359, 1997.

[17] K. Price, R. M. Storn, and J. A. Lampinen, *Differential Evolution: An Approach to Global Optimization*, 1st ed. Springer, 2005.

[18] T. Cover and P. Hart, "Nearest neighbor pattern classification," *IEEE Transactions on Information Theory*, vol. 13, no. 1, pp. 21–27, 1967.

[19] J. A. Cuff and G. J. Barton, "Evaluation and improvement of multiple sequence methods for protein secondary structure prediction," *Proteins: Structure, Function, and Genetics*, vol. 34, no. 4, pp. 508–519, Mar 1999.

[20] W. S. Mcculloch and W. Pitts, "A logical calculus of the ideas immanent in nervous activity," *The Bulletin of Mathematical Biophysics*, vol. 5, no. 4, pp. 115–133, 1943.

[21] F. Chollet *et al.*, "Keras," https://keras.io, 2015.

[22] K. Dierik P and J. Ba, "Adam: A method for stochastic optimization," in *3rd International Coference for Learning Representations*. ICLR, 2015.

[23] J. Santos and H. Rivas, "Evolution of amino acid properties in the context of protein secondary structure prediction," in *2021 IEEE Congress on Evolutionary Computation (IEEE-CEC)*. IEEE, Jun 2021, pp. 426–433.