Neuroevolución: entrenamiento con metaheurísticas

Neuroevolution: Training with Metaheuristics

Autor
Cerezo, Elías
Máster ISIA
Universidad de Málaga
Málaga, España
elias2603@uma.es

Director
Alba, Enrique
Universidad de Málaga
Dpto. de Lenguajes y Ciencias de la Computación
Málaga, España
eat@lcc.uma.es

***Resumen*— En este trabajo se realiza un estudio del concepto de neuroevolución en el entrenamiento de redes neuronales artificiales usando aprendizaje supervisado. Algoritmos genéticos, estrategias evolutivas y recocido simulado han sido las metaheurísticas seleccionadas para realizar implementaciones de redes neuronales artificiales evolucionadas. También se realiza un análisis estadístico detallado sobre las métricas proporcionadas por estas redes neuronales artificiales evolucionadas comparándolas con las métricas proporcionadas tanto por una red neuronal artificial entrenada usando propagación hacia atrás como por una red neuronal artificial entrenada mediante búsqueda aleatoria. Los datos para las pruebas han sido seleccionados del repositorio UCI de aprendizaje automático. Los resultados muestran que los algoritmos genéticos y las estrategias evolutivas mejoran notablemente el enfoque clásico de la propagación hacia atrás como método de entrenamiento.**

***Abstract*— In this document we study the concept of neuroevolution for artificial neural network training from the supervised learning point of view. Genetic algorithms, evolutionary strategies and simulated annealing are the selected metaheuristics in order to perform artificial neural network implementations evolved by them. A detailed statistical analysis over the metrics provided by these evolved artificial neural networks is carried out as well as a comparison between the metrics provided by backpropagation and random search as training methods and the evolved artificial neural networks outcomes in this matter. The testbed has been chosen from the UCI machine learning repository. The results show that genetic algoritms and evolutionary strategies improve noticeably the clasical backpropagation approach for neural network training.**

***Keywords*—neuroevolution; metaheuristics; neural networks; supervised learning.**

I. Introducción

En este trabajo fin de máster se pretende investigar en un campo muy interesante como lo es la inteligencia artificial. El tema propuesto es la *neuroevolución* a nivel de *entrenamiento* [Floreano et al., 2008] [Alba et al., 1993] y la profundidad que se le desea conferir es la de un trabajo fin de máster.
Hay muchos campos interesantes dentro de la inteligencia artificial, entre los cuales se encuentran: *procesamiento de datos de forma inteligente* [Hernández and Stolfo, 1998] [Khayyat et al., 2015], *detección de errores* [Kemeny and Snell, 1976] [West and Bhattacharya, 2016], y las *redes neuronales artificiales*. [Haykin, 1994] [Miikkulainen et al., 2019].

En este trabajo, nos vamos a centrar en el último de ellos, las redes neuronales artificiales (RNAs) que han ganado mucho protagonismo en los últimos años y se han convertido en herramientas muy potentes para una gran cantidad de aplicaciones, entre ellas: *visión por computador* [Wang et al., 2017a] [Bojarski et al., 2016], *procesamiento del lenguaje natural* [Goldberg, 2017] [Hirschberg and Manning, 2015], *ayuda a diagnóstico médico* [Amato et al., 2013] [Jiang et al., 2010] y *detección de errores y fraude* [Fu et al., 2016] [Wang et al., 2017b]. Este campo, en el año 2019, tenía un valor de 8.3 billones de dólares [Markets, 2020] y se prevé un aumento en años venideros, superando la cifra de 50 billones de dólares en 2025. Se debe tener en cuenta que este es un estudio internacional y que estas cifras desde el punto de vista del mercado español serán inferiores por un abundante margen.

Las RNAs se componen de una serie de capas compuestas a su vez por neuronas artificiales conectadas entre ellas por enlaces. Cada enlace lleva asociado una ponderación llamada *peso* que ajusta los datos de entrada. Las RNAs se componen de una *capa de entrada*, una o varias *capas ocultas* de neuronas y una *capa de salida*. La información fluye desde la entrada a la salida pasando por las capas ocultas. El error de una RNA depende de muchos factores. En la mayoría de los casos depende de la estructura de la RNA y del uso que se le vaya a dar. Una posible función de error de una RNA es la siguiente:

$$1/n \sum_{i=0}^{n} (y - \hat{y})^2 \tag{1}$$

1. $\hat{y}$: Es el resultado aportado por la RNA al pasarle unos datos de entrada.
2. y: Es el resultado esperado para los datos de entrada.

Esta función no sirve para todos los casos ni para todos los supuestos. La función de error nos indica que tan lejos estamos de nuestro objetivo final, y normalmente es una función que debe ser minimizada[1].

Esta función no sirve para todos los casos ni para todos los supuestos. La función de error nos indica que tan lejos estamos de nuestro objetivo final, y normalmente es una función que debe ser minimizada[2].

En inteligencia artificial, se conoce como *aprendizaje supervisado* a el grupo de técnicas y problemas en los que, para cada valor o valores de entrada, se conoce el valor o valores de salida [Conneau et al., 2017]. A su vez existe el *aprendizaje no supervisado* en el cual este conocimiento no se tiene y se trabaja sin él [Srivastava et al., 2015]. La función de error definida en la ecuación 1, está indicada para problemas de aprendizaje supervisado.

Los problemas a los que nos enfrentamos en este trabajo son: por un lado, el *estancamiento en mínimos locales* [Gori and Tesi, 1992] por parte de las RNAs entrenadas con *propagación hacia atrás* [Rumelhart et al., 1986]. Por otro lado, nos enfrentamos a la *falta de escalabilidad* de las RNAs entrenadas con esta técnica [Tesauro, 1987], ya que cuando se aumenta la dimensionalidad de los problemas que resolver, la propagación hacia atrás no proporciona buenos resultados finales. Por último, nos enfrentamos al *sobreajuste* [Dietterich, 1995] [Hawkins, 2004], de las RNAs entrenadas con propagación hacia atrás.

Las soluciones propuestas a estos problemas son diversas. Por un lado, para los problemas de estancamiento en mínimos locales y falta de escalabilidad se tienen entre otras: adaptación de la *función de activación* [Wang et al., 2004], modificación de las funciones de error [Bi et al., 2005] o la aplicación de *metaheurísticas* [Soria et al., 2006] como métodos de entrenamiento. Por otro lado, para el sobreajuste se proponen, entre otras: usar *dropout* [Srivastava et al., 2014], *detener el entrenamiento con antelación* [Caruana et al., 2001] o *aumentar la cantidad de datos disponible* [Shorten and Khoshgoftaar, 2019]

Se propone realizar un estudio de uno de los enfoques citados: usar metaheurísticas para el entrenamiento de las RNAs. En concreto se plantea estudiar, contrastar y verificar tres metaheurísticas para compararlas con el método tradicional de la propagación hacia atrás: *algoritmos genéticos* (Genetic Algoritms) (GAs) [Davis, 1991], *estrategias evolutivas* (Evolutionary Strategies) (ES) [Wierstra et al., 2008] y *recocido simulado* (Simulated Annealing) (SA) [Van Laarhoven and Aarts, 1987]. Todo ello desde el punto de vista de un trabajo fin de máster y sin ser este un artículo científico.

La metodología que se va a usar a lo largo del trabajo es el método científico [Newton, 1687], el cual vamos a seguir paso por paso. El método científico tiene varias fases: formulación de la pregunta, observación, construcción de la hipótesis, experimentación, medición y falsabilidad.

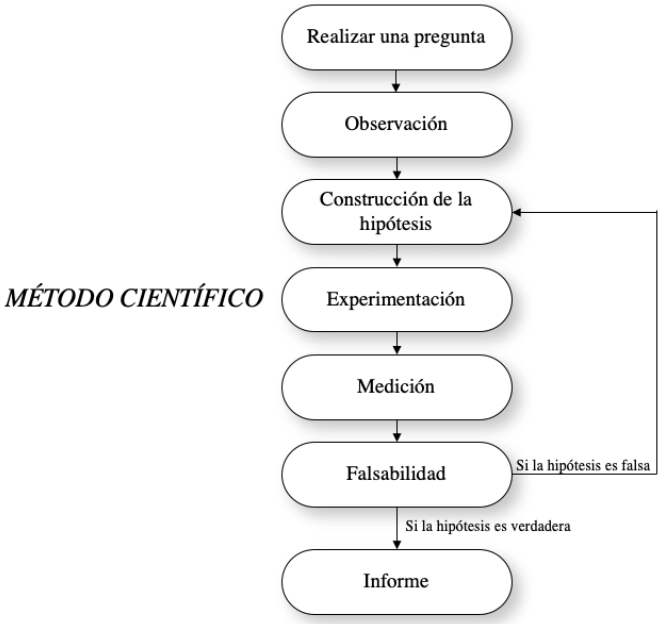

Figura 1. Fases del método científico

Este trabajo se ha estructurado de forma que se siguen cada una de las fases de este.

Los objetivos a los que se aspira con la elaboración de este trabajo son:

- O1: Estudiar y comprender el concepto de neuroevolución.
- O2: Estudiar el funcionamiento la técnica conocida como propagación hacia atrás para el entrenamiento de RNAs.
- O3: Estudiar y comprender los GAs como metaheurística.
- O4: Estudiar las implicaciones que los GAs tienen en el entrenamiento de las RNAs respecto al método tradicional.
- O5: Estudiar y comprender las ES como metaheurísticas.
- O6: Estudiar las implicaciones que las ES tienen en el entrenamiento de las RNAs respecto al método tradicional.
- O7: Estudiar y comprender el recocido simulado como metaheurística.
- O8: Estudiar las implicaciones que el recocido simulado tiene en el entrenamiento de RNAs respecto al método tradicional.
- O9: Realizar un estudio estadístico con los resultados obtenidos de los objetivos O2, O4, O6 y O8.

La formulación de la hipótesis para este trabajo siguiendo el método científico es la siguiente:

Las metaheurísticas mejoran el entrenamiento de las redes neuronales artificiales respecto al método tradicional de propagación hacia atrás.

[1]La función de error puede tratarse de una entropía, un error, o una función de coste entre otras.

[2]La función de error puede tratarse de una entropía, un error, o una función de coste entre otras.

La contribución de este trabajo es original por poner a prueba ideas de otros investigadores anteriores con contribuciones originales. Además, se plantea con un enfoque comparativo, y los datos extraídos de las ejecuciones de los distintos experimentos son únicos.

En lo que resta de este trabajo, se encuentran la definición del estado del arte actual, propuestas técnicas para neuroevolución, una descripción del experimento, la elaboración del experimento, una explicación de los resultados obtenidos y por último las conclusiones extraídas del mismo y trabajos futuros.

II. Estado del arte

En esta sección se revisan algunos trabajos referentes a búsquedas de mínimos globales, centrándonos en aquellos que afectan a este trabajo como lo son las metaheurísticas y el *descenso de gradiente estocástico* [Bottou, 2010] usado como método de búsqueda del mínimo en la propagación hacia atrás. Esto se hace con el objetivo de utilizar estos conceptos en la creación de técnicas que permitan reducir el error de las RNAs.

Para justificar el interés de este tema, se debe decir que es muy importante para el resto del trabajo dado que tanto las metaheurísticas como el procedimiento de propagación hacia atrás se hacen con un fin único: encontrar el mínimo global en el espacio de búsqueda definido por la función del error.

La sección *proceso de búsqueda del mínimo global* guarda una relación importante con todo el trabajo pues las técnicas que van a permitir entrenar las RNAs usadas en este experimento y que serán evaluadas por la calidad de los resultados aportados consisten en minimizar la función de error de las distintas RNAs.

II-A. Proceso de búsqueda del mínimo global

El entrenamiento de RNAs se basa en minimizar una función que se define antes de comenzar el proceso, llamada función de pérdida o función de error (*loss function*).

El proceso de búsqueda del mínimo consiste en cambiar una serie de parámetros en la RNA. Estos parámetros son los pesos (*weights*) y los umbrales (*biases*). La función de error define el espacio de búsqueda en el que nos movemos, y la metodología con la que decidimos como cambiamos estos parámetros a cada paso definen la estrategia de búsqueda que tomamos al movernos por este espacio.

El *descenso del gradiente estocástico* [Polyak and Juditsky, 1992] usado en el entrenamiento mediante propagación hacia atrás, toma a cada paso decisiones que intentan llevar a la solución a un punto menor en la función del error. Es por esto por lo que este método tiene el problema del estancamiento en mínimos locales, pues cuando se llega a un mínimo local no es capaz de salir de él [Gori and Tesi, 1992].

Las metaheurísticas, por el contrario, toman enfoques distintos. Por un lado, las hay que exploran el espacio de búsqueda con varias soluciones a la vez, como los GAs [Davis, 1991].

Por otro lado, las hay que desplazan una o varias soluciones ayudándose de poblaciones exploradoras, como las ES [Back, 1996] [Hansen et al., 2015].

Además, las hay que comienzan con una libertad de movimiento por el espacio de búsqueda muy alta y con el paso del tiempo van reduciendo la libertad de movilidad de la solución, como el recocido simulado [Van Laarhoven and Aarts, 1987].

Las metaheurísticas comparten una característica importante para la resolución de todo tipo de problemas y es que disponen de dos fases. En primer lugar, la fase de *exploración* en la cual la metaheurística se mueve por el espacio de búsqueda libremente. En segundo lugar, la fase de *intensificación o explotación* en la cual la metaheurística se centra en minimizar (o maximizar, según el caso) el valor de la función objetivo o función fitness. El equilibrio entre estas dos fases afecta la velocidad de convergencia del algoritmo.

Usando cualquiera de estas técnicas podemos diseñar un proceso de búsqueda de un mínimo, en función de la implementación que realicemos podremos alcanzar mínimos globales con cierta frecuencia o demostrar que para todas las ejecuciones se alcanza un mínimo global.

III. Propuestas técnicas

En esta sección se muestran las propuestas técnicas en lo que respecta a RNAs y procedimientos de búsqueda del mínimo, tanto metaheurísticos como clásicos.

Esta sección se compone de subsecciones dedicadas a: las RNAs (O2), GAs (O3), RNAs evolucionadas usando GAs, ES (O5), RNAs evolucionadas usando ES, recocido simulado (O7) y RNAs evolucionadas usando recocido simulado en ese orden.

El interés de esta sección reside en el cumplimiento de los objetivos entre paréntesis indicados anteriormente.

Todas las subsecciones giran en torno a la idea de buscar mínimos globales en la función del error de una RNA. De forma general, esta sección presenta las implementaciones que serán utilizadas en próximas secciones.

III-A. Redes neuronales artificiales

Hoy en día las RNAs son un modelo de aprendizaje habitual para análisis de datos de todos los tipos [Krizhevsky et al., 2012] [LeCun et al., 2015] [Bojarski et al., 2016].

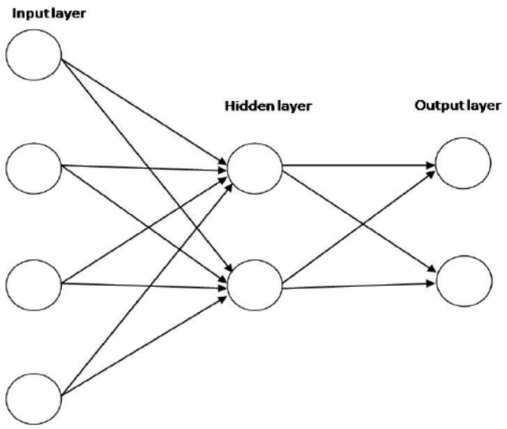

Figura 2. Ejemplo de RNA completa con 3 capas [Magoc and Magoc, 2011]

Éstas se componen de una serie de elementos llamados neuronas artificiales y un conjunto de conexiones entre ellas. Las neuronas se agrupan por capas, como se puede ver en la figura 2 y la información fluye desde la capa de entrada, pasando por la capa o capas ocultas hacia la capa de salida. Si nos centramos en una de esas unidades que componen la RNA observamos lo siguiente:

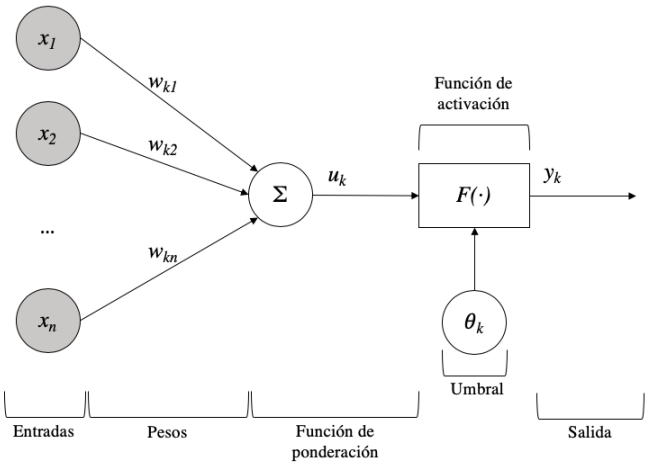

Figura 3. RNA [Ramis, 1943]

El funcionamiento de una neurona como la representada en la figura 3 depende de una serie de elementos:

- Entradas: Representadas en la figura 3 como x_1, x_2 y x_n, simbolizan los datos de entrada para la neurona en cuestión. Estos datos pueden tener dos orígenes: o bien pertenecen al conjunto de datos que se usa para el entrenamiento, validación o pruebas, o bien pertenecen a una neurona que precede a la actual.
- Pesos: Representados en la figura 3 como w_{k1}, w_{k2}, y w_{kn} corresponden a una ponderación que se les aplica a los datos de entrada independientemente de su origen.
- Función de ponderación: Representada en la figura 3 como Σ simboliza la función que aplica las ponderaciones de los pesos a las entradas.
- Umbral: Representado en la figura 3 como Θ_k es un término que regula la predisposición de la neurona a ser activada, independientemente de los valores proporcionados por la función de ponderación. La predisposición de que la neurona sea activada es inversamente proporcional al valor de este umbral. También se identifican con la letra *b*.
- Función de activación: Representada en la figura por *F(·)* es una función que recibe por argumentos los resultados de la función de ponderación y el umbral asignado a la neurona, realiza la operación punto, y aplica el umbral. El valor de esta función normalmente es definido como:

$$F(y) = x_{ki} \times w_{ki} + b_{ki} \quad (2)$$

En la ecuación 2, x son los valores de entrada a la neurona, w los pesos que ponderan dichas entradas dentro de la neurona, y b el umbral. Los subíndices k e i simbolizan la capa en la que se encuentra la neurona y la posición que ocupa esta dentro de esa capa respectivamente. Por tanto, la ecuación se lee: ."El valor de la función de activación para una neurona es igual a las entradas de la neurona ponderadas por sus pesos mas el umbral asignado".

Cuando se agrupan varias neuronas sin conexiones entre ellas, estas forman una capa. Al enlazar varias capas entre ellas se obtiene una RNA. Un ejemplo de RNA lo vemos en la figura 2.

Es de vital importancia señalar que la capa de entrada, etiquetada en la figura 2 como *input layer* no realiza ponderación, activación o aplicación de umbral algunas a los datos que recibe, sólo los distribuye hacia la capa oculta (*hidden layer* en la figura 2) en la que sí son procesados con estas operaciones. Por tanto, la representación de la neurona de la figura 3 se aplica a aquellas neuronas de la capa oculta y de la capa de salida en el ejemplo de la figura 2 permitiendo a la capa de entrada hacer honor a su nombre y actuar como entrada a las capas de neuronas sucesivas.

Las RNAs se sitúan dentro del subcampo de las ciencias de la computación conocido como *aprendizaje automático* [Turing, 2009]. En este subcampo encontramos distintos tipos de aprendizaje, siendo los dos más conocidos el aprendizaje supervisado y el aprendizaje no supervisado. Las RNAs que se van a tratar en este documento pertenecen al primero, lo cual tiene implicaciones en el proceso de entrenamiento.

El entrenamiento de una neurona, al igual que el de una RNA, consiste en ajustar pesos y umbrales para conseguir la salida deseada. En el modelo de aprendizaje tratado en este trabajo, se necesita conocer la salida para cada uno de los ejemplos para ajustar pesos y umbrales, de ahí el nombre de supervisado. Un ejemplo de conjunto de datos para entrenar la neurona de la figura 3 en el caso en el que tuviéramos dos entradas hacia la función de ponderación es el siguiente:

Cuadro I
EJEMPLO DE CONJUNTO DE DATOS (XOR)

i_1	i_2	o
0	0	0
0	1	1
1	0	1
1	1	0

Gracias al conjunto de datos retratado en la tabla I, se puede enseñar a una neurona a realizar la operación XOR utilizando ejemplos. Un comportamiento concreto, puede ser aprendido indistintamente por una neurona que por una RNA. Igual que una RNA sencilla es capaz de aprender la función XOR, una RNA de un cierto tamaño puede aprender problemas más complejos, como, por ejemplo: a distinguir flores dadas unas características [Xin and Embrechts, 2001], a distinguir letras o números escritos a mano en imágenes [Cohen et al.,

2017] o asignar una probabilidad de recidiva a un cáncer de mama una vez superado [Ravdin et al., 1992].

El entrenamiento en RNAs, para aprendizaje supervisado, de forma clásica [HECHT-NIELSEN, 1992] [Soria et al., 2006] se realiza por iteraciones; estas iteraciones se llaman *épocas*. Por cada época, se tienen 2 operaciones: el paso hacia delante (*forward pass*) y la propagación hacia atrás (*backpropagation*).

En cada paso hacia delante se utilizan las entradas de uno de los ejemplos, se pasan por la RNA usando las ponderaciones, funciones de activación (ecuación 2) y umbrales de las neuronas que la componen y se compara con el valor que debería devolver la RNA (para esto se necesita conocer la salida). Con esto se calcula el error que hay entre el valor proporcionado por la RNA y el valor real.

En cada propagación hacia atrás [Rumelhart et al., 1986] se toma el valor del error generado por el paso hacia delante y se actualizan los pesos. Para realizar la actualización de los pesos, para cada uno de ellos y desde la capa de salida hacia la de entrada se aplica la ecuación 3

$$*w_x = w_x - \alpha(\frac{\partial error}{\partial w_x}) \tag{3}$$

- $*w_x$ es el peso actualizado una vez aplicada la propagación hacia atrás.
- w_x es el peso antes de la actualización.
- α es la *tasa de aprendizaje*. Una tasa de aprendizaje demasiado pequeña hará que el sistema se estanque en mínimos locales, y una tasa muy grande generará un comportamiento caótico.
- $(\frac{\partial error}{\partial w_x})$ es la derivada parcial del error con respecto al peso que se está actualizando. Para este cálculo se necesita la derivada de la función de activación usada en el paso hacia delante.

El estancamiento en mínimos locales es un problema importante del que sufren las RNAs optimizadas por propagación hacia atrás, sobre todo cuando el problema se vuelve más complejo y se tiene una alta dimensionalidad en consecuencia, pues debido a la propagación hacia atrás del error y a una tasa de aprendizaje pequeña, una vez llegado a cierto mínimo, la RNA no es capaz de escapar de este. En esta línea hay varios trabajos [Gori and Tesi, 1992] [Wang et al., 2004] que proponen soluciones a este problema.

La RNA que se pretende usar en el experimento que llevar a cabo en el presente trabajo es de tipo completo (*fully-connected*), esto significa que cada neurona de una capa está conectada con todas las neuronas de la capa siguiente, así sucesivamente de principio a fin como se puede ver en la figura 2.

Este diseño tiene varias implicaciones, la más importante es que son aplicables a todo tipo de problemas independientemente de los tipos de datos o complejidad de estos. Como contrapartida, son más lentas que RNAs de propósito específico preparadas para cierto tipo de dato o complejidad como las *RNAs convolucionales* para imágenes [Krizhevsky et al., 2012] o los *autoencoders* [Baldi, 2012].

III-B. Algoritmos genéticos

Las metaheurísticas [Glover and Kochenberger, 2006] [Alba, 2005] [Alba et al., 2013] son un método heurístico para resolver un tipo de problema computacional general. Normalmente se aplican a problemas que no tienen una heurística que dé una solución satisfactoria. Encontrar un mínimo global en la función del error del entrenamiento de una RNA es un problema que no tiene una heurística única definida actualmente, lo que hace interesante comprobar si un GA puede tratar este problema de forma satisfactoria.

En particular las metaheurísticas usadas para el diseño y entrenamiento de RNAs han dado lugar a un campo muy importante actualmente denominado neuroevolución [Floreano et al., 2008].

Un GA [Goldberg et al., 1989] [Holland et al., 1975] es un tipo de metaheurística que se vale de una población, de una función objetivo y de una serie de operadores genéticos para generar descendencia.

La población está definida por individuos, cada uno de ellos se puede componer como se desee. Las implementaciones pueden oscilar entre cosas tan sencillas como un número entero, y tan complejas como una RNA completa con varias capas. Un ejemplo de representación de la población para el problema ONEMAX es la definida en la figura 4:

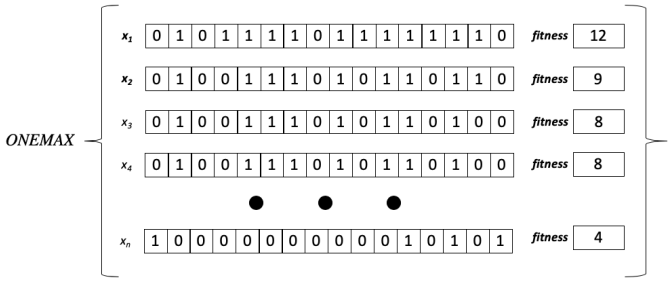

Figura 4. Representación de la población del problema ONEMAX

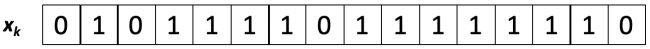

Figura 5. Representación de un *cromosoma* del problema ONEMAX

Además de la población, el GA también se vale de una función para clasificar los individuos y serie de operadores genéticos. El valor de la primera define la bondad del individuo al que se aplica. Los últimos se usan para modificar esa población y producir descendencia. Los operadores siguen las leyes propuestas por Darwin [Darwin and Bynum, 2009], y Mendel [Mendel, 1965] respecto a la selección natural, la mutación y el cruce de individuos [Fox and McMahon, 1991].

- Operadores de selección: Este tipo de operadores realizan la selección al estilo de como pasa en la naturaleza. En el caso de los GAs pasamos de generación (sobreviven) los individuos que tienen mayor (o menor según el caso) valor de la función fitness, aunque en la mayoría de

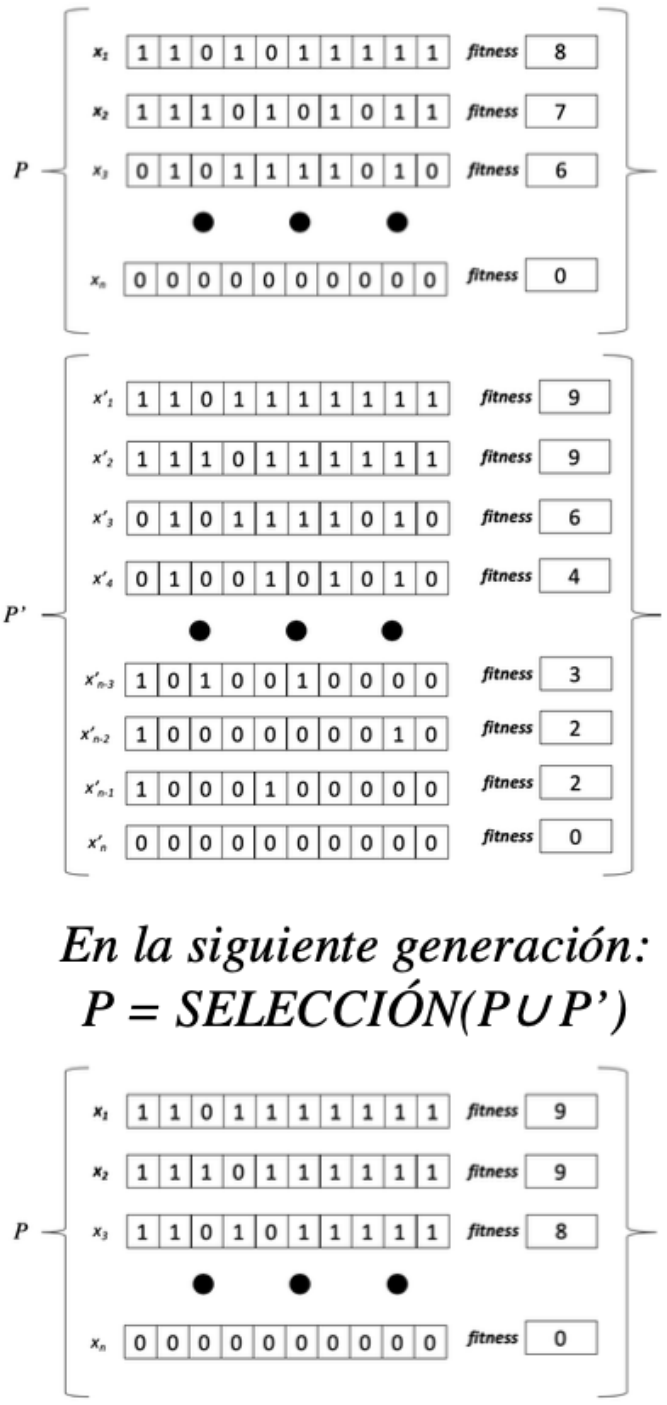

Figura 6. Representación del reemplazo de poblaciones en el GA

las ocasiones también es conveniente mantener algunos resultados que no sean los mejores para que el algoritmo no tienda a converger a mínimos locales y mantenga la variabilidad y las opciones de exploración durante las iteraciones del algoritmo.

- Operadores de mutación: Este tipo de operadores realizan cambios en los diferentes individuos siguiendo una probabilidad. El propósito de estos operadores es el de modificar la estructura del individuo de forma que realice un desplazamiento en el espacio de búsqueda en el que se encuentra.
- Operadores de cruce: Este tipo de operadores realizan recombinaciones entre distintos individuos en base a una probabilidad. Los cruces más comunes son aquellos que afectan a dos individuos de la población en cada aplicación.
- Función fitness: Define la calidad del individuo evaluado. En función de la implementación del GA, la función puede implementarse de forma que haya de maximizarse o bien minimizarse. Un ejemplo de función fitness que maximizar para el problema *ONEMAX* [Schaffer and Eshelman, 1991] es la mostrada en la ecuación 4:

$$-1/n \sum_{i=0}^{n} x[i] \tag{4}$$

x es el *fenotipo* de un individuo.

Con estos elementos se puede construir un GA como el que se puede apreciar en el algoritmo 1.

Algorithm 1: Pseudocódigo del GA

```
Result: población_final
pob := inicializa_población(100)
iter := 0
max_iter := 3000
while iter < max_iter do
    foreach individual individual of the population
     pob do
        foreach element elem of the individual
         individual do
            p := número_aleatorio(0,100)
            if p < probabilidad_mutación then
                pob := mutación(pob)
            end
        end
        p := número_aleatorio(0,100)
        if p < probabilidad_cruce then
            pob := cruce(pob)
        end
    end
    pob := selección(pob)
    iter := iter + 1
end
población_final := pob
```

Para la elaboración de este experimento se han utilizado probabilidades de mutación, selección y cruce constantes.

III-C. Red neuronal artificial evolucionada con algoritmo genético

Si unimos los conceptos de RNA y GA aparecen características interesantes. Una de las posibilidades que brinda esta combinación es la de representar las RNAs como individuos y utilizar los GAs para entrenarlas, buscando los pesos y

umbrales que hagan el valor de la función de error mínimo, función de error que actuará como función fitness.

Desde el punto de vista del GA, necesitamos: la representación de los individuos, la función fitness y los operadores de mutación, selección, cruce y reemplazo.

El genotipo compone a un individuo, es el conjunto de elementos que lo representan. Para la elaboración de este experimento se ha seleccionado una representación de la RNA desde un punto de vista de los parámetros que deben ser modificados para conseguir el valor mínimo de la función fitness. Para esto, dada una estructura de *l* capas que simbolizan una RNA, se tendrán *l* matrices de pesos[3], *l* vectores de umbrales[4] y un valor de la función fitness. Estas variables exceptuando el valor de la función fitness que para un conjunto de pesos y umbrales concreto no varía, cambiarán a lo largo del tiempo según se vayan aplicando los distintos operadores genéticos.

El fenotipo constituye la representación concreta de un individuo, esto es las matrices de pesos con valores para una cierta estructura de neuronas y los vectores de umbrales con valores para esa estructura de neuronas. Esta representación del individuo concreta se usará para evaluarla con la función fitness a lo largo de la ejecución del algoritmo, para obtener la bondad de la solución y eventualmente descartarla o pasarla a la siguiente generación.

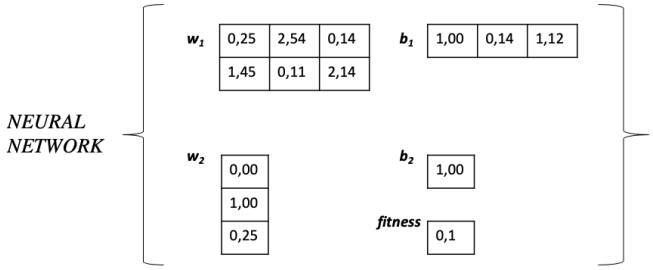

Figura 7. Representación de un individuo de la RNA evolucionada con tres capas

En la figura 7 figura se puede ver el fenotipo de un individuo en una iteración cualquiera de la RNA evolucionada. Esta representación corresponde a una RNA con una estructura (2-3-1), esto es dos neuronas en la capa de entrada, tres neuronas en la capa oculta y una neurona de salida. Es por esto por lo que w_1 es una matriz de dimensiones 2×3 y w_2 es una matriz de dimensiones 3×1. Asimismo, los umbrales adoptan formas vectoriales con el mismo número de elementos que de neuronas tiene la capa siguiente. Es por esto por lo que b_1 contiene tres elementos y b_2 contiene uno. Nótese que no existen ni la matriz w_0 ni el vector de umbrales b_0; esto es así por lo que se comentó anteriormente sobre la capa de entrada, cuya función es servir de entrada para las neuronas de capas sucesivas. Como se puede ver, las matrices contienen números reales. La población por tanto se compone de una serie de individuos con estas representaciones y con variaciones en los números de las matrices dado que son inicializados aleatoriamente y las mutaciones que pueden sufrir a lo largo de la ejecución también son aleatorias.

La función fitness de ejemplo definida anteriormente en la ecuación 1, es la diferencia en cada una de las iteraciones entre los valores arrojados por la RNA y el valor real que se está buscando obtener. Anteriormente en la propagación hacia atrás usábamos ese valor para realizar una actualización de los pesos de la RNA. El uso que se le va a dar ahora es para clasificar qué tan cerca está un individuo del mínimo global.

Algorithm 2: Pseudocódigo de operador de mutación

```
Result: población_mutada
elem := individuo
población_mutada := pob
operador := selecciona_operador_mutación()
mutado := operador(elem)
población_mutada := pob + nuevo_individuo(mutado)
```

El algoritmo 2 es el encargado de realizar la mutación de un individuo, la selección del operador de mutación se realiza teniendo en cuenta una lista de operadores definidos previamente. Los operadores de mutación implementados en este experimento son los siguientes:

- *randomize_a_weight*: Asigna un valor aleatorio a un peso o umbral.
- *add_a_weighted_random_value*: Suma un valor aleatorio aleatoriamente ponderado a un peso o umbral.
- *add_or_substract_a_random_value*: Suma o resta aleatoriamente un valor aleatorio a un peso o umbral.
- *change_sign_of_a_weight_or_bias*: Cambia el signo a un peso o umbral.
- *zero_a_weight_or_bias*: Asigna a cero un peso o umbral.

Todos los operadores se encuentran en una lista en memoria y en cada uso del algoritmo 2 se selecciona un índice aleatorio de esa lista usando un generador de números aleatorios. Con estos operadores conseguimos modificar el individuo de una forma aleatoria, y como se ve en el algoritmo 2 se genera un nuevo individuo para cada una de las mutaciones. Esto es importante ya que de otra forma perderíamos los individuos originales, con la pérdida de los puntos de partida como fuente de exploración y explotación del espacio de búsqueda [Gen and Lin, 2007].

Algorithm 3: Pseudocódigo de operador de selección

```
Result: población_reducida
población_reducida := inicializa_población(0)
población_objetivo := pob_objetivo
población_reducida := población_reducida + take(pob,
  población_objetivo/2)
población_reducida := población_reducida +
  take_randomly(pob, población_objetivo/2)
```

[3]Las dimensiones de las matrices de pesos variarán en función de la estructura definida

[4]Las dimensiones de los vectores de umbrales variarán en función de la estructura definida

En lo que respecta al algoritmo de selección 3, este realiza una reducción de la población a un objetivo concreto definido a nivel de clase. La función *take* trunca los *k* elementos de una población[5] y la función *take_randomly* toma *k* elementos de la población de forma aleatoria.
El algoritmo de cruce 4 toma dos individuos diferentes,

Algorithm 4: Pseudocódigo de operador de cruce

```
Result: población_cruzada
individuos := selecciona_2_aleatoriamente(población)
elem1 := individuos[0]
elem2 := individuos[1]
población_cruzada := []
operador := selecciona_operador_cruce()
cruces := operador(elem1,elem2)
población_cruzada := pob + nuevo_individuo(cruces[0])
  + nuevo_individuo(cruces[1])
```

seleccionados aleatoriamente de la población, realiza una selección de operador, realiza el cruce y se generan dos nuevos individuos, por cada una de las recombinaciones de los individuos. para este experimento se han implementado dos operadores diferentes de cruce:

- *single_point_crossover*: Realiza el cruce en un punto. Consiste en seleccionar un punto concreto de la matriz de pesos, o del vector de umbrales (según sea el caso) de forma aleatoria y realizar una recombinación de las partes que quedan separadas por dicho punto.
- *uniform_crossover*: Realiza un cruce uniforme. Consiste en iterar por la matriz de pesos o por el vector de umbrales (según sea el caso) y por cada elemento decidir si se intercambia por el elemento que ocupa la misma posición en el individuo opuesto.

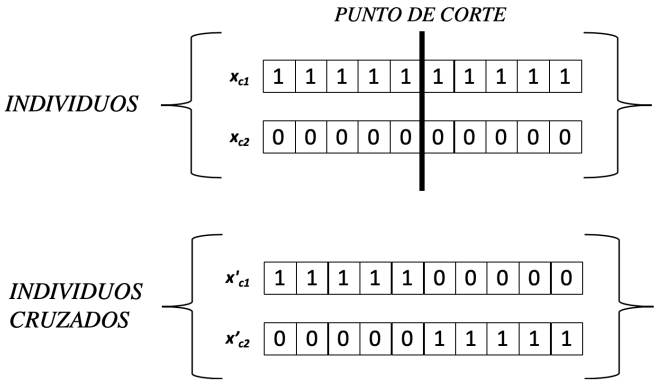

Figura 8. Ejemplo de cruce en un punto para dos individuos del problema ONEMAX

Las figuras 8 y 9 muestran como funcionan estos cruces de una forma más gráfica. La aplicación en la representación de RNAs es exactamente igual. En el caso de las matrices de

[5]La población ha de estar ordenada por fitness antes de llegar a este operador de selección

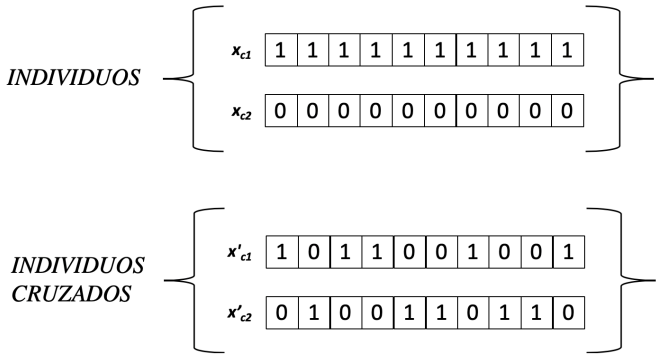

Figura 9. Ejemplo de cruce uniforme para dos individuos del problema ONEMAX

pesos, estas se convierten a vector para seleccionar el punto de corte y tras el cruce se vuelven a convertir a matrices de las dimensiones correctas; esta operación no es necesaria en el caso de los vectores de umbrales, que ya tienen este formato.

La función fitness de esta implementación, es la entropía cruzada [Shore and Johnson, 1980] [Odegua, 2020], pues es la función correcta cuando se quiere entrenar sobre una distribución multinomial, que es lo que se pretende hacer en este trabajo, pues los conjuntos de datos que se van a usar son todos conjuntos que resuelven problemas de clasificación.

De forma intuitiva, se pretende penalizar los casos en los que se obtiene la respuesta incorrecta de forma que el valor de esta función crece exponencialmente según nos alejamos de la respuesta correcta. El error cuadrático medio [Wang and Bovik, 2009] sin embargo no tiene en cuenta lo suficiente las clasificaciones erróneas y por ello está contraindicado para este tipo de problemas. La fórmula que describe la entropía cruzada usada en el desarrollo del experimento es la siguiente:

$$-1/n \sum_{i=1}^{n}((\log \hat{y}) * y) + ((1-y) * \log(1-\hat{y})) \qquad (5)$$

- $\hat{y}$: Son los resultados obtenidos tras realizar el paso hacia delante[6].
- y: Simboliza resultados reales que corresponden al ejemplo concreto que se le pasa a la RNA, referenciado como o en la tabla I.
- n: Corresponde a la longitud del vector y.

La búsqueda del mínimo global para este problema no difiere en absoluto de la búsqueda del mínimo global para otros problemas resueltos por esta metaheurística. Se inicializan todos los individuos y se itera un número de veces en las cuales se mutan, cruzan, se seleccionan y se reemplazan a los individuos hasta llegar al mínimo global o bien llegar al máximo de iteraciones.

Lo que si que cambia es la representación de los individuos en función del problema.

[6]Los valores de las variables y de la ecuación completa se calculan por cada individuo en cada iteración del algoritmo

Las probabilidades de los operadores de selección, mutación y cruce se mantienen constantes durante toda la ejecución en el experimento realizado en este trabajo.

III-D. Estrategias evolutivas

Las ES [Back, 1996] son unas metaheurísticas que pertenecen a la clase de los algoritmos evolutivos, al igual que éstos, utiliza operadores genéticos para modificar una población en búsqueda del mínimo global en un espacio de búsqueda definido por una función fitness.

Las ES aplicadas en este trabajo son una derivación de las ES naturales [Wierstra et al., 2008] donde los individuos se generan a partir de una única solución usando ruido gaussiano [Rasmussen, 2003].

El genotipo y el fenotipo de la población de una ES no tiene por qué diferir del genotipo y el fenotipo de un GA, dado que, son una forma de representar la realidad del problema. En este caso, si se estuviera tratando el problema ONEMAX, la representación de la población en un estadio cualquiera del problema sigue la misma estructura definida en la figura 5.

La implementación sin embargo sí difiere de la de un GA clásico. En este caso, se trabaja con una o varias soluciones a partir de la cual o las cuales se genera una población de individuos basados en ruido gaussiano con la que se opera. Los operadores genéticos aplicados aquí son los de mutación, los operadores de cruce y selección se dejan a un lado. En esta implementación se usa la misma función fitness que en el caso del GA para la resolución del problema ONEMAX, la definida en la ecuación 4.

Algorithm 5: Pseudocódigo del algoritmo ES

```
Result: solución_final
sol_inicial := inicializa_solución()
tam_población := 100
iter := 0
max_iter := 3000
σ := 0,5
tasa_apredizaje := 0.01
while iter < max_iter do
    N := inicializa_población(tam_población)
    R := inicializa_individuos_vacíos(tam_población)
    idx := 0
    foreach individual i of the population N do
        jittered := sol_inicial + (σi)
        R[idx] := evalúa_individuo(jittered)
        idx := idx + 1
    end
    A := normaliza_resultados(R)
    sol_inicial := actualiza_solución(sol_inicial, A,
      tasa_apredizaje)
    iter := iter + 1
end
solución_final := sol_inicial
```

En el algoritmo 5 que representa el ciclo de evolución de las ES, se puede ver que es claramente diferente al algoritmo 1, que es el que representa al GA.

- N: Es una variable intermedia que contiene una inicialización completamente aleatoria de una población de $tam\_población$ individuos en cada iteración.
- R: Es una variable temporal que almacenará los valores arrojados por la RNA.
- A: Valores de R normalizados. La normalización realizada desplaza los valores a un intervalo entre cero y uno.
- $actualiza\_solución$: Función que toma la solución o las soluciones iniciales, R y la tasa de aprendizaje. Realiza un desplazamiento de la solución o las soluciones iniciales hacia la dirección definida por los valores de R.

Como se puede ver la forma de operación es bastante distinta de la del GA, siendo la diferencia más clara la del número de soluciones con las que se trabajan en cada momento. Mientras que en el GA teníamos una población de n individuos, aquí trabajamos con una o varias soluciones por separado. Las implicaciones en la estrategia de búsqueda sin embargo no son tantas como se podría pensar en un primer momento, dado que σ que es el parámetro que permite definir cómo de lejos de la solución inicial se generan los individuos de la población exploradora. Por tanto si damos un valor suficientemente grande de σ, no se pierde la capacidad exploratoria del algoritmo.

III-E. Red neuronal artificial evolucionada con estrategias evolutivas

Al igual que con los GAs, al combinar las RNAs con ES aparecen posibilidades interesantes. Si representamos la RNA como un individuo, podemos desplazarlo por el espacio de búsqueda definido por la función fitness.

Aquí se reutilizan elementos de la implementación de la RNA evolucionada mediante GA: la función fitness, el genotipo, el fenotipo (figura 7), la inicialización aleatoria de la población y la evaluación de esta.

La función fitness es la entropía cruzada descrita en la ecuación 5, que crece exponencialmente si nos alejamos de la solución correcta.

Con estos elementos el único restante es el algoritmo que se usa para llegar al mínimo. En este caso es el algoritmo 5 pero en vez de utilizar la representación de individuos para el problema ONEMAX, se usa la representación fenotípica de la figura 7.

III-F. Recocido simulado

El SA [Van Laarhoven and Aarts, 1987] es una metaheurística que se basa en el proceso de recocido del acero, las cerámicas y los vidrios. El proceso en cuestión consiste en calentar y enfriar lentamente el material para variar sus propiedades físicas, en el ejemplo del acero, para aumentar su dureza. Al calentarse, los átomos aumentan su energía (y su entropía en consecuencia) y pueden desplazarse con mayor libertad desde sus posiciones iniciales. Según la temperatura va descendiendo, éstos terminan adoptando sus posiciones finales, alcanzando un estado de menor energía en el proceso.

En esta descripción la alta movilidad de los átomos, en nuestro caso soluciones, representa la habilidad de escapar de

mínimos locales, para después, adoptar una configuración de menor energía, el mínimo global.

Para la aplicación de esta metaheurística en este trabajo, se ha escogido una variante en la que se realiza un decremento de la temperatura logarítmico, lo cual permite un movimiento menos restrictivo en las primeras iteraciones y una posterior estabilización una vez alcanzado un cierto número de iteraciones. La transición entre la fase de exploración y la fase de intensificación es suave en consecuencia.

En esta metaheurística se trabaja con una solución en cada momento, y nos ayudamos de los operadores genéticos de mutación para explorar.

Como ejemplo volvemos a tomar en consideración el problema ONEMAX, en el que usamos como representación de la solución la figura 5 y como función fitness la definida por la ecuación 4.

Algorithm 6: Pseudocódigo de SA

```
Result: solución_final
sol_inicial := inicializa_solución()
iter := 0
max_iter := 3000
constante_boltzmann := 0.05
temperatura := 100
factor_de_descenso := 0.1
coste := evalua_coste(sol_inicial)
while iter < max_iter do
    actualiza_resultado := False
    nuevo_estado := operador_mutación(sol_inicial)
    nuevo_coste := evalúa_estado(nuevo_estado)
    if nuevo_coste < coste then
        actualiza_resultado := True
    else
        if acepta_resultado(coste, nuevo_coste,
          constante_boltzmann, temperatura) then
            actualiza_resultado := True
        end
    end
    if actualiza_resultado then
        coste := nuevo_coste
        sol_inicial = nuevo_estado
    end
    iter := iter + 1
    temperatura := temperatura · (1 -
      factor_de_descenso)
end
solución_final := sol_inicial
```

Si nos centramos ahora en el algoritmo 6 las funciones desconocidas son:

- operador_mutación: Función que toma un individuo, itera sobre él elemento a elemento y muta cada uno de estos elementos con cierta probabilidad.
- acepta_resultado: Función que toma el coste, el nuevo coste calculado del individuo mutado, la constante de Boltzmann [Aarts and Korst, 1988] definida y la temperatura en esa iteración y en base a una probabilidad modelada por la ecuación 6 donde k es la constante de Boltzmann y T la temperatura, se decide tomar o no la solución como válida, aunque esta sea peor. Este es el factor que ayuda a escapar de mínimos locales.

$$e^{-\triangle c/kT} \tag{6}$$

Como se puede observar, este algoritmo procede de forma diferente ante los resultados peores, usando una función probabilística. Al ir descendiendo la temperatura con el tiempo, la probabilidad de aceptación de un resultado peor también disminuye en consecuencia, aunque nunca es cero.

III-G. Red neuronal artificial evolucionada con recocido simulado

Como se ha visto anteriormente, el SA ofrece una transición entre exploración y explotación suave, lo cual es muy interesante para el entrenamiento de RNAs, ya que la exploración del espacio de parámetros, que crece con la complejidad de la RNA, es difícil de ajustar. La ventaja que puede brindar este método es muy interesante [Rere et al., 2015].

En esta implementación se reutilizan muchos elementos de implementaciones anteriores: La representación del fenotipo de la RNA (figura 7), la inicialización aleatoria de individuos, la implementación de los distintos operadores de mutación del GA y la función del error (aquí referenciada como coste) definida por la ecuación 5.

Con estos elementos y usando el algoritmo 6, se tiene lo necesario para entrenar una RNA usando SA como procedimiento de entrenamiento.

IV. El experimento

En esta sección se va a proporcionar información sobre los conjuntos de datos, los parámetros de entrenamiento de las distintas RNAs, estén o no evolucionadas mediante metaheurísticas, tratamientos previos del conjunto de datos e información sobre las tecnologías utilizadas.

La relación que esta sección guarda con el trabajo es vital, dado que es la descripción del experimento propiamente dicha.

El interés de esta sección reside en añadir detalles sobre la ejecución del experimento de cara a la publicación y puesta en perspectiva de los resultados de este.

El experimento que se pretende realizar consiste en una serie de pruebas usando las implementaciones de las propuestas técnicas de la sección anterior, enfrentándolas a una serie de conjuntos de datos, comparando los resultados del error, tiempo de entrenamiento y sobreajuste obtenidos con las RNAs evolucionadas con distintas metaheurísticas a los obtenidos con una RNA entrenada de forma clásica, esto es, usando propagación hacia atrás.

En lo que respecta a los conjuntos de datos, éstos vienen dados en ficheros de estructura *Comma Separated Value* (CSV). Estos ficheros se dividen de forma vertical en columnas, cada una de las cuales contiene los valores para una determinada variable y horizontalmente en filas a las que a partir de ahora nos referiremos como *ejemplos*. Por cada ejemplo se tienen una serie de variables que se usarán como entradas en las

distintas RNAs (evolucionadas o no por metaheurísticas) y una variable que representa la clase en la que se incluye a ese ejemplo. Ésta variable en función del conjunto de datos tendrá más o menos valores diferentes. Esto es importante porque define la cantidad de neuronas que cada una de las RNAs (evolucionadas o no con metaheurísticas) tendrán en la capa de salida. Esto es así porque se utiliza la entropía cruzada [Rubinstein and Kroese, 2013] como función de error[7] y fitness[8].

Los conjuntos de datos que se van a usar han sido seleccionados del repositorio UCI de aprendizaje automático. En concreto se han seleccionado los siguientes: *iris*, *breast cancer in Wisconsin*, *wine* y *heart disease*. Algunos de estos conjuntos de datos han sido preprocesados asociando atributos alfanuméricos con números, para el correcto funcionamiento de las distintas RNAs. Concretamente se han realizado las siguientes modificaciones como parte del preprocesamiento:

- Se ha sustituido la variable alfanumérica que da el nombre de la flor del iris por números entre el cero y el dos.
- Se ha eliminado una columna desconocida del conjunto de datos wine que no venía descrita en su fichero correspondiente.
- Se ha modificado el valor numérico que identifican el cáncer (valor dos para benigno y cuatro para maligno) por los valores cero y uno en el conjunto de datos del cancer de mama en Wisconsin.
- Se ha eliminado la columna *sample_number* del conjunto de datos del cancer de mama en Wisconsin ya que no aporta nada al problema, solo es identificativo.
- Se han eliminado los ejemplos que tuvieran algún valor desconocido de todos los conjuntos de datos.

Junto al resto de modificaciones, antes de entrenar y definir las RNAs, se realiza un procesamiento adicional de los datos. Debido a que se toma como función del error la entropía cruzada en el caso de la propagación hacia atrás y se toma esta misma función como fitness en las distintas metaheurísticas, se debe proporcionar una probabilidad ente cero y uno en cada resultado arrojado por la RNA, siendo este la probabilidad de pertenencia a la clase que se intenta predecir. Los problemas a los que enfrentamos las distintas RNAs en este trabajo tienen mas de una clase a la que pueden pertenecer cada uno de los ejemplos. La solución a este inconveniente para que las redes que usan la entropía cruzada como función de error funcionen correctamente es utilizar una neurona en la capa de salida por cada valor de clasificación diferente que pueda tomar un ejemplo y que cada una de esas neuronas arroje un resultado entre cero y uno siendo esta cada una de las probabilidades de pertenencia a cada una de las distintas clases en las que se clasifican los ejemplos. Debido a esto, los valores esperados han de ser convertidos a *variables categóricas*. Un resumen de las neuronas de la capa de entrada y salida, así como de cuántos ejemplos incluye cada conjunto de datos se puede ver en la tabla II.

Cuadro II
RESUMEN DE LOS CONJUNTOS DE DATOS

	Iris	HD	BCW	Vino
Número de atributos de entrada	4	13	9	11
Número de valores posibles de salida	3	5	2	3
Número de muestras	150	298	684	179

La primera fila de la tabla se corresponde con:

- Iris: Conjunto de datos de las flores del iris.
- HD: Conjunto de datos de las enfermedades del corazon (Heart Disease).
- BCW: Conjunto de datos del cancer de mama en Wisconsin (Breast Cancer in Wisconsin).
- Vino: Conjunto de datos del vino.

sepal_length	sepal_width	petal_length	petal_width	y
5.1	3.5	1.4	0.2	Iris-setosa
4.9	3.0	1.4	0.2	Iris-setosa
6.3	3.3	6.0	2.5	Iris-virginica
5.8	2.7	5.1	1.9	Iris-virginica
7.0	3.2	4.7	1.4	Iris-versicolor
6.4	3.2	4.5	1.5	Iris-versicolor

Cuadro III
EXTRACTO DEL CONJUNTO DE DATOS DE LAS FLORES DEL IRIS ANTES DE SER PROCESADO

sepal_length	sepal_width	petal_length	petal_width	new_y
5.1	3.5	1.4	0.2	[1,0,0]
4.9	3.0	1.4	0.2	[1,0,0]
6.3	3.3	6.0	2.5	[0,1,0]
5.8	2.7	5.1	1.9	[0,1,0]
7.0	3.2	4.7	1.4	[0,0,1]
6.4	3.2	4.5	1.5	[0,0,1]

Cuadro IV
EXTRACTO DEL CONJUNTO DE DATOS DE LAS FLORES DEL IRIS TRAS SER PROCESADO

Para poner esto en perspectiva, vamos a tomar como ejemplo el conjunto de datos de las flores del iris del cual se tiene un extracto en la tabla III. Este conjunto de datos contiene para cada ejemplo cuatro atributos que se usan para la clasificación de este: longitud del sépalo, ancho del sépalo, longitud del pétalo y ancho del pétalo; estos atributos serán los valores que la RNA tome como entradas. En el conjunto de datos encontramos que para cada combinación de estos valores hay una clasificación de tipo alfanumérico: el tipo de flor. Hay tres tipos de flores en este conjunto de datos: *iris setosa*, *iris virginica* e *iris versicolor*, estos valores compondrán la salida de la RNA. La RNA resultante se entrenará con propagación hacia atrás y será evolucionada mediante GAs, SA y ES.

Como se ha dicho anteriormente, por cada posible clasificación debe existir una neurona en la capa de salida, y por cada atributo que se quiera usar para predecir dicha clasificación se debe usar una neurona en la capa de entrada, por tanto, la estructura de la RNA resultante sería (4-x-3) donde x compone la capa o capas ocultas de la RNA. Si x se compone de

[7]Usada en la RNA que usa propagación hacia atrás

[8]Usada en las RNAs evolucionadas mediante alguna de las metaheurísticas

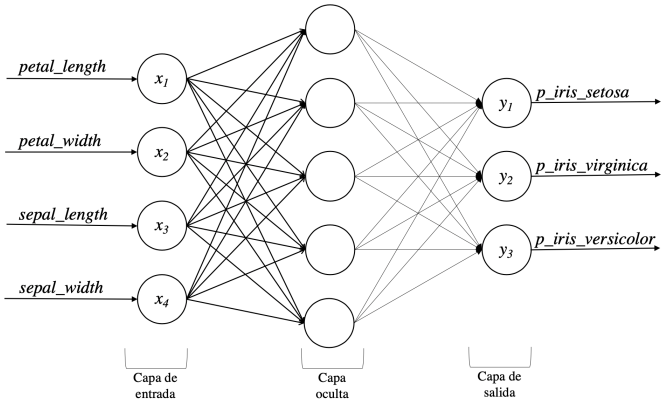

Figura 10. Ejemplo de RNA para el conjunto de datos de las flores del iris con una capa oculta de 5 neuronas

una capa con 5 neuronas, tenemos la RNA de la figura 10. Antes de proceder al entrenamiento de esta RNA, se tiene que realizar el preprocesamiento de los datos y a la conversión de las variables de clasificación a forma categórica, como se puede ver en la tabla IV.

En lo que respecta a tecnologías usadas, se ha elegido *python* por la variedad de paquetes que tiene a su disposición, los más interesantes y los que se han usado han sido el paquete estándar de Python, *numPy* y *pandas*. NumPy por su parte aporta las operaciones con vectores y matrices, y lo hace con buena velocidad ya que este paquete esta programado y compilado en C, que es mucho mas rápido que el intérprete de python. Pandas por su parte, aporta la posibilidad de trabajar los conjuntos de datos en formato estándar CSV. Además, provee la capacidad de modificarlos de una forma muy sencilla lo que ayuda al preprocesamiento, y de extraer métricas de los datos. También se ha usado para construir las tablas y estadísticas de los distintos entrenamientos paralelos.

Junto al código anterior, se han implementado pruebas unitarias para todas las RNAs diferentes evolucionadas o no mediante metaheurísticas. Para esto, se ha utilizado *pyTest*, un paquete sencillo en contiene todo lo necesario para hacer pruebas unitarias, incluyendo *mocking* para la realización de las pruebas lógicas.

Junto a estos ficheros de código y pruebas, también se han incluido en el repositorio de este trabajo una serie de cuadernillos realizados con *Jupyter* en los cuales se exponen ejemplos usando las RNAs implementadas de cara a la comunidad y al claro entendimiento de lo que se ha realizado en este trabajo. Dichos cuadernillos están en inglés de cara a llegar a una porción más amplia de la comunidad.

Debido a que se quiere realizar un análisis estadístico de los resultados obtenidos, en este experimento realizaremos treinta ejecuciones con cada uno de los modelos y conjuntos de datos, cambiando entre cada una de las ejecuciones la semilla del generador de números aleatorios. Para la selección de semillas se han tomado números primos diferentes en magnitud para asegurar un mejor resultado, pero todos ellos dentro del rango entero, esto es porque numPy no acepta semillas mayores de $2^{32} - 1$. Se hacen treinta debido a que es el mínimo número de ejecuciones necesarias para que sea estadísticamente significativo.

Para la realización del experimento se ha preparado un fragmento de código paralelizado, ya que, en unas pruebas previas, se ha visto que la diferencia entre paralelizar la ejecución del experimento y no hacerlo hay un 48 % de diferencia en tiempo. Esto es debido a que la maquina que se encarga de ejecutar este código del experimento cuenta con 16 núcleos de procesamiento y se puede ejecutar el entrenamiento de las distintas RNAs al mismo tiempo. En caso de que este experimento se ejecute en una máquina mononúcleo, no habrá diferencia significativa entre una ejecución paralelizada y una ejecución secuencial.

Por cada algoritmo y conjunto de datos se realiza *validación cruzada de K iteraciones* [Browne, 2000], para ello se toma el conjunto de datos con el que queremos entrenar la RNA y se divide en K partes de las cuales una se utiliza para validar el entrenamiento de la RNA y $K - 1$ se utilizan para el entrenamiento de la red. En este experimento, para todas las ejecuciones $K = 5$, es decir, que se utiliza un 80 % de los datos del conjunto de datos concreto para el entrenamiento y un 20 % para validación.

V. Realización del experimento

En esta sección se van a presentar los datos extraídos de las ejecuciones del experimento con los conjuntos de datos mencionados anteriormente.

El interés de esta sección reside en mostrar los datos extraídos del experimento descrito en la sección anterior y que está basado en todo lo comentado en este trabajo. Este interés se comparte con el interés de proporcionar argumentos para en la siguiente sección determinar si la hipótesis que se propone al principio del trabajo es cierta o no, coincidiendo con la fase de medición del método científico.

La relación que esta sección guarda es muy importante con el resto del trabajo pues es la sección en la que se muestran los resultados del experimento que se viene definiendo en el resto de este.

En esta sección se trabajan los objetivos O4, O6, O8 y O9.

Para la elaboración de este experimento se toman todas las propuestas técnicas propuestas con anterioridad y se ponen a prueba con lo conjuntos de datos comentados en la sección anterior. Al usar todos estos recursos podemos formar el algoritmo 7 que nos servirá para realizar la extracción de métricas sobre el rendimiento de las distintas RNAs, tanto aquellas configuraciones que se entrenan usando propagación hacia atrás como aquellas que se evolucionan mediante alguna de las metaheurísticas propuestas.

En el algoritmo 7 tenemos una serie de elementos clave que son:

- *create_global_dataframes*: Crea unas estructuras de datos con pandas de forma global de modo que son accesibles por todas las RNAs en cualquier momento de la ejecución. Estas estructuras están protegidas para escritura con

Algorithm 7: Pseudocódigo del que realiza la ejecución de uno de los experimentos

```
create_global_dataframes()
num_épocas := 5
num_folds := 5
primos := [3,11,101,5003, ...
 ,849347,1849283,71849363]
labels_lists, inputs_lists := inicializa_datasets()
foreach primo p of the list primos do
    random_seed := p
    foreach number idx in
     rango(length(labels_list)) do
        training_folds, testing_folds :=
         KFold(inputs_lists[idx], labels_lists[idx],
         num_folds) foreach training_fold,
         testing_fold in
         enumerate(training_folds, testing_folds)
         do
            t1 := new_thread(basic_nn_tenant)
            t2 := new_thread(random_nn_tenant)
            t3 := new_thread(genetic_nn_tenant)
            t4 := new_thread(strategy_nn_tenant)
            t5 := new_thread(annealed_nn_tenant)
            t1.start(training_fold, testing_fold,
             num_épocas)
            t2.start(training_fold, testing_fold,
             num_épocas)
            t3.start(training_fold, testing_fold,
             num_épocas)
            t4.start(training_fold, testing_fold,
             num_épocas)
            t5.start(training_fold, testing_fold,
             num_épocas)
            t1.join()
            t2.join()
            t3.join()
            t4.join()
            t5.join()
        end
    end
end
save_data()
```

una estructura de control de concurrencia conocida como *lock* [Kleiman et al., 1996].

- *inicializa_datasets*: Función que procesa los conjuntos de datos y los prepara para el entrenamiento realizando los cambios descritos en la sección anterior y la conversión a variables categóricas tal y como queda en la tabla IV
- *labels_lists, input_lists*: Listas que contienen los datos de los conjuntos de datos procesados en inicializa_datasets. La primera variable contiene los valores de las variables clasificatorias en forma categórica y la segunda los valores de los atributos de entrada para la RNA ya procesados en caso de que fuera necesario.
- *random_seed*: Asignación de la semilla del generador de números aleatorios.
- *training_folds, testing_folds*: Listas de las distintas particiones realizadas mediante la validación cruzada de K iteraciones. En la primera variable tenemos una lista de las particiones del conjunto de entrenamiento y en la segunda tenemos una lista de particiones para el conjunto de pruebas.
- *basic_nn_tenant*: Función que contiene la ejecución del experimento completa para el caso de la RNA entrenada con propagación hacia atrás. En esta función se crea la RNA con la estructura definida, se entrena con el número de épocas definido al principio del algoritmo, se extraen las estadísticas y se añaden a la estructura de datos global definida en *create_global_dataframes*.
- *random_nn_tenant*: Función que contiene la ejecución del experimento completa para el caso de la RNA entrenada con búsqueda aleatoria que servirá como control en este experimento. En esta función se crea la RNA con la estructura definida, se entrena con el número de épocas definido al principio del algoritmo, se extraen las estadísticas y se añaden a la estructura de datos global definida en *create_global_dataframes*.
- *genetic_nn_tenant*: Función que contiene la ejecución del experimento completa para el caso de la RNA evolucionada mediante GAs. En esta función se crea la RNA con la estructura definida, se entrena con el número de épocas definido al principio del algoritmo, se extraen las estadísticas y se añaden a la estructura de datos global definida en *create_global_dataframes*. Junto a las estadísticas globales, también se extraen unas estadísticas sobre el fitness en cada época que se añade a una estructura de datos diferente y exclusiva para el GA de cara a analizar su comportamiento tras la ejecución del experimento.
- *strategy_nn_tenant*: Función que contiene la ejecución del experimento completa para el caso de la RNA evolucionada mediante ES. En esta función se crea la RNA con la estructura definida, se entrena con el número de épocas definido al principio del algoritmo, se extraen las estadísticas y se añaden a la estructura de datos global definida en *create_global_dataframes*. Junto a las estadísticas globales, también se extraen unas estadísticas sobre el fitness en cada época que se añade a una estructura de datos diferente y exclusiva para las ES de cara a analizar su comportamiento tras la ejecución del experimento.
- *annealed_nn_tenant*: Función que contiene la ejecución del experimento completa para el caso de la RNA evolucionada mediante SA. En esta función se crea la RNA con la estructura definida, se entrena con el número de épocas definido al principio del algoritmo, se extraen las estadísticas y se añaden a la estructura de datos global definida en *create_global_dataframes*. Junto a las estadísticas globales, también se extraen unas estadísticas sobre el fitness en cada época que se añade a una

estructura de datos diferente y exclusiva para el SA de cara a analizar su comportamiento tras la ejecución del experimento.

- *save_data* Función que toma las estructuras de datos globales (por eso no se le pasa ningún argumento) y las guarda en ficheros estándar CSV.

Los experimentos que se han realizado en este trabajo constan de cuatro experimentos diferentes con parámetros distintos y diferentes configuraciones de RNAs. Cada experimento consiste en 30×4 entrenamientos por parte de cada una de las RNAs. Treinta son las semillas diferentes utilizadas en el experimento para que sea estadísticamente significativo y cuatro son los conjuntos de datos utilizados en cada experimento.

Los factores que diferencian a estos cuatro experimentos son las estructuras de las RNAs y el número de épocas en las que han sido entrenadas.

Los parámetros internos con los que se han trabajado en las distintas implementaciones de las RNAs quedan resumidos en las tablas: V, VI, VII y VIII.

Cuadro V
PARÁMETROS DE EJECUCIÓN DEL EXPERIMENTO PARA LA RNA ENTRENADA MEDIANTE PROPAGACIÓN HACIA ATRÁS

RNA con propagación hacia atrás	
Parámetro	Valor
Tasa de aprendizaje	0.05

Cuadro VI
PARÁMETROS DE EJECUCIÓN DEL EXPERIMENTO PARA LA RNA EVOLUCIONADA MEDIANTE GA

RNA con GA	
Parámetro	Valor
Tamaño de población	100
Prob. Mutación	40 %
Prob. Cruce	20 %
Prob. Selección	100 %

Cuadro VII
PARÁMETROS DE EJECUCIÓN DEL EXPERIMENTO PARA LA RNA EVOLUCIONADA MEDIANTE ES

RNA con ES	
Parámetro	Valor
Tamaño de población	50
σ	0.5
Tasa de aprendizaje	0.1

Cuadro VIII
PARÁMETROS DE EJECUCIÓN DEL EXPERIMENTO PARA LA RNA EVOLUCIONADA MEDIANTE SA

RNA con SA	
Parámetro	Valor
Temperatura inicial	5
Factor de caída	0,2
Constante de Bolzmann	50
Probabilidad de mutación	40 %

Durante la ejecución de todos los experimentos estos parámetros se han mantenido constantes.

Los experimentos realizados se han estructurado de la siguiente forma:

- *Experimento 1*: Consiste en utilizar una estructura con cinco neuronas en la capa oculta y cada una de las redes resultantes con los distintos procedimientos durante cinco épocas. Para el caso del conjunto de datos de las flores del iris la RNA resultante es la representada en la figura 10. Sin embargo, es importante notar que esta RNA no tendrá esa estructura para todos los conjuntos de datos ya que cada conjunto de datos tiene un número diferente de atributos de entrada y un número distinto de valores para la salida como se puede apreciar en la tabla II, por tanto, las estructuras van a variar tanto en las capas de entrada como la de salida siendo lo único constante la capa oculta de cinco neuronas. Este comportamiento se repetirá a lo largo del resto de experimentos.
- *Experimento 2*: Consiste en utilizar una estructura con cinco neuronas en la capa oculta y entrenar cada una de las redes resultantes con los distintos procedimientos durante diez épocas.
- *Experimento 3*: Consiste en utilizar una estructura con diez neuronas en la capa oculta y entrenar cada una de las redes resultantes con los distintos procedimientos durante cinco épocas.
- *Experimento 4*: Consiste en utilizar una estructura con diez neuronas en la capa oculta y entrenar cada una de las redes resultantes con los distintos procedimientos durante diez épocas.

Las tablas numeradas consecutivamente desde IX hasta XXIV, ambas incluidas contienen algunos resultados de los cuatro experimentos realizados. Cada tabla corresponde a un conjunto de datos de un experimento; esto se ha hecho así para ver el rendimiento de cada uno de los métodos de entrenamiento a la hora de entrenar una RNA con una misma estructura. En cada tabla encontramos lo siguiente:

- *Método de entrenamiento*: Contiene los métodos de entrenamiento con los que la RNA ha sido entrenada o evolucionada según el caso.
- *MFE*: Corresponde al valor medio del fitness (o de la función del error según corresponda) durante el entrenamiento.
- *DTFE*: Corresponde a la desviación típica del fitness (o de la función del error según corresponda) durante el entrenamiento.
- *MFV*: Corresponde al valor medio del fitness (o de la función del error según corresponda) en la fase de validación[9].
- *DTFV*: Corresponde a la desviación típica del fitness (o de la función del error según corresponda) en la fase de validación[10].

[9]Para la fase de validación se usa el 20 % de los datos como ya se ha comentado en la sección *El experimento*

[10]Para la fase de validación se usa el 20 % de los datos como ya se ha comentado en la sección *El experimento*

Cuadro IX
MÉTRICAS PARA LAS RNAS ENTRENADAS EN EL CONJUNTO DE DATOS DE LAS FLORES DEL IRIS EN EL EXPERIMENTO 1

Método de entrenamiento	MFE	DTFE	MFV	DTFV	MTE	DTTE	Sobreajuste
Propagación hacia atrás	3.0839	0.8863	5.3631	2.4694	0.0033	0.0022	2.2792
Búsqueda aleatoria	2.1938	0.241	2.2087	0.2875	**0.0031**	0.001	**0.0149**
SA	3.242	1.8278	4.8615	2.5887	0.0129	**0.0007**	1.6195
ES	2.1246	0.2746	2.1635	0.3214	22.6945	1.0703	0.0389
GA	**1.6613**	**0.1023**	**1.6775**	**0.1366**	35.8627	2.5753	0.0162

Cuadro X
MÉTRICAS PARA LAS RNAS ENTRENADAS EN EL CONJUNTO DE DATOS DE LAS ENFERMEDADES DEL CORAZÓN EN EL EXPERIMENTO 1

Método de entrenamiento	MFE	DTFE	MFV	DTFV	MTE	DTTE	Sobreajuste
Propagación hacia atrás	5.1219	1.8066	11.322	5.5038	0.0073	0.0022	6.2001
Búsqueda aleatoria	3.2823	0.6111	3.2756	0.6199	**0.0059**	0.0025	0.0067
SA	5.7829	3.2704	7.8181	4.0871	0.018	**0.0014**	2.0352
ES	3.3793	0.779	3.3767	0.7993	4.2725	0.2334	**0.0026**
GA	**2.2651**	**0.1065**	**2.289**	**0.1868**	6.039	0.4258	0.0239

Cuadro XI
MÉTRICAS PARA LAS RNAS ENTRENADAS EN EL CONJUNTO DE DATOS DE CÁNCER DE MAMA EN WISCONSIN EN EL EXPERIMENTO 1

Método de entrenamiento	MFE	DTFE	MFV	DTFV	MTE	DTTE	Sobreajuste
Propagación hacia atrás	1.9675	0.6093	inf	N/A	0.0073	0.0021	inf
Búsqueda aleatoria	1.4327	0.1544	1.4279	0.1744	**0.0071**	0.0028	0.0048
SA	2.4561	1.9719	4.1038	2.5332	0.0146	**0.0009**	1.6477
ES	1.4036	0.1695	1.4094	0.1793	3.5718	0.2082	0.0058
GA	**1.073**	**0.0921**	**1.0736**	**0.1114**	4.8825	0.3014	**0.0006**

Cuadro XII
MÉTRICAS PARA LAS RNAS ENTRENADAS EN EL CONJUNTO DE DATOS DEL VINO EN EL EXPERIMENTO 1

Método de entrenamiento	MFE	DTFE	MFV	DTFV	MTE	DTTE	Sobreajuste
Propagación hacia atrás	3.0527	0.9027	6.4208	3.3492	0.0042	0.0022	3.3681
Búsqueda aleatoria	2.17	0.2003	2.2009	0.2321	**0.0033**	**0.001**	0.0309
SA	3.3972	1.9026	5.4808	3.2966	0.0161	**0.001**	2.0836
ES	2.1686	0.2664	2.1635	0.2656	27.5046	1.4561	**0.0051**
GA	**1.8569**	**0.0574**	**1.8798**	**0.0899**	43.5808	3.3569	0.0229

Cuadro XIII
MÉTRICAS PARA LAS RNAS ENTRENADAS EN EL CONJUNTO DE DATOS DE LAS FLORES DEL IRIS EN EL EXPERIMENTO 2

Método de entrenamiento	MFE	DTFE	MFV	DTFV	MTE	DTTE	Sobreajuste
Propagación hacia atrás	3.047	0.9616	4.8865	2.2169	**0.0031**	0.002	1.8395
Búsqueda aleatoria	2.0612	0.1678	2.053	0.1822	0.0063	0.0105	**0.0082**
SA	3.3459	2.347	5.6158	3.1195	0.0241	**0.0018**	2.2699
ES	1.8802	**0.0735**	1.8986	**0.0785**	91.7441	3.4822	0.0184
GA	**1.5049**	0.122	**1.5333**	0.138	124.3472	7.0259	0.0284

Cuadro XIV
MÉTRICAS PARA LAS RNAS ENTRENADAS EN EL CONJUNTO DE DATOS DE LAS ENFERMEDADES DEL CORAZÓN EN EL EXPERIMENTO 2

Método de entrenamiento	MFE	DTFE	MFV	DTFV	MTE	DTTE	Sobreajuste
Propagación hacia atrás	5.0974	1.7002	10.8183	5.2886	**0.0074**	**0.0019**	5.7209
Búsqueda aleatoria	3.0574	0.4375	3.0528	0.4775	0.0091	0.0021	0.0046
SA	5.4472	3.3897	9.0481	4.9417	0.0336	0.0027	3.6009
ES	2.4329	0.2387	2.435	0.2622	16.0305	0.6756	**0.0021**
GA	**2.154**	**0.0652**	**2.1675**	**0.1441**	20.6245	0.9993	0.0135

Cuadro XV
MÉTRICAS PARA LAS RNAS ENTRENADAS EN EL CONJUNTO DE DATOS DE CÁNCER DE MAMA EN WISCONSIN EN EL EXPERIMENTO 2

Método de entrenamiento	MFE	DTFE	MFV	DTFV	MTE	DTTE	Sobreajuste
Propagación hacia atrás	2.0242	0.7008	629928546.9314	1524535077.7644	**0.0074**	**0.002**	629928544.9072
Búsqueda aleatoria	1.3536	0.1106	1.3544	0.1159	0.0109	0.0026	**0.0008**
SA	1.9522	1.0509	3.7634	2.2201	0.0274	0.0026	1.8112
ES	1.2451	**0.0715**	1.2486	**0.0834**	14.022	0.463	0.0035
GA	**0.9424**	0.1066	**0.9521**	0.1171	16.8523	0.8942	0.0097

Cuadro XVI
MÉTRICAS PARA LAS RNAS ENTRENADAS EN EL CONJUNTO DE DATOS DEL VINO EN EL EXPERIMENTO 2

Método de entrenamiento	MFE	DTFE	MFV	DTFV	MTE	DTTE	Sobreajuste
Propagación hacia atrás	3.0947	0.8748	6.9053	3.2411	**0.0036**	**0.0017**	3.8106
Búsqueda aleatoria	2.0829	0.1516	2.0864	0.1936	0.0069	0.0135	**0.0035**
SA	2.9389	1.3659	5.7334	3.5333	0.0299	0.0025	2.7945
ES	1.9152	**0.0272**	1.934	**0.0526**	111.5076	3.3963	0.0188
GA	**1.7776**	0.0886	**1.8021**	0.1233	150.981	8.0304	0.0245

Cuadro XVII
MÉTRICAS PARA LAS RNAS ENTRENADAS EN EL CONJUNTO DE DATOS DE LAS FLORES DEL IRIS EN EL EXPERIMENTO 3

Método de entrenamiento	MFE	DTFE	MFV	DTFV	MTE	DTTE	Sobreajuste
Propagación hacia atrás	3.4713	1.2094	9.5676	4.1881	0.0039	0.002	6.0963
Búsqueda aleatoria	2.3471	0.374	2.3677	0.4516	**0.0036**	0.0023	0.0206
SA	3.8334	2.2115	5.6189	2.8824	0.0159	**0.0016**	1.7855
ES	2.1928	0.316	2.2145	0.3298	12.1933	1.4932	0.0217
GA	**1.616**	**0.1223**	**1.6205**	**0.1859**	17.4045	1.4103	**0.0045**

Cuadro XVIII
MÉTRICAS PARA LAS RNAS ENTRENADAS EN EL CONJUNTO DE DATOS DE LAS ENFERMEDADES DEL CORAZÓN EN EL EXPERIMENTO 3

Método de entrenamiento	MFE	DTFE	MFV	DTFV	MTE	DTTE	Sobreajuste
Propagación hacia atrás	5.9344	2.6232	143165596.4026	773553042.5087	0.0088	0.0024	143165590.4682
Búsqueda aleatoria	3.5641	0.7955	3.5853	0.8041	**0.0079**	0.0028	0.0212
SA	6.3161	3.5932	9.4552	4.7701	0.0251	**0.0017**	3.1391
ES	3.4823	0.8521	3.4853	0.8448	5.7192	0.3468	**0.003**
GA	**2.293**	**0.112**	**2.3163**	**0.2145**	8.0273	0.533	0.0233

Cuadro XIX
MÉTRICAS PARA LAS RNAS ENTRENADAS EN EL CONJUNTO DE DATOS DE CÁNCER DE MAMA EN WISCONSIN EN EL EXPERIMENTO 3

Método de entrenamiento	MFE	DTFE	MFV	DTFV	MTE	DTTE	Sobreajuste
Propagación hacia atrás	2.4786	1.0061	inf	N/A	**0.0081**	0.0021	inf
Búsqueda aleatoria	1.5145	0.2178	1.5247	0.2329	0.0095	0.0027	0.0102
SA	2.7894	2.1242	4.723	2.8709	0.0193	**0.0018**	1.9336
ES	1.4589	0.2541	1.4538	0.2576	4.7382	0.3696	**0.0051**
GA	**1.0707**	**0.0875**	**1.0906**	**0.1145**	6.2675	0.4376	0.0199

Cuadro XX
MÉTRICAS PARA LAS RNAS ENTRENADAS EN EL CONJUNTO DE DATOS DEL VINO EN EL EXPERIMENTO 3

Método de entrenamiento	MFE	DTFE	MFV	DTFV	MTE	DTTE	Sobreajuste
Propagación hacia atrás	3.6968	1.4248	13.929	6.9431	0.0043	0.0019	10.2322
Búsqueda aleatoria	2.3772	0.35	2.3803	0.3752	**0.0037**	0.0018	**0.0031**
SA	4.0882	2.7267	6.2878	3.5714	0.0218	**0.0017**	2.1996
ES	2.3186	0.4353	2.3274	0.4549	14.1927	0.8952	0.0088
GA	**1.8455**	**0.0814**	**1.8729**	**0.1135**	21.228	1.6921	0.0274

Cuadro XXI
MÉTRICAS PARA LAS RNAS ENTRENADAS EN EL CONJUNTO DE DATOS DE LAS FLORES DEL IRIS EN EL EXPERIMENTO 4

Método de entrenamiento	MFE	DTFE	MFV	DTFV	MTE	DTTE	Sobreajuste
Propagación hacia atrás	3.4096	1.097	9.2039	3.9554	**0.004**	0.0018	5.7943
Búsqueda aleatoria	2.1307	0.2389	2.1405	0.2826	0.0057	**0.001**	**0.0098**
SA	3.4653	1.7207	6.5049	3.8526	0.0295	0.0029	3.0396
ES	1.8697	**0.0751**	1.8889	**0.087**	47.0626	3.9039	0.0192
GA	**1.4285**	0.1344	**1.4387**	0.1542	59.393	3.8174	0.0102

Cuadro XXII
MÉTRICAS PARA LAS RNAS ENTRENADAS EN EL CONJUNTO DE DATOS DE LAS ENFERMEDADES DEL CORAZÓN EN EL EXPERIMENTO 4

Método de entrenamiento	MFE	DTFE	MFV	DTFV	MTE	DTTE	Sobreajuste
Propagación hacia atrás	6.1854	2.511	229064942.3195	968300733.8014	**0.0084**	**0.0019**	229064936.1341
Búsqueda aleatoria	3.2083	0.5769	3.2495	0.6335	0.0113	0.0025	0.0412
SA	5.9212	2.8426	9.9493	5.1545	0.0477	0.0029	4.0281
ES	2.3775	0.2175	2.3888	0.2393	22.2372	0.7828	**0.0113**
GA	**2.1609**	**0.0706**	**2.183**	**0.1498**	27.2777	1.3155	0.0221

Cuadro XXIII
MÉTRICAS PARA LAS RNAS ENTRENADAS EN EL CONJUNTO DE DATOS DE CÁNCER DE MAMA EN WISCONSIN EN EL EXPERIMENTO 4

Método de entrenamiento	MFE	DTFE	MFV	DTFV	MTE	DTTE	Sobreajuste
Propagación hacia atrás	2.4469	1.0292	1889785622.0241	2139107970.8255	**0.0087**	**0.0017**	1889785619.5772
Búsqueda aleatoria	1.3938	0.144	1.3941	0.1583	0.0137	0.0027	**0.0003**
SA	2.3528	1.7736	4.8148	2.7066	0.0353	0.0025	2.462
ES	1.2497	**0.0823**	1.2569	**0.0947**	18.1007	0.7056	0.0072
GA	**0.9432**	0.1026	**0.9498**	0.1287	21.7233	1.1075	0.0066

Cuadro XXIV
MÉTRICAS PARA LAS RNAS ENTRENADAS EN EL CONJUNTO DE DATOS DEL VINO EN EL EXPERIMENTO 4

Método de entrenamiento	MFE	DTFE	MFV	DTFV	MTE	DTTE	Sobreajuste
Propagación hacia atrás	3.6279	1.4805	13.7481	6.5889	**0.0044**	**0.0018**	10.1202
Búsqueda aleatoria	2.2015	0.2258	2.195	0.2486	0.0063	**0.0018**	**0.0065**
SA	3.7982	2.7097	6.4182	3.5255	0.04	0.0028	2.62
ES	1.9239	**0.0357**	1.9501	**0.0773**	55.2781	2.8636	0.0262
GA	**1.7685**	0.0903	**1.801**	0.1244	73.4908	4.5724	0.0325

- *MTE*: Corresponde a la media del tiempo de ejecución de los entrenamientos.
- *DTTE*: Corresponde a la desviación típica del tiempo de ejecución de los entrenamientos.

Antes de entrar a valorar cada uno de los experimentos con sus resultados por separado se van a extraer unas conclusiones generales que son aplicables a la totalidad de los experimentos.

Para valorar lo bien que funciona un algoritmo de entrenamiento de una RNA, la métrica mas importante es el valor de la función fitness (o función de error según el método de entrenamiento del que estemos hablando) tanto en entrenamiento como en validación, aunque la segunda es más importante y la primera no muy significativa. La prueba del verdadero aprendizaje de la RNA se realiza en la fase de validación donde enfrentamos la RNA a datos que esta no ha procesado antes y con los cuales no se actualizan los pesos y umbrales de esta. Las métricas extraídas de esta fase nos muestran cómo va a reaccionar la RNA ante nuevos datos que no haya procesado con anterioridad; en esencia, cómo funcionará una vez que comencemos a usarla en el emplazamiento para el que ha sido diseñada. Esta métrica puede no corresponder a la realidad en algunos casos dado que en muchas ocasiones hay datos duplicados en el conjunto de datos y puede ser que en la selección de los datos de validación se incluyan datos que la RNA ya haya procesado durante el entrenamiento debido a esta duplicación. Esto no sucede aquí dado que los conjuntos de datos seleccionados están preparados para evaluar el rendimiento de las RNAs y por tanto no contienen datos duplicados.

En esta métrica es en la que la RNA evolucionada mediante GAs destaca por encima del resto de metaheurísticas y del método clásico de la propagación hacia atrás manteniendo el valor menor de entre todos los métodos en todos los experimentos con todos los conjuntos de datos. Esto se puede comprobar en las tablas desde IX hasta XXIV ambas incluidas, donde se puede ver en negrita que los valores mas bajos de la función fitness de media tanto en entrenamiento como en validación los aporta esta metaheurística como método de entrenamiento.

Estos valores de la función fitness aportados por la metaheurística se piensa que son tan buenos por las tasas de mutación y cruce empleadas, además de la variedad de los operadores de mutación y selección. Gracias a las primeras mantenemos un balance entre exploración y explotación.

Gracias a los segundos, en primer lugar, desplazamos las soluciones en el espacio de búsqueda lo suficiente y de formas suficientemente variadas y en segundo lugar, no sólo nos quedamos con los mejores resultados de la población sino que se mantienen resultados que pueden estar en otro lugar del espacio de búsqueda, una vez mas, para mantener la exploración por el espacio de búsqueda. El tamaño de la población también se considera que ayuda, sin embargo, se piensa que puede ser grande, pues en los tiempos de entrenamiento de todos los experimentos queda demostrado que es el método más lento de entre todos los propuestos.

Como se puede observar en la tabla X en la columna *MFV* el GA es, junto a la RNA evolucionada mediante ES y a la RNA entrenada mediante búsqueda aleatoria el que menor diferencia tiene entre *MFV* y *MFE*, lo que indica que no padece de sobreajuste, como si lo sufren las RNAs entrenadas mediante propagación hacia atrás y aquellas entrenadas mediante SA. Este sobreajuste por parte de la RNA entrenada mediante propagación hacia atrás se repite a lo largo de todas las tablas y conjuntos de datos para todos los experimentos, siendo el conjunto de datos en el que menos se aprecia este comportamiento el conjunto de datos de las flores del iris, y en el que esto es mas notable, el conjunto de datos del cáncer de mama en Wisconsin. Este comportamiento de la RNA entrenada mediante propagación hacia atrás es sorprendente, pues no se esperaba que se mostrara tan claramente el sobreajuste que adolece.

Se han explorado una serie de publicaciones relacionadas con el sobreajuste: [Hawkins, 2004], [Dietterich, 1995], [Lameski et al., 2015] y [Weigend, 1994]. En ninguna de ellas se ha encontrado una definición clara de cómo medir el sobreajuste, por tanto, se va a definir una métrica para cuantificar el sobreajuste y poder hacer comparaciones en base a esto. El sobreajuste queda definido por la ecuación 7.

$$Overfitting = |MFV - MFE| \tag{7}$$

Con esta ecuación podemos comparar los diferentes métodos de entrenamiento en lo que respecta al sobreajuste y se pueden hacer afirmaciones categóricas al respecto. Para mostrar esto de una forma mas sencilla en todas las tablas de todos los experimentos anteriormente expuestas se puede ver una columna con el valor del sobreajuste que presentan los métodos de entrenamiento para el conjunto de datos que en la tabla se expone.

El sobreajuste de la RNA entrenada mediante propagación hacia atrás anteriormente mencionado podría parecer que tuviera un origen en la división del conjunto de datos en el conjunto de entrenamiento y el conjunto de validación. Sin embargo, gracias a que se ha usado validación cruzada de cinco iteraciones en cada una de las 30 ejecuciones realizadas, se descarta la posibilidad de que pueda ser por la selección de datos para validación del conjunto de datos, que además es aleatoria y la semilla para su división cambia en cada ejecución del experimento al igual que cambia la semilla del generador de números aleatorios.

Algo que también es sorprendente son los buenos y consistentes resultados que arroja la RNA entrenada mediante búsqueda aleatoria, con unos valores de sobreajuste que sorprenden ya que en algunos de los conjuntos de datos son los mejores valores (más cercanos a cero). En cuanto a valores de la función de error, sus resultados son mejores que los mostrados por la RNA entrenada mediante propagación hacia atrás, mejor que el SA como metaheurística y a la par con la implementación de ES realizada, en los experimentos *1 y 3*. Respecto al SA, también sorprenden los pobres resultados que arroja esta metaheurística, con unos valores de sobreajuste en muchas ocasiones por encima de los mostrados por la RNA entrenada con propagación hacia atrás. La implementación de esta metaheurística como método de entrenamiento no ha sido la mas acertada.

Algo que se esperaba era que tanto el GA como las ES arrojaran buenos resultados dado que en pruebas previas realizadas entrenando una RNA sencilla usando el conjunto de datos XOR de la tabla I, antes de enfrentar estas dos metaheurísticas al entrenamiento de una RNA usando los conjuntos de datos del experimento, se había visto que la búsqueda del mínimo de ambas era consistente y mejor que el entrenamiento mediante propagación hacia atrás. También se esperaba que estas dos metaheurísticas fueran las más lentas, dado que además de operaciones con numPy, estas metaheurísticas usaban instrucciones del intérprete de python que son más lentas que aquellas instrucciones que numPy ejecuta por la naturaleza interpretada de este lenguaje, comportamiento que se ha visto ratificado.

Revisamos el *Experimento 1*, al cual corresponden las tablas: IX, X, XI y XII. Estas tablas están elaboradas tomando como cabecera el conjunto de datos para el cual se prueban los distintos métodos de entrenamiento. Esto se hace para comparar el rendimiento de cada uno de los métodos en un entorno común. En ellas se puede observar que el GA es el que más destaca en cuanto a valor del fitness en todos los conjuntos de datos como se ha dicho anteriormente[11] seguido por las ES y estas a su vez seguidas muy de cerca por la búsqueda aleatoria. Sin embargo, en tiempo de entrenamiento, el GA no destaca, pues es el más lento de todos los métodos de entrenamiento propuestos.

En este experimento lo que más sorprende es el buen rendimiento mostrado por la búsqueda aleatoria como método de entrenamiento, llegando, en el conjunto de datos de las enfermedades del corazón en la tabla X a mejorar el comportamiento mostrado por las ES.

Las métricas de validación arrojadas por la propagación hacia atrás también sorprenden, cuyo sobreajuste desmesurado se presenta en la tabla XI.

En este experimento se verifica todo lo mencionado en las conclusiones generales extraídas anteriormente.

Las gráficas mostradas desde la figura 11 hasta 18 ambas incluidas nos permiten ver una evolución de los valores de la función fitness en las metaheurísticas.

[11]En estos experimentos la función fitness en las metaheurísticas, y la función del error en la propagación hacia atrás son funciones que minimizar

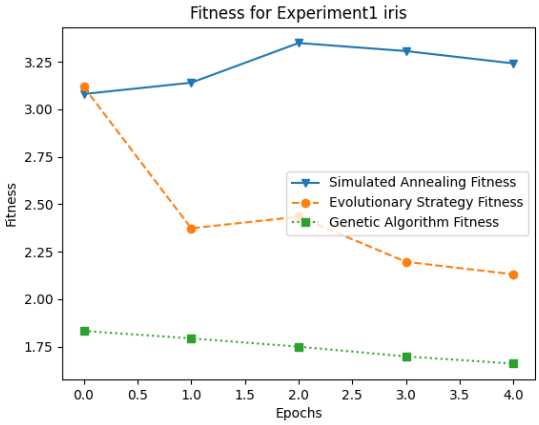

Figura 11. Representación del fitness durante el entrenamiento por épocas en para el *Experimento 1* en el conjunto de datos de las flores del iris

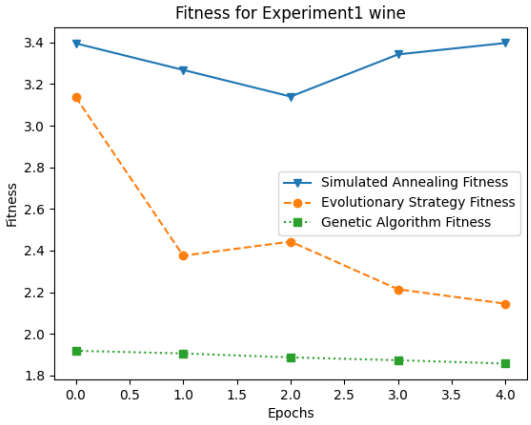

Figura 14. Representación del fitness durante el entrenamiento por épocas en para el *Experimento 1* en el conjunto de datos del vino

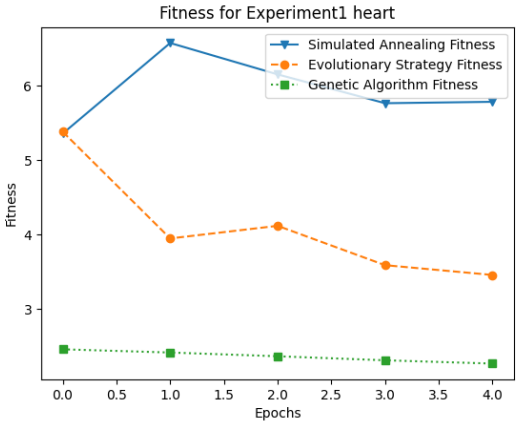

Figura 12. Representación del fitness durante el entrenamiento por épocas en para el *Experimento 1* en el conjunto de datos de las enfermedades del corazón

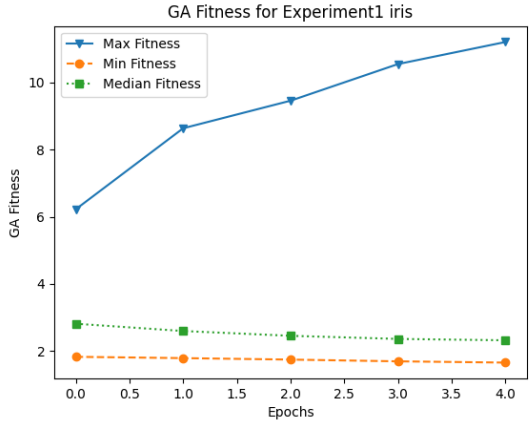

Figura 15. Representación del fitness durante el entrenamiento del GA por épocas en para el *Experimento 1* en el conjunto de datos de las flores del iris

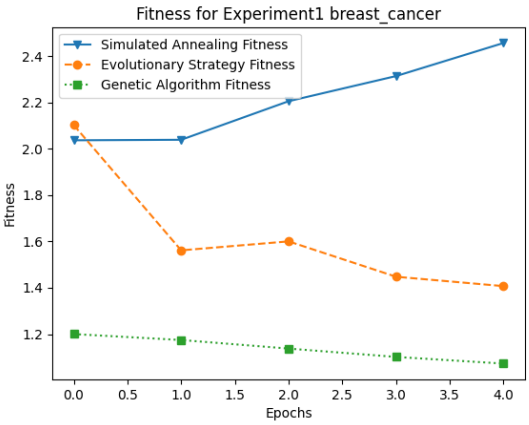

Figura 13. Representación del fitness durante el entrenamiento por épocas en para el *Experimento 1* en el conjunto de datos del cáncer de mama en Wisconsin

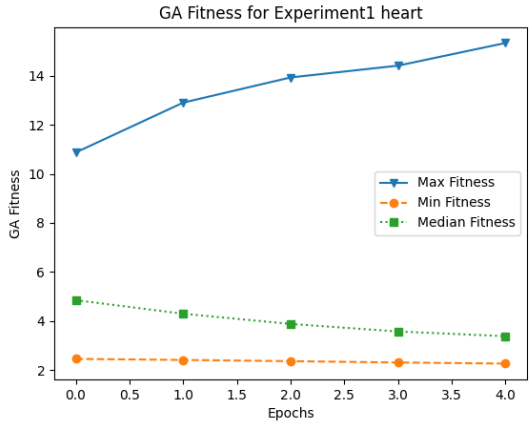

Figura 16. Representación del fitness durante el entrenamiento del GA por épocas en para el *Experimento 1* en el conjunto de datos de las enfermedades del corazón

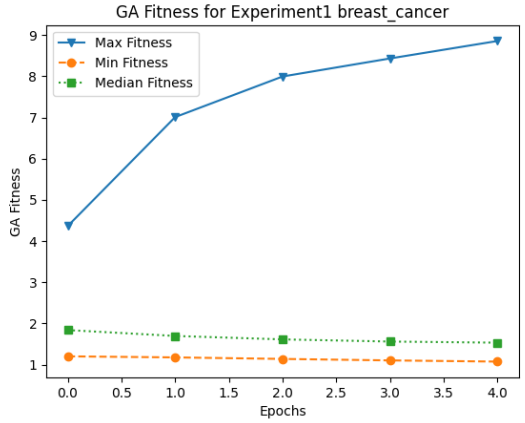

Figura 17. Representación del fitness durante el entrenamiento del GA por épocas en para el *Experimento 1* en el conjunto de datos del cáncer de mama en Wisconsin

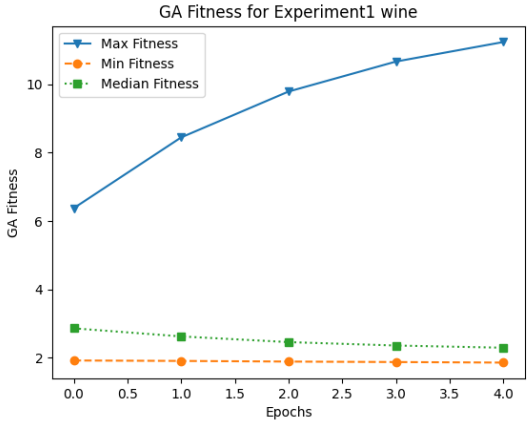

Figura 18. Representación del fitness durante el entrenamiento del GA por épocas en para el *Experimento 1* en el conjunto de datos del vino

En primer lugar en las figuras 11, 12, 13 y 14 se muestra el valor de la función fitness para las tres metaheurísticas implementadas. En ellas claramente se ve que el GA como método de entrenamiento es mejor durante toda la ejecución, comenzando en un valor menor de fitness, lo cual se justifica por simple probabilidad, recordemos que el GA trabaja con una población de 100 individuos iniciados aleatoriamente mientras que la implementación de ES y SA trabajan con una única solución aleatoria. Algo que es visible claramente es que el SA no solo escapa de mínimos locales, sino que en todos los casos termina con un valor del fitness mayor del valor con el que comienza. Esto se cree que es debido a un mal ajuste de los parámetros, pero en diversas pruebas realizadas antes de llegar a los resultados mostrados con los mismos conjuntos de datos en todas sucedía esto mismo, que los valores finales del fitness eran peores que los de partida. En cuanto a las ES se puede ver cómo avanzan hacia la dirección del mínimo, pero al tratarse de una solución única, hay poblaciones exploradoras que a lo largo de la ejecución la hacen explorar soluciones que son peores que las anteriormente exploradas, aunque se puede ver en todas las figuras que el valor final del fitness es bastante menor que el de partida.

En segundo lugar en las figuras 15, 16, 17 y 18 se muestra cómo evoluciona la población del GA a lo largo de las épocas. En las figuras se puede ver claramente como el valor máximo del fitness medio de todas las ejecuciones crece con el paso del tiempo, lo cual nos indica que la exploración de la RNA entrenada con esta metaheurística se mantiene. El valor mínimo del fitness medio de todas las ejecuciones prácticamente se reduce de forma lineal y el valor de la mediana del fitness también se reduce a lo largo del tiempo, lo que indica que tanto los operadores de cruce que explotan el espacio de búsqueda como el operador de selección funcionan correctamente y reducen el valor del fitness en la población en general, aunque el valor máximo del fitness se mantenga alto.

Se va a revisar el *Experimento 2*, cuyas tablas de resultados son: XIII, XIV, XV y XVI. Una vez más las tablas se realizan por conjunto de datos para poder comparar el rendimiento de los distintos métodos de entrenamiento en cada uno de los conjuntos de datos.

En las tablas se puede observar que, al igual que en el *Experimento 1* el GA como método de entrenamiento sobresale con respecto al resto en cuanto al valor de la función fitness medio seguido de cerca por las ES. Sin embargo, éstas son mas estables que el GA, porque como se puede ver en las tablas XIII, XV y XVI la desviación típica de las ES es menor que la que proporciona el GA.

Al igual que en el *Experimento 1* la búsqueda aleatoria sorprende por los buenos resultados que arroja, mejor que propagación hacia atrás y SA, pero en este caso ya no está tan a la par con las ES como estaba en el *Experimento 1*. Esto puede tener su explicación en la misma naturaleza del experimento ya que en este tenemos un mayor número de épocas, el doble que en el *Experimento 1* y las ES son capaces de cubrir una mayor porción del espacio de búsqueda, mientras que la búsqueda aleatoria sigue siendo aleatoria y, aun teniendo el doble de intentos, el hecho de no tener una heurística de cobertura del espacio de búsqueda hace que se separe de las métricas arrojadas por las ES.

Lo más sorprendente sin duda de este experimento son las métricas de validación arrojadas por la propagación hacia atrás, cuyo sobreajuste brilla una vez mas por su presencia y por el alto valor de este indicador, sobre todo en la tabla XV.

En este experimento se vuelve a verificar todo lo anteriormente comentado en las conclusiones generales.

Las gráficas mostradas desde la figura 19 hasta 26 ambas incluidas nos permiten ver una evolución de los valores de la función fitness en las metaheurísticas a lo largo del *Experimento 2*.

En primer lugar en las figuras 19, 20, 21 y 22 se muestra el valor de la función fitness para las tres metaheurísticas implementadas. En ellas claramente se ve que el GA como método de entrenamiento es mejor durante toda la ejecución,

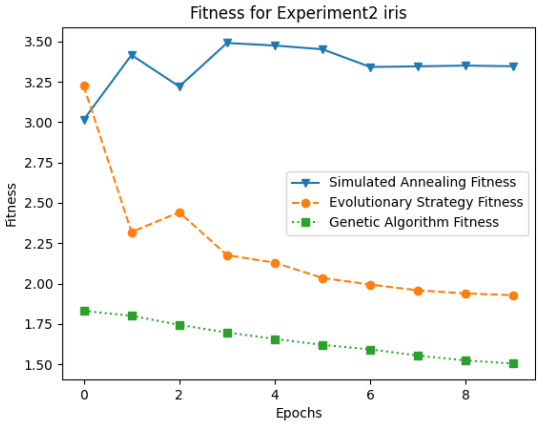

Figura 19. Representación del fitness durante el entrenamiento por épocas en para el *Experimento 2* en el conjunto de datos de las flores del iris

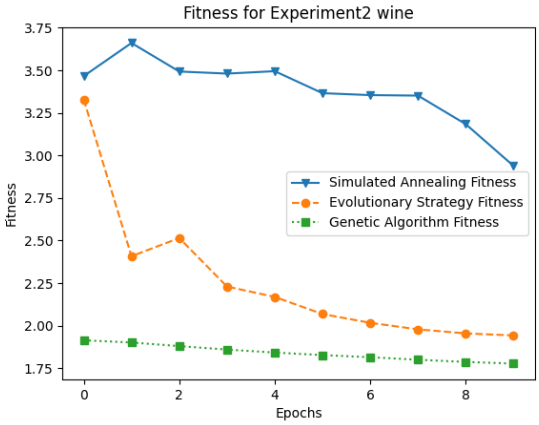

Figura 22. Representación del fitness durante el entrenamiento por épocas en para el *Experimento 2* en el conjunto de datos del vino

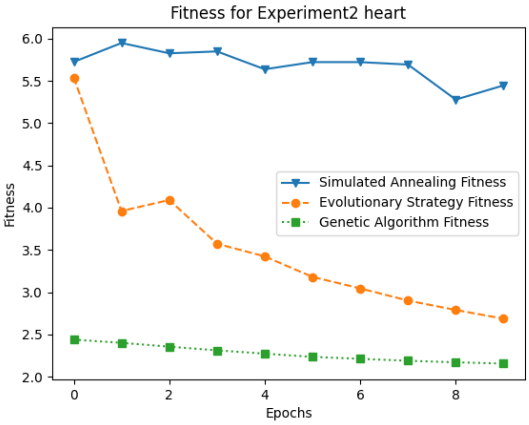

Figura 20. Representación del fitness durante el entrenamiento por épocas en para el *Experimento 2* en el conjunto de datos de las enfermedades del corazón

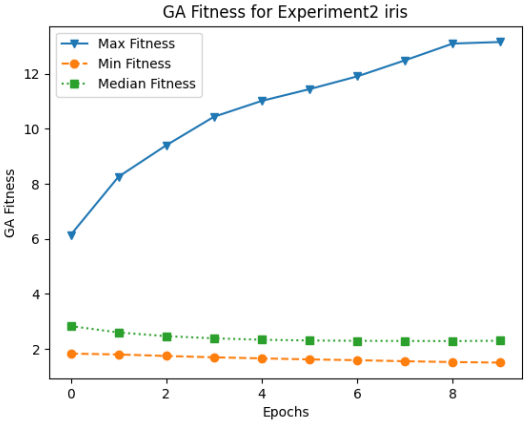

Figura 23. Representación del fitness durante el entrenamiento del GA por épocas en para el *Experimento 2* en el conjunto de datos de las flores del iris

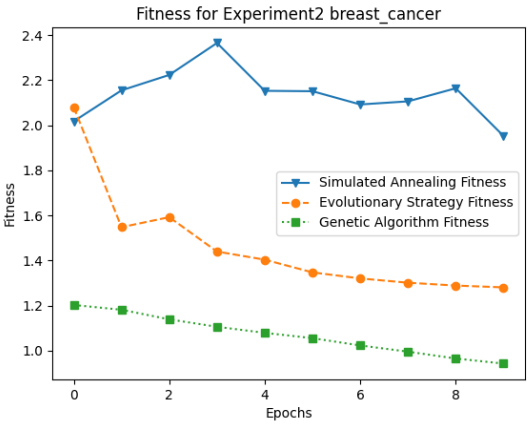

Figura 21. Representación del fitness durante el entrenamiento por épocas en para el *Experimento 2* en el conjunto de datos del cáncer de mama en Wisconsin

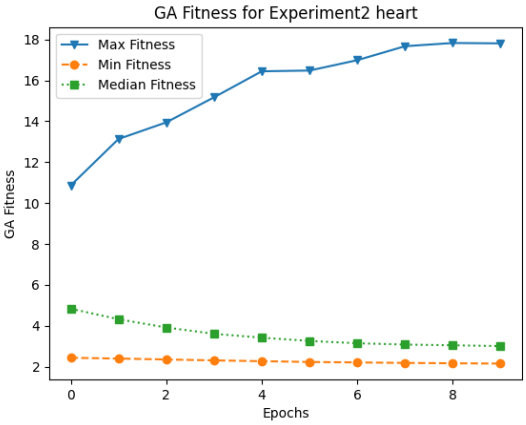

Figura 24. Representación del fitness durante el entrenamiento del GA por épocas en para el *Experimento 2* en el conjunto de datos de las enfermedades del corazón

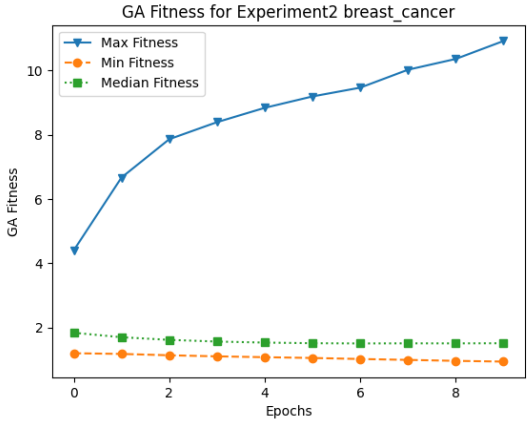

Figura 25. Representación del fitness durante el entrenamiento del GA por épocas en para el *Experimento 2* en el conjunto de datos del cáncer de mama en Wisconsin

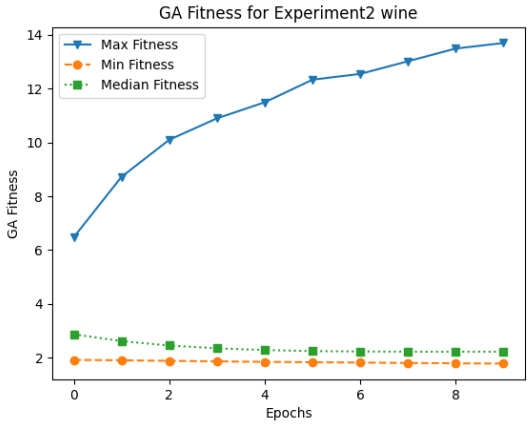

Figura 26. Representación del fitness durante el entrenamiento del GA por épocas en para el *Experimento 2* en el conjunto de datos del vino

comenzando en un valor menor de fitness por lo anteriormente comentado sobre la inicialización aleatoria. En este caso, el recocido simulado mejora la inicialización inicial a lo largo del tiempo, si bien es verdad que esto no pasa en el conjunto de datos de las flores del iris mostrado en la figura 19 Esto se cree que es debido al mayor número de épocas de las que dispone este método que al ser mayor que en el experimento anterior puede cubrir una mayor porción del espacio de búsqueda. En cuanto a las ES se puede ver cómo avanzan hacia la dirección del mínimo, al igual que en el *Experimento 1*, en este caso, con un mayor número de épocas se puede apreciar claramente el descenso en el valor de la función fitness, aunque a lo largo de la ejecución del entrenamiento se exploren soluciones peores que las anteriores.

En segundo lugar en las figuras 23, 24, 25 y 26 se muestra cómo evoluciona la población del GA a lo largo de las épocas. En las figuras se puede ver claramente como el valor máximo del fitness medio de todas las ejecuciones crece con el paso del tiempo al igual que sucedía en el *Experimento 1*, lo cual nos indica que la exploración de la RNA entrenada con esta metaheurística se mantiene. El valor mínimo del fitness medio de todas las ejecuciones prácticamente se reduce de forma lineal y el valor de la mediana del fitness también se reduce a lo largo del tiempo igual que en el experimento anterior, llegando en algunos casos a ser casi una línea paralela muy cercana al mínimo como se puede apreciar en las figuras 25 y 26

Se va a revisar el *Experimento 3*, cuyas tablas de resultados son: XVII, XVIII, XIX y XX. Las tablas realizadas se crean teniendo en cuenta los conjuntos de datos de cara a comparar el rendimiento de cada uno de los métodos de entrenamiento en un medio común.

En las tablas se puede observar, que al igual que en el *Experimento 1* el GA como método de entrenamiento destaca sobre el resto de las alternativas en lo que respecta a la media del valor de la función fitness. En este caso, además es el método de entrenamiento mas consistente pues es el que menor valor de la desviación típica respecto al valor del fitness aporta.

Siguiendo la tendencia de los anteriores experimentos, la búsqueda aleatoria sorprende por los valores del fitness que aporta y lo estable que es, además del nulo sobreajuste que presenta. Este método en este experimento vuelve a estar a la par de la implementación de ES y esto es debido a que el número de épocas sobre las que itera el algoritmo vuelve a ser el mismo que en *Experimento 1*. Sorprende también, pero esta vez para mal el sobreajuste desmesurado que presenta la propagación hacia atrás como método de entrenamiento llegando en la tabla XVIII y XIX a un valor infinito. En este experimento crece ese sobreajuste y se presenta en conjuntos de datos en los que anteriormente no aparecía porque en este experimento el número de neuronas en la capa oculta es mayor, y el sobreajuste crece con la dimensionalidad de la RNA [Bischl et al., 2012].

Al igual que en anteriores experimentos el funcionamiento de ES y GA se esperaba positivo y en este experimento también son los que mejores resultados arrojan, aunque seguidos muy de cerca por la RNA entrenada mediante búsqueda aleatoria, lo que puede indicar que el número de épocas no es suficiente, ya que como se vio en el *Experimento 2* ES comenzaba a distanciarse de la búsqueda aleatoria con un mayor número de épocas.

Los resultados de todos los métodos de entrenamiento en general son un poco peores que los arrojados por los mismos en el *Experimento 1*. Esto se cree que es debido a que, en el mismo número de épocas, con potencialmente el mismo número de ejecuciones de los operadores genéticos en el caso de metaheurísticas, se tienen que optimizar muchos mas parámetros (el doble en el caso del conjunto de datos de las flores del iris, por ejemplo) y por tanto es mas difícil alcanzar los mismos valores del fitness que anteriormente. A esto le debemos sumar que la RNA es mas compleja y por tanto hay mas posibilidades que durante el entrenamiento se sobreajuste la RNA a los datos.

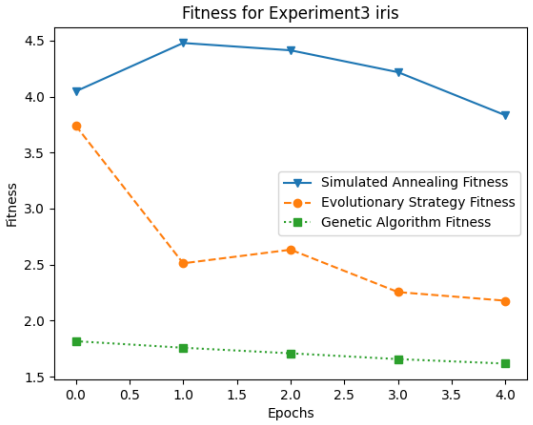

Figura 27. Representación del fitness durante el entrenamiento por épocas en para el *Experimento 3* en el conjunto de datos de las flores del iris

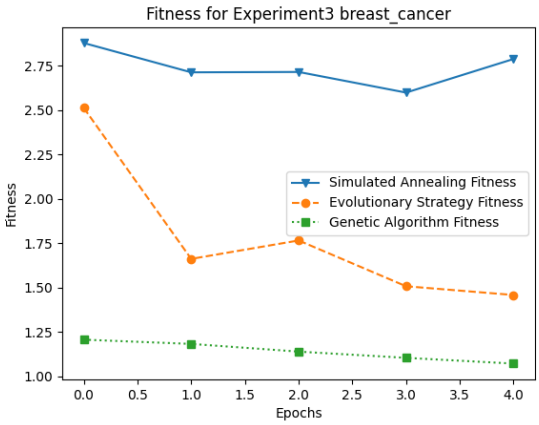

Figura 29. Representación del fitness durante el entrenamiento por épocas en para el *Experimento 3* en el conjunto de datos del cáncer de mama en Wisconsin

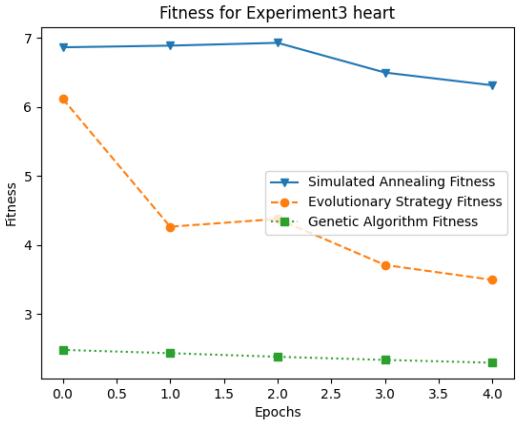

Figura 28. Representación del fitness durante el entrenamiento por épocas en para el *Experimento 3* en el conjunto de datos de las enfermedades del corazón

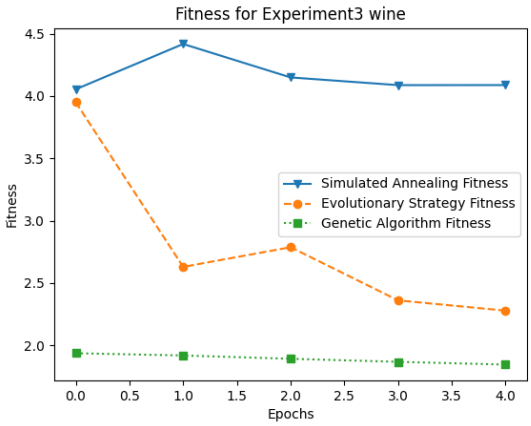

Figura 30. Representación del fitness durante el entrenamiento por épocas en para el *Experimento 3* en el conjunto de datos del vino

En este experimento se vuelve a verificar todo lo anteriormente comentado en las conclusiones generales.

Las gráficas mostradas desde la figura 27 hasta 34 ambas incluidas nos permiten ver una evolución de los valores de la función fitness en las metaheurísticas a lo largo del *Experimento 3*.

En primer lugar en las figuras 27, 28, 29 y 30 se muestra el valor de la función fitness para las tres metaheurísticas implementadas. En este caso volvemos a ver que de forma diferente al *Experimento 1*, aun teniendo el mismo número de épocas el SA es capaz de mejorar sus resultados o al menos mantener los valores del fitness de la inicialización aleatoria, lo cual es sorprendente ya que al tener un mayor número de variables que modificar se esperaría el efecto contrario, que el comportamiento mostrado en el *Experimento 1* se mostrara de forma más acentuada. Tanto ES como GA siguen en la línea de los resultados mostrados en el *Experimento 1* aunque con unos resultados un poco peores justificados por tener el mismo número de épocas, pero una cantidad mayor de

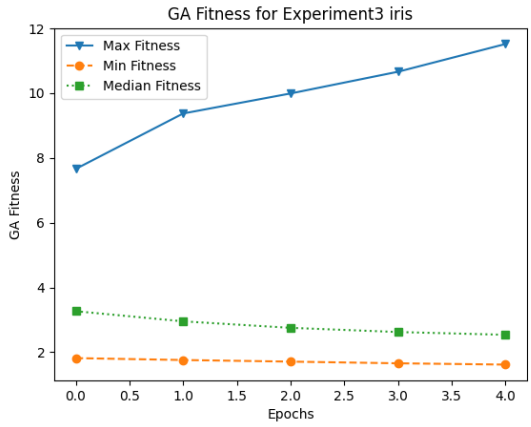

Figura 31. Representación del fitness durante el entrenamiento del GA por épocas en para el *Experimento 3* en el conjunto de datos de las flores del iris

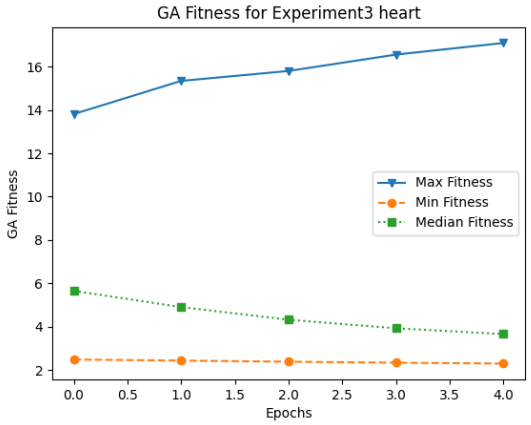

Figura 32. Representación del fitness durante el entrenamiento del GA por épocas en para el *Experimento 3* en el conjunto de datos de las enfermedades del corazón

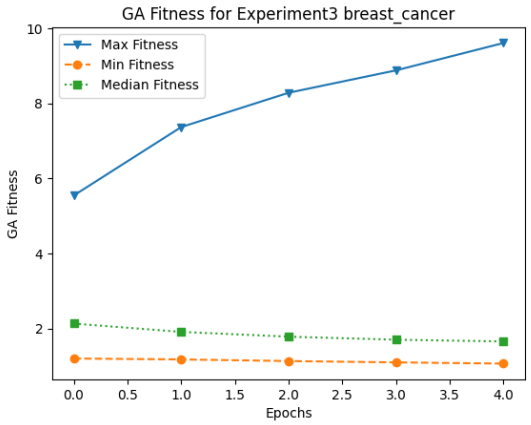

Figura 33. Representación del fitness durante el entrenamiento del GA por épocas en para el *Experimento 3* en el conjunto de datos del cáncer de mama en Wisconsin

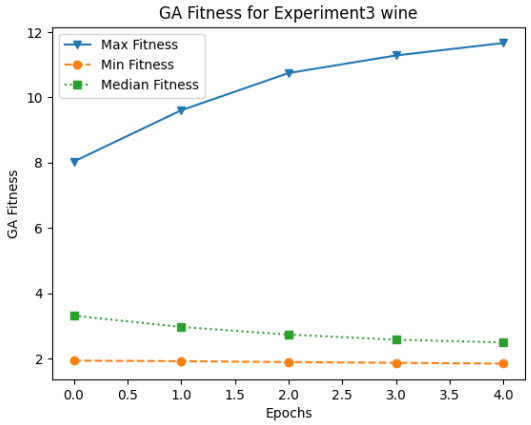

Figura 34. Representación del fitness durante el entrenamiento del GA por épocas en para el *Experimento 3* en el conjunto de datos del vino

variables modificables.

En segundo lugar en las figuras 31, 32, 33 y 34 se muestra cómo evoluciona la población del GA a lo largo de las épocas. En las figuras se puede ver claramente como el valor máximo del fitness medio de todas las ejecuciones crece con el paso del tiempo al igual que sucedía en el *Experimento 1*, lo cual nos indica que la exploración de la RNA entrenada con esta metaheurística se mantiene. El valor mínimo del fitness medio de todas las ejecuciones prácticamente se reduce de forma lineal y el valor de la mediana del fitness también se reduce a lo largo del tiempo igual que en el *Experimento 1*, pero esta vez sin alcanzar esos valores tan similares a aquellos mostrados por el valor mínimo del fitness.

Analizamos ahora el *Experimento 4*, cuyas tablas de resultados son: XXI, XXII, XXIII y XXIV. Estas tablas de resultados están realizadas por conjunto de datos para poder ver el rendimiento de los distintos métodos de entrenamiento en un medio común.

En las tablas se puede apreciar, al igual que en el *Experimento 2* que el GA como método de entrenamiento es el más efectivo de todos los propuestos alcanzando el valor medio del fitness más bajo, seguido de cerca por las ES. De igual manera, las ES vuelven a ser el método de entrenamiento más consistente alcanzando el mínimo de desviación típica en las tablas XXI, XXIII y XXIV. Ninguno de estos dos métodos son los más eficientes en tiempo, siendo los últimos de la tabla en este aspecto.

Al igual que en el resto de experimentos, la búsqueda aleatoria sorprende una vez más por el valor del fitness tan bajo que proporciona, aunque en este caso, del mismo modo que sucedía en el *Experimento 2*, al existir un mayor número de épocas para que ES cubra una porción mayor del espacio de búsqueda, la metaheurística alcanza unos valores menores de la función fitness, lo que la aleja de los valores alcanzados por la búsqueda aleatoria como método de entrenamiento. El sobreajuste superlativo que sufre la propagación hacia atrás vuelve a sorprender en este experimento, pues no se esperaba que la propagación hacia atrás mostrara un ajuste tan fino a los datos de entrenamiento. Este sobreajuste tan grande se debe principalmente a que la RNA que se emplea tiene una mayor cantidad de neuronas en la capa oculta y el sobreajuste crece con la dimensionalidad de la RNA [Bischl et al., 2012]. A esto se debe añadir que se tienen un número mayor de épocas de forma que la propagación hacia atrás puede ajustar de forma todavía más precisa los datos de entrenamiento resultando en un sobreajuste mayúsculo.

En este experimento se vuelve a verificar todo lo anteriormente comentado en las conclusiones generales.

Las gráficas mostradas desde la figura 35 hasta 42 ambas incluidas nos permiten ver una evolución de los valores de la función fitness en las metaheurísticas a lo largo del *Experimento 4*.

En primer lugar en las figuras 35, 36, 37 y 38 se muestra el valor de la función fitness para las tres metaheurísticas implementadas. En este caso volvemos a ver que de forma similar al *Experimento 2*, el SA consigue reducir el valor

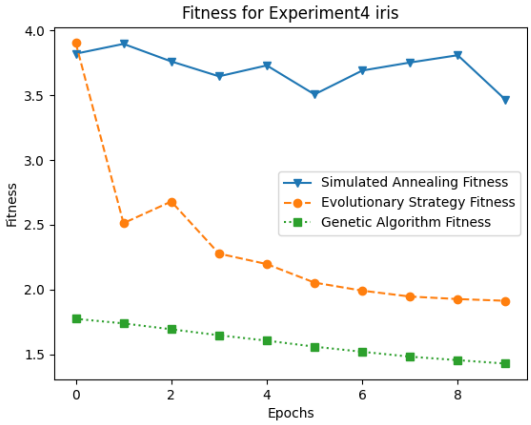

Figura 35. Representación del fitness durante el entrenamiento por épocas en para el *Experimento 4* en el conjunto de datos de las flores del iris

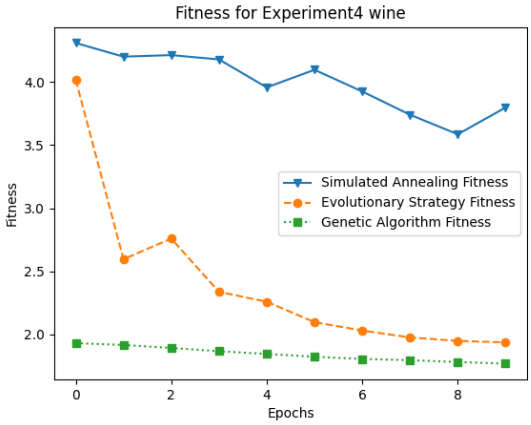

Figura 38. Representación del fitness durante el entrenamiento por épocas en para el *Experimento 4* en el conjunto de datos del vino

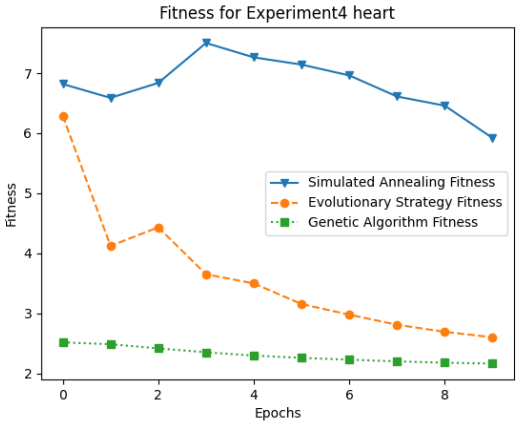

Figura 36. Representación del fitness durante el entrenamiento por épocas en para el *Experimento 4* en el conjunto de datos de las enfermedades del corazón

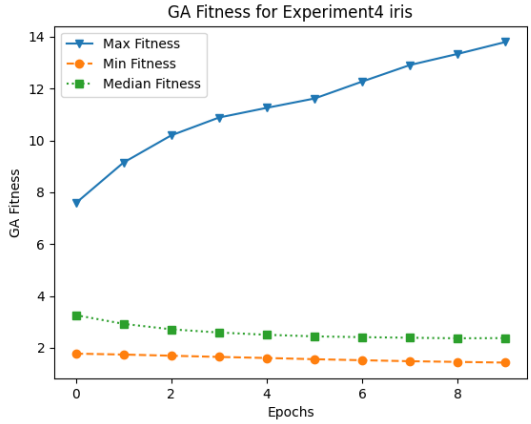

Figura 39. Representación del fitness durante el entrenamiento del GA por épocas en para el *Experimento 4* en el conjunto de datos de las flores del iris

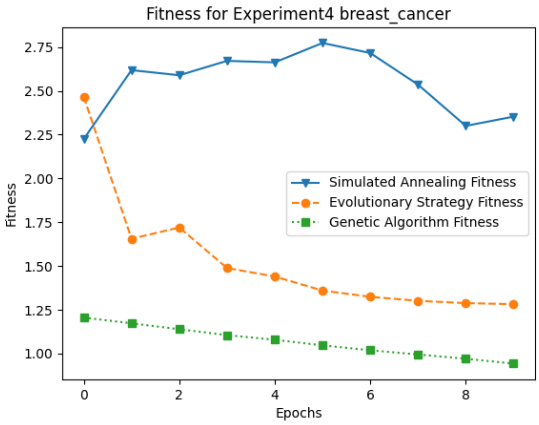

Figura 37. Representación del fitness durante el entrenamiento por épocas en para el *Experimento 4* en el conjunto de datos del cáncer de mama en Wisconsin

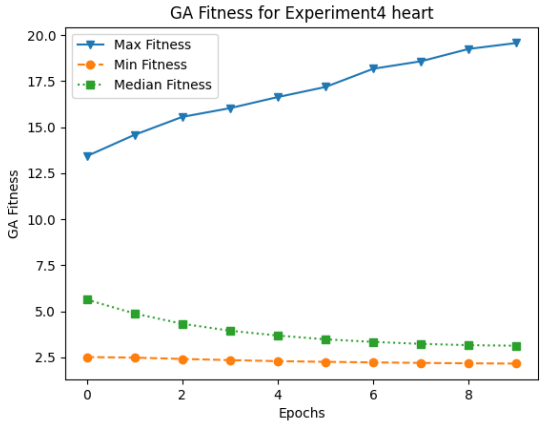

Figura 40. Representación del fitness durante el entrenamiento del GA por épocas en para el *Experimento 4* en el conjunto de datos de las enfermedades del corazón

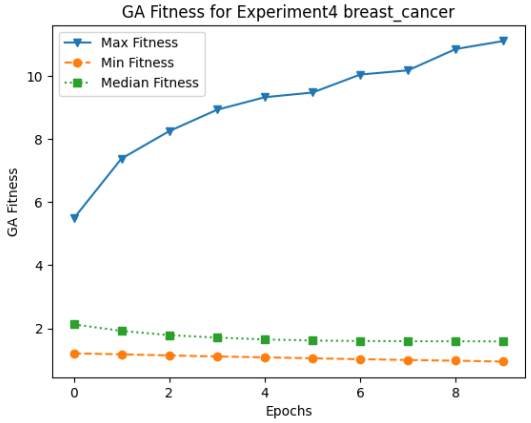

Figura 41. Representación del fitness durante el entrenamiento del GA por épocas en para el *Experimento 4* en el conjunto de datos del cáncer de mama en Wisconsin

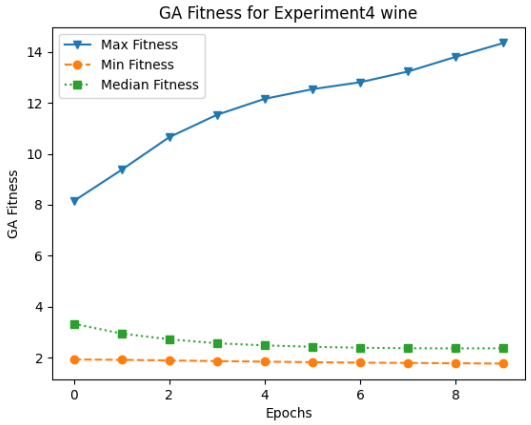

Figura 42. Representación del fitness durante el entrenamiento del GA por épocas en para el *Experimento 4* en el conjunto de datos del vino

del fitness desde su inicialización aleatoria, ayudado por un número mayor de épocas, aunque no en todos los casos se logra esto. Como se puede ver en la figura 37 la inicialización aleatoria de media es mejor que el valor final aportado por esta metaheurística de media. Tanto ES como GA siguen en la línea de los buenos resultados y se puede apreciar que la línea que traza el valor del fitness en el GA tiene una pendiente algo más pronunciada de lo que encontramos en el resto de los experimentos. Esto se cree que es debido al mayor número de pesos y umbrales que poseen tanto este experimento como el anterior ya que la inicialización aleatoria puede ser peor. Como en el caso del Experimento 3 se ha encontrado que los resultados arrojados por las metaheurísticas son un poco peores que en los experimentos anteriores y esto se cree que se debe una vez más al mayor número de pesos y umbrales de estas RNAs.

En segundo lugar en las figuras 39, 40, 41 y 42 se muestra cómo evoluciona la población del GA a lo largo de las épocas. En las figuras se puede ver claramente como el valor máximo del fitness medio de todas las ejecuciones crece con el paso del tiempo al igual que sucedía con anteriores experimentos, lo cual nos indica que la exploración de la RNA entrenada con esta metaheurística se mantiene. El valor mínimo del fitness medio de todas las ejecuciones prácticamente se reduce de forma lineal y el valor de la mediana del fitness también se reduce a lo largo del tiempo, pero no queda tan cerca de la línea del mínimo como sí pasaba en el *Experimento 2*

Cuadro XXV
VALORES DE ANOVA *F* Y *P* PRESENTADOS POR LOS CONJUNTOS DE DATOS DE ENTRENAMIENTO CON LOS DISTINTOS MÉTODOS DE ENTRENAMIENTO EN EL EXPERIMENTO 1

Parámetro	Valor
F	187.4639505421199
P	1.4572759761710294e-143

Cuadro XXVI
VALORES DE ANOVA *F* Y *P* PRESENTADOS POR LOS CONJUNTOS DE DATOS DE VALIDACIÓN CON LOS DISTINTOS MÉTODOS DE ENTRENAMIENTO EN EL EXPERIMENTO 1

Parámetro	Valor
F	580.4701749008882
P	0.0

Cuadro XXVII
VALORES DE ANOVA *F* Y *P* PRESENTADOS POR LOS CONJUNTOS DE DATOS DE ENTRENAMIENTO CON LOS DISTINTOS MÉTODOS DE ENTRENAMIENTO EN EL EXPERIMENTO 2

Parámetro	Valor
F	215.81820953853975
P	6.415687691979981e-163

Cuadro XXVIII
VALORES DE ANOVA *F* Y *P* PRESENTADOS POR LOS CONJUNTOS DE DATOS DE VALIDACIÓN CON LOS DISTINTOS MÉTODOS DE ENTRENAMIENTO EN EL EXPERIMENTO 2

Parámetro	Valor
F	622.9984552714786
P	0.0

Cuadro XXIX
VALORES DE ANOVA *F* Y *P* PRESENTADOS POR LOS CONJUNTOS DE DATOS DE ENTRENAMIENTO CON LOS DISTINTOS MÉTODOS DE ENTRENAMIENTO EN EL EXPERIMENTO 3

Parámetro	Valor
F	233.10919262323483
P	1.8958984040808326e-174

Cuadro XXX
VALORES DE ANOVA *F* Y *P* PRESENTADOS POR LOS CONJUNTOS DE DATOS DE VALIDACIÓN CON LOS DISTINTOS MÉTODOS DE ENTRENAMIENTO EN EL EXPERIMENTO 3

Parámetro	Valor
F	873.205973047988
P	0.0

Cuadro XXXI
VALORES DE ANOVA *F* Y *P* PRESENTADOS POR LOS CONJUNTOS DE DATOS DE ENTRENAMIENTO CON LOS DISTINTOS MÉTODOS DE ENTRENAMIENTO EN EL EXPERIMENTO 4

Parámetro	Valor
F	303.9073509740202
P	1.2061784354723878e-219

Cuadro XXXII
VALORES DE ANOVA *F* Y *P* PRESENTADOS POR LOS CONJUNTOS DE DATOS DE VALIDACIÓN CON LOS DISTINTOS MÉTODOS DE ENTRENAMIENTO EN EL EXPERIMENTO 4

Parámetro	Valor
F	876.0287799617927
P	0.0

Antes de pasar a las conclusiones, se ha elaborado el test estadístico *ANOVA* de un solo factor [Girden, 1992] que se usa para comparar las medias de más de dos grupos de valores. En este caso se quieren comparar los valores medios obtenidos por cada uno de los métodos de entrenamiento durante la fase de entrenamiento y validación de cara a saber si los algoritmos implementados pueden ser considerados como diferentes. Dado que el valor de este test es un valor global, para poder comparar cada par de métodos de entrenamiento y los valores aportados por estos, se ha realizado el *test de Tukey* [Abdi and Williams, 2010] para extraer los diferentes valores que nos indican si los algoritmos que proporcionan esos valores son diferentes con un intervalo de confianza concreto.

Las tablas consecutivas desde la XXV hasta la XXXII ambas incluidas incluyen los valores del test *ANOVA* para los valores de la función de error (o función fitness según corresponda) de todos los métodos de entrenamiento sobre el conjunto de entrenamiento de cada experimento (tablas XXV, XXVII, XXIX y XXXI) y sobre el conjunto de validación de cada experimento(tablas XXVI, XXVIII, XXX y XXXII).

ANOVA es el ratio entre la varianza de las medias de los grupos y el promedio de la varianza dentro de los grupos. *ANOVA* plantea dos hipótesis:

- Hipótesis nula: No hay diferencias entre las medias de los diferentes grupos.
- Hipótesis alternativa: Al menos un par de medias son significativamente distintas la una de la otra.

Estas hipótesis desde el punto de vista de este trabajo significan que los algoritmos que arrojan esos valores son los mismos y que los algoritmos que retornan esos valores son diferentes respectivamente.

Estas tablas contienen tanto el valor *F* como el valor *P* que ANOVA devuelve al ser aplicado. El valor *P* mide la evidencia de la hipótesis nula. Un valor *P* más bajo proporciona una evidencia más fuerte en pro de la hipótesis alternativa. El valor *F* es usado para calcular el valor *P* y un valor F suficientemente grande indica que el modelo estadístico es significativo. Cuanto menor sea el valor de *P* y mayor sea el valor de *F*, más seguros podemos estar de que la hipótesis alternativa se cumple.

Si analizamos los valores de las tablas consecutivas desde la XXV hasta la XXXII ambas incluidas vemos que en todas ellas el valor *P* es ridículamente pequeño, llegando a ser cero al aplicar ANOVA a los conjuntos de datos de validación en todos los experimentos. Además de esto, se puede observar que el valor de *F* es grande, aunque no se tiene una referencia de un valor significativamente grande para compararlo con éste y valorar si esos valores son suficientemente grandes. Sin embargo, aunque estos valores ayudan, estamos comparando los cinco métodos de entrenamiento entre ellos, con lo que no sabemos hasta que punto cada algoritmo es único respecto al resto. Para este propósito se ha aplicado el *test de Tukey*.

Las tablas consecutivas desde la XXXIII hasta la XL ambas incluidas muestran los valores del *test de Tukey* para cada experimento con los conjuntos de datos de entrenamiento (tablas XXXIII, XXXIV, XXXV y XXXVI) y validación (tablas XXXVII, XXXVIII, XXXIX y XL). Cada una de esas tablas contiene lo siguiente:

- *group1*: Primer método de entrenamiento que se está comparando.
- *group2*: Segundo método de entrenamiento que se está comparando.
- *lower*: Valor inferior del intervalo de confianza.
- *upper*: Valor superior del intervalo de confianza.
- *meandiff*: $(lower + upper)/2$.
- *p-adj*: Valor *P* ajustado para comparaciones múltiples.
- *reject*: Valor booleano que nos indica si se rechaza o no la hipótesis nula.

Los valores que aparecen tanto en las columnas *group1* como en las columnnas *group2* se corresponden con:

- *backprop_training*: Entrenamiento con propagación hacia atrás.
- *randomnn_training*: Entrenamiento con búsqueda aleatoria.
- *es_training*: Entrenamiento con ES.
- *ga_training*: Entrenamiento con GA.
- *sa_training*: Entrenamiento con SA.
- *backprop_testing*: Validación con propagación hacia atrás.
- *randomnn_testing*: Validación con búsqueda aleatoria.
- *es_testing*: Validación con ES.
- *ga_testing*: Validación con GA.
- *sa_testing*: Validación con SA.

De todas las medidas que contienen las tablas, las que más nos interesan son, en primer lugar *reject* ya que es el indicativo de si se puede o no rechazar la hipótesis nula, y relacionado con él *p-adj* pues nos va a dar el valor *P* para cada dos comparaciones.

Los valores de *P* de ANOVA son todos menores que el intervalo de confianza para el que se quiere demostrar que las distribuciones de medias son diferentes (0.05) con lo que se concluye que hay diferencias significativas entre los valores medios que arrojan los distintos métodos de entrenamiento, es

Cuadro XXXIII
COMPARACIÓN POR PARES DE LOS DISTINTOS MÉTODOS DE ENTRENAMIENTO SOBRE LOS DATOS DE ENTRENAMIENTO USANDO EL TEST DE TUKEY EN EL EXPERIMENTO 1

group1	group2	meandiff	p-adj	lower	upper	reject
backprop_training	es_training	-1.0375	**0.001**	-1.2709	-0.8041	**True**
backprop_training	ga_training	-1.5925	**0.001**	-1.8258	-1.3591	**True**
backprop_training	random_training	-1.0368	**0.001**	-1.2702	-0.8034	**True**
backprop_training	sa_training	0.413	**0.001**	0.1797	0.6464	**True**
es_training	ga_training	-0.555	**0.001**	-0.7883	-0.3216	**True**
es_training	random_training	0.0007	0.9	-0.2326	0.2341	False
es_training	sa_training	1.4505	**0.001**	1.2172	1.6839	**True**
ga_training	random_training	0.5557	**0.001**	0.3223	0.789	**True**
ga_training	sa_training	2.0055	**0.001**	1.7721	2.2388	**True**
random_training	sa_training	1.4498	**0.001**	1.2165	1.6832	**True**

Cuadro XXXIV
COMPARACIÓN POR PARES DE LOS DISTINTOS MÉTODOS DE ENTRENAMIENTO SOBRE LOS DATOS DE ENTRENAMIENTO USANDO EL TEST DE TUKEY EN EL EXPERIMENTO 2

group1	group2	meandiff	p-adj	lower	upper	reject
basic_training	es_training	-1.4475	**0.001**	-1.6696	-1.2254	**True**
basic_training	ga_training	-1.7211	**0.001**	-1.9431	-1.499	**True**
basic_training	randomnn_training	-1.1771	**0.001**	-1.3991	-0.955	**True**
basic_training	sa_training	0.1052	0.6716	-0.1169	0.3273	False
es_training	ga_training	-0.2736	0.007	-0.4957	-0.0515	**True**
es_training	randomnn_training	0.2704	0.008	0.0483	0.4925	**True**
es_training	sa_training	1.5527	**0.001**	1.3306	1.7748	**True**
ga_training	randomnn_training	0.544	**0.001**	0.3219	0.7661	**True**
ga_training	sa_training	1.8263	**0.001**	1.6042	2.0483	**True**
randomnn_training	sa_training	1.2823	**0.001**	1.0602	1.5043	**True**

Cuadro XXXV
COMPARACIÓN POR PARES DE LOS DISTINTOS MÉTODOS DE ENTRENAMIENTO SOBRE LOS DATOS DE ENTRENAMIENTO USANDO EL TEST DE TUKEY EN EL EXPERIMENTO 3

group1	group2	meandiff	p-adj	lower	upper	reject
basic_training	es_training	-1.5322	**0.001**	-1.8074	-1.2569	**True**
basic_training	ga_training	-2.189	**0.001**	-2.4642	-1.9137	**True**
basic_training	randomnn_training	-1.4445	**0.001**	-1.7198	-1.1693	**True**
basic_training	sa_training	0.3615	0.0032	0.0863	0.6367	**True**
es_training	ga_training	-0.6568	**0.001**	-0.932	-0.3816	**True**
es_training	randomnn_training	0.0876	0.9	-0.1876	0.3628	False
es_training	sa_training	1.8937	**0.001**	1.6184	2.1689	**True**
ga_training	randomnn_training	0.7444	**0.001**	0.4692	1.0196	**True**
ga_training	sa_training	2.5505	**0.001**	2.2752	2.8257	**True**
randomnn_training	sa_training	1.806	**0.001**	1.5308	2.0813	**True**

Cuadro XXXVI
COMPARACIÓN POR PARES DE LOS DISTINTOS MÉTODOS DE ENTRENAMIENTO SOBRE LOS DATOS DE ENTRENAMIENTO USANDO EL TEST DE TUKEY EN EL EXPERIMENTO 4

group1	group2	meandiff	p-adj	lower	upper	reject
basic_training	es_training	-2.0622	**0.001**	-2.3118	-1.8127	**True**
basic_training	ga_training	-2.3422	**0.001**	-2.5917	-2.0926	**True**
basic_training	randomnn_training	-1.6839	**0.001**	-1.9334	-1.4343	**True**
basic_training	sa_training	-0.0331	0.9	-0.2826	0.2165	False
es_training	ga_training	-0.2799	0.0189	-0.5295	-0.0304	**True**
es_training	randomnn_training	0.3784	**0.001**	0.1288	0.6279	**True**
es_training	sa_training	2.0292	**0.001**	1.7796	2.2787	**True**
ga_training	randomnn_training	0.6583	**0.001**	0.4087	0.9078	**True**
ga_training	sa_training	2.3091	**0.001**	2.0596	2.5587	**True**
randomnn_training	sa_training	1.6508	**0.001**	1.4013	1.9004	**True**

Cuadro XXXVII
COMPARACIÓN POR PARES DE LOS DISTINTOS MÉTODOS DE ENTRENAMIENTO SOBRE LOS DATOS DE VALIDACIÓN USANDO EL TEST DE TUKEY EN EL EXPERIMENTO 1

group1	group2	meandiff	p-adj	lower	upper	reject
basic_testing	es_testing	-6.3069	**0.001**	-6.7786	-5.8352	**True**
basic_testing	ga_testing	-6.8605	**0.001**	-7.3322	-6.3888	**True**
basic_testing	randomnn_testing	-6.3068	**0.001**	-6.7785	-5.8351	**True**
basic_testing	sa_testing	-2.9755	**0.001**	-3.4472	-2.5038	**True**
es_testing	ga_testing	-0.5536	0.012	-1.0253	-0.0819	**True**
es_testing	randomnn_testing	0.0001	0.9	-0.4717	0.4718	False
es_testing	sa_testing	3.3313	**0.001**	2.8596	3.8031	**True**
ga_testing	randomnn_testing	0.5537	0.012	0.0819	1.0254	**True**
ga_testing	sa_testing	3.885	**0.001**	3.4132	4.3567	**True**
randomnn_testing	sa_testing	3.3313	**0.001**	2.8596	3.803	**True**

Cuadro XXXVIII
COMPARACIÓN POR PARES DE LOS DISTINTOS MÉTODOS DE ENTRENAMIENTO SOBRE LOS DATOS DE VALIDACIÓN USANDO EL TEST DE TUKEY EN EL EXPERIMENTO 2

group1	group2	meandiff	p-adj	lower	upper	reject
basic_testing	es_testing	-6.6455	**0.001**	-7.1242	-6.1668	**True**
basic_testing	ga_testing	-6.9094	**0.001**	-7.3881	-6.4307	**True**
basic_testing	randomnn_testing	-6.3817	**0.001**	-6.8604	-5.903	**True**
basic_testing	sa_testing	-2.4684	**0.001**	-2.9471	-1.9897	**True**
es_testing	ga_testing	-0.2639	0.5515	-0.7426	0.2148	False
es_testing	randomnn_testing	0.2638	0.5519	-0.2149	0.7425	False
es_testing	sa_testing	4.177	**0.001**	3.6983	4.6557	**True**
ga_testing	randomnn_testing	0.5277	0.0222	0.049	1.0064	**True**
ga_testing	sa_testing	4.4409	**0.001**	3.9622	4.9196	**True**
randomnn_testing	sa_testing	3.9133	**0.001**	3.4346	4.392	**True**

Cuadro XXXIX
COMPARACIÓN POR PARES DE LOS DISTINTOS MÉTODOS DE ENTRENAMIENTO SOBRE LOS DATOS DE VALIDACIÓN USANDO EL TEST DE TUKEY EN EL EXPERIMENTO 3

group1	group2	meandiff	p-adj	lower	upper	reject
basic_testing	es_testing	-13.2253	**0.001**	-13.9878	-12.4628	**True**
basic_testing	ga_testing	-13.9066	**0.001**	-14.6691	-13.1441	**True**
basic_testing	randomnn_testing	-13.1308	**0.001**	-13.8933	-12.3683	**True**
basic_testing	sa_testing	-9.0506	**0.001**	-9.8131	-8.2881	**True**
es_testing	ga_testing	-0.6813	0.1054	-1.4438	0.0812	False
es_testing	randomnn_testing	0.0946	0.9	-0.6679	0.8571	False
es_testing	sa_testing	4.1747	**0.001**	3.4122	4.9372	**True**
ga_testing	randomnn_testing	0.7759	0.0438	0.0134	1.5384	**True**
ga_testing	sa_testing	4.856	**0.001**	4.0935	5.6185	**True**
randomnn_testing	sa_testing	4.0801	**0.001**	3.3176	4.8426	**True**

Cuadro XL
COMPARACIÓN POR PARES DE LOS DISTINTOS MÉTODOS DE ENTRENAMIENTO SOBRE LOS DATOS DE VALIDACIÓN USANDO EL TEST DE TUKEY EN EL EXPERIMENTO 4

group1	group2	meandiff	p-adj	lower	upper	reject
basic_testing	es_testing	-13.7741	**0.001**	-14.5566	-12.9915	**True**
basic_testing	ga_testing	-14.0446	**0.001**	-14.8272	-13.262	**True**
basic_testing	randomnn_testing	-13.3744	**0.001**	-14.1569	-12.5918	**True**
basic_testing	sa_testing	-8.5763	**0.001**	-9.3589	-7.7937	**True**
es_testing	ga_testing	-0.2705	0.8698	-1.0531	0.512	False
es_testing	randomnn_testing	0.3997	0.6142	-0.3829	1.1823	False
es_testing	sa_testing	5.1978	**0.001**	4.4152	5.9803	**True**
ga_testing	randomnn_testing	0.6702	0.1331	-0.1123	1.4528	False
ga_testing	sa_testing	5.4683	**0.001**	4.6857	6.2509	**True**
randomnn_testing	sa_testing	4.7981	**0.001**	4.0155	5.5806	**True**

por esto que se realiza el *test de Tukey* para comparar cada par de métodos de entrenamiento para comprobar si para todos ellos se cumplen las diferencias significativas. Las hipótesis nula y alternativa se aceptan o rechazan con el mismo nivel de confianza que el test ANOVA (0.05).

Si analizamos las tablas del *Experimento 1* podemos ver que, según este test, exceptuando el par de métodos (ES-búsqueda aleatoria) el resto muestran diferencias significativas, lo que significa que ES y búsqueda aleatoria tanto en el conjunto de datos de entrenamiento como en el conjunto de datos de validación son el mismo algoritmo.

Si revisamos las tablas de datos del *Experimento 1* y nos centramos en los datos referentes a ES y a búsqueda aleatoria, no sorprende que en este test se acepte la hipótesis nula pues no hay una diferencia significativa entre los valores de estas tablas para estos métodos de entrenamiento ES y búsqueda aleatoria, aunque sí sorprende el alto valor de P con el que se relacionan ($P = 0{,}9$).

Si analizamos las tablas del *Experimento 2* podemos ver que según este test, exceptuando el par de métodos (SA-propagación hacia atrás) en el conjunto de datos de entrenamiento y los pares (ES-GA) y (ES-búsqueda aleatoria) en el conjunto de validación el resto muestran diferencia significativas, lo que significa que para el conjunto de datos de entrenamiento SA y propagación hacia atrás se comportan de forma similar y para el conjunto de validación (ES-GA) y (ES-búsqueda aleatoria) muestran un comportamiento similar.

En los resultados de este test para el *Experimento 2*, no resulta sorprendente que se relacione en el conjunto de datos de entrenamiento a los métodos SA y propagación hacia atrás y que se acepte la hipótesis nula en consecuencia, ya que ambos presentan unas medias y desviaciones típicas muy similares. Además de esto, tampoco es una sorpresa que en el conjunto de datos de validación se acepte la hipótesis nula con un valor de $P = 0{,}55$ para el par (ES-GA) porque como se comentaba en los resultados del *Experimento 2*, gracias a este mayor número de épocas las ES se acercaban a los mejores resultados, resultados que están monopolizados durante todos los experimentos por los GAs como método de entrenamiento. Sin embargo, sorprende que en este mismo conjunto de datos de validación se acepte la hipótesis nula con un valor $P = 0{,}55$ para el par (ES-búsqueda aleatoria), ya que como se comentaba en los resultados del experimento, gracias a este mayor número de épocas, ES se separaba de la búsqueda aleatoria en resultados, como consecuencia P baja, pero siguen sin considerarse algoritmos diferentes.

Si analizamos las tablas del *Experimento 3* podemos ver que según este test, al igual que en el *Experimento 1*, exceptuando el par de métodos (ES-búsqueda aleatoria) en el conjunto de datos de entrenamiento y los pares (ES-GA) y (ES-búsqueda aleatoria) en el conjunto de validación el resto muestran diferencia significativas, lo que significa que para el conjunto de datos de entrenamiento, al igual que sucedía en el *Experimento 1* ES y búsqueda aleatoria se comportan de forma similar y para el conjunto de validación (ES-GA) y (ES-búsqueda aleatoria) muestran un comportamiento similar.

De forma similar a lo que ocurría en el *Experimento 1*, no resulta sorprendente que se relacione en el conjunto de datos de entrenamiento al par (ES-búsqueda aleatoria) ya que en este experimento se vuelve a tener un menor número de épocas y se cree que ES no cubre una porción suficiente del espacio de búsqueda como se comentó anteriormente en el análisis de los datos de las tablas extraídas de este experimento. Se cree que es debido a esto por lo que el valor *P* es tan alto $P = 0{,}9$. Lo que sí resulta sorprendente es que para el conjunto de datos de validación de este experimento se relacione a ES con GA, aunque el valor de *P* es relativamente bajo $P = 0{,}1$ aunque suficiente para descartar la hipótesis nula.

Si analizamos las tablas del *Experimento 4* podemos ver que según este test, al igual que en el *Experimento 2*, exceptuando el par de métodos (SA-propagación hacia atrás) en el conjunto de datos de entrenamiento y los pares (ES-GA), (ES-búsqueda aleatoria) y (GA-búsqueda aleatoria) en el conjunto de validación el resto muestran diferencia significativas, lo que significa que para el conjunto de datos de entrenamiento, al igual que sucedía en el *Experimento 2* SA y propagación hacia atrás se comportan de forma similar y para el conjunto de validación ES, GA y búsqueda aleatoria muestran un comportamiento similar.

En los resultados de este test para el *Experimento 4* no resulta sorprendente el hecho de que se relacionen SA y propagación hacia atrás pues en primer lugar aportan medias muy similares, aun siendo métodos que operan de forma completamente distinta, y además en el *Experimento 2* ya aparecía este comportamiento. En lo que respecta al conjunto de datos de validación, no es sorprendente que ES se relacione con el GA ya que como se dijo anteriormente tanto en este experimento como en el *Experimento 2* ES mejoraba el valor del fitness mostrado acercándose más a GA. Lo que sí resulta sorprendente es la relación que ambas metaheurísticas presentan, según este test, con la búsqueda aleatoria ya que si observamos las tablas pertenecientes al *Experimento 4* se aprecian bastantes diferencias en los valores de MFV para todos los conjuntos de datos, sobre todo en el GA. Se cree que es por esto por lo que el valor de *P* para el par (GA-búsqueda aleatoria) es menor que para el par (ES-búsqueda aleatoria)

VI. Conclusiones y trabajos futuros

En esta sección se va a dar respuesta a la hipótesis planteada al principio de este trabajo y se van a proponer mejoras para el trabajo realizado. Esta sección se corresponde con la fase de falsabilidad del método científico.

El interés de esta sección reside en conocer si la hipótesis planteada es cierta o no además de conocer las líneas futuras de investigación posibles relacionadas con el presente trabajo.

La relación que esta sección guarda con el resto del trabajo es tremendamente importante ya que se valora la hipótesis planteada, se valora el cumplimiento de los objetivos y se proponen algunas líneas de trabajos futuros que pueden ser interesantes.

La hipótesis planteada al principio del trabajo rezaba lo siguiente:

las metaheurísticas mejoran el entrenamiento de las redes neuronales artificiales respecto al método tradicional de propagación hacia atrás.

Para dar respuesta a esta hipótesis se plantean las siguientes afirmaciones:

En primer lugar, se considera que las metaheurísticas sí mejoran el entrenamiento de las RNAs respecto al método tradicional. Desde el punto de vista de los valores de la función del error (función fitness en el caso de las metaheurísticas) tanto los GAs como las ES presentan mejores valores de esta función que el entrenamiento mediante propagación hacia atrás. También obtienen mejores resultados de la desviación típica en este aspecto.

En segundo lugar, al principio del trabajo se proponía el problema del sobreajuste que acusan las RNAs entrenadas con propagación hacia atrás. Según los valores que encontramos en las tablas desde la IX hasta XXIV ambas incluidas, tanto los GAs como las ES no presentan sobreajuste en ninguno de los experimentos realizados como sí lo muestra el método clásico de la propagación hacia atrás, con lo cual en este aspecto también se considera que las metaheurísticas sí mejoran el entrenamiento de las RNAs respecto al método de propagación hacia atrás.

En tercer lugar, al principio del trabajo se proponía el problema del estancamiento en mínimos locales por parte de las RNAs entrenadas con propagación hacia atrás. Este problema se puede ver reflejado en los datos de los experimentos, en las tablas consecutivas desde la IX hasta XXIV ambas incluidas. En los datos, se puede ver que el valor de la función del error proporcionado por la propagación hacia atrás queda en un punto más alto que el proporcionado por la búsqueda aleatoria, ES y GA.

En cuarto lugar, al principio del trabajo se mencionaba el problema de la escalabilidad de la propagación hacia atrás como método de entrenamiento. Si nos centramos en las tablas consecutivas desde la IX hasta XXIV ambas incluidas, cuando añadimos más neuronas en la capa oculta, aumentando en consecuencia la dimensionalidad de la RNA, no se aprecia un deterioro mayor en los valores de la función del error proporcionados por la RNA entrenada con propagación hacia atrás que el que se aprecia en cualquiera de los otros métodos de entrenamiento. En lo que sí repercute este aumento de dimensionalidad es en el sobreajuste. Sin embargo, esto no quiere decir que al utilizar este método de entrenamiento con una RNA en la que se tengan varias capas ocultas con varias neuronas cada una de ellas este problema no sea notable.

En quinto lugar, la implementación de SA sí sufre de los problemas de sobreajuste y arroja peores valores de la función fitness con respecto a las otras dos metaheurísticas. En este aspecto se considera que la implementación de las metaheurísticas no mejora el entrenamiento de las RNAs respecto al método tradicional.

En sexto lugar, tanto GA como ES son métodos de entrenamiento mucho mas lentos en tiempo que la implementación de la propagación hacia atrás. En este aspecto se considera que la implementación de las metaheurísticas no mejora el entrenamiento de las RNAs respecto al método tradicional.

En lo que respecta al cumplimiento de los objetivos planteados al inicio del trabajo se aportan las siguientes afirmaciones:

En primer lugar el concepto de neuroevolución se ha aprendido a lo largo de todo el trabajo, con especial énfasis en la sección de propuestas técnicas, en las subsecciones de redes neuronales evolucionadas con GA, ES y SA.

En segundo lugar, al realizar la implementación de la red neuronal entrenada con propagación hacia atrás se ha comprendido como funciona esta. También se encuentran detalles de esta implementación y de cómo funciona en la subsección *Redes neuronales artificiales* de la sección de propuestas técnicas.

En tercer lugar en lo que respecta a GAs, tanto en la subsección asociada a ellos en las propuestas técnicas como las conclusiones extraídas anteriormente en esta misma sección, se considera que se han cumplido los objetivos 3 y 4.

En cuarto lugar en lo que respecta a ES, tanto en la subsección asociada a ellas en las propuestas técnicas como las conclusiones extraídas anteriormente en esta misma sección, se considera que se han cumplido los objetivos 5 y 6.

En cuarto lugar en lo que respecta a SA, tanto en la subsección asociada a este método en las propuestas técnicas como las conclusiones extraídas anteriormente en esta misma sección, se considera que se han cumplido los objetivos 7 y 8.

Por último, gracias al análisis estadístico de la sección *Realización del experimento* se considera que el objetivo 9 está cumplido.

Para terminar esta sección se plantean algunas propuestas de investigación y trabajos futuros.

En primer lugar, como se ha visto en la sección *Realización del experimento* los tiempos de entrenamiento proporcionados por las redes neuronales evolucionadas mediante metaheurísticas, con especial atención a aquellos proporcionados por ES y GA, están muy lejos de aquellos proporcionados por la propagación hacia atrás. Una propuesta muy interesante consiste en hacer esos algoritmos paralelos [Alba, 2005] [Alba et al., 2013] y usar un procesador de gráficos para la ejecución de su entrenamiento, ya que estos procesadores ya son usados hoy para acelerar entrenamientos de RNAs [Song et al., 2018].

En segundo lugar, como se ha visto en la sección *El experimento*, se tiene que realizar un preprocesamiento de los datos alfanuméricos dado que la implementación realizada de la RNA no acepta datos que no sean numéricos. Una propuesta interesante es mejorar esa implementación para que este tipo de dato sea aceptado por la red neuronal.

En tercer lugar, como se mencionó en la introducción, una solución posible al sobreajuste de las RNAs entrenadas con propagación hacia atrás es usar una técnica llamada *dropout* [Ba and Frey, 2013] que consiste en anular algunas de las conexiones entre las capas de neuronas. Una propuesta

interesante es implementar esta solución, pero no sólo para las RNAs entrenadas con propagación hacia atrás, sino también para aquellas evolucionadas mediante metaheurísticas.

En cuarto lugar, en la sección *Realización del experimento* se ha trabajado con redes neuronales que contienen tres capas de neuronas, la capa de entrada, la capa oculta y la capa de salida. Una propuesta interesante es realizar estos experimentos con redes neuronales más complejas y con un mayor número de capas y comprobar si las metaheurísticas siguen arrojando los mejores valores en cuanto a la función fitness.

En quinto y ultimo lugar, en la sección *El experimento* se muestran los hiperparámetros utilizados en el entrenamiento de las distintas implementaciones de las RNAs. Sin embargo, estos hiperparámetros que se usan no tienen por qué ser los mejores para cada una de las RNAs presentadas. Se propone implementar una optimización de los hiperparámetros automática [Li et al., 2017] [Loshchilov and Hutter, 2016] [Diaz et al., 2017].

AGRADECIMIENTOS

A mi madre, a José Manuel y a Ismael por estar ahí siempre. A Enrique Alba por todo lo aprendido. A Álvaro por enseñarme a confiar en mi mismo.

REFERENCIAS

[Aarts and Korst, 1988] Aarts, E. and Korst, J. (1988). Simulated annealing and boltzmann machines.

[Abdi and Williams, 2010] Abdi, H. and Williams, L. J. (2010). Newman-keuls test and tukey test. *Encyclopedia of Research Design. Thousand Oaks, CA: Sage*, pages 1–11.

[Alba, 2005] Alba, E. (2005). *Parallel metaheuristics: a new class of algorithms*, volume 47. John Wiley & Sons.

[Alba et al., 1993] Alba, E., Aldana, J., and Troya, J. (1993). Genetic algorithms as heuristics for optimizing ann design. In *Artificial Neural Nets and Genetic Algorithms*, pages 683–690. Springer.

[Alba et al., 2013] Alba, E., Luque, G., and Nesmachnow, S. (2013). Parallel metaheuristics: recent advances and new trends. *International Transactions in Operational Research*, 20(1):1–48.

[Amato et al., 2013] Amato, F., López, A., Peña-Méndez, E. M., Vaňhara, P., Hampl, A., and Havel, J. (2013). Artificial neural networks in medical diagnosis.

[Ba and Frey, 2013] Ba, J. and Frey, B. (2013). Adaptive dropout for training deep neural networks. In *Advances in neural information processing systems*, pages 3084–3092.

[Back, 1996] Back, T. (1996). *Evolutionary algorithms in theory and practice: evolution strategies, evolutionary programming, genetic algorithms*. Oxford university press.

[Baldi, 2012] Baldi, P. (2012). Autoencoders, unsupervised learning, and deep architectures. In *Proceedings of ICML workshop on unsupervised and transfer learning*, pages 37–49.

[Bi et al., 2005] Bi, W., Wang, X., Tang, Z., and Tamura, H. (2005). Avoiding the local minima problem in backpropagation algorithm with modified error function. *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, 88(12):3645–3653.

[Bischl et al., 2012] Bischl, B., Mersmann, O., Trautmann, H., and Weihs, C. (2012). Resampling methods for meta-model validation with recommendations for evolutionary computation. *Evolutionary Computation*, 20(2):249–275.

[Bojarski et al., 2016] Bojarski, M., Del Testa, D., Dworakowski, D., Firner, B., Flepp, B., Goyal, P., Jackel, L. D., Monfort, M., Muller, U., Zhang, J., et al. (2016). End to end learning for self-driving cars. *arXiv preprint arXiv:1604.07316*.

[Bottou, 2010] Bottou, L. (2010). Large-scale machine learning with stochastic gradient descent. In *Proceedings of COMPSTAT'2010*, pages 177–186. Springer.

[Browne, 2000] Browne, M. W. (2000). Cross-validation methods. *Journal of mathematical psychology*, 44(1):108–132.

[Caruana et al., 2001] Caruana, R., Lawrence, S., and Giles, C. L. (2001). Overfitting in neural nets: Backpropagation, conjugate gradient, and early stopping. In *Advances in neural information processing systems*, pages 402–408.

[Cohen et al., 2017] Cohen, G., Afshar, S., Tapson, J., and Van Schaik, A. (2017). Emnist: Extending mnist to handwritten letters. In *2017 International Joint Conference on Neural Networks (IJCNN)*, pages 2921–2926. IEEE.

[Conneau et al., 2017] Conneau, A., Kiela, D., Schwenk, H., Barrault, L., and Bordes, A. (2017). Supervised learning of universal sentence representations from natural language inference data. *arXiv preprint arXiv:1705.02364*.

[Darwin and Bynum, 2009] Darwin, C. and Bynum, W. F. (2009). *The origin of species by means of natural selection: or, the preservation of favored races in the struggle for life*. Penguin Harmondsworth.

[Davis, 1991] Davis, L. (1991). Handbook of genetic algorithms.

[Diaz et al., 2017] Diaz, G. I., Fokoue-Nkoutche, A., Nannicini, G., and Samulowitz, H. (2017). An effective algorithm for hyperparameter optimization of neural networks. *IBM Journal of Research and Development*, 61(4/5):9–1.

[Dietterich, 1995] Dietterich, T. (1995). Overfitting and undercomputing in machine learning. *ACM computing surveys (CSUR)*, 27(3):326–327.

[Floreano et al., 2008] Floreano, D., Dürr, P., and Mattiussi, C. (2008). Neuroevolution: from architectures to learning. *Evolutionary intelligence*, 1(1):47–62.

[Fox and McMahon, 1991] Fox, B. and McMahon, M. (1991). Genetic operators for sequencing problems. In *Foundations of genetic algorithms*, volume 1, pages 284–300. Elsevier.

[Fu et al., 2016] Fu, K., Cheng, D., Tu, Y., and Zhang, L. (2016). Credit card fraud detection using convolutional neural networks. In *International Conference on Neural Information Processing*, pages 483–490. Springer.

[Gen and Lin, 2007] Gen, M. and Lin, L. (2007). Genetic algorithms. *Wiley Encyclopedia of Computer Science and Engineering*, pages 1–15.

[Girden, 1992] Girden, E. R. (1992). *ANOVA: Repeated measures*. Number 84. Sage.

[Glover and Kochenberger, 2006] Glover, F. W. and Kochenberger, G. A. (2006). *Handbook of metaheuristics*, volume 57. Springer Science & Business Media.

[Goldberg et al., 1989] Goldberg, D. E., Korb, B., Deb, K., et al. (1989). Messy genetic algorithms: Motivation, analysis, and first results. *Complex systems*, 3(5):493–530.

[Goldberg, 2017] Goldberg, Y. (2017). Neural network methods for natural language processing. *Synthesis Lectures on Human Language Technologies*, 10(1):1–309.

[Gori and Tesi, 1992] Gori, M. and Tesi, A. (1992). On the problem of local minima in backpropagation. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, (1):76–86.

[Hansen et al., 2015] Hansen, N., Arnold, D. V., and Auger, A. (2015). Evolution strategies. In *Springer handbook of computational intelligence*, pages 871–898. Springer.

[Hawkins, 2004] Hawkins, D. M. (2004). The problem of overfitting. *Journal of chemical information and computer sciences*, 44(1):1–12.

[Haykin, 1994] Haykin, S. (1994). *Neural networks: a comprehensive foundation*. Prentice Hall PTR.

[HECHT-NIELSEN, 1992] HECHT-NIELSEN, R. (1992). Iii.3 - theory of the backpropagation neural network**based on "nonindent" by robert hecht-nielsen, which appeared in proceedings of the international joint conference on neural networks 1, 593–611, june 1989. © 1989 ieee. In Wechsler, H., editor, *Neural Networks for Perception*, pages 65 – 93. Academic Press.

[Hernández and Stolfo, 1998] Hernández, M. A. and Stolfo, S. J. (1998). Real-world data is dirty: Data cleansing and the merge/purge problem. *Data mining and knowledge discovery*, 2(1):9–37.

[Hirschberg and Manning, 2015] Hirschberg, J. and Manning, C. D. (2015). Advances in natural language processing. *Science*, 349(6245):261–266.

[Holland et al., 1975] Holland, J. H. et al. (1975). *Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence*. MIT press.

[Jiang et al., 2010] Jiang, J., Trundle, P., and Ren, J. (2010). Medical image analysis with artificial neural networks. *Computerized Medical Imaging and Graphics*, 34(8):617–631.

[Kemeny and Snell, 1976] Kemeny, J. G. and Snell, J. L. (1976). *Markov chains*. Springer-Verlag, New York.

[Khayyat et al., 2015] Khayyat, Z., Ilyas, I. F., Jindal, A., Madden, S., Ouzzani, M., Papotti, P., Quiané-Ruiz, J.-A., Tang, N., and Yin, S. (2015). Bigdansing: A system for big data cleansing. In *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*, pages 1215–1230.

[Kleiman et al., 1996] Kleiman, S., Shah, D., and Smaalders, B. (1996). *Programming with threads*. Sun Soft Press Mountain View.

[Krizhevsky et al., 2012] Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105.

[Lameski et al., 2015] Lameski, P., Zdravevski, E., Mingov, R., and Kulakov, A. (2015). Svm parameter tuning with grid search and its impact on reduction of model over-fitting. In *Rough Sets, Fuzzy Sets, Data Mining, and Granular Computing*, pages 464–474. Springer.

[LeCun et al., 2015] LeCun, Y., Bengio, Y., and Hinton, G. (2015). Deep learning. *nature*, 521(7553):436–444.

[Li et al., 2017] Li, L., Jamieson, K., DeSalvo, G., Rostamizadeh, A., and Talwalkar, A. (2017). Hyperband: A novel bandit-based approach to hyperparameter optimization. *The Journal of Machine Learning Research*, 18(1):6765–6816.

[Loshchilov and Hutter, 2016] Loshchilov, I. and Hutter, F. (2016). Cma-es for hyperparameter optimization of deep neural networks. *arXiv preprint arXiv:1604.07269*.

[Magoc and Magoc, 2011] Magoc, T. and Magoc, D. (2011). Neural network to identify individuals at health risk. *Computing Research Repository - CORR*, 2.

[Markets, 2020] Markets, R. a. (2020). The global $50billionneuralnetworksoftwaremarket - 2020 - 2025outlookreport$.

[Mendel, 1965] Mendel, G. (1965). *Experiments in plant hybridisation*. Harvard University Press.

[Miikkulainen et al., 2019] Miikkulainen, R., Liang, J., Meyerson, E., Rawal, A., Fink, D., Francon, O., Raju, B., Shahrzad, H., Navruzyan, A., Duffy, N., et al. (2019). Evolving deep neural networks. In *Artificial Intelligence in the Age of Neural Networks and Brain Computing*, pages 293–312. Elsevier.

[Newton, 1687] Newton, I. (1687). Philosophiæ naturalis principia mathematica (mathematical principles of natural philosophy). *London (1687)*, 1687.

[Odegua, 2020] Odegua, R. (2020). Building a neural network from scratch using python (part 1).

[Polyak and Juditsky, 1992] Polyak, B. T. and Juditsky, A. B. (1992). Acceleration of stochastic approximation by averaging. *SIAM journal on control and optimization*, 30(4):838–855.

[Ramis, 1943] Ramis, M. (1943). Neurona artificial. https://proyectoidis.org/neurona-artificial/.

[Rasmussen, 2003] Rasmussen, C. E. (2003). Gaussian processes in machine learning. In *Summer School on Machine Learning*, pages 63–71. Springer.

[Ravdin et al., 1992] Ravdin, P. M., Clark, G. M., Hilsenbeck, S. G., Owens, M. A., Vendely, P., Pandian, M., and McGuire, W. L. (1992). A demonstration that breast cancer recurrence can be predicted by neural network analysis. *Breast Cancer Research and Treatment*, 21(1):47–53.

[Rere et al., 2015] Rere, L. R., Fanany, M. I., and Arymurthy, A. M. (2015). Simulated annealing algorithm for deep learning. *Procedia Computer Science*, 72(1):137–144.

[Rubinstein and Kroese, 2013] Rubinstein, R. Y. and Kroese, D. P. (2013). *The cross-entropy method: a unified approach to combinatorial optimization, Monte-Carlo simulation and machine learning*. Springer Science & Business Media.

[Rumelhart et al., 1986] Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1986). Learning representations by back-propagating errors. *nature*, 323(6088):533–536.

[Schaffer and Eshelman, 1991] Schaffer, J. and Eshelman, L. (1991). On Crossover as an Evolutionary Viable Strategy. In Belew, R. and Booker, L., editors, *Proceedings of the 4th International Conference on Genetic Algorithms*, pages 61–68. Morgan Kaufmann.

[Shore and Johnson, 1980] Shore, J. and Johnson, R. (1980). Axiomatic derivation of the principle of maximum entropy and the principle of minimum cross-entropy. *IEEE Transactions on information theory*, 26(1):26–37.

[Shorten and Khoshgoftaar, 2019] Shorten, C. and Khoshgoftaar, T. M. (2019). A survey on image data augmentation for deep learning. *Journal of Big Data*, 6(1):60.

[Song et al., 2018] Song, W., Zou, S., Tian, Y., and Fong, S. (2018). A gpu-based training of bp neural network for healthcare data analysis. In *Advanced Multimedia and Ubiquitous Engineering*, pages 193–198. Springer.

[Soria et al., 2006] Soria, E., Martín, J. D., and Lisboa, P. J. G. (2006). *Classical Training Methods*, pages 7–36. Springer US, Boston, MA.

[Srivastava et al., 2014] Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2014). Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958.

[Srivastava et al., 2015] Srivastava, N., Mansimov, E., and Salakhudinov, R. (2015). Unsupervised learning of video representations using lstms. In *International conference on machine learning*, pages 843–852.

[Tesauro, 1987] Tesauro, G. (1987). Scaling relationships in back-propagation learning: Dependence on training set size. *Complex Systems*, 1(2):367–372.

[Turing, 2009] Turing, A. M. (2009). Computing machinery and intelligence. In *Parsing the Turing Test*, pages 23–65. Springer.

[Van Laarhoven and Aarts, 1987] Van Laarhoven, P. J. and Aarts, E. H. (1987). Simulated annealing. In *Simulated annealing: Theory and applications*, pages 7–15. Springer.

[Wang et al., 2017a] Wang, F., Jiang, M., Qian, C., Yang, S., Li, C., Zhang, H., Wang, X., and Tang, X. (2017a). Residual attention network for image classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3156–3164.

[Wang et al., 2017b] Wang, S., Liu, C., Gao, X., Qu, H., and Xu, W. (2017b). Session-based fraud detection in online e-commerce transactions using recurrent neural networks. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 241–252. Springer.

[Wang et al., 2004] Wang, X., Tang, Z., Tamura, H., Ishii, M., and Sun, W. (2004). An improved backpropagation algorithm to avoid the local minima problem. *Neurocomputing*, 56:455–460.

[Wang and Bovik, 2009] Wang, Z. and Bovik, A. C. (2009). Mean squared error: Love it or leave it? a new look at signal fidelity measures. *IEEE signal processing magazine*, 26(1):98–117.

[Weigend, 1994] Weigend, A. (1994). On overfitting and the effective number of hidden units. In *Proceedings of the 1993 connectionist models summer school*, volume 1, pages 335–342.

[West and Bhattacharya, 2016] West, J. and Bhattacharya, M. (2016). Intelligent financial fraud detection: a comprehensive review. *Computers & security*, 57:47–66.

[Wierstra et al., 2008] Wierstra, D., Schaul, T., Peters, J., and Schmidhuber, J. (2008). Natural evolution strategies. In *2008 IEEE Congress on Evolutionary Computation (IEEE World Congress on Computational Intelligence)*, pages 3381–3387. IEEE.

[Xin and Embrechts, 2001] Xin, J. and Embrechts, M. J. (2001). Supervised learning with spiking neural networks. In *IJCNN'01. International Joint Conference on Neural Networks. Proceedings (Cat. No. 01CH37222)*, volume 3, pages 1772–1777. IEEE.