SEP
SECRETARÍA DE EDUCACIÓN PÚBLICA

TECNOLÓGICO NACIONAL DE MÉXICO
Instituto Tecnológico de Ciudad Madero

INSTITUTO TECNOLÓGICO DE CIUDAD MADERO
División de Estudios de Posgrado e Investigación

"POR MI PATRIA Y POR MI BIEN"

APLICACIÓN DE TÉCNICAS DE ALGORITMOS HÍBRIDOS EVOLUTIVOS CON REDES NEURONALES PARA EL DIAGNÓSTICO MÉDICO

Opción 1
TESIS

Que para obtener el grado de:
Maestro en Ciencias de la Computación

Presenta:
I.S.C. Erick Estrada Patiño
G10070518

Director:
Dr. Guadalupe Castilla Valdez

Codirector:
Dr. Juan Frausto Solís

Cd. Madero Tamaulipas **Mayo 2018**

SEP
SECRETARÍA DE EDUCACIÓN PÚBLICA

TECNOLÓGICO NACIONAL DE MÉXICO
Instituto Tecnológico de Ciudad Madero

Cd. Madero, Tams., a 24 de Abril de 2018

OFICIO No.: U5.042/18
ÁREA: DIVISIÓN DE ESTUDIOS DE POSGRADO E INVESTIGACIÓN
ASUNTO: AUTORIZACIÓN DE IMPRESIÓN DE TESIS.

C. ING. ERICK ESTRADA PATIÑO
No. DE CONTROL G10070518
P R E S E N T E

Me es grato comunicarle que después de la revisión realizada por el Jurado designado para su Examen de Grado de Maestro en Ciencias de la Computación, se acordó autorizar la impresión de su tesis titulada:

"APLICACIÓN DE TÉCNICAS DE ALGORITMOS HÍBRIDOS EVOLUTIVOS CON REDES NEURONALES PARA EL DIAGNÓSTICO MÉDICO "

El Jurado está integrado por los siguientes catedráticos:

PRESIDENTE :	DRA.	LAURA CRUZ REYES
SECRETARIO:	DR.	HÉCTOR JOAQUÍN FRAIRE HUACUJA
VOCAL:	DRA.	GUADALUPE CASTILLA VALDEZ
SUPLENTE:	DR.	JUAN FRAUSTO SOLÍS
DIRECTORA DE TESIS :	DRA.	GUADALUPE CASTILLA VALDEZ
CO-DIRECTOR DE TESIS:	DR.	JUAN FRAUSTO SOLÍS

Es muy satisfactorio para la División de Estudios de Posgrado e Investigación compartir con Usted el logro de esta meta. Espero que continúe con éxito su desarrollo profesional y dedique su experiencia e inteligencia en beneficio de México.

A T E N T A M E N T E
Excelencia en Educación Tecnológica®
"POR MI PATRIA Y POR MI BIEN"®

DR. JOSÉ AARÓN MELO BANDA
ENCARGADO DE LA DIVISIÓN DE ESTUDIOS DE POSGRADO E INVESTIGACIÓN

SECRETARÍA DE EDUCACIÓN PÚBLICA
TECNOLÓGICO NACIONAL DE MÉXICO
INSTITUTO TECNOLÓGICO DE CIUDAD MADERO
DIVISIÓN DE ESTUDIOS DE POSGRADO E INVESTIGACIÓN

c.c.p.- Archivo
Minuta

JAMB 'JAMF 'mdcoa*

Contenido

Índice de Figuras

Índice de Graficas

Índice de Tablas

Índice de ecuaciones

Capítulo 1. Introducción

1.1 Introducción

Los diagnósticos médicos no siempre resultan sencillos, la complejidad en la detección de los síntomas, la diversidad de pacientes, el medio ambiente y la automedicación complican el diagnóstico.

Un mal diagnóstico podría llevar a complicaciones muy graves, existen dos posibles errores al diagnosticar; un falso positivo y un falso negativo. Un falso positivo es cuando el médico diagnostica una enfermedad que no tiene el paciente, mientras que un falso negativo es cuando el médico no diagnostica una enfermedad que si tiene el paciente. Un falso positivo llevaría a una medicación y/o cirugía innecesaria, mientras que un falso negativo puede llegar a complicar la salud del paciente al no atender la enfermedad a tiempo. En este último caso, y dependiendo de la enfermedad, esto puede llevar al paciente a una medicación tardía, cirugía de emergencia y la muerte.

La inteligencia artificial ha ayudado en el área de la salud a realizar diagnósticos para varias enfermedades (Bustamante & Chavarría, 1992). Dentro de las estrategias de inteligencia artificial, las redes neuronales han tenido diversas aplicaciones en el área de diagnóstico médico (Delgado, 1999).

Pese a los esfuerzos previos siempre se desea tener una mayor certeza en el diagnóstico de enfermedades. En este trabajo se desarrollará una red neuroevolutiva para el diagnóstico médico y se evaluará su desempeño comparándose con el de una red neuronal estándar y alguna otra técnica de inteligencia artificial utilizando al menos un caso de estudio médico. Con base en el trabajo de Diana Ortiz (Ortiz & Villa, 2007), se espera que la red neuroevolutiva presente en general un mejor desempeño que las otras implementaciones de este trabajo de investigación.

1.2 Planteamiento del problema

El diagnóstico médico es el resultado de la clasificación de una sintomatología, signos y evidencia médica que un profesional de la salud evalúa para determinar un padecimiento.

El profesional de la salud hará una recopilación de todos los datos que tenga disponibles, y en base a esto y a su experiencia determinará un diagnostico descartando padecimientos con los que la sintomatología no coincide y tomando como posibilidad aquellos en los que la sintomatología sea similar. (Houghton, 2011)

Este proceso de diagnóstico funciona de manera similar a los procedimientos de clasificación utilizados en la minería de datos. Dado que se busca hacer un match con la sintomatología presentada y un padecimiento el cual tenga una sintomatología conocida similar o igual. Relacionando ambos ámbitos, la aplicación de un proceso de inteligencia artificial para la realización de diagnóstico médico resultaría en un sistema experto.

Las redes neuronales tradicionales son una técnica de inteligencia artificial capaces de poder generalizar el conocimiento y dar respuestas aceptables, inclusive a situaciones anómalas o desconocidas. Esto se debe a un entrenamiento previo significativo.

Las redes neuronales evolutivas son una variante de la red neuronal clásica, con la particularidad de tener un mejor acoplamiento al problema, sin perder la capacidad de responder ante situaciones desconocidas. (Pino, Gomez, & De Abajo, 2001)

Se pretende probar la eficiencia de las redes neuronales evolutivas sobre las redes neuronales clásicas en el ámbito del diagnóstico médico.

1.3 Justificación y beneficios

Existen enfermedades cuyo diagnóstico en etapas tempranas por un médico no especialista es complicado, porque la sintomatología no es aún evidente, presenta síntomas atípicos, los signos son difusos, y las pruebas clínicas y de gabinete no otorgan aún suficiente información.

Si bien en muchos casos con el pasar del tiempo y la observación, la sintomatología converge y se puede determinar un diagnóstico, en otros casos existe información contradictoria que descarta un diagnóstico especifico y puede generar dudas de diagnóstico aún con la evaluación de un médico especialista.

Un diagnóstico certero en etapas tempranas evitaría que el paciente presentará complicaciones y que el padecimiento o enfermedad se atendiera en etapas tardías con el riesgo de secuelas o muerte.

1.4 Objetivos

A continuación, se detallan los objetivos, los cuales se utilizan para especificar y determinar la dirección que toma el desarrollo de este trabajo de tesis.

1.4.1 Objetivo general

Clasificar una sintomatología sospechosa de un cuadro clínico y obtener un diagnóstico con un alto nivel de confianza.

1.4.2 Objetivos específicos

- Construir una representación que caracterice las variables para el diagnóstico.
- Obtener instancias de los casos de estudio.
- Diseñar e implementar una red neuronal clásica.
- Diseñar e implementar una red neuroevolutiva.

- Realizar un estudio de al menos tres técnicas neuroevolutivas para incluirlas en la red neuronal evolutiva.
- Realizar una evaluación experimental.

1.5 Hipótesis

Se plantea como hipótesis de esta investigación de que la calidad de predicción obtenida con una red neuroevolutiva para el diagnóstico médico es superior a la calidad de predicción obtenida de una red neuronal clásica.

1.6 Alcances y limitaciones

A continuación, se describen los alcances y las limitaciones del trabajo.

- Alcances
  - Se desarrollará una estructura de estructura base de red neuronal de la cual partirán los dos tipos de red neuronal a implementar.
  - Se hará la comparación de los resultados de ambas redes mediante la prueba estadística de Wilcoxon.
- Limitaciones
  - No se pretende realizar un entorno grafico de usuario para la interacción con la red neuronal.
  - La red neuronal evolutiva solo implementará técnicas de neuroevolución para las que se encuentren herramientas disponibles para fines de comparación.
  - Solamente se utilizarán instancias que hayan sido validadas clínicamente y/o disponibles en la literatura.

1.7 Contenido de la tesis

Capítulo 2. **Marco conceptual**: En este capítulo están fundamentadas las bases teóricas relacionadas con este trabajo, como los conceptos de redes neuronales, algoritmos evolutivos, así como redes neuronales evolutivas.

Capítulo 3. **Descripción del problema y estado del arte**: En esta sección se describe el problema a abordar en este trabajo, además se presentan los trabajos relevantes relacionados con esta investigación.

Capítulo 4. **Propuesta de solución**: En esta sección se presenta el modelo propuesto en forma general y se describen con detalle los elementos y estrategias que lo integran

Capítulo 5. **Experimentación y resultados**: Contiene una descripción amplia de todo el proceso de experimentación que se realiza, así como las condiciones, los resultados obtenidos y el análisis estadístico de los resultados obtenidos.

Capítulo 6. **Conclusiones y trabajos futuros**: Se describen las conclusiones derivadas del proceso experimental, los logros y metas alcanzados; también se describen posibles trabajos o propuestas que permitirían dar continuidad a este trabajo de tesis.

Capítulo 2. Marco conceptual

2.1 Inteligencia artificial y minería de datos

En ciencias de la computación la aplicación de diversos algoritmos que pueden emular una capacidad humana de aprendizaje, reconocimiento o toma de decisiones se les puede considerar como algoritmos de inteligencia artificial. Dentro de este conjunto de algoritmos se encuentran aquellos que hacen el procesamiento de gran cantidad de datos logrando descubrir patrones de manera eficaz, a estas técnicas se les conoce como técnicas de minería de datos

Ambos enfoques han tenido aplicaciones en el diagnóstico médico como por ejemplo como auxiliares en el diagnostico o sistemas expertos. A continuación, se amplían los conceptos mencionados anteriormente

2.1.1 Inteligencia artificial

El término de inteligencia ha sido discutido desde la antigüedad, debido a que su definición no es lo suficientemente clara para muchos. La inteligencia no es exclusiva de los seres humanos, sino de todo ser vivo.

La inteligencia es una capacidad de todo ser vivo de pensar, tomar decisiones, razonar y entender. Esta capacidad es heredada junto con un cumulo de conocimiento limitado por los organismos padres, y es desarrollado y complementado a lo largo del tiempo y experiencia. (Maureira, 2017)

En las ciencias de la computación se dice que una máquina o computadora es inteligente si puede realizar alguna actividad que realiza el ser humano sin la intervención del mismo.

En 1950 Alan Turing público lo que se considera el comienzo de la inteligencia artificial en su libro *“Computing machinery and intelligence”* donde propone que una máquina o sistema puede ser considerado como inteligente si un humano puede interactuar con él y el humano desconoce si esta interactuando con un computador o con otro ser humano. A esta prueba le denominó *“Test de Turing”.* (Inteligencia artificial : fundamentos, práctica y aplicaciones, 2012).

Para aprobar el Test de Turing, una máquina debe tener las siguientes características (Inteligencia artificial : fundamentos, práctica y aplicaciones, 2012).

- Reconocimiento del Lenguaje Natural
- Razonamiento
- Aprendizaje
- Representación del conocimiento

Es muy difícil concebir un sistema con tales características, por lo que actualmente cada una de ellas se considera una rama de la inteligencia artificial.

Podemos definir entonces que la inteligencia artificial es la capacidad de una máquina de emular comportamientos humanos de forma natural, es decir no secuenciales si no acorde a las situaciones presentadas.

2.1.2 Minería de datos

La minería de datos es un proceso de descubrimiento del conocimiento mediante técnicas que analizan información en grandes volúmenes o volúmenes relevantes.

El objetivo de la minería de datos es el descubrimiento de patrones, relaciones en la información, que en el caso del diagnóstico médico resultan difíciles de detectar fácilmente en humanos, o bien es imposible de realizar sin herramientas computacionales debido al gran volumen de información que se requiere manejar para lograr dichos objetivos. Afortunadamente, tales objetivos se pueden lograr mediante un proceso de extracción de conocimientos comúnmente conocido como KDD (del inglés*, Knowledge Discovery Data*) (Durán & Costaguta*,* 2007).

Entre las técnicas de extracción de la información más relevantes se encuentran el algoritmo KNN, perceptrón, redes neuronales multicapa, clasificadores bayesianos y árboles de decisión (Hernández, 2004).

2.2 Redes neuronales

Las redes neuronales son herramientas de clasificación y predicción basadas en el comportamiento del cerebro humano. Entre sus principales características se encuentra la simplicidad de su estructura, debido a que están compuestas por dos elementos básicos: las neuronas, las cuales son unidades simples de procesamiento que están limitadas a hacer cálculos matemáticos relativamente sencillos y las conexiones de las neuronas, es decir la forma en la que ellas se comunican; el proceso de conexión es lo que hace que esta estructura simple se convierta en una herramienta muy poderosa. Las redes neuronales pueden definirse como un conjunto de nodos o neuronas interconectadas entre sí que transforman un conjunto de valores de entrada en uno o varios valores de salida. (Gurney, 2003)

De manera más simple podemos decir que las neuronas son unidades simples de procesamiento que reciben información y la procesan, para producir algún valor que será propagado a otras neuronas. La topología de la red es la manera en que estas neuronas están interconectadas con otras y transmiten la información de la entrada, procesándola varias veces según la cantidad de neuronas que existan hasta llegar a la salida o las salidas.

Una de las principales características que tienen las redes neuronales es que son autoadaptables, es decir; la sintonización de sus parámetros para producir las salidas se hace de manera automática en un proceso de aprendizaje basado en ejemplos. Este funcionamiento evita utilizar probabilidades de ocurrencia o reglas de decisión, debido a que la red se ajusta a los casos que haya aprendido y es capaz de dar una respuesta aceptable a situaciones completamente desconocidas. (Sempere, Gallego, Llorens, Pujol, & Rizo, 2005)

2.2.1 Similitudes con la neurona biológica

Las redes neuronales fueron diseñadas originalmente para emular el funcionamiento de las neuronas en el cerebro, por tanto, tienen una similitud con

el mismo. El cerebro humano está compuesto de unidades muy básicas llamadas neuronas que en conjunto con las interconexiones entre ellas forman un sistema muy complejo y poderoso.

La neurona es una célula especializada del sistema nervioso con características transmisoras y receptoras, con un núcleo que decide si transfiere una señal o no, dependiendo si un umbral es alcanzado. Están compuestas por tres elementos característicos: los axones, el núcleo y las dendritas (Lafarja, 1994). Esto se ilustra en la *Figura 1* a continuación.

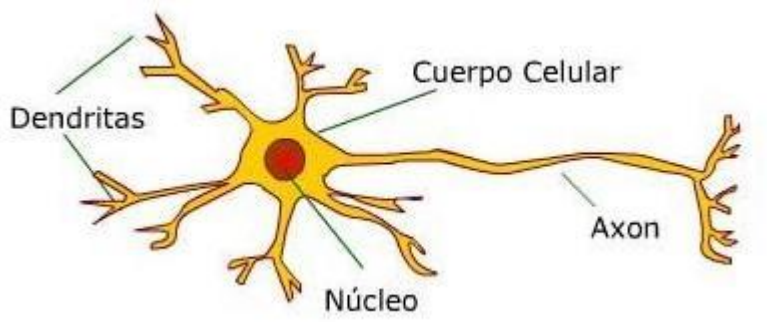

Figura 1. Estructura biológica neuronal (EducarChile, 2016).

Las dendritas son conexiones de entrada que se utilizan para recibir impulsos eléctricos de otras neuronas; a cada conexión de entrada le corresponde una dendrita y ésta tiene una fuerza de conexión. La fuerza de conexión es la influencia que tiene esa neurona sobre la otra.

Al proceso de conexión eléctrica de dos neuronas se le conoce como sinapsis, la cual ocurre cuando un impulso eléctrico es transmitido por la dendrita, y viaja a través de un líquido neurotransmisor al axón de otra neurona.

Cuando una neurona recibe impulsos eléctricos a través de sus dendritas, estos impulsos son evaluados en el núcleo de la célula. Si el impulso eléctrico de entrada supera un umbral determinado propio de la neurona, se propaga a las demás neuronas con las cuales tiene conexión.

La propagación de los impulsos eléctricos se hace por igual a través de todas las ramificaciones del axón que tiene la neurona, de esta forma el impulso se va propagando a las demás neuronas.

2.2.2 Historia y orígenes de las redes neuronales

Las redes neuronales artificiales son una técnica que data de la década de 1940; sin embargo, su historia no ha sido totalmente exitosa, ya que por muchas décadas fueron consideradas obsoletas e ineficientes, por su incapacidad de resolver problemas no linealmente separables.

En 1943 McCulloch y Pitts propusieron un modelo basado en las neuronas biológicas. En su trabajo proponen que no se tome en consideración aspectos fisiológicos, sino su funcionamiento y capacidades, entonces la neurona puede ser vista como una entidad lógica, y por tanto puede modelarse matemáticamente (McCulloch & Pitts, 1943).

El primer modelo neuronal de McCulloch y Pitts consiste en un conjunto de neuronas que aceptan solo estados binarios; es decir, encendidas (1) o apagadas (0). Los valores se reciben y transmiten mediante una conexión o sinapsis, la cual tiene valores de influencia fijos. Este modelo, aunque muy básico resultó la base para que las redes neuronales continuaran evolucionando.

En 1949 Donald Hebb demostró que un conjunto de neuronas tiene la capacidad de aprender, percibir y almacenar información. Su idea era que la activación de una neurona produce la activación de un conjunto de neuronas que se encuentren interconectadas. Cada activación y propagación de información entre las neuronas produce que su conexión se refuerce (Hebb, 1949).

Fue hasta 1958 que Rosenblatt propuso el primer modelo de red neuronal, el perceptrón simple Este modelo se basó en el funcionamiento ocular donde la intensidad de un haz de luz incide en el foto-receptor y este propaga una señal dependiendo de la cantidad de luz incidente. Trasladando estos principios a una neurona, esta podría adquirir un rango de valores dependiendo de las diferentes

intensidades de señal que sobre ella inciden, así las neuronas pierden las propiedades binarias propuestas por McCulloch y pueden tratar valores reales dentro de un rango determinado (Rosenblatt, 1958).

Las redes neuronales hasta ese entonces se habían usado para varias aplicaciones lineales, como por ejemplo eliminar ecos en líneas telefónicas. Sin embargo Marvin Minsky y Seymour Papert en 1969 publicaron un libro en el cual indicaban la debilidad del perceptrón, probando mediante métodos matematicos que las redes neuronales solamente podían resolver problemas lineales, y en problemas no lineales tenían un alto porcentaje de error puesto que la división del espacio de soluciones solo podía segmentarse en dos (Minsky & Papert, 1972).

Sin embargo, en 1985 John Hopfield propone un modelo de red neuronal que da una buena solución para el problema del agente viajero. El determina que las redes neuronales con un alto número de conexiones y una división neuronal en forma de capas pueden resolver problemas no lineales y otorgar buenos resultados (Hopfield & Tank, 1985).

En 1986 los investigadores Rumerlhart y Hinton propusieron un modelo de aprendizaje en el cual el ajuste de los pesos se hace partiendo de las neuronas de salida hacia las neuronas de entrada, y le llamaron método de propagación de error hacia atrás o *Back Propagation*, esto otorga un proceso de aprendizaje rápido y resultados más precisos (Rumelhart, Hinton, & Williams, 1986).

2.2.3 Estructura neuronal

La neurona o neurona artificial es la unidad básica de procesamiento dentro de una red neuronal. Su estructura interna es muy simple, y básicamente puede dividirse en tres boques de procesamiento los cuales se ilustran en la *Figura 2*.

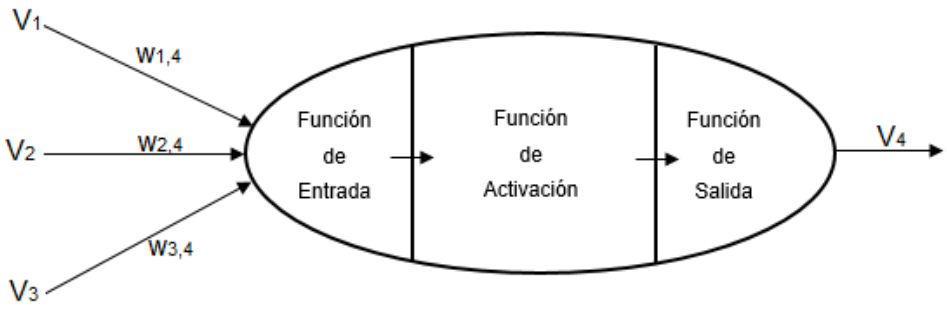

Figura 2. Representación de una neurona artificial.

Estos bloques son: la función de entrada, la función de activación y la función de salida. Estos bloques trabajan secuencialmente transformando un conjunto de valores que inciden en la neurona para convertirlos en una salida.

Cada incidencia en la neurona tiene una magnitud asociada equivalente a la fuerza de la conexión $w_{i,j}$. Dicha magnitud altera el valor que proviene de las neuronas previas V_i hacia la neurona actual j, a esta magnitud se le conoce como peso de la conexión. Las conexiones pueden poseer pesos positivos, negativos o cero, dicho de otra forma; la conexión puede favorecer la activación de la neurona, inhibirla o no influir.

El comportamiento de la neurona se puede describir mediante los siguientes pasos: una función de entrada procesa los datos que ingresan a la neurona, y el resultado es utilizado por la función de activación que decide si se activará la neurona, posteriormente se procesa la salida de la función de activación aplicando otra función que determinará el valor que será propagado a las siguientes neuronas.

Las neuronas necesitan procesar la información de las conexiones que inciden en ellas. Este procesamiento aplica una función a todas las entradas que indicen en la neurona y generalmente aplica un decremento de un umbral al resultado. El valor de entrada será procesado después por la función de activación.

Algunas de las funciones más importantes que se aplican en la neurona se describen enseguida. Es importante destacar que la aplicación de esta depende directamente del problema sobre el cual se esté trabajando.

Función sumatoria

Esta función realiza la suma aritmética de todos los valores que inciden en la neurona, multiplicados por su respectivo peso o fuerza de conexión. Luego, al valor resultante de la sumatoria se le resta el umbral de la neurona tal como lo indica en la *ecuación 1*.

$$VFE = \sum (v_{origen,\ actual} * w_{origen,\ actual}) - \ umbral_{actual}$$

Ecuación 1. Función de entrada sumatoria

Donde:
$actual =$ Neurona actual
$origen =$ neurona que incide en ella
$umbral =$ umbral de la neurona
$v =$ valor enviado de la neurona origen a la neurona actual
$w =$ peso o significancia de la conexión de la neurona origen a la neurona actual
$VFE =$ Valor de la función de entrada

Función productora

Esta función realiza una multiplicación aritmética de todos los valores que inciden en la neurona, multiplicados por su respectivo peso o fuerza de conexión. Luego al valor resultante se le resta el umbral de la neurona; como se muestra en la *ecuación 2*.

$$VFE = \prod (v_{origen,\ actual} * w_{origen,\ actual}) \ - \ umbral_{actual}$$

Ecuación 2. Función de entrada productora

Donde:
$actual =$ Neurona actual
$origen =$ neurona que incide en ella
$umbral \ =$ umbral de la neurona
$v =$ valor enviado de la neurona origen a la neurona actual
$w =$ peso o significancia de la conexión de la neurona origen a la neurona actual
$VFE =$ Valor de la función de entrada

Función argumento máximo

Esta función elige el valor más grande entre todos los valores que inciden en ella, multiplicados por su respectivo peso. Después de identificar el valor más grande se le resta el umbral correspondiente de la neurona actual, tal como lo indica la *ecuación 3*.

$$VFE = argMAX(\{v_{origen,\ actual} * w_{origen,\ actual}\}) - umbral_{actual}$$

Ecuación 3. Función de entrada por argumento máximo

Donde:
$actual =$ Neurona actual
$origen =$ neurona que incide en ella
$umbral =$ umbral de la neurona
$v =$ valor enviado de la neurona origen a la neurona actual
$w =$ peso o significancia de la conexión de la neurona origen a la neurona actual
$VFE =$ Valor de la función de entrada

Funciones de activación

Las funciones de activación transforman el valor de entrada en un nuevo valor que determinará si la entrada es suficientemente significativa como para activar la neurona y para identificar el valor con el que será activada, para que continúe propagando información. La función de activación puede ser lineal o no lineal.

Función lineal

Esta función transmite sin modificaciones el valor de la función de entrada hacia la función de salida, como puede observarse en la *ecuación 4.*

$$VFA = VFE$$

Ecuación 4. Función de activación lineal

Donde:
$VFE =$ Valor de la función de entrada
$VFA =$ Valor de la función de activación

Función lineal positiva

Esta función asegura que solo se envíen a la función de salida valores mayores o iguales que cero, como indica la *ecuación 5.*

$$VFA = \begin{cases} 0; & VFE < 0 \\ VFE; & VFE \geq 0 \end{cases}$$

Ecuación 5. Función de activación lineal positiva.

Donde:
$VFE =$ Valor de la función de entrada
$VFA =$ Valor de la función de activación

Función lineal saturada

Acota la posible salida entre 0 y 1. Cuando recibe un valor de la función de entrada lo envía a la función de salida directamente si su valor está entre el rango de valores, en caso contrario, envía la cota más cercana. En la *ecuación 6* se muestra este tipo de función.

$$VFA = \begin{cases} 0; & VFE < 0 \\ VFE; & 0 \leq VFE \leq 1 \\ 1; & VFE > 1 \end{cases}$$

Ecuación 6. Función de activación lineal saturada

Donde:
$VFE =$ Valor de la función de entrada
$VFA =$ Valor de la función de activación

Función limitador fuerte

Esta función se limita a regresar el valor de cero o el valor de uno, dependiendo el valor producido en la función de entrada, La *ecuación 7* corresponde a este tipo de función.

$$VFA = \begin{cases} 0; & VFE < 0 \\ 1; & VFE \geq 0 \end{cases}$$

Ecuación 7. Función de activación de limitador fuerte

Donde:
$VFE =$ Valor de la función de entrada
$VFA =$ Valor de la función de activación

Función limitador fuerte simétrico

Esta función se limita a regresar el valor de -1 o 1, dependiendo el valor producido en la función de entrada, y se utiliza la *ecuación 8* para esta función.

$$VFA = \begin{cases} -1; & VFE < 0 \\ +1; & VFE \geq 0 \end{cases}$$

Ecuación 8. Función de activación de limitador fuerte simétrico

Donde:
$VFE =$ Valor de la función de entrada
$VFA =$ Valor de la función de activación

Función competitiva

La función competitiva involucra a todas las demás neuronas que tienen el mismo nivel topológico. A la neurona que tenga el valor máximo o el valor mínimo (Dependiendo del problema) se le asigna el valor de 1, todas las demás neuronas que integren la capa tendrán un valor de 0. La *ecuación 9* aplica para la ecuación competitiva.

$$VFA = \begin{cases} 1; & argMax(\{VFE\}) \\ 0; & \neg argMax(\{VFE\}) \end{cases}$$

Ecuación 9. Función de activación competitiva.

Donde:
$VFE =$ Valor de la función de entrada
$VFA =$ Valor de la función de activación

Función tangente sigmoidea hiperbólica

Esta función calcula la tangente sigmoidea del valor de la función de entrada para producir un valor de salida no lineal. La *ecuación 10* representa esta función.

$$VFA = \frac{e^{VFE} - e^{-VFE}}{e^{VFE} + e^{-VFE}}$$

Ecuación 10. Función de activación de tangente sigmoidea hiperbólica.

Donde:
$VFE =$ Valor de la función de entrada
$VFA =$ Valor de la función de activación

Función sigmoidea logarítmica

Esta función calcula el valor sigmoideo de la función de entrada para producir un valor de salida no lineal. La *ecuación 11* muestra este tipo de función.

$$VFA = \frac{1}{1 + e^{-VFE}}$$

Ecuación 11. Función de activación sigmoidea logarítmica.

Dónde:
$VFE =$ Valor de la función de entrada
$VFA =$ Valor de la función de activación

2.2.4 Topologías

La topología de las redes neuronales es la forma en la que la red está estructurada, es decir el número de capas de conexión, número de neuronas, número de neuronas por capa, cantidad de conexiones, combinaciones de conexiones, número de neuronas de entrada y número de neuronas de salida.

El número de neuronas de la capa de entrada es dependiente del problema al que sean aplicadas. Así el número está dado por la longitud del vector representativo de entrada del problema.

De la misma manera, el número de neuronas de la capa de salida es directamente dependiente de la salida de nuestro problema, es decir va a ser igual a la longitud del vector representativo de la salida del problema o, dicho de otra forma, la cantidad de variables de salida del problema.

La estructura interna, en caso de que las halla, contiene el número de capas y de conexiones y no depende de la entrada ni la salida del problema. Los valores de

estos parámetros deben ser sintonizados con el fin de obtener un porcentaje de error de clasificación mínimo.

La topología de una red neuronal se puede clasificar de varias maneras, a continuación, se listan algunas de ellas. (Flórez & Miguel, 2008)

De acuerdo al número de capas:

Las redes neuronales se clasifican en dos estructuras, las redes monocapa y las redes multicapa.

Las redes monocapa consisten en neuronas cuyas conexiones son laterales o recurrentes, es decir que están compuestas por una única capa de neuronas y usualmente se usan para realizar agrupamiento de datos.

Las redes multicapa consisten en neuronas organizadas por capas. Una capa puede definirse como un conjunto de neuronas que recibe información exclusivamente de una capa anterior y transmite información a una capa posterior.

Las redes neuronales multicapa contienen una capa de entrada, una capa oculta y una capa de salida. La capa oculta puede tener una o más capas de neuronas ocultas. Lo anterior se ilustra en la *Figura 3*.

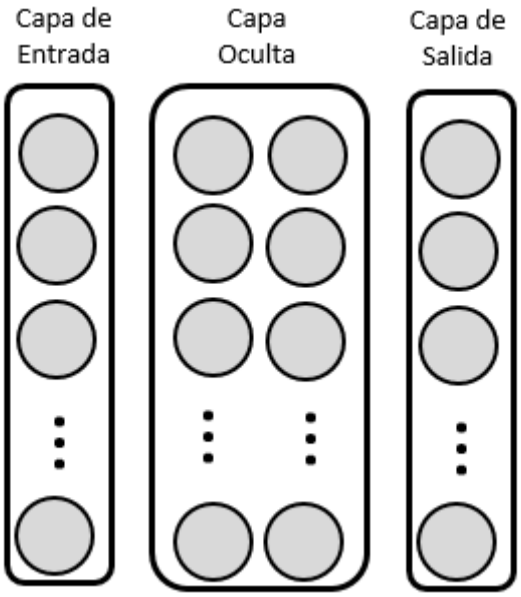

Figura 3. Representación de la organización por capas de una red neuronal.

Las redes multicapa son especialmente útiles para resolver problemas linealmente no separables. En 1972 se demostró que la red de una capa era muy débil para resolver problemas reales, los cuales usualmente caen en la categoría antes mencionada. (Minsky & Papert, 1972)

Las redes de una capa solo pueden resolver problemas lineales. Un ejemplo es el problema del OR exclusivo ilustrado en la *Figura 4*, el cual es imposible de clasificar apropiadamente usando redes de una capa ya que el espacio de soluciones para este problema no es lineal.

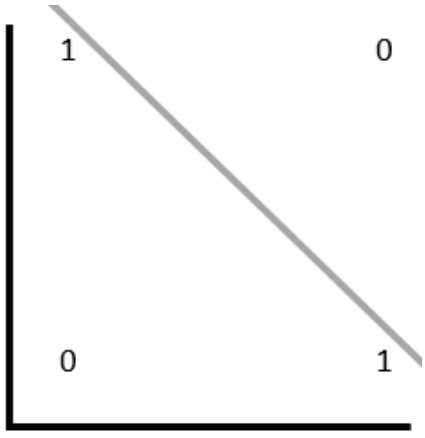

Figura 4. Representación de la división del espacio de soluciones para el problema XOR

La solución a este problema se implementó añadiendo capas a la red. Entonces el número de capas en la red neuronal depende de la complejidad del espacio de soluciones del problema a resolver. (Hopfield & Tank, 1985)

De acuerdo al tipo de flujo de los datos:

Las redes feedforward son aquellas que solo tienen conexiones hacia neuronas de capas superiores. Se dividen en dos tipos; redes feedforward y redes feedforward multicapa.

Las redes feedforward no tienen una estructura fija, si bien las conexiones solo son con capas superiores, no necesariamente la conexión se da con la capa inmediata, si no que las conexiones se pueden dar con neuronas de otras capas siempre y cuando sean capas superiores.

Las redes feedforward multicapa tienen conexiones exclusivamente con las neuronas de la capa superior inmediata.

Por otra parte, las redes feedback tienen conexiones neuronales hacia neuronas de capas superiores, hacia neuronas de la misma capa o hacia neuronas de la capa anterior. Cuando las neuronas tienen una conexión hacia neuronas de una capa anterior y se forma un ciclo, entonces podemos hablar de que se tratan de redes recurrentes.

2.2.5 Mecanismos de aprendizaje

Para que las redes neuronales funcionen eficientemente, se les debe someter a un proceso de entrenamiento en el cual la red aprende de los ejemplos que se le proporcionan, autoajustando sus parámetros. En este proceso se refuerzan o debilitan las conexiones neuronales, y esto es totalmente dependiente del problema a resolver y de los ejemplos de entrenamiento dados.

Aprendizaje Heebiano

Para redes de una neurona, el aprendizaje Heebiano es una opción para el ajuste de los pesos de la red. En este aprendizaje se inicializan los pesos en cero, una

vez iniciado el proceso de entrenamiento se calcula la salida para cada ejemplo, esta se iguala con la salida deseada y el ajuste de los pesos se obtiene utilizando la *ecuación 12*.

$$w_i^{new} = w_i^{old} + x_i(y_d - y_o)$$

Ecuación 12. Aprendizaje heebiano

Donde:
w_i^{new} = Valor del peso nuevo
w_i^{old} = Valor del peso anterior
y_d = Valor deseado
y_o = Valor obtenido
x_i = Valor de entrada de la neurona

El proceso de entrenamiento se da por terminado cuando el error de todo el conjunto de entrenamiento es cero o se llega a un criterio de paro.

Aprendizaje perceptrón

Para redes de una neurona, el aprendizaje perceptrón es un método muy popular de aprendizaje. A diferencia del aprendizaje Heebiano, para el aprendizaje perceptrón debemos definir un valor de umbral y una tasa de aprendizaje.

El valor de umbral es una cota que debe ser superada por el valor de las entradas para que la neurona sea activada; en otras palabras, es el valor que debe ser rebasado por el resultado de la función de entrada para que la neurona pueda procesar la información.

La tasa de aprendizaje es un valor acotado entre cero y uno que representa el grado de cambio en la neurona; esto es de vital importancia ya que cambiar bruscamente los valores en cada prueba llevaría a un proceso tardío de ajuste, sin embargo, al reducir el cambio, se provoca un aprendizaje más suave.

En el entrenamiento se inicializan los pesos y umbrales aleatoriamente, se procesa cada ejemplo obteniendo su salida y una vez obtenido el valor de la salida se aplica la *ecuación 13.*

$$w_i^{new} = w_i^{old} + \propto (y_d - y_o)x_i$$

Ecuación 13. Aprendizaje perceptrón

Dónde:
w_i^{new} = Valor del peso nuevo
w_i^{old} = Valor del peso anterior
α = Tasa de aprendizaje
y_d = Valor deseado
y_o = Valor obtenido
x_i = Valor de entrada de la neurona

Aprendizaje por propagación del error hacia atrás (Backpropagation)

Este tipo de aprendizaje es de los más usados, consiste en propagar el error desde la capa de neuronas de salida hasta la capa de neuronas de entrada, así cada conexión de las capas ocultas recibe un ajuste.

Este aprendizaje se utiliza para redes de varias capas y feedforward, es decir que no tenga conexiones hacia atrás o laterales, en redes de muchas capas el error tiende a perderse conforme se transmite a capas anteriores por lo que solo es recomendable su uso en redes de pocas capas.

En el entrenamiento se inician los pesos y umbrales aleatoriamente y se procesa cada ejemplo obteniendo su salida. Luego se obtiene el error por cada neurona de la capa de salida con la *ecuación 14.*

$$\delta_{pk}^0 = (y_{dk} - y_{ok}) f_k^0(net_{pk}^0)$$

Ecuación 14. Calculo de delta para la capa de salida

Donde:
δ_{pk}^0 = Delta de la neurona de la capa de salida
f_k^0 = función derivable (sigmoidea para salidas binarias y lineal para las demás)
y_{dk} = Valor deseado
y_{ok} = Valor obtenido
net_{pk}^0 = Valor de la neurona

Tratándose de una capa oculta, el error se calcula de otra manera ya que el valor *delta del error* depende directamente del resultado de la capa de salida. La *ecuación 15* muestra el proceso.

$$\delta_{pj}^{h} = f_{j}^{h}(net_{pj}^{h}) \sum_{k} \delta_{pk}^{i} w_{jk}^{i}$$

Ecuación 15. Calculo de delta para las capas ocultas

Donde:
δ_{pj}^{h} = Delta de la neurona de la capa de la capa oculta
f_{j}^{h} = función derivable (sigmoidea para salidas binarias y lineal para las demás)
δ_{pk}^{i} = Delta de la neurona i perteneciente a una capa superior
w_{kj}^{i} = Peso de la conexión de la neurona k a la neurona j
net_{pj}^{h} = Valor de la neurona

Así, se observa que el error en las capas ocultas depende directamente del error en las capas de salida; por este procedimiento se conoce al aprendizaje como propagación de error hacia atrás. (Valencia & Márquez, 2006)

Una vez calculado el error, se debe proceder a realizar el ajuste de los pesos, para este ajuste aplica la *ecuación 16.*

$$w_{i}^{new} = w_{i}^{old} + \propto \delta_{pk}^{h} y_{j}^{p}$$

Ecuación 16. Regla delta

Donde:
w_{i}^{new} = Valor del peso nuevo
w_{i}^{old} = Valor del peso anterior
$\boldsymbol{\alpha}$ = Tasa de aprendizaje
δ_{pk}^{h} = Delta del error
y_{j}^{p} = Valor de entrada de la neurona

2.3 Algoritmos evolutivos

Los algoritmos evolutivos son estrategias inspiradas en la evolución de las especies, los cuales han sido aplicados con éxito a la optimización (Reeves C. R., 2010).

En este tipo de algoritmos, las especies o poblaciones sufren un proceso de adaptación a su ambiente, mejorando o adquiriendo características necesarias y perdiendo características no deseables. Esto se debe a que, en cada generación los nuevos individuos heredan características de sus padres y para adaptarse a

su ambiente sufren pequeños cambios o perturbaciones. Los más aptos sobreviven para reproducirse y seguir transmitiendo sus características a los nuevos individuos mientras que los menos adaptados mueren (Coello, Gary, & Van Veldhuizen, 2007).

2.3.1 Algoritmo genético

Pertenecientes a los algoritmos evolutivos basan su proceso de mejora en la codificación de un genoma y una cruza de cromosomas. Comúnmente se les conoce como metaheurísticas debido a que utilizan conocimiento del problema dentro de heurísticas subordinado a un proceso maestro iterativo con el fin de aplicar estrategias que eviten el estancamiento en óptimos locales (Reeves C. R., 2010).

Las especies sufren un proceso de adaptación a su ambiente, mejorando o adquiriendo características necesarias y perdiendo características no deseables. Esto se debe a que, en cada generación, los nuevos individuos heredan características de sus padres, y para adaptarse a su ambiente sufren pequeños cambios o mutaciones, los más aptos sobreviven para reproducirse y seguir heredando sus características a los nuevos individuos, mientras que los menos aptos mueren.

2.3.1.1 Historia

En 1960 los alemanes Ingo Rechenberg y Hans-Paul Schwefel desarrollaron la idea de lo que ellos llamaron evolución estratégica, mientras que en Estados Unidos también en 1960 Fogel propuso un modelo que fue llamado “programación evolutiva” (Rechenberg, 1978) (Fogel, 1964).

El biólogo Alexander Fraser fue el primero en trabajar en una representación binaria emulando la estructura genética, la cual simulaba un proceso de evolución mediante una selección de individuos en una población y un mecanismo de cruza. En su libro *“Computer Models in Genetics”* anticipa los principios básicos de los

algoritmos genéticos, hablando de una función para calcular el valor de aptitud y de un criterio de convergencia del modelo (Fraser & Burnell, 1970).

El término de algoritmo genético o *GA* fue introducido por primera vez por John Holland en 1975 en su libro “Adaptación Natural y Sistemas Artificiales”. Holland influyó en el desarrollo de este tema de manera muy importante, sin embargo, otros científicos con diferente perspectiva desarrollaron ideas similares. Las ideas comunes de estas propuestas eran la selección y mutación, conceptos que habían sido tomados de las teorías neo darwinistas de la evolución.

El uso de algoritmos genéticos para la optimización es muy popular para la solución de problemas altamente complejos en los entornos actuales.

2.3.1.2 Estructura

Los algoritmos genéticos siguen una estructura general básica, aunque pueden existir variantes u otros operadores añadidos. La estructura incluye los siguientes procesos: La generación de la población inicial, evaluación de la aptitud de la población, procedimiento de selección de individuos, proceso de cruza y proceso de mutación. Esta estructura se muestra en la *Figura 5*.

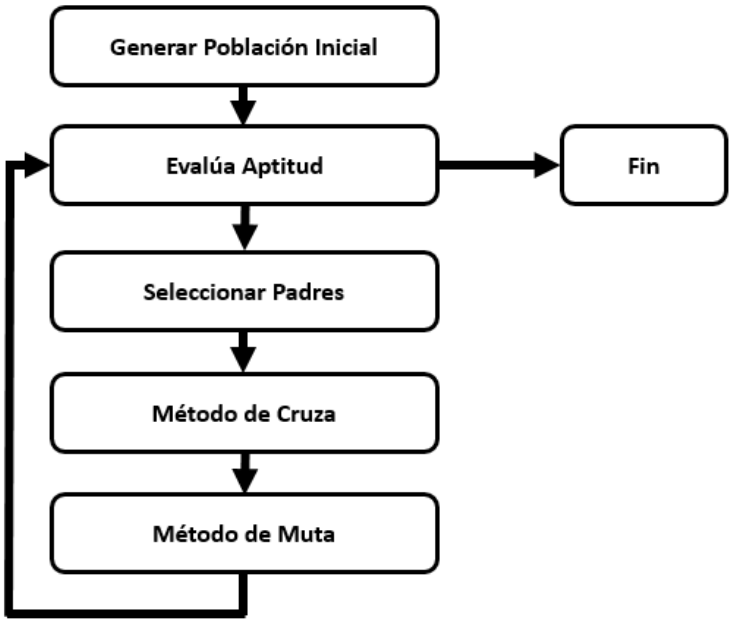

Figura 5. Estructura general de los algoritmos genéticos.

La población inicial en los algoritmos genéticos representa a un conjunto de individuos que comúnmente son generados de manera aleatoria, o en otros casos se emplean algoritmos heurísticos para inicializar la población con elementos que tengan una buena aptitud.

El individuo es la representación de la solución de un problema en forma de cromosoma y de preferencia se busca que sea factible. La población evoluciona a través de las operaciones de cruza y mutación mejorando su aptitud y aproximándose a lo largo de las iteraciones a la solución óptima del problema a tratar.

Un genotipo es la codificación de un individuo en el algoritmo genético; por el contrario, mientras que el fenotipo representa al individuo no codificado.

Al tratarse de un algoritmo poblacional, es importante la definición del tamaño de la población, ya que poblaciones pequeñas no explorarían suficientemente el espacio de soluciones y poblaciones grandes aumentan el costo computacional de procesamiento, se debe encontrar entonces un equilibro entre la eficacia para encontrar una buena solución y la eficiencia para hacerlo rápidamente.

Con respecto a la función de aptitud, esta corresponde a la evaluación de un individuo para conocer su desempeño en el problema. La aptitud depende directamente del problema en que se trabaja. Por ejemplo, en un problema de clasificación podría estar dada en función del porcentaje de elementos que fueron clasificados correctamente.

El proceso de selección está directamente asociado con la valoración de la aptitud, el objetivo es seleccionar un conjunto de padres con un buen valor de aptitud. Sin embargo, podría añadirse al proceso de selección alguna estrategia para la selección de valores dispersos o malos. Esto para explorar con mayor amplitud el espacio de soluciones y evitar el problema de estancamiento.

Una vez seleccionado una pareja de individuos, el procedimiento de cruza se aplicará a estas soluciones para generar nuevos individuos que dentro del

contexto de evoluciones corresponderían a los hijos. En la cruza los padres se combinan con el objetivo de transmitir las características de ambos padres (generación actual) hacia la nueva generación (soluciones hijas).

Sin embargo, se debe mencionar que el proceso de cruza no siempre genera hijos de buena calidad, inclusive si los dos padres son soluciones muy buenas su combinación podría producir hijos de menor calidad.

El método de mutación en esencia, consiste en hacer modificaciones pequeñas o perturbación a un individuo, esto con el fin de incorporar una mayor diversidad en la población y evitar una convergencia prematura.

El proceso de mutación se aplica utilizando una probabilidad para decidir si esta ópera o no sobre un individuo, debido a que, en el proceso de evolución de los seres vivos, la mutación ocurre solo esporádicamente. La selección del método de mutación depende directamente del método de codificación usado en el problema; por ejemplo, puede ser el intercambio de posiciones dentro del vector, la negación de algunos valores o un pequeño incremento o decremento en los valores contenidos en su cromosoma.

Usualmente a la estructura del algoritmo genético suelen añadirse otros operadores con el fin de mejorar su desempeño. Por ejemplo, se puede incorporar una búsqueda local con el fin de explora el vecindario para encontrar el valor óptimo local, así nos garantiza que la solución, aunque pueda no ser el óptimo global es el mejor valor dentro de su vecindario, a este algoritmo se le denomina *algoritmo memetico*.

2.3.2 Algoritmo de búsqueda dispersa

La heurística de búsqueda dispersa o *Scatter Search* es un método que está basado en estrategias de combinación y mejora de un conjunto de soluciones. Pertenece a la familia de algoritmos heurísticos evolutivos, ya incluye una evolución de un subconjunto de soluciones para generar mejores soluciones (Gowda, Manjunath, & Jayaram, 2011).

El algoritmo guarda mucha similitud con los algoritmos genéticos; sin embargo, basa su funcionamiento en estrategias sistemáticas, en lugar de estrategias aleatorias como se manejan en los algoritmos genéticos.

El algoritmo utiliza dos conjuntos de soluciones principales; un conjunto de soluciones iniciales ***P*** y un conjunto de referencia, comúnmente conocido como ***RefSet***.

En comparación de un algoritmo genético que trabaja con muchas soluciones, el conjunto de trabajo de *Scatter Search* es el conjunto ***RefSet*** que usualmente está constituido por un rango que oscila entre 10 y 15 soluciones; la mitad de estas soluciones inicialmente son seleccionadas de las mejores soluciones del conjunto **P** (criterio de calidad), mientras que la otra mitad se elige aplicando un criterio de diversidad; es decir, se seleccionan las soluciones más dispersas o lejanas con respecto a las soluciones de calidad que fueron añadidas en la primer etapa del proceso de construcción.

La búsqueda dispersa implementa estrategias de combinación sobre soluciones tomadas del conjunto de referencia que pueden ser parejas, ternas o subconjuntos mayores. A las soluciones que fueron generadas mediante la combinación en el anterior paso se les aplica una búsqueda local que tiene por objetivo mejorar las soluciones recién creadas. Es decir, se modifican estas soluciones mediante estrategias de mejora o búsqueda local para obtener óptimos locales.

Glover (Glover & Laguna, 2000) propuso una combinación ponderada como proceso de generación de nuevas soluciones, en la cual se introduce la idea de combinar soluciones muy buenas con soluciones dispersas y los resultados de este tipo de combinación resultaron superiores.

2.4 Redes neuronales evolutivas

Las redes neuronales evolutivas son el resultado de la hibridación de redes neuronales artificiales con algún algoritmo evolutivo como los algoritmos

genéticos que se utiliza para generar una evolución en un conjunto de redes neuronales.

En la neuroevolución, el algoritmo genético estará compuesto por una población de redes neuronales, las cuales mediante los operadores del algoritmo genético se recombinarán para obtener mejores o iguales redes a las anteriores.

Sempere nos dice que *"Al utilizar un algoritmo genético para evolucionar una población, se está realizando una búsqueda en el espacio de los comportamientos, con la intención de encontrar un comportamiento óptimo"* (Sempere, Gallego, Llorens, Pujol, & Rizo, 2005).

Esto significa que, dado un problema, la población se adapta paulatinamente a éste, encontrando finalmente un mejor desempeño. A diferencia de las redes neuronales tradicionales las redes neuroevolutivas tienen una adaptación superior al problema, y son menos propensas a presentar el problema de sobre entrenamiento que sufren las redes neuronales tradicionales.

El proceso de evolución para estas redes se puede dividir en 3 bloques: La evolución sináptica, la evolución topológica y la evolución de las reglas de aprendizaje (Castillo, Castellano, Merelo, & Prieto, 2001).

En el proceso de evolución sináptica, los valores se van ajustando con el objetivo de aumentar el desempeño. Este tipo de aprendizaje tiene el mismo objetivo que el algoritmo *Back Propagation*; inclusive éste puede utilizarse como mecanismo de ajuste y tiene como principal objetivo es alcanzar un valor óptimo local.

La evolución de las reglas de aprendizaje se refiere a la selección automática de la función de aprendizaje por cada neurona. Esta forma de aplicar la selección permite ajustar de manera individual y diferente cada segmentación del espacio de soluciones.

La evolución topológica ajusta el número de neuronas en la red, el número de capas, las neuronas por capa y la forma en que estas están organizadas. El

objetivo es encontrar un equilibrio y buen ajuste topológico para no caer en entrenamiento limitado o sobre entrenamiento.

2.4.1 Tipos de codificación

Para poder evolucionar los pesos de la red y su estructura, la red debe ser sometida a un proceso de codificación. Básicamente existen dos tipos de codificación: La codificación directa y la codificación indirecta.

La diferencia de ambas codificaciones recae en la cantidad de información que conforma la codificación. La codificación de la red a detalle se denomina codificación directa, por el contrario, la codificación de solo algunos aspectos relevantes de la red se denomina codificación indirecta.

La elección del tipo de codificación es un proceso que depende directamente del tipo y la complejidad del problema que se esté tratando.

Típicamente en la codificación directa, un conjunto N de neuronas es almacenado en una matriz de incidencia de tamaño $N * N$. Donde cada posición c_{ij} representa una conexión de la neurona N_i con la neurona N_j. Esta representación puede ser binaria o de números reales real, incluyendo los pesos directamente como el valor de la conexión.

En caso de redes del tipo feedforward solamente se utiliza el triángulo inferior de la matriz para evitar crear conexiones entre neuronas de la misma capa o capas anteriores.

En la codificación indirecta solamente se estructuran determinadas características de la red neuronal. La codificación está basada directamente en el conocimiento que se tiene del problema a resolver. Esta codificación si bien produce un genotipo de tamaño pequeño comparado con la codificación directa, reduce el espacio de búsqueda al tener una arquitectura reducida de la red.

Un aspecto inherente a la codificación de la red es la manera de representación de los genes. Típicamente la codificación binaria y la codificación real son las más

utilizadas, ambas tienen cualidades importantes, pero también tienen problemas o características que deben ser tomadas en cuenta para la elección de la codificación.

Los enfoques innovadores han representado la codificación de las redes neuronales en estructuras más complejas como por ejemplo objetos o listas de objetos donde se representan los elementos de la red.

La codificación de las redes neuronales da lugar a que puedan ser procesadas y evolucionadas mediante un algoritmo evolutivo. Típicamente, las redes neuronales tienen una selección de topología fija y manual. Este tipo de redes, solo se entrenaban para ajustar los pesos de la red, lo que puede generar un mal desempeño debido a que no permite buscar el equilibrio optimo entre número de capas y numero de neuronas.

Considerando lo anterior, las redes con muy pocas neuronas no serían capaces de generalizar el conocimiento y tener un aprendizaje ideal. Las redes con un número muy grande de neuronas sufrirán un sobreajuste, otorgando resultados erróneos cuando se presentan ejemplos desconocidos por la red.

2.4.2 Evolución topológica

Existen varios tipos de evolución topológica, un ejemplo de ellos es la evolución topológica incrementa. Este proceso de evolución inicia con topologías básicas, sin capas ocultas y poco a poco a lo largo del proceso de evolución se generan los elementos internos.

Como ya se mencionó, el procedimiento de evolución topológica tiene el mismo fin que en los algoritmos evolutivos tradicionales. De esta manera los operadores internos del algoritmo evolutivo tienen el mismo fin, aunque la metodología o manera de realizarlo pueda diferir un poco.

En el procedimiento de cruza neuronal existen algunas estrategias para realizar la combinación, tales como seleccionar las coincidencias topológicas para aplicar la

cruza sobre ellas, considerar similitudes o tomar en cuenta los números de innovación que indica la iteración en que una topología fue generada, entre otras.

El proceso de mutación consiste en realizar pequeñas modificaciones al cromosoma de la red. Típicamente estas modificaciones pueden realizarse con el fin de modificar la conectividad de la red; ya sea agregando neuronas, conexiones, capas o modificando los valores sinápticos o funciones de aprendizaje.

2.5 Diagnóstico Médico

Un diagnóstico médico es el resultado de las conclusiones que el médico determina en base a su experiencia y conocimiento sobre la presencia de un conjunto de síntomas, y signos. Houghton lo define como *"La explicación más racional de síntomas y signos"* (Houghton, 2011).

En algunos casos, resulta muy evidente para el médico el resultado de un diagnóstico; sin embargo, cuando existe incertidumbre y no puede emitirse un diagnóstico con la información que se tiene al momento, el médico puede apoyarse en un interrogatorio al paciente y/o en exámenes especializados.

Los diagnósticos no son reglas. La sintomatología y signos presentados no siempre son definitivos en un diagnóstico y los cuadros pueden confundirse con facilidad; es aquí donde la experiencia del médico entra en juego para hacer un diagnóstico con un alto nivel de confianza.

Dada la dificultad que conllevan los diagnósticos médicos, pueden establecerse falsos diagnósticos. Un diagnostico falso negativo se da cuando se tiene la presencia de un padecimiento y el médico no logra diagnosticarlo. Un diagnostico falso positivo se da cuando el médico diagnostica en el paciente un padecimiento que no tiene.

Ambos tipos de errores son perjudiciales para el paciente, el cual recibe consecuencias que van desde una medicación o cirugía innecesaria, el

agravamiento de un padecimiento no detectado, secuelas o en algunos casos la muerte. Se estima que entre el 15% y 20% de los errores de diagnóstico causan este tipo de complicaciones en el paciente (Cerian, 2015).

2.5.1 Caso de estudio: Diabetes mellitus

La diabetes mellitus es causada por la deficiencia de producción o nula producción de insulina, lo que produce un trastorno en el uso de la glucosa por parte del cuerpo. Tébar la define como *"Un grupo de enfermedades o síndromes metabólicos caracterizados por la aparición de hipoglucemia secundaria a efectos de la secreción de insulina, de la acción de la insulina o de ambas"* (Tébar & Escobar, 2009)*;* este es un concepto más actual y que engloba la alta complejidad del padecimiento.

La diabetes mellitus es un padecimiento progresivo que no tiene cura y causará un deterioro progresivo en quien la padezca hasta la muerte (Sabán, 2012). El grado de deterioro se reduce significativamente si se aplica un tratamiento, pero puede aumentar con la presencia de otros factores como la presión arterial elevada, la obesidad u otros padecimientos.

La diabetes mellitus causa una afectación universal; es decir, que toda célula del organismo se ve afectada por la alteración metabólica. Esta causa secuelas cuya gravedad depende del grado de la hipoglucemia que presente el paciente, algunas de las secuelas que se presentan con mayor frecuencia son: La pérdida total o parcial de la visión, la afectación renal que puede evolucionar dependiendo del grado de hipoglucemia en insuficiencia renal, la insuficiencia arterial en las extremidades, sobre todo las extremidades inferiores que puede llegar a causar la amputación de estas, el infarto al miocardio y afectaciones en el sistema nervioso central y periférico.

Existen diversas clasificaciones de diabetes mellitus (Díaz, Fernandez, & Parede, 1997). La diabetes mellitus del tipo **I** se caracteriza en que los pacientes no secretan insulina y esta debe ser suministrada bajo estricta supervisión, ya que de suspenderse llevaría al paciente a la muerte.

La diabetes mellitus tipo II se caracteriza en que el organismo de los pacientes tiene un grado de resistencia a la acción de la insulina y este no necesariamente disminuye la producción de esta.

La diabetes gestacional se da cuando el cuadro de diabetes se presenta por primera vez durante el embarazo, después del periodo gestacional se debe evaluar el cuadro para clasificarlo en los dos tipos de diabetes anteriores.

2.5.2 Caso de estudio: Cáncer

El cuerpo humano está compuesto por células, estas realizan funciones específicas, y no migran a otros lugares fuera de su lugar de función, se reproducen y mueren de manera ordenada.

En el proceso de reproducción de la célula, la célula crea una o dos células idénticas con una información genética específica. Sin embargo, puede existir el caso en que se presente una alteración genética por daño o mutación; esto produciría una célula con una formación y comportamiento atípico, la cual se multiplicará e invadirá zonas adyacentes sin control alguno. A esto se le conoce como célula cancerígena (Alatorre, 2004) (Macarulla, Ramos, & Tabernero, 2011).

Una de las funciones de los leucocitos es eliminar las células cancerígenas; sin embargo, algunas veces estas células no pueden ser eliminadas. Cuando ese conjunto de células se reproduce sin control, se comienza a presentar la formación de un tumor maligno.

Las causas por las que la reproducción de una célula resulta en una célula cancerígena pueden ser internas o externas. Las causas externas son factores como la contaminación, radiación, sobrepeso, tabaquismo, exposición a sustancias químicas, etc. que alteran el proceso de reproducción celular.

Las causas internas se refieren a que en la estructura genética del individuo se encuentran oncogenes, que son genes marcados por la ascendencia del individuo que presentó un cuadro de cáncer. El tipo de cáncer no necesariamente es el

mismo en ambas personas, esto quiere decir que puede presentarse en diferentes órganos (Alatorre, 2004).

Cuando se detecta algún crecimiento tumoral o la aparición de sintomatología atípica de una enfermedad, el médico realiza estudios para detectar un posible tumor. De resultar positivo y una vez ubicado, se extrae una pequeña parte de ese tumor para ser analizada por un conjunto de especialistas y determinar un diagnóstico.

De acuerdo a la invasión celular cancerígena, el padecimiento suele clasificarse en cinco etapas (NCI, 2015). La etapa cero se presenta cuando comienza a darse la presencia de células con formación y funciones irregulares, estas aun no pueden ser consideradas como cáncer.

En la primera etapa comienza a presentarse un cúmulo celular irregular o tumor maligno. En la segunda etapa la tumoración es considerable, y esta comienza a extenderse más allá del órgano afectado oprimiendo todos aquellos órganos que lo rodean.

Para la tercera etapa, las células cancerígenas ya han invadido los organismos adyacentes al tumor y su expansión continua. Para la cuarta etapa la extensión de las células cancerígenas ya abarca varios órganos del cuerpo humano, que pueden ser adyacentes o no adyacentes al tumor, dada esta condición se dice que el paciente tiene una condición de metástasis.

Solamente las primeras dos etapas tienen un tratamiento no definitivo en el que es posible erradicar la enfermedad, aunque el cáncer puede reaparecer. Las etapas tres y cuatro reciben tratamiento para mejorar la calidad de vida, por lo que es importante un diagnóstico temprano para evitar la evolución del padecimiento.

Capítulo 3. Descripción del problema y estado del arte

3.1 Descripción del problema

Los problemas de clasificación buscan encontrar un modelo o formulación matemática o lógica robusta y representativa que pueda dividir de manera precisa el espacio de soluciones; con el fin de decidir a qué categoría pertenece cada elemento evaluado.

Esta tarea sin embargo puede tornarse difícil, debido a la complejidad de la información, la ausencia de ella, una mala selección o evaluación de la información, o que el espacio de soluciones se encuentre muy segmentado.

Las redes neuronales representan una herramienta para la evaluación de conjuntos de datos; estas segmentan el espacio de soluciones de acuerdo a una topología de capas y neuronas por cada capa, y estas son ajustadas en un proceso de entrenamiento.

Sin embargo, el proceso de selección del número de capas y neuronas por capa es un proceso manual que se realiza comúnmente a prueba y error para determinar que configuración otorga un mejor resultado. Esto resulta ineficiente y tiende a presentar problemas como lentitud en la evaluación, una división del espacio de soluciones que no es lo suficientemente representativo y que traería como consecuencia un entrenamiento pobre o por el lado contrario, una alta división del espacio de soluciones el cual traería como consecuencia un problema de sobre entrenamiento.

No existen reglas para la elección de una topología de red, ya que eso depende directamente de la información de entrenamiento.

Actualmente se han propuesto varios enfoques, los cuales buscan una sintonización topológica automática, la cual evite los problemas mencionados anteriormente y sea apegada al problema. Típicamente se realiza mediante un proceso de evolución que se explora varios tipos de topología y evalúa para elegir aquella que presente un desempeño superior.

Estos sin embargo pueden sufrir otro tipo de afectaciones. Las topologías generadas durante el proceso de evolución pueden ser, dependiendo el enfoque topologías sin una organización por capas donde en cada generación se pueden agregar o combinar neuronas y arcos.

Esto representa una problemática, debido a que el proceso evolutivo no contempla un ajuste sináptico. Algunos algoritmos plantean un proceso de combinación y mutación sináptica; sin embargo, esto no se realiza por cada topología generada si no tiene cierto nivel de aleatoriedad.

Se plantea entonces, un algoritmo que realice un proceso de sintonización topológica y sináptica automática dependiendo de las características de los datos de entrenamiento. Así, cada elemento generado recibirá un ajuste sináptico y se evaluará de forma más competitiva el despeño de cada topología generada para elegir la topología que presente una mejor división del espacio de soluciones.

3.2 Estado del arte

Dada la descripción del problema presentado y el enfoque de resolución de casos médicos que presenta este trabajo, se realizó una búsqueda bibliográfica de los trabajos que aplican redes neuronales evolutivas para el problema de diagnóstico médico; en especial que utilizaran alguno de los conjuntos de datos que se planeó utilizar en este trabajo.

A continuación, se presentan dos trabajos, los cuales evalúan enfoques de redes neuronales evolutivas para el CONJUNTO DE DATOS DIABETES; además de esto uno de ellos presenta la evaluación de un enfoque clásico y varios enfoques evolutivos para instancias de estudio.

3.2.1 Aplicación de un algoritmo genético para la optimización de pesos de conexión en redes neuronales para el diagnóstico médico de Pima Indians Diabetes

(Gowda, Manjunath, & Jayaram, 2011) En este trabajo donde Gowda propone la hibridación de un algoritmo genético con *Back Propagation* para el entrenamiento de una red neuronal con topología fija. Se encontró que el proceso de hibridación otorga mejores resultados que el aprendizaje *Back Propagation*.

En este trabajo el algoritmo genético es utilizado como un método de diversificación de los valores sinápticos de la red, con el fin de evitar el estancamiento prematuro que produce *Back Propagation*.

La estrategia consiste en proporcionar a *Back Propagation* una población de valores sinápticos para ser ajustados; el resultado obtenido por este algoritmo se evalúa y mientras no cumpla los criterios de paro el algoritmo genético genera una nueva población que será nuevamente evaluada por *Back Propagation*. El algoritmo cumple el criterio de paro cuando se obtiene una buena representación sináptica.

En el trabajo que se describe se implementan estrategias de depuración de entradas neuronales que logran mejorar el desempeño de clasificación, de acuerdo a las conclusiones de sus autores. Sin embargo, estas estas no son del interés de este trabajo pues se busca emular las condiciones de las redes con fines de comparación del desempeño.

3.2.2 Una nueva cruza basada en similitudes para la evolución de redes neuronales artificiales

(Dragoni, Azzini, & Tettamanzi, 2014) En este trabajo, Dragoni propone un nuevo método de cruza topológica, la cual evalúa similitudes para determinar qué soluciones participan en la cruza. Sus resultados muestran que el método de cruza inteligente produce mejores resultados que los métodos: aprendizaje clásico, evolución sináptica y evolución mediante topologías incrementales.

En su trabajo, proponen un método de evolución topológica que conserva la estructura de división por capas. Este tipo de evolución se siguió en lugar del enfoque de topologías incrementales con el fin de aplicar una cruza topológica que, en lugar de involucrar la totalidad de los padres, solo involucra dos capas por cada uno de ellos.

La cruza propuesta por Dragoni la hace asignando un ranking a cada neurona involucrada. Este tipo de cruza inteligente creará las conexiones más significativas para la nueva red neuronal.

Cabe mencionar que cada una de las redes generadas se entrena mediante el algoritmo *Back Propagation* buscando llegar a un buen desempeño sináptico es decir considera solo el ajuste de los pesos.

Otro aspecto relevante de este trabajo es que realiza una evaluación de los enfoques propuestos comparándolos con un conjunto amplio y diverso de elementos disponibles en el estado del arte. Dentro de esos conjuntos se encuentra el conjunto Pima Indians Diabetes que es del interés de este trabajo.

Capítulo 4. Propuesta de solución

4.1 Red neuronal híbrida evolutiva

Una red neuronal típicamente está compuesta por neuronas, conexiones, umbrales, pesos de conexiones, funciones de activación neuronal, topología y un método de aprendizaje de red entre otros elementos. El procedimiento común es la definición topológica manual de neuronas y conexiones y el entrenamiento regularmente se realiza con Backpropagation. Terminando este proceso, la red esta lista y sintonizada para procesar nuevos casos y dar una respuesta a ellos.

En este trabajo de tesis, el enfoque de solución del problema de diagnóstico médico se utilizando redes neuronales evolutivas en las cuales tanto la topología como los valores sinápticos son elementos a ajustar que dependen de los datos de entrenamiento del problema.

4.1.1 Doble evolución como mecanismo de aprendizaje

El algoritmo desarrollado incluye un modelo de doble evolución anidada que utiliza un algoritmo genético para evolucionar las topologías de una población de redes neuronales; cada topología generada realiza el ajuste de los pesos mediante un proceso evolutivo que se implementa mediante la metaheurística de búsqueda dispersa. El algoritmo genético diseñado promueve un ajuste topológico que evita el sobre entrenamiento al trabajar con topologías diversas y mejora el ajuste neuronal de los pesos que es un problema de las redes neuronales clásicas. El ajuste de los valores sinápticos obtenido mediante búsqueda dispersa remplaza a la búsqueda local Backpropagation y añade características que permiten escapar de valores óptimos locales.

Ambos procesos evolutivos funcionan como un mecanismo de aprendizaje de la red, ya que ajustan su estructura y valores a la información de entrenamiento.

4.2 Tratamiento del conjunto de datos

El objetivo de tratar la información es homogeneizar los datos de entrada para que todos los valores se encuentren dentro de un rango limitado, así como hacer las

codificaciones pertinentes de la información que no puede ser tratada como un valor numérico. Estos procesos se realizan mediante una operación de normalización y cuando es necesario un filtrado.

El algoritmo obtiene la información para entrenamiento, validación y pruebas de un conjunto de datos, este conjunto se introduce en el algoritmo en forma de archivo de texto en formato *arff* propuesto por *weka*, el cual será descrito en la siguiente sección (Waikato, 2002).

4.2.1 Formato de lectura

Como se mencionó en la sección anterior, el conjunto de casos con el que se entrena y evalúa el algoritmo se deben introducir a este mediante un fichero de texto plano con formato y extensión *arff*.

El formato arff es un formato propuesto por la Universidad de Waikato para el software *Weka*; se utiliza este formato debido a que los conjuntos de datos del estado del arte se encuentran reportados bajo este formato (Waikato, 2002).

En este tipo de fichero el modo en que la información se presenta permite que el conjunto de datos sea fácilmente entendible. En el *Algoritmo 1* se describen las secciones del fichero.

```
@relation automovil
@attribute 'cilindrada' numeric
@attribute 'largo' numeric
@attribute 'ancho' numeric
@attribute 'ejes' numeric
@attribute 'class' {auto, camión}
@data
12,232,4213,22,auto
14,563,345,877,auto
1947,503,280,2938,camión
```

Algoritmo 1. Estructura general de archivos ARFF.

Las líneas que comienzan con el carácter **%** son líneas de comentarios, las cuales son ignoradas al momento de realizar la lectura de datos. La etiqueta ***@relation***

describe el nombre de la relación o conjunto de datos. Con esta etiqueta se comienza la lectura y en líneas subsecuentes se encuentran los atributos del conjunto.

Los atributos son identificados por la etiqueta ***@attribute***, la cual indica entre comillas simples el nombre del atribute seguido del tipo de dato al que corresponde ese atributo; por ejemplo, numérico, nominal, fecha, etc.

Es importante que el atributo de clase se coloque al final de la lista de atributos, y su identificador es ***class***.

La etiqueta ***@data*** marca el inicio de los casos; es decir, después de la etiqueta solo va la información de los casos. Los atributos de cada caso son separados por una coma simple, y cada caso es separado de otro a través de un *"Enter"*. En caso de ausencia de un valor, este no debe omitirse ya que provocaría errores en la ejecución del algoritmo. En estos casos se utiliza el símbolo "***?***", el cual indica el desconocimiento del valor del atributo para ese caso determinado.

4.2.2 Transformación de atributos categóricos

Se consideran variables o atributos categóricos a todos aquellos atributos que pueden tomar un valor que se encuentra definido en un rango de valores no continuo; estos pueden ser numéricos o nominales (Center, s.f.).

Un ejemplo de este caso se presenta con el atributo "*embarazo"*, este indica si una mujer se encuentra en estado de embarazado o no, y sus posibles valores se representan como: ***embarazo* {Si, No}**.

El convertir este atributo a un atributo numérico no reflejaría la representación real del atributo, ya que no se puede representar en una numeración continua si una mujer se encuentra embarazada o no se encuentra.

La estrategia para tratar este tipo de atributos es dividirlo en tantas secciones como opciones tenga el atributo, por eso es necesario que el atributo tenga especificado todos los posibles valores que pueden ser asignados.

Así, el atributo embarazo se dividiría en el atributo ***embarazoSi*** y **embarazoNo**; estos se tratarían como valores binarios mutuamente excluyentes entre sí; es decir, cuando uno de ellos adquiera el valor de 1, el otro no puede adquirir ese valor y adquiriría el valor de 0.

4.2.3 Normalización de datos

El proceso de normalización de los datos consiste en llevar a cada atributo a un rango definido. Los datos pueden o no tener picos o distancias muy grandes entre ellos, lo que dificulta su tratamiento y procesamiento.

El funcionamiento de las redes neuronales es a través de funciones de entrada, activación y umbrales de propagación; estos suelen trabajar en rangos de [-1,1] o [0,1], por lo que valores grandes pueden inhabilitar partes de la red neuronal y el resultado sería un enteramiento y desempeño pobre de la red.

En el algoritmo desarrollado los valores se normalizaron en el rango de [0,1]. El proceso de normalización se realizó por cada atributo, uno a la vez. Primero se obtienen los valores máximos y mínimos del atributo; enseguida, se aplica la fórmula de *normalización básica a la unidad* mediante la *ecuación 17*.

$$v' = \frac{v - V_{min}}{V_{max} - V_{min}}$$

Ecuación 17. Formula de normalización básica a la unidad

Dónde:
v' es el valor normalizado
v es el valor original
V_{max} es el valor máximo del atributo
V_{min} es el valor mínimo del atributo

De esta manera se garantizó que todos los valores de entrada serán procesables en la red neuronal. En este trabajo, al tratarse clasificaciones categóricas es innecesario el proceso de des-normalización.

4.2.4 Representación de la información

A partir de la lectura del archivo de texto y el tratamiento de atributos nominales, los datos son almacenados en una matriz numérica que tiene como dimensión en columnas el número de atributos totales y como dimensión en filas el número de casos leídos. Los nombres de los atributos son almacenados en un vector que tiene como longitud el número de atributos totales.

La matriz después de ser leída entra a un proceso de “agitación”. En un proceso de agitación que consiste en producir una distribución lo más aleatoria posible sin que se pierda ninguno de los valores de sus atributos. Mediante este proceso se evita el fenómeno de datos ordenados que conllevan a un pobre desempeño de la red neuronal.

La representación en forma de matriz permite tener un acceso más eficiente a los datos. Esto se ve reflejado en un menor tiempo de ejecución del algoritmo. La matriz trabaja con datos del tipo *double* cuyo espacio de ocupación en memoria depende directamente de la cantidad de atributos y casos a tratar en el conjunto de datos.

Aunque el conjunto de datos pertenece a una sola matriz, se definen índices de acceso; es decir, se divide en bloques. En este caso la matriz se dividió en dos bloques uno para entrenamiento y otro para validación. Cada parte del algoritmo solo tiene acceso al bloque correspondiente y el tamaño de cada bloque se toma de acuerdo al tamaño recomendado por la literatura especializada del área.

4.3 Definición de la población neuronal

El algoritmo consta de una población de redes neuronales, la cual evoluciona a lo largo de diversos ciclos y generaciones de un algoritmo genético. La población se define independientemente del algoritmo y se mantiene, aunque termine algún ciclo del algoritmo genético o algún bloque de validación cruzada; es decir, la población nace antes del aprendizaje evoluciona con este y no se elimina al terminar este.

La población de redes neuronales está constituida por un conjunto de redes neuronales que comparten características con respecto a la capa de entrada y salida, pero no necesariamente lo hacen en la capa o capas intermedias o en los valores sinápticos.

El algoritmo usa evolución topológica mediante topologías incrementales; es decir, inicia con topologías mínimas o pequeñas, y estas crecen ajustándose al problema. Las topologías incrementales además hacen eficiente el procesamiento debido a que al inicio los tiempos son reducidos y solo se requerirán tiempos altos cuando las topologías han evolucionado a topologías grandes. Aunado a esta mejora en reducción de tiempo se busca un buen ajuste sináptico desde topologías pequeñas con el fin de evitar el sobre entrenamiento del algoritmo.

Cuando se genera la población inicial, se construye un conjunto de redes neuronales básicas con topologías idénticas, pero con valores sinápticos aleatorios. Dicho de otra forma, se crean redes neuronales totalmente conectadas y sin capas ocultas, y el peso de cada conexión se establece de manera aleatoria entre [0,1].

En la *Figura 6* se muestra un ejemplo de la definición de la población inicial.

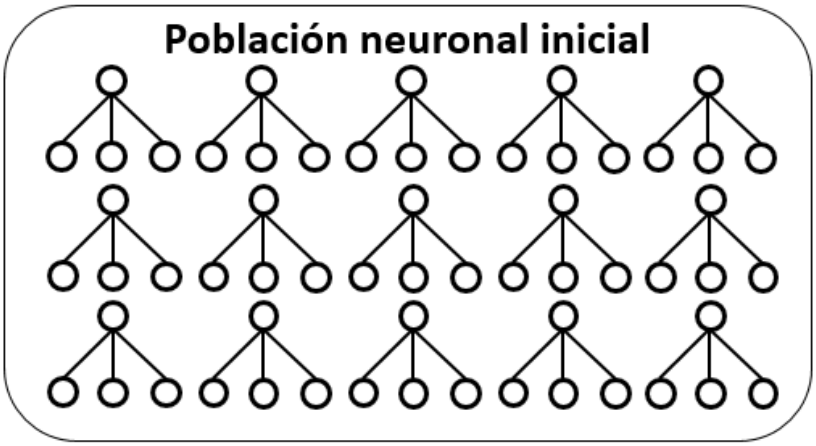

Figura 6. Representación de la población inicial de redes neuronales

4.3.1 Estructura de red neuronal

La red neuronal es una estructura de neuronas y arcos que, dado un valor de entrada produce un valor de salida. Internamente la red neuronal tiene estructuras,

tanto de organización topológica para propagar la información, como neuronal para transformar un valor de entrada en un valor de salida.

Cada neurona de la red con excepción de las neuronas de la capa de salida, tiene al menos un arco de incidencia hacia otra neurona de la red. Este arco representa una relación de conexión en la que se propagan los valores. Debido al tipo de estructura de la red, una neurona no incidir en una neurona que tenga incidencia con ella o con sus antecesoras, además una neurona no puede tener incidencia hacia sí misma.

Estas restricciones se hacen con el fin de garantizar la integridad de la red neuronal; evitando comportamientos cíclicos o redundantes. Este tipo de red solo acepta conexiones hacia delante y se le conoce como red feedforward (Mathivet, 2017).

Con respecto a la estructura neuronal, cada neurona tiene la tarea de transformar un valor o conjunto de valores que son adyacentes a ella, en un valor de salida.

Como se describió en el marco conceptual, cada neurona tiene dos funciones de tratamiento de la información, la función de entrada y la función de activación neuronal.

Este algoritmo utiliza la función de entrada la función sumatoria y de activación la función sigmoidea como se describen en la *ecuación 18*.

$$V_{sj} = \frac{1}{1 + e^{-(\sum V_{eij} * w_{eij} - \theta)}}$$

Ecuación 18. Formula de entrada y activación neuronal

Donde:

$Veij$ = Valor entrada de neurona i a neurona j

$Weij$ = Valor sináptico de neurona i a neurona j

Vsj = Valor salida de la neurona j

4.3.2 Codificación de una red neuronal en un genotipo

En el proceso evolutivo, es necesario codificar los objetos a evolucionar para llevar a cabo operaciones de una manera más sencilla. Trasladando los conceptos de

algoritmos evolutivos a este problema, se debe realizar una codificación de la red neuronal a modo de genotipo o cromosoma, con el fin de poder utilizar los operadores genéticos.

Dicho esto, se puede considerar a una red neuronal como un fenotipo, y a la codificación de esta en una estructura para el algoritmo como un genotipo. En la *Figura 7* se ilustra el enunciado.

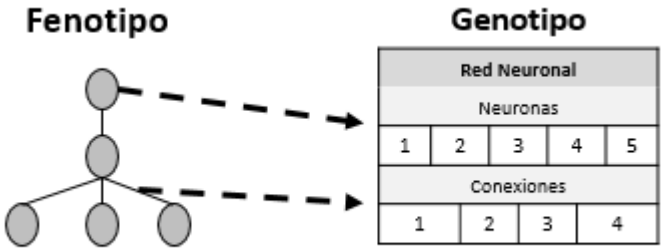

Figura 7. Codificación de un fenotipo a un genotipo de red neuronal.

La estructura del genotipo propuesto contiene dos bloques de información en su interior. Cabe destacar que no se utiliza un vector simple como representación de la red neuronal, si uno una estructura representativa. En la *Figura 8* podemos observar la estructura del genotipo propuesto.

Red Neuronal			
Neuronas			
ID: 1 ID Hist: 1 Tipo: Entrada	ID: 2 ID Hist: 2 Tipo: Entrada	ID: 3 ID Hist: 7 Tipo: Oculta	ID: 4 ID Hist: 3 Tipo: Salida
Conexiones			
ID Origen: 1 ID Destino: 4 Peso: 0.4724 Habilitada: Si	ID Origen: 2 ID Destino: 4 Peso: 0.3278 Habilitada: No	ID Origen: 2 ID Destino: 3 Peso: -0.4835 Habilitada: Si	ID Origen: 3 ID Destino: 4 Peso: 0.9264 Habilitada: Si

Figura 8. Representación de una red neuronal en un genotipo de red.

El primer bloque del genotipo almacena todas y cada una de las neuronas de la red. Cada neurona almacena la información inherente a ella: su identificador, el identificador histórico y el tipo de neurona.

El identificador de la neurona es un número entero único en la red, el propósito del número es ubicar los arcos que llegan o salen de esa neurona. El identificador histórico hace referencia a la aparición de esa neurona en el proceso de evolución, es un identificador global e irrepetible en la población; así cada que una nueva neurona es generada, se asigna un identificador histórico superior al identificador asignado anteriormente. El tipo de neurona describe si la neurona se trata de un nodo de entrada, de salida o de capa oculta.

El segundo bloque del genotipo almacena todas y cada una de las conexiones neuronales de la red. Cada conexión almacena información inherente a ella como la neurona de origen de la conexión, la neurona destino de la conexión, el peso sináptico y si la conexión se encuentra habilitada o deshabilitada.

La neurona de origen indica el punto de partida de la conexión sináptica. La neurona destino indica el punto de llegada de la conexión sináptica. Ambas hacen referencia a una neurona que debe estar especificada en el bloque de neuronas; además deben cumplir con la estructura neuronal feedforward que restringe las conexiones para que solo tengan un flujo hacia adelante, con lo que no es posible que existan conexiones a neuronas anteriores o a sí mismas.

El valor del peso sináptico es la influencia o la importancia de ese arco sináptico, su intervalo de valores va de [-1,1] e inicia aleatoriamente. Cada arco puede estar habilitado o deshabilitado; se dice que un arco esta deshabilitado cuando existe registro en el genotipo de su existencia, sin embargo, no es tomado en cuenta en la evaluación de la red.

4.4 Entrenamiento de la red neuronal

Definida la topología de una red neuronal, esta puede comenzar a dar respuesta a entradas dadas; sin embargo, estas respuestas muy probablemente sean valores lejanos a los deseados, es por eso que la red neuronal debe sufrir un proceso de entrenamiento para que las respuestas sean más favorables.

En este algoritmo, el proceso de entrenamiento ajusta la cantidad de neuronas, sus posiciones en la red, la cantidad de conexiones y los valores sinápticos. Así, a diferencia de enfoques típicos donde solo se ajustan valores sinápticos para obtener buenos resultados, se ajusta la topología y los valores sinápticos.

Un fenómeno que puede presentarse al momento de realizar el entrenamiento de un algoritmo es el sobre entrenamiento; este sucede cuando el algoritmo se ajusta demasiado a la información dada para entrenar, o esta no es lo suficientemente representativa del espacio de soluciones. Este problema nos llevaría obtener un buen desempeño con el conjunto de datos de entrenamiento, no así para cualesquier otros conjuntos ya sea de validación o prueba.

El entrenamiento de la red ayudará a encontrar una buena configuración topológica y sináptica. Para evitar el problema de sobre aprendizaje se utilizan estrategias como la validación cruzada que se describe a continuación.

4.4.1 Entrenamiento por validación cruzada

El objetivo de la validación cruzada es hacer una evaluación interna con casos del conjunto de entrenamiento, los cuales son desconocidos para el algoritmo, esto con el fin de medir el desempeño.

La validación cruzada divide en N bloques el conjunto de casos de entrenamiento; después hará N ejecuciones donde cada ejecución toma N-1 bloques, estos sirven como conjunto de entrenamiento y el bloque sobrante correspondiente a la iteración de la ejecución funciona como bloque de validación. Terminadas las ejecuciones se obtiene un promedio del desempeño de acuerdo a los resultados de la validación. El procedimiento se ilustra en la *Figura 9*.

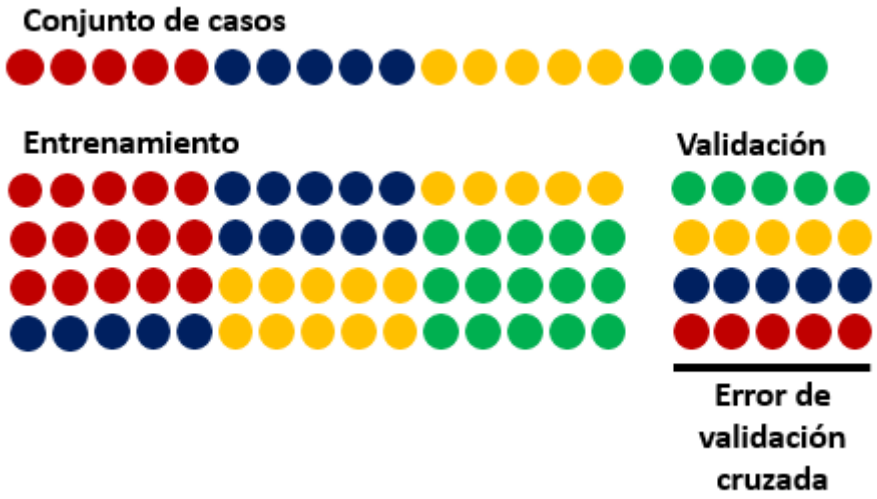

Figura 9. Representación de la evaluación mediante la técnica de validación cruzada.

Cabe mencionar que el proceso de validación es interno a la validación cruzada y no es el mismo que el proceso de validación general del algoritmo. El intercalar los bloques en el proceso de entrenamiento ayudará a evitar el sobre aprendizaje y estancamiento prematuro del algoritmo.

En cada evaluación de bloques del proceso de validación cruzada, se llama al algoritmo genético encargado de realizar la evolución topológica, este proceso se describe a continuación.

4.6 Algoritmo genético de evolución topológica

El proceso de evolución topológica se realiza a través de un algoritmo genético. Este es el encargado de explorar nuevas topologías a partir de mutaciones (perturbaciones) y cruzas (combinaciones) de estas.

El algoritmo trabaja con la población general y su número de generaciones debe sintonizado junto con los demás parámetros del algoritmo.

Por cada generación de individuos, se obtienen ***N*** cantidad de hijos donde ***N*** es el tamaño de la población, así al final de cada generación se obtendrán tantos hijos como padres hay en la población.

A continuación, se describen los procesos que se realizan dentro del Algoritmo Genético.

4.6.1 Selección de padres

El proceso de selección de padres consiste en elegir dos genotipos de la red neuronal, los cuales fungirán como padres en la generación de un nuevo individuo.

Esto se hace a través de la técnica de selección mediante torneo binario, en la cual se eligen los padres de acuerdo a una comparación de su calidad. Para elegir un padre se toman de forma aleatoria dos genotipos de la población neuronal, se debe validar que los genotipos seleccionados sean diferentes, se compara la calidad de ambos y se elige como padre aquel que tenga mayor calidad, en caso de empate se elige uno aleatoriamente. Para la elección del segundo padre se realiza la misma operación y se verifica que los dos padres seleccionados no tengan el mismo genotipo.

El procedimiento de selección descrito se ve ilustrado en la *Figura 10*. Cabe destacar que el procedimiento de selección binaria nos garantiza que existe diversidad en los hijos generados, además de evitar el estancamiento prematuro. Así, cualquier miembro de la población tiene posibilidad de ser seleccionado como padre en la generación de un nuevo individuo; con excepción del peor elemento de la población.

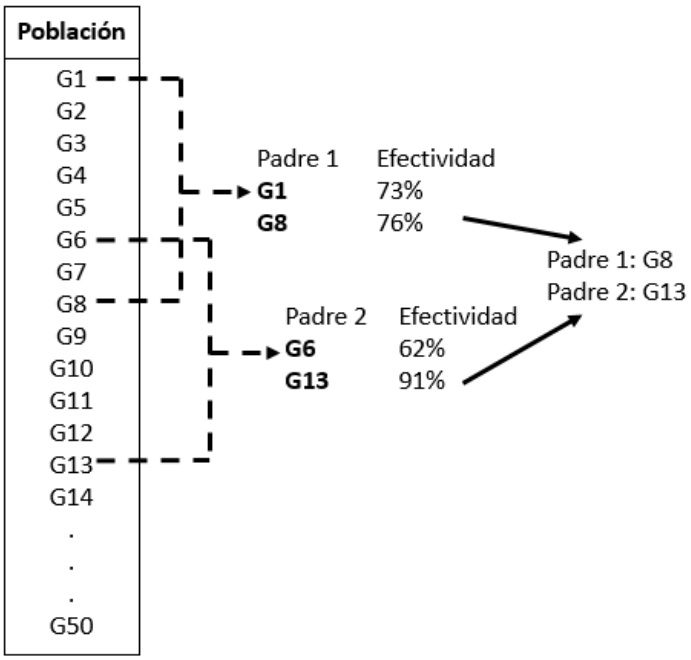

Figura 10. Representación de la selección evolutiva de genotipos de red

4.6.2 Procedimiento de cruza

El procedimiento de cruza consiste en la generación de una nueva topología hija, la cual está constituida a partir de la información genética de los padres seleccionados. El objetivo de este proceso es la combinación de las estructuras con el fin de poder explorar el espacio de soluciones.

Como se mencionó, la nueva topología generada tiene una estructura que se obtuvo a partir de los padres seleccionados. Se debe tener especial cuidado ya que una combinación de los padres podría generar topologías infactibles y conducir al algoritmo a un error; para ello se propone una cruza basada en identificadores neuronales históricos y una revisión exhaustiva topológica.

Una vez que se obtienen los dos padres, se genera una nueva topología hija, al generarla solo se agregan las neuronas de entrada y salida debido a que estas son exactamente las mismas para cada una de las topologías. Acto seguido se agregan todas las neuronas de ambos padres se analiza su número histórico y en caso de que ambas tengan el mismo número histórico, solo se agregan una sola vez.

Hasta este punto, no se tiene ninguna conexión neuronal, solo se cuenta con un conjunto de neuronas en la red hija. Como paso siguiente se analizan los arcos de ambos padres, con el fin de darle conectividad a la red hija.

Se procede a copiar todos los arcos de uno de los dos padres, esto se hace haciendo referencia a los identificadores históricos de cada neurona para garantizar la integridad de la red neuronal. Deben copiarse en su totalidad no importando si estos están habilitados o deshabilitados.

En esta fase la red neuronal ya tiene algunas conexiones, pero puede darse el caso que algunas neuronas aún se encuentren no conexas; en este caso se conectan en la siguiente operación del procedimiento.

Se analiza cada arco del siguiente padre. En caso de no existir en la red neuronal hija se analiza si genera una red infactibles, en tal situación no se agrega el arco a la nueva topología y en otro caso se añade el correspondiente arco.

Si el arco ya existe en la red hija, se compara con este y si alguno de los dos está deshabilitado por defecto la red hija tendrá ese acto deshabilitado, en caso contrario la red hija adquiere aleatoriamente uno de los dos arcos. El proceso se ilustra en la *Figura 11*.

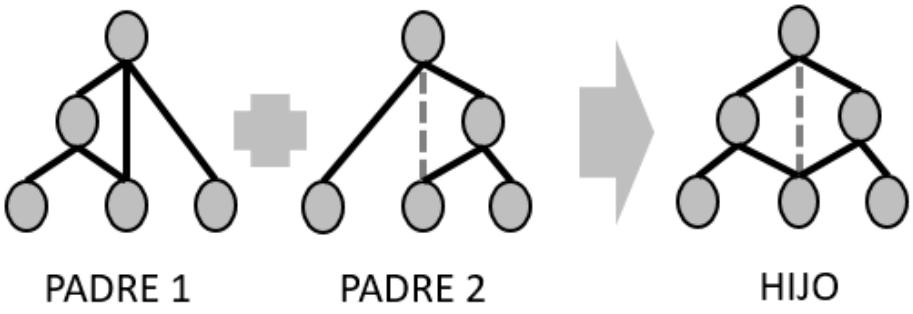

Figura 11. Representación del procedimiento de cruza topológica

El procedimiento anterior garantiza la obtención de una red neuronal hija integra que tiene todas sus neuronas conexas. Este tipo de cruza topológica se realiza mediante la estructura de genotipos y produce un nuevo genotipo.

4.6.3 Perturbación topológica

Este proceso es equivalente a la de mutación que se aplica en el campo de los algoritmos genéticos, su objetivo es hacer una pequeña alteración a la topología generada en el proceso anterior, con el fin de explorar el espacio de soluciones de una mejor manera. Por otro lado, este proceso es el que promueve principalmente la expansión de topologías al incluir mecanismos que agregan neuronas y conexiones dentro de la red.

La perturbación no se aplica a todas las topologías hijas generadas, sino que se realiza de manera aleatoria de acuerdo a un porcentaje definido que constituye un parámetro a sintonizar para el algoritmo.

Se definieron tres tipos diferentes de perturbación sináptica: 1. agregar una conexión sináptica, 2. agregar una neurona, y 3, eliminar una neurona. Estas se describen más adelante.

Cuando una red neuronal entra al proceso de perturbación y es elegida para participar en él se selecciona de forma aleatoria alguno de los métodos de perturbación mencionados en el párrafo anterior. Los tres métodos tienen la misma probabilidad de ser elegidos y solamente se aplica uno de los tres.

La primera perturbación consiste en intentar añadir un arco inexistente en la red. Esta perturbación no siempre se puede realizar, ya que podría generar topologías infactibles o intentar añadir una conexión ya existente. Para esto se aplican diversos tipos de validación dentro del procedimiento.

En la primera etapa del procedimiento, se seleccionan aleatoriamente dos neuronas de la topología, enseguida se verifica si existe un arco entre ellas, si existe entonces el procedimiento termina ya que no tiene sentido agregarlo; en caso de ser negativo, se puede añadir un arco, pero debe verificarse que esa conexión no cause un ciclo en la red. En caso de pasar la verificación se agrega un nuevo arco entre esas neuronas con un valor sináptico aleatorio. El procedimiento se ilustra en la *Figura 12*.

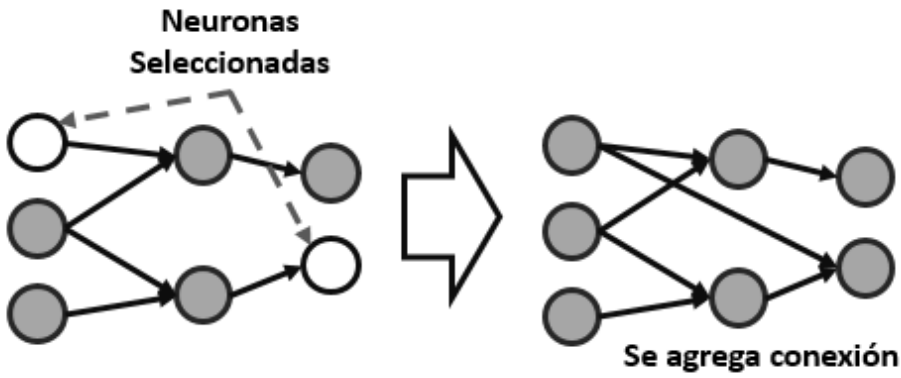

Figura 12. Perturbación mediante incorporación de un arco en la red

La segunda clase de perturbación consiste en añadir una nueva neurona a la red. Esta perturbación no tiene restricción para su aplicación y aplica en cualquier topología, ya que no se corre el riesgo de afectar la integridad estructural de la red neuronal. Consiste en elegir aleatoriamente un arco de la red, y añadir una nueva neurona a la red neuronal. El arco seleccionado se deshabilita de la red neuronal; y el origen del arco deshabilitado será el origen de un nuevo arco hacia la nueva neurona; de igual manera, el destino del arco deshabilitado se convertirá en el destino de un nuevo arco que parte de la nueva neurona hacia este. Los nuevos arcos creados tendrán valores sinápticos aleatorios. El proceso se ilustra en la *Figura 13*.

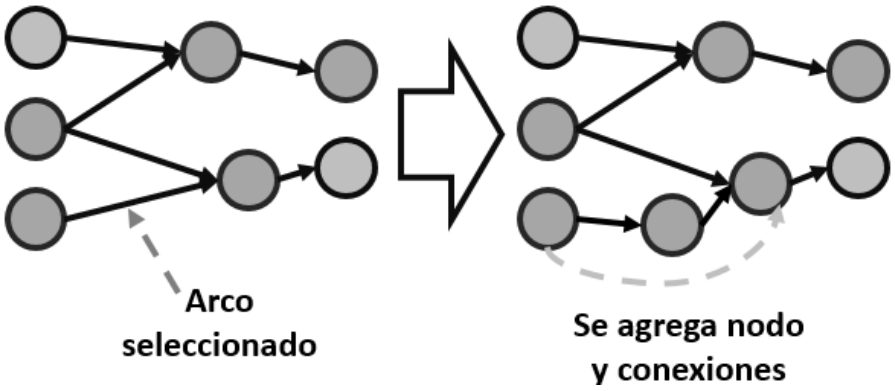

Figura 13. Perturbación por incorporación de una neurona en la red

La tercera clase de perturbación consiste en eliminar una neurona de la red neuronal. Esta perturbación es posible realizarla con la restricción de eliminar solo

neuronas que pertenezcan a la capa oculta y que además la eliminación no provoque la desconexión de otras neuronas.

La eliminación de una neurona consiste en seleccionar de forma aleatoria una neurona de la capa oculta, como siguiente paso se verifican las neuronas que inciden a la neurona seleccionada, si alguna de ellas pierde la conexión con el resto de la red al eliminar esa neurona no es posible realizar la eliminación y el proceso termina. De la misma manera se comprueban las neuronas con las cuales la neurona seleccionada tiene una conexión hacia delante, si estas solo reciben conexión de la neurona seleccionada no es posible realizar la eliminación y el proceso termina.

En caso de ser candidata a ser eliminada, la neurona y los arcos que llegan y salen de ella son eliminados de la red neuronal. El proceso de eliminación se ilustra en la *Figura 14*.

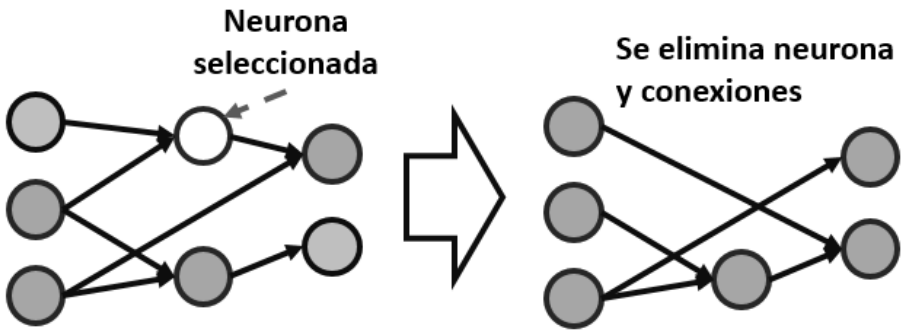

Figura 14. Perturbación mediante eliminación de una neurona

4.6.4 Ajuste sináptico

Hasta este punto se ha generado una topología hija mediante la cruza, la cual pudo haber pasado por un procedimiento de perturbación sináptica; sin embargo, esto no garantiza que los valores sinápticos adquiridos en los dos procedimientos anteriores sean buenos.

Dicho de otra forma; la generación de una topología y la perturbación pueden tomar valores sinápticos buenos, regulares o malos. El principal problema de esto es que

una topología mala puede tomar valores sinápticos buenos, lo que la llevaría a competir por permanecer en la población; de la misma manera una topología buena puede tomar un ajuste sináptico malo, por tanto, su desempeño no será bueno y podría quedar eliminada de la población.

Para resolver este problema se plantea una técnica de ajuste sináptico, que toma la topología generada y le realiza un ajuste de valores sinápticos con el fin de mejorar la calidad de cada topología y así hacerla competitiva en la población.

Para implementar este proceso de ajuste se desarrolló un algoritmo meta heurístico poblacional evolutivo que garantiza obtener una solución de buena calidad en un periodo de tiempo razonable. Este procedimiento de ajuste sináptico se trata con mayor profundidad en la sección "*4.7 Algoritmo de Búsqueda Dispersa"* descrito en este trabajo.

4.6.5 Actualización de la población

Una vez que se produce la siguiente generación los hijos generados son seleccionados e insertados en una tabla, de igual manera los elementos de la población del algoritmo son seleccionados e insertados en la misma tabla, de aquí se seleccionan los elementos que constituirán la población para la siguiente generación algorítmica.

La tabla que contiene a la totalidad de los individuos debe ser ordenada, el criterio de ordenamiento considera el porcentaje de efectividad en la clasificación de los casos de entrenamiento; estos son ordenados en forma descendente, es decir, de la mejor efectividad hacia la peor efectividad.

Después de ordenar la tabla se seleccionan ***N*** cantidad de individuos, donde ***N*** es el tamaño de la población de genotipos; los elementos seleccionados se toman de la parte superior de la tabla ordenada. De esta manera, solo permanecerán en la población aquellos elementos cuya calidad sea buena.

Cabe mencionar que el proceso sináptico puede producir redes con topologías idénticas y valores sinápticos idénticos; en ese caso se elimina una de ella

remplazándola por una topología básica con valores sinápticos aleatorios evitándose la convergencia prematura. Este proceso de remediación se realiza en caso de ser necesario previo a la evaluación y selección de los elementos de la población.

4.6.6 Criterios de terminación del algoritmo

Se definieron dos criterios de terminación del algoritmo genético, uno es el número de generaciones y el otro es el cumplimiento de un numero de generaciones sin mejora del mejor elemento de la población.

Dentro de los parámetros a ajustar en el algoritmo, uno de ellos es el número de generaciones genéticas. El algoritmo genético producirá en total tantas generaciones como se especifique, a menos de que la condición que verifica la mejora termine la ejecución del algoritmo.

El criterio de iteraciones sin mejora se establece debido a que el algoritmo pudo haberse estancado o encontrado una solución difícil de superar en un periodo de tiempo. En caso de que la mejor solución no cambie o mejore en un número específico de iteraciones a sintonizar, el algoritmo se dará por terminado.

4.6.7 Acotamiento topológico

El tipo de evolución topológica utilizada añade neuronas y arcos; esto lo hace de manera aleatoria y no mediante división de capas; así, todas las neuronas tienen posibilidad de crecer lo que favorece un crecimiento horizontal y vertical de la red. Un ejemplo de topologías generadas se muestra en la *Figura 15*.

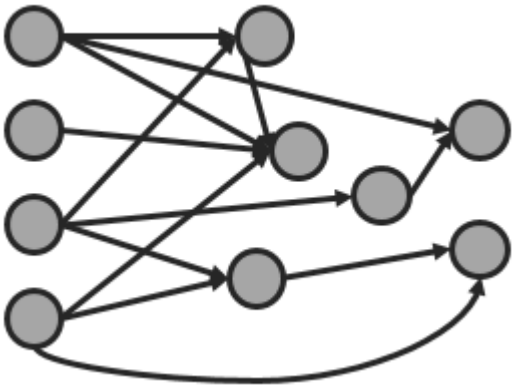

Figura 15. Ejemplo de topología generada en el proceso evolutivo.

Cuando los tiempos de evolución son largos, las topologías tienden a hacerse muy grandes. Este tipo de crecimiento representa un problema, debido a que los tiempos de procesamiento son mayores y el espacio de soluciones se secciona de tal manera que podría presentarse un problema de sobre entrenamiento.

Por ello, se establecieron límites que acotan de tamaño de las topologías, que si bien no limitan hacia donde crece la red neuronal, si limita el número de neuronas dentro de la topología; estos límites bloquean la red neuronal para que, en caso de alcanzar el límite de neuronas se reinicie la topología convirtiéndose en una topología básica.

El reinicio de la topología se realiza para dar a la población mayor diversidad y evitar que los demás individuos sean saturados con un gran número de neuronas, ya que el individuo saturado podría formar parte en un procedimiento de cruza.

4.7 Algoritmo de Búsqueda Dispersa

Para implementar el proceso de ajuste sináptico se desarrolla un algoritmo de búsqueda dispersa. Este algoritmo evolutivo utiliza una población de trabajo que se genera al iniciar el algoritmo y se destruye cuando el algoritmo termina.

El algoritmo diseñado obtiene un vector de valores sinápticos acorde a la topología, mediante mecanismos de combinación y búsqueda local. Además, incorpora

estrategias de paralelización para mejorar la velocidad de procesamiento del algoritmo. A continuación, se describen los procesos que se llevan a cabo dentro del algoritmo.

4.7.1 Representación de valores sinápticos en el algoritmo

Los valores sinápticos de la red neuronal son codificados en un vector, para lo cual se crea un vector sináptico de tamaño ***N***, donde ***N*** representa el número de conexiones sinápticas de la red. Cada posición del vector representa el valor de un arco de la red. En la *Figura 16*, se ilustra la equivalencia de los valores sinápticos de la red con el vector sináptico.

Figura 16. Codificación de los valores sinápticos en un vector numérico

4.7.2 Generación del conjunto inicial *P*

El *conjunto P* está formado por un grupo de vectores sinápticos iniciales, los cuales sirven para construir el conjunto de referencia *RefSet* del algoritmo. Este conjunto está compuesto por un número determinado de vectores, el tamaño del conjunto es un parámetro a sintonizar en el algoritmo.

El primer elemento del conjunto corresponde a los valores sinápticos de la red sobre la que se trabajará. Aunque no es necesario que se tomen en cuenta esos valores, es de utilidad añadirlos ya que podrían ser valores de buena calidad que acerquen al algoritmo a una buena solución.

Los elementos restantes del conjunto son generados aleatoriamente en el rango de (-1,1).

Una vez que se construye el conjunto se hace una evaluación del porcentaje de efectividad en la clasificación de los casos de entrenamiento, asignando a la topología los vectores sinápticos y evaluando su efectividad. Así, se registra la efectividad de cada vector en la red neuronal.

4.7.3 Selección del conjunto de referencia *RefSet*

El conjunto de referencia *RefSet* es el conjunto con el cual trabaja el algoritmo, su tamaño es definido como un parámetro ajustable en el algoritmo y tiene la particularidad de ser un conjunto élite y disperso.

Se inicia construyéndolo a partir del conjunto **P**; donde la mitad de RefSet contiene los vectores sinápticos con mejor efectividad de P. La segunda mitad se construye midiendo la dispersión de los elementos restantes en P con respecto a los elementos actuales en RefSet; aquellos elementos que tengan un mayor valor de dispersión serán seleccionados para pertenecer al conjunto de referencia.

El valor de dispersión medio de la diferencia absoluta de cada posición del vector con respecto a un vector del conjunto de referencia se utiliza para constituir la segunda mitad del conjunto de referencia. Esta medida se ilustra en la *ecuación 19*. El valor de dispersión total del vector es la media de todas las dispersiones calculadas para ese vector.

$$vm = \frac{\sum_{i=0}^{N}(|V_i^a - V_i^e|)}{N}$$

Ecuación 19. Evaluación del grado de dispersión de un vector

Donde:

vm = Valor de dispersión medio.

V_i^a = Valor del vector actual en la posición i

V_i^e = Valor del vector evaluado en la posición i

N = Tamaño del vector

Este conjunto no es definitivo, si no que se actualiza a lo largo de la ejecución del algoritmo.

4.7.4 Proceso de evolución

Una vez que se obtiene el conjunto de referencia, el algoritmo inicia el ciclo de evolución. En esta etapa el conjunto de referencia será sometido a operaciones para mejorar las soluciones.

El ciclo evolutivo no tiene un límite de generaciones y su condición de parada consiste en verificar un numero de iteraciones sin mejora de la mejor solución.

Cada iteración producirá tantos hijos como parejas de soluciones sean generadas y ambos competirán para tomar un lugar en el conjunto de referencia. A continuación, se detallan los procedimientos utilizados.

4.7.5 Selección de parejas

En la selección de parejas se realizan todas las posibles combinaciones de dos soluciones tomadas del conjunto de referencia con excepción de las parejas donde ambas soluciones son iguales.

El proceso de generación de parejas a pesar de ser exhaustivo no resulta costoso computacionalmente, esto es debido a que el conjunto RefSet de donde se toman las soluciones es un conjunto de dimensión muy reducida, regularmente de 10 elementos (Laguna & Cunquero, 2003).

4.7.6 Combinación y búsqueda local mediante paralelismo

Una vez generadas las parejas, se realiza un proceso de combinación y de búsqueda local. Este proceso se aplica para cada una de las parejas generadas y típicamente se realiza de manera secuencial, es decir cuando se termina de procesar una pareja se prosigue con la siguiente hasta terminar con ellas. Sin embargo, en el algoritmo diseñado el proceso de combinación de parejas y mejora mediante búsqueda local se realizó por lotes con el fin de aprovechar la ventaja de aplicar estos procesos en forma paralela a cada una de las parejas del lote y seleccionar al final las mejores 5 soluciones considerando su calidad y las 5 soluciones más diversas.

La estrategia para implementar el paralelismo en estas operaciones representa un ahorro considerable de tiempo; además gracias a la estrategia utilizada no se depende de la arquitectura ni de la cantidad de núcleos por procesador o cantidad de procesadores. La *Figura 17* se ilustra el proceso paralelo para implementar la combinación y búsqueda local de las parejas generadas.

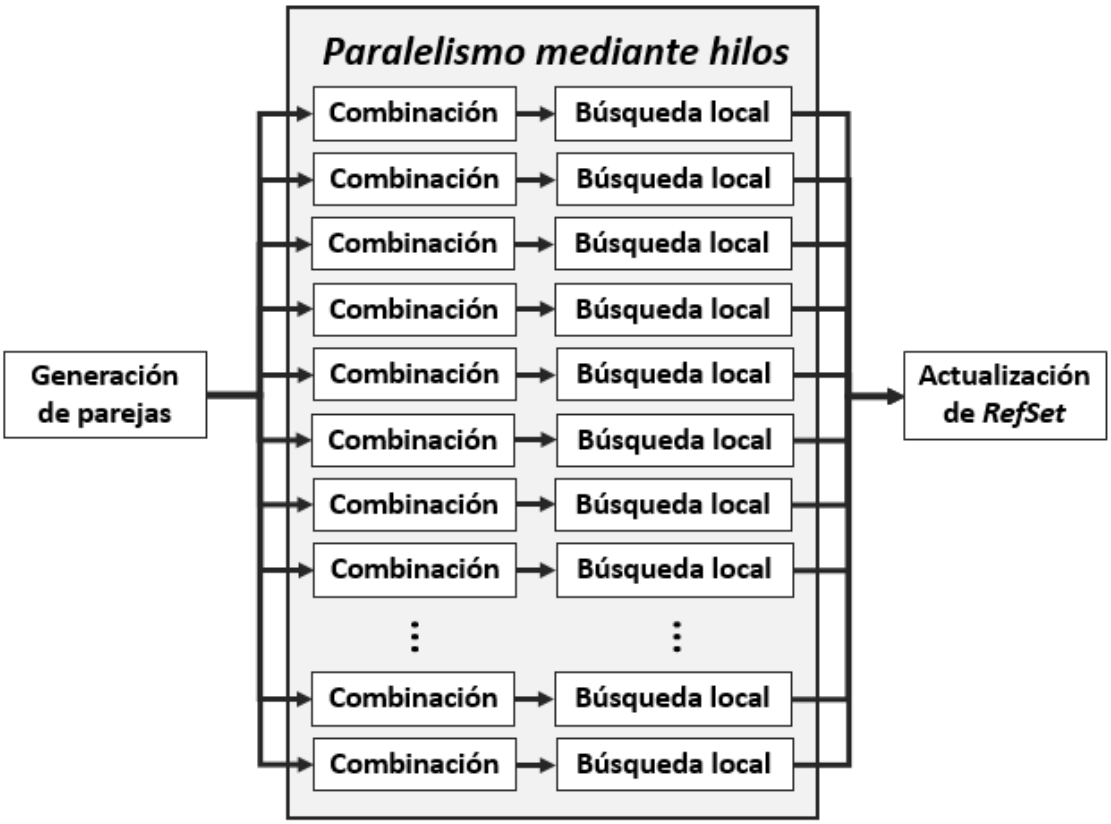

Figura 17. Representación del proceso de ejecución del algoritmo en un entorno de multiprocesamiento

La paralelización de estos procesos se implementa utilizando la librería Thread de Java (Oracle, s.f.). Esta librería permite crear hilos de proceso, los hilos son pequeños subprocesos independientes entre sí que se resuelven en el procesador. Independientemente del número de núcleos o procesadores disponibles, los hilos pueden ejecutarse a partir de un núcleo.

De esta forma, el incremento en la velocidad de resolución depende solo del número de procesadores disponibles, pero no existe un mínimo o máximo determinado para que se realice la ejecución.

En el algoritmo se crean tantos hilos como parejas de padres sean generadas, y a cada hilo se le asigna la tarea a realizar y se le envía la pareja de soluciones correspondiente. Posteriormente el algoritmo adquiere un estado de espera en el que aguarda la terminación de todos los procesos con el fin de continuar con la ejecución.

4.7.7 Generación de un individuo mediante combinación

Una vez que se eligen dos soluciones, se genera una nueva solución a partir de estos; esto se realiza a través de un método de combinación. En el algoritmo desarrollados se implementaron tres diferentes métodos de combinación, de los cuales es elegido uno aleatoriamente para generar a la nueva solución. Los métodos de combinación son por posición aleatoria, por valor medio y por punto de corte.

El método de combinación por posición aleatoria itera por cada posición del nuevo vector hijo, donde el valor que adquiere es el resultado de la elección aleatoria de alguno de los dos padres en la misma posición del vector hijo. El procedimiento es ilustrado en la *Figura 18*.

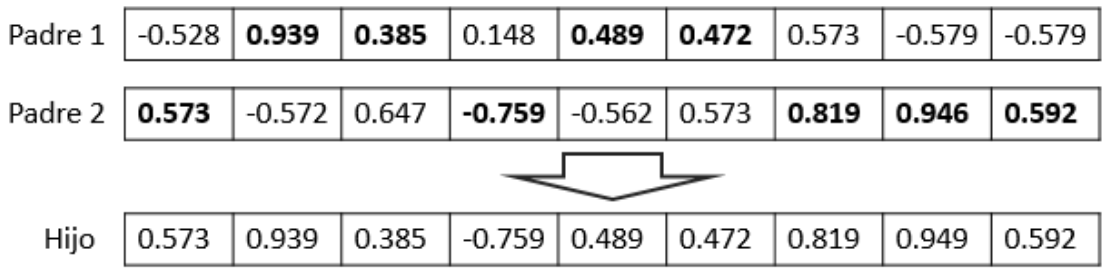

Figura 18. Ejemplo de cruza sináptica aleatoria.

El método de combinación por valor medio itera por cada posición del nuevo vector hijo, y el valor que adquiere es el valor medio de la posición de ambos padres. Este tipo de cruza se ejemplifica en la *Figura 19* mostrada a continuación.

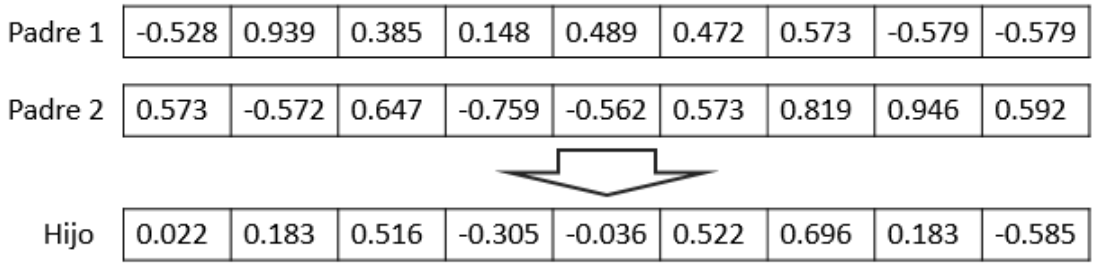

Figura 19. Ejemplo de cruza sináptica por valor medio.

El método de combinación por punto de corte elige un valor aleatorio acotado entre cero y el tamaño del vector hijo. Elegido el valor selecciona aleatoriamente a uno de los padres y copia su información desde el inicio hasta el punto de corte. El siguiente paso es copiar la información del segundo padre desde el punto de corte hasta el final del vector. El proceso se ve ilustrado en la *Figura 20*.

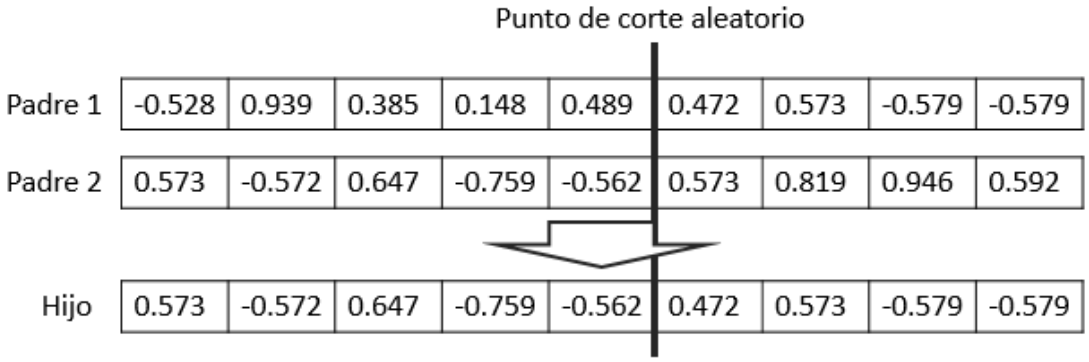

Figura 20. Ejemplo de cruza sináptica por punto de corte.

4.7.8 Procedimiento de búsqueda local

Este proceso tiene como objetivo ajustar una solución generada en el proceso de combinación para acercarla a un valor optimo local. La búsqueda encamina la solución haciendo pequeños cambios o perturbaciones controladas en el vector. Esto se ilustra en la Figura 21.

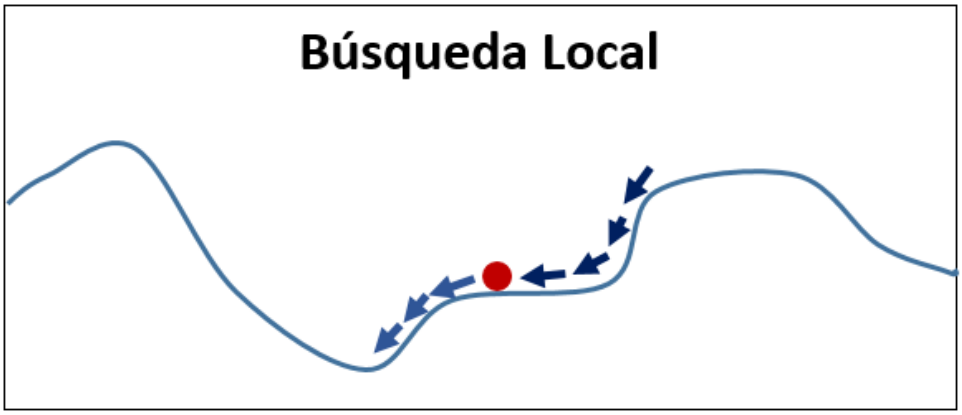

Figura 21. Representación del proceso de búsqueda local.

El procedimiento de búsqueda local está estrictamente ligado a la estructura de la red neuronal. Algunos procedimientos tales como Backpropagation no podrían ser utilizados, a pesar de ser ampliamente efectivos debido al tipo de topología generada, en la que no existe una jerarquización mediante capas.

El tipo de datos contenidos en el vector sináptico son números reales, por lo que fue necesario la integración de una búsqueda de números reales en el algoritmo.

Se utilizó una búsqueda local con encaminamiento tipo gradiente. Se itera por cada posición del vector y se hace una pequeña modificación aleatoria en un intervalo definido en el algoritmo a cada posición del vector; terminado este paso se acepta solo el mejor cambio realizado. Este procedimiento se repite tantas veces como mejoras en el vector sináptico existan. En la *Figura 22* se ilustra el proceso.

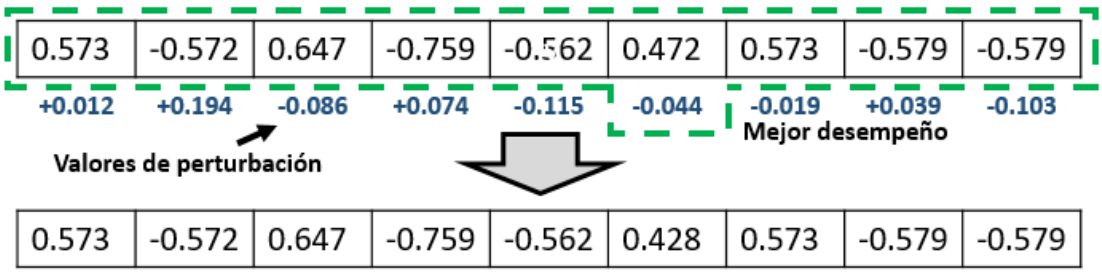

Figura 22. Representación del proceso de búsqueda local.

4.7.9 Criterios de terminación del algoritmo

El algoritmo no tiene definido un número de generaciones, sin embargo, el proceso de terminación del algoritmo evitar que este se quedase estancado por periodos muy largos de tiempo o infinitamente.

Cuando se obtiene una misma mejor solución un número determinado de veces, el algoritmo da por terminado su proceso evolutivo y termina. El número de repeticiones de la mejor solución es un parámetro sintonizable en el algoritmo.

Capítulo 5. Experimentación y resultados

5.1 Condiciones de experimentación

En el presente capitulo se describe el proceso experimental que se llevó a cabo para evaluar los algoritmos desarrollados. En primer término, se describen los valores de los parámetros utilizados en cada uno de los algoritmos desarrollados, las condiciones experimentales y una descripción detallada de las instancias de prueba utilizadas. Además, se presentan los resultados de la experimentación y del análisis estadístico no paramétrico desarrollado. Por otra parte, también se realiza una comparativa con los resultados reportados en el estado del arte para otros enfoques de solución basados en redes neuronales.

5.1.1 Parámetros para las estrategias neuroevolutivas

Se desarrollaron y evaluaron tres estrategias de solución basadas en redes neuroevolutivas para el problema de diagnóstico médico: el primer algoritmo utiliza una red neuroevolutiva con ajuste sináptico completo **(AASC)**; el segundo es una red neuroevolutiva que incorpora evolución para el ajuste sináptico **(ASAS)** y finalmente el tercero es una red neuronal doblemente evolutiva que utiliza un algoritmo genético para el ajuste topológico y un algoritmo de búsqueda dispersa para el ajuste sináptico; este algoritmo de doble evolución se denominó algoritmo con ajuste sináptico sin búsqueda local **(AASSBL)**.

Las *Tablas 1, 2 y 3* contienen la descripción de los parámetros utilizados en la experimentación para cada uno de los algoritmos desarrollados.

Tabla 1. Parámetros de configuración del algoritmo ASAS

Algoritmo sin ajuste sináptico ASAS			
Conjunto	**Pima Indians Diabetes**	**Breast Cancer**	**Heart Statlog**
División de los datos			
Entrenamiento (%) / Validación (%)	60/40	60/40	60/40
Población y topología de la red			
Tamaño de la población	100	100	100
Número de neuronas máximo	25	80	25
Validación cruzada			
Número de bloques	5	5	5
Máximo de generaciones totales	1000	1000	1000
Evolución topológica			
Número de generaciones	50	50	50
Probabilidad de mutación	0.3	0.3	0.3

Tabla 2. Parámetros de configuración del algoritmo AASSBL

Algoritmo con ajuste sináptico sin búsqueda local AASSBL			
	Pima Indians Diabetes	**Breast Cancer**	**Heart Statlog**
División de los datos			
Entrenamiento (%) / Validación (%)	60/40	60/40	60/40
Topologías de red y Población			
Tamaño de la población	100	100	100
Número de neuronas máximo	25	80	25
Validación cruzada			
Número de bloques	5	5	5
Máximo de generaciones totales	1000	1000	1000
Evolución topológica			

Número de generaciones	50	50	50
Probabilidad de mutación	0.3	0.3	0.3
Evolución sináptica			
Tamaño conjunto *P*	100	100	100
Tamaño conjunto *RefSet*	10	10	10

Tabla 3. Parámetros de configuración del algoritmo AASC

Algoritmo con ajuste sináptico completo AASC			
	Pima Indians Diabetes	**Breast Cancer**	**Heart Statlog**
División de los datos			
Entrenamiento (%) / Validación (%)	60/40	60/40	60/40
Topologías de red y Población			
Tamaño de la población	100	100	100
Número de neuronas máximo	25	80	25
Validación cruzada			
Número de bloques	5	5	5
Máximo de generaciones totales	1000	1000	1000
Evolución topológica			
Número de generaciones	50	50	50
Probabilidad de mutación	0.3	0.3	0.3
Evolución sináptica			
Tamaño conjunto *P*	100	100	100
Tamaño conjunto *RefSet*	10	10	10
Intervalo de búsqueda local	±0.1	±0.1	±0.1

5.1.2 Ambiente experimental

El algoritmo fue compilado en lenguaje Java en su versión 7.0. El paradigma de programación fue mediante objetos; además se utilizó la técnica de programación mediante hilos.

El entorno de programación utilizado fue el software Netbeans versión 8.1 y tanto el compilador como el entorno de desarrollo estaban soportados por el sistema operativo Ubuntu versión 16.4.

La ejecución del algoritmo se realizó en el clúster Ehécatl ubicado en el campus 3 del Instituto Tecnológico de Ciudad Madero. Este dispositivo cuenta con el sistema operativo Linux CentOs, la máquina virtual de Java 7.0 y sus características de hardware son: 4 nodos de trabajo con procesadores INTEL XEON y 64gb de memoria RAM (Madero, s.f.).

5.2 Conjuntos de datos de prueba

La selección de los conjuntos de prueba de acuerdo a la temática del trabajo de investigación se realizó de los conjuntos de datos disponibles en el estado del arte. Se utilizaron los conjuntos de diabetes, cáncer y corazón los cuales fueron tomados del sitio "*UC Irvine Machine Learning Repository*" y son descritos a continuación (UC Irvine Machine Learning Repository, s.f.).

5.2.1 Pima Indians Diabetes

El conjunto de datos Pima Indians Diabetes analiza a un conjunto de mujeres de al menos 21 años con ascendencia Pima, las cuales tienen o no el padecimiento diabetes.

El conjunto de datos está compuesto por 768 casos, de los cuales 500 casos son positivos para diabetes y 268 casos son negativos para diabetes.

La estructura está compuesta por 8 atributos numéricos y una clase categórica. Los atributos son el número de embarazos, concentración de glucosa en plasma a

2 horas, presión arterial diastólica, espesor de pliegue cutáneo en tríceps, insulina sérica de 2 horas, índice de masa corporal, función pedigrítica de la diabetes y la edad. El atributo de clase es el diagnóstico y éste puede ser positivo o negativo.

Los atributos son numéricos no acotados, además se encuentran algunos datos faltantes dentro del conjunto de datos

5.2.2 Breast Cancer

El conjunto de datos Breast Cancer analiza a un conjunto de individuos, los cuales padecieron cáncer de pecho con anterioridad y pueden o no tener reincidencia del padecimiento.

El conjunto se compone de 286 casos, de los cuales 201 no tuvieron reincidencia del padecimiento y 85 de ellos sí tuvieron reincidencia del padecimiento. La estructura está compuesta por 9 atributos numéricos y categóricos, así como una clase categórica. Los atributos son la edad, edad de inicio de la menopausia, el tamaño del tumor, nodos implicados, nodos caps., grado de tumor, pecho afectado, cuadrante afectado y si el paciente ha sido irradiado. El atributo de clase indica la reincidencia o no reincidencia del padecimiento.

Como se mencionó, algunos atributos representan valores categóricos, los cuales son transformados por el algoritmo.

5.2.3 Heart Statlog

El conjunto Heart Statlog analiza a un conjunto de individuos, los cuales pueden o no presentar una afección cardiaca.

El conjunto se compone de 270 casos, de los cuales 150 no presentan un padecimiento o afección cardiaca y 120 si presentan este tipo de afección. La estructura está compuesta por 13 atributos numéricos y categóricos, así como una clase categórica. Los atributos son: la edad, el género, el tipo de dolor en el pecho, presión sanguínea, colesterol, concentración de azúcar en sangre, resultados electrocardiográficos, frecuencia cardiaca máxima, agina inducida por

ejercicio, depresión de ST inducida por el ejercicio, pendiente pico del segmento ST inducido por el ejercicio, número de vasos principales y grado de daño.

Como se mencionó, algunos atributos representan valores categóricos, los cuales son transformados por el algoritmo.

5.3 Resultados del algoritmo

Para obtener resultados del algoritmo se realizaron 30 ejecuciones por cada conjunto de datos evaluado. En cada ejecución se respetaron los parámetros de configuración, sin embargo, se utilizó una semilla aleatoria para la generación de valores aleatorios donde sea necesario.

Los resultados muestran el porcentaje de casos correctamente clasificados; es decir el porcentaje de efectividad de clasificación. Por cada conjunto evaluado se muestra el valor medio y la mediana del total de ejecuciones, así como también se muestra el peor valor encontrado, el mejor valor encontrado y la desviación estándar del conjunto de datos.

En la *Tabla 4* se muestran los resultados para el conjunto de casos Pima Indians Diabetes en las tres variantes del algoritmo. También en las *Gráficas 1, 2 y 3* se muestran los histogramas de frecuencias de cada uno de los algoritmos.

Tabla 4. Resultados de la comparativa entre ASAS, AASSBL y AASC para el conjunto Diabetes.

Pima Indians Diabetes					
	Mejor valor	Peor valor	Valor medio	Mediana	Desviación estándar
Algoritmo *sin ajuste sináptico* ***ASAS***	72.4%	64.2%	67.7%	67.9%	0.0200
Algoritmo *con ajuste sináptico sin búsqueda local* ***AASSBL***	82.1%	75%	77.6%	77.5%	0.0177
Algoritmo con *ajuste sináptico completo* ***AASC***	82.1%	77.2%	79.1%	78.8%	0.0149

.

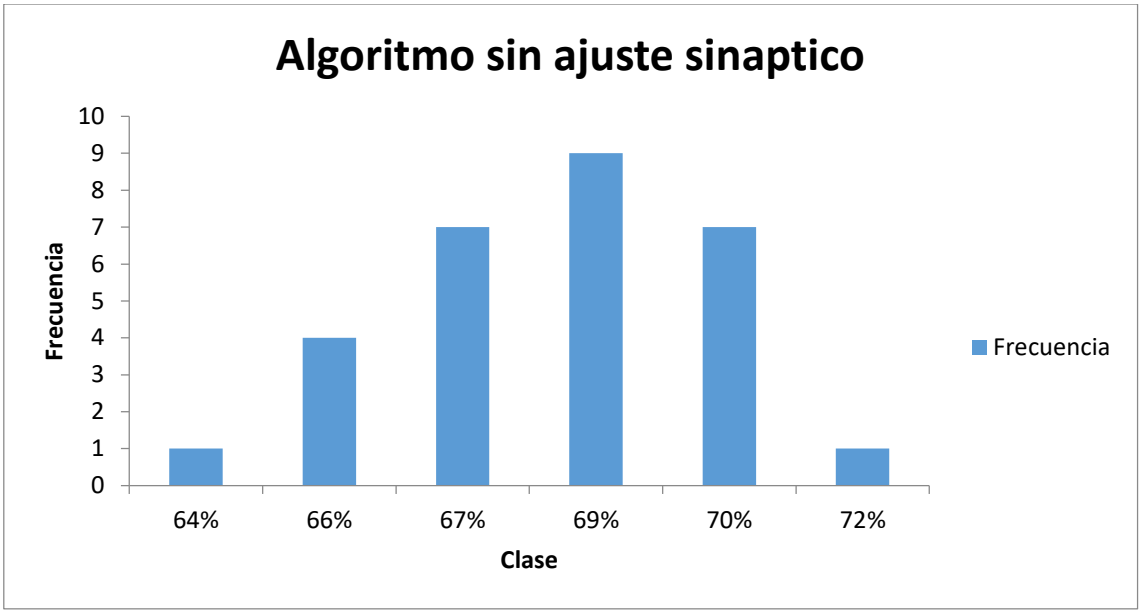

Gráfico 1. Histograma de frecuencias de ASAS para el conjunto Diabetes.

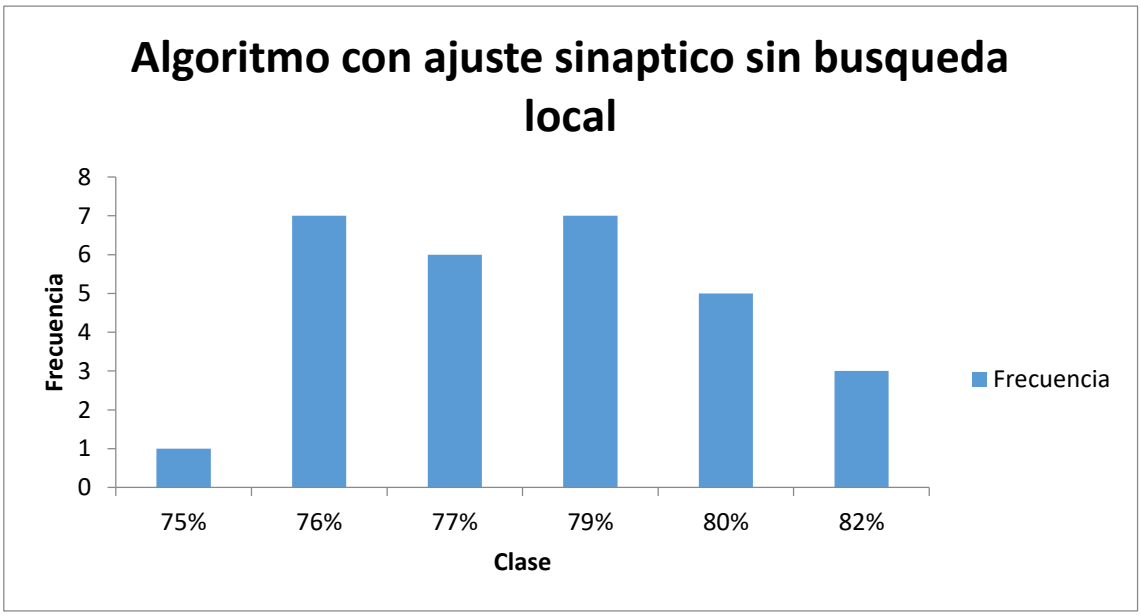

Gráfico 2. Histograma de frecuencias de AASSBL para el conjunto Diabetes.

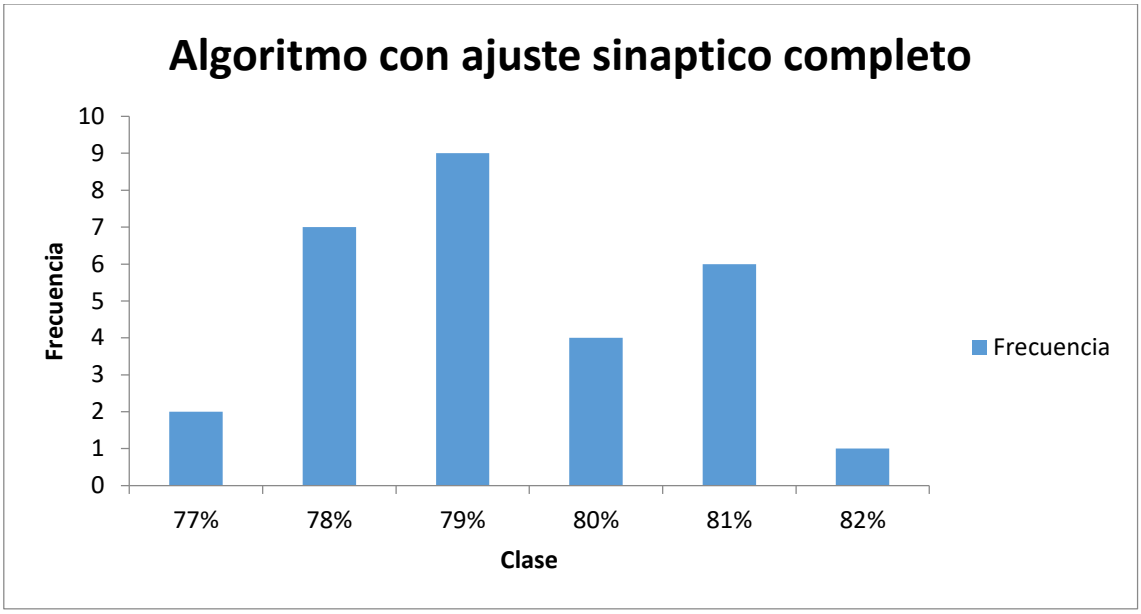

Gráfico 3. Histograma de frecuencias de AASC para el conjunto Diabetes.

Se observa que el algoritmo AASC y el algoritmo AASSBL obtuvieron un mejor resultado con **82.1%** de efectividad de clasificación por el **72.4%** de efectividad del algoritmo ASAS.

Sin embargo, la efectividad media de los algoritmos nos indica que el algoritmo AASC tiene una mejor efectividad con **79.1%** contra el **77.6%** del algoritmo AASSBL. La diferencia entre los dos algoritmos se puede ver de manera más clara en las *Gráficas 2 y 3*, los cuales muestran que el algoritmo AASSBL tiene un rango de clases mucho mayor que el algoritmo AASC. Esto también puede observarse gracias a los valores de desviación estándar donde el algoritmo AASC tiene un mejor valor de desviación, por lo tanto, los valores resultantes se encuentran menos dispersos del valor medio con respecto a los resultados del algoritmo AASSBL.

Esta información, sin embargo, aún no es determinante para saber si las dos variantes tienen alguna diferencia significativa de desempeño. En la siguiente sección se realiza una validación mediante una prueba estadística no paramétrica para comprobarlo.

Para el conjunto de datos Breast Cancer los resultados se muestran en la *Tabla 5*, donde de igual manera son evaluadas las tres estrategias algorítmicas. También en las *Gráficas 4, 5 y 6* se muestran los histogramas de frecuencias para cada algoritmo evaluado.

Tabla 5. Resultados de la comparativa entre ASAS, AASSBL y AASC para el conjunto Cancer.

Breast Cancer					
	Mejor valor	Peor valor	Valor medio	Mediana	Desviación estándar
Algoritmo *sin ajuste sináptico* ***ASAS***	83.4%	70.4%	76.1%	76%	0.0268
Algoritmo *con ajuste sináptico sin búsqueda local* ***AASSBL***	84.3%	73.9%	77.9%	77.3%	0.0242
Algoritmo con *ajuste sináptico completo* ***AASC***	85.2%	72.1%	79.8%	80%	0.0321

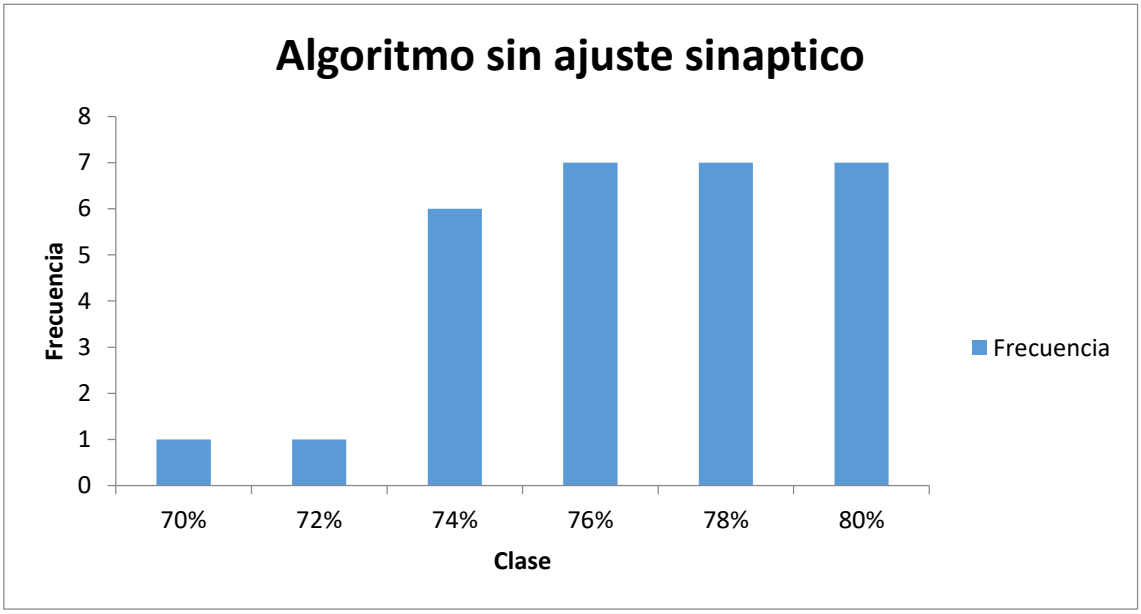

Gráfico 4. Histograma de frecuencias de ASAS para el conjunto Cancer.

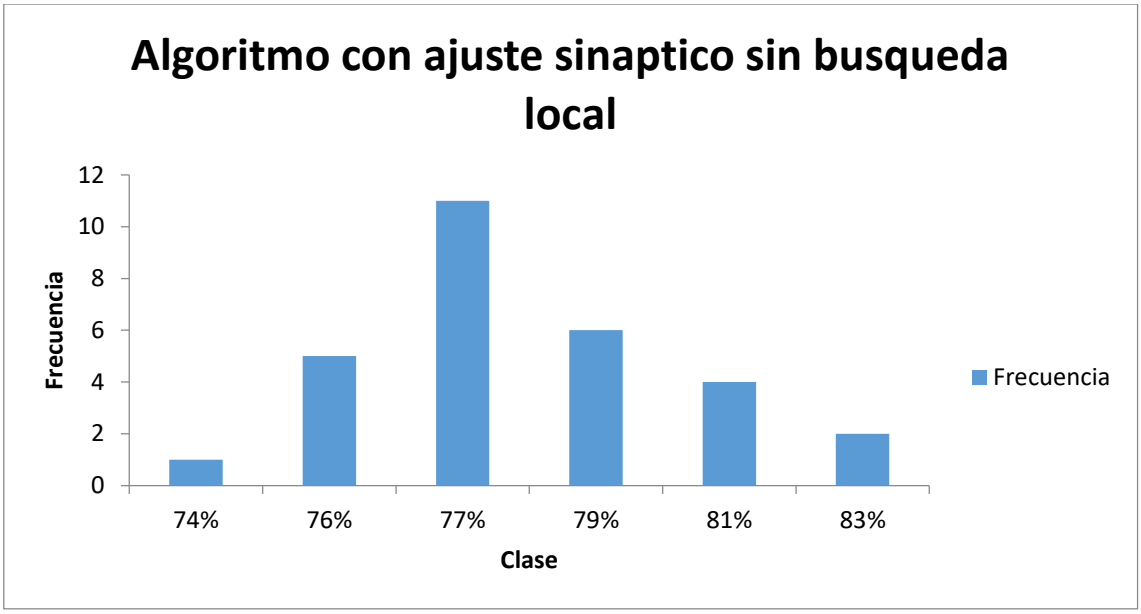

Gráfico 5. Histograma de frecuencias de AASSBL para el conjunto Cancer.

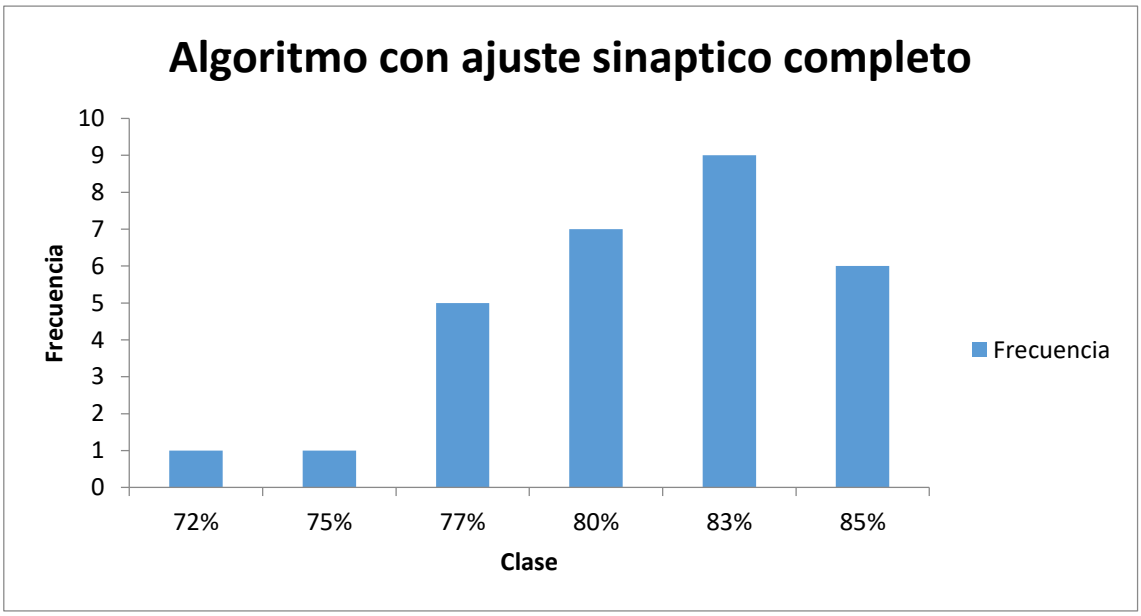

Gráfico 6. Histograma de frecuencias de AASC para el conjunto Cancer

Se observa en la *Tabla 5* que los tres algoritmos analizados obtienen para el *mejor valor de efectividad* valores muy similares; donde el AASC es superior con **85.2%**, el AASSBL con **84.3%** y al final el ASAS con **83.4%**.

Aunque los tres algoritmos tengan un comportamiento similar respecto al *mejor valor* encontrado, su diferencia en desempeño se hace más evidente analizando

el valor medio de desempeño donde el algoritmo AASC tiene un valor medio de **79.8%** contra el **77.3%** y **76.1%** de AASSBL y ASAS respectivamente.

Por otro lado, analizando las representaciones mostradas en los *Gráficos 4, 5 y 6*, se puede observar que el algoritmo ASAS tiene la mayor frecuencia en valores en el rango de **74%** a **80%** de efectividad, obteniendo muy pocos resultados por debajo de ese rango; el algoritmo AASSBL tiene la mayor frecuencia de resultados en el rango que va del **76%** al **79%** de efectividad, obteniendo casos aislados de menor y de mayor efectividad; el algoritmo AASC tiene su mayor frecuencia de resultados en el rango del **77%** al **85%** , obteniendo el mínimo número de valores menores a este rango.

Se puede analizar la dispersión de los datos respecto a la media observando los valores de desviación estándar. Como puede verse a partir de los datos de la desviación que se presentan en la columna 6 de la Tabla 5, el algoritmo AASSBL contiene una menor dispersión de los resultados y el algoritmo ASAS a pesar de ser el algoritmo con el mejor valor de desempeño y una mejor media de desempeño contiene una dispersión más alta en sus resultados.

Para el conjunto de datos Heart Statlog los resultados se muestran en la *Tabla 6*, donde de igual manera son evaluadas las tres variantes del algoritmo. También en las *Gráficas 7, 8 y 9* se muestran los histogramas de frecuencias para cada variante del algoritmo evaluada.

Tabla 6. Resultados de la comparativa entre ASAS, AASSBL y AASC para el conjunto Heart.

Heart Statlog					
	Mejor valor	Peor valor	Valor medio	Mediana	Desviación estándar
Algoritmo *sin ajuste sináptico* ***ASAS***	87%	60.1%	73.2%	.74%	0.0713
Algoritmo *con ajuste sináptico sin búsqueda local* ***AASSBL***	77.7%	58.3%	71%	71.2%	0.0452

Algoritmo con *ajuste sináptico completo* ***AASC***	90.7%	77.7%	84.2%	83.7%	0.0257

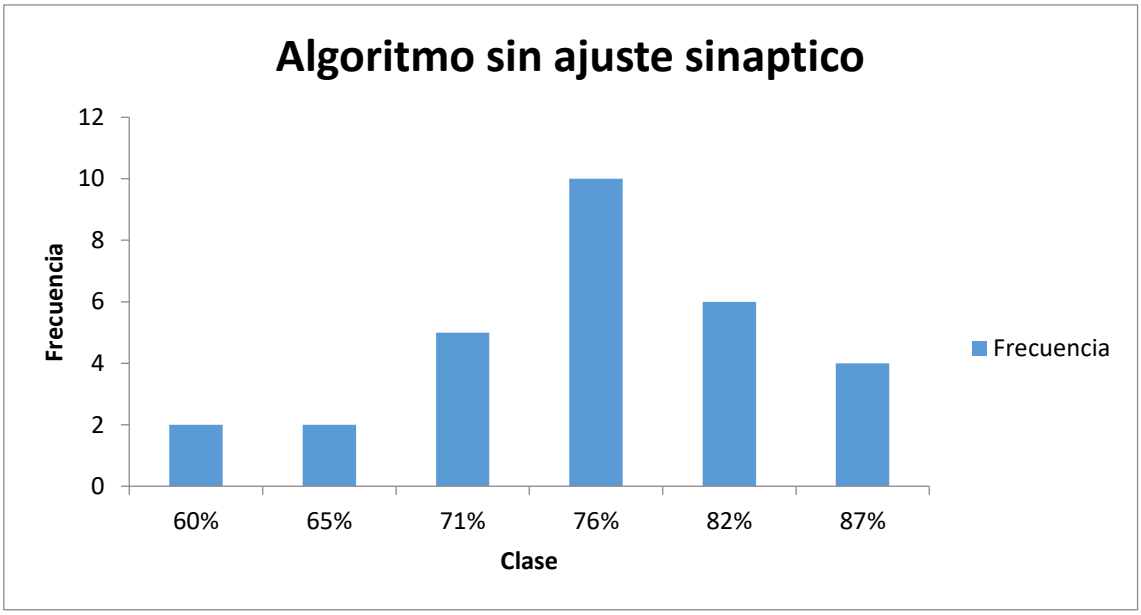

Gráfico 7. Histograma de frecuencias de ASAS para el conjunto Heart

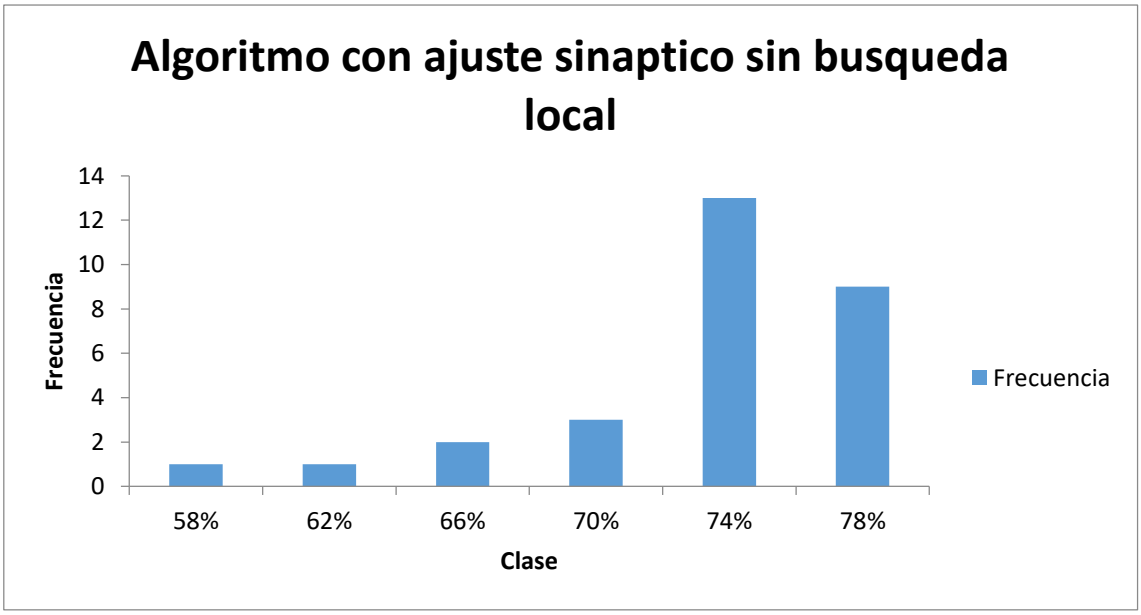

Gráfico 8. Histograma de frecuencias de AASSBL para el conjunto Heart

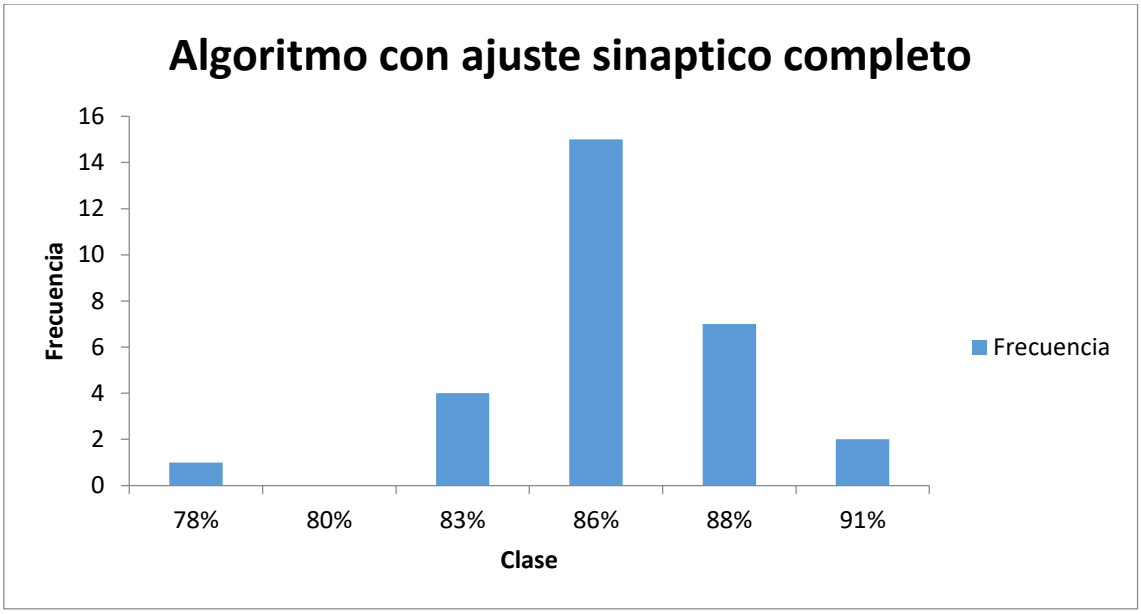

Gráfico 9. Histograma de frecuencias de AASC para el conjunto Heart

Se observa en la *Tabla 6* que el algoritmo AASC obtuvo un mejor valor de desempeño que los otros dos algoritmos, aunque el algoritmo ASAS obtuvo un valor cercano con **90.7%** y **87%** respectivamente.

Analizando las medias de valores nos encontramos que el algoritmo AASC obtuvo un valor de **77.3%** contra un **60.1%** de ASAS y por último un **58.3%** AASSBL. Las diferencias de valores tan grandes están dadas por una gran dispersión de los resultados en esta instancia. Esto se puede observar en la desviación estándar ya que se obtuvieron valores altos.

También observando los *Gráficos 7, 8 y 9* podemos visualizar que el algoritmo ASAS obtiene la mayoría de los resultados en el rango de **71%** a **82%** con una incidencia menor en rangos inferiores y superiores. El algoritmo AASSBL obtiene la mayor parte de sus resultados en el rango de **74%** a **78%**, sin embargo, obtiene una importante cantidad de resultados en rangos menores que se encuentran dispersos desde **58%** hasta **73%**. El algoritmo AASC obtiene el grueso de sus datos de **83%** a **88%** con una menor incidencia de valores menores y mayores siendo el algoritmo con una desviación estándar menor.

5.4 Análisis estadístico

Los resultados obtenidos en la fase de experimentación aun no nos son determinantes para poder concluir si el algoritmo propuesto es superior a alguna de sus variantes o es equivalente a estas en desempeño.

Para determinar si existen diferencias estadísticas en el desempeño de los algoritmos se utilizó la prueba estadística de Wilcoxon por pares. Se eligió utilizar esta prueba no paramétrica debido a que no se puede suponer la normalidad de los resultados de los algoritmos comparados y por tanto no se sabe si se ajusta a una curva normal. La evaluación se hizo con el solver *SciStatCalc* que dispone de varias herramientas estadísticas online (SciStatCalc, s.f.).

Tabla 7. Parámetros de configuración para la evaluación estadística Wilcoxon

Parámetros Wilcoxon	
α: nivel de significancia	0.05
H_0: hipótesis nula	"Los algoritmos tienen un desempeño equivalente"
H_1: Hipótesis alterna	"Existe una diferencia significativa en el desempeño de los algoritmos"

En la Tabla 7 se muestran los parámetros utilizados en la prueba de Wilcoxon, los cuales fueron idénticos para cada validación realizada en cada pareja de conjuntos de datos evaluados. El nivel de significancia α se estableció en **0.05**; así, si el valor resultante de P-value es menor que ***α***, entonces la hipótesis nula es rechazada y se acepta la hipótesis alterna, de lo contrario la hipótesis nula es aceptada.

Los resultados obtenidos gracias al P value nos indican si existe una diferencia significativa o no, pero no nos permiten saber cuál algoritmo es mejor si existiese una diferencia significativa. Esto se determina gracias al valor de suma del rango de diferencia positiva y negativa (*SPDR y SNDR respectivamente*); de tener una mayor diferencia positiva entonces el algoritmo en la *fila1* tendría un mejor

desempeño, si se tiene una mayor diferencia negativa el algoritmo en la columna 2 tiene un mayor desempeño.

Tabla 8. Resultados de la validación estadística entre AASC, AASSBL y ASAS para el conjunto Diabetes.

Prueba de Wilcoxon para el conjunto Pima Indians Diabetes			
	AASC	**AASSBL**	**ASAS**
AASC		P value: **0.000004** *"Se rechaza H_0, existe diferencia significativa"* SPDR: **406** SNDR: **0** *"Algoritmo AASC tiene mejor desempeño"*	P value: **0.000002** *"Se rechaza H_0, existe diferencia significativa"* SPDR: **465** SNDR: **0** *"Algoritmo AASC tiene mejor desempeño"*
AASSBL	P value: **0.000004** *"Se rechaza H_0, existe diferencia significativa"* SPDR: **0** SNDR: **406** *"Algoritmo AASC tiene mejor desempeño"*		P value: **0.000002** *"Se rechaza H_0, existe diferencia significativa"* SPDR: **465** SNDR: **0** *"Algoritmo AASSBL tiene mejor desempeño"*
ASAS	P value: **0.000002** *"Se rechaza H_0, existe diferencia significativa"* SPDR: **0** SNDR: **465** *"Algoritmo AASC tiene mejor desempeño"*	P value: **0.000002** *"Se rechaza H_0, existe diferencia significativa"* SPDR: **0** SNDR: **465** *"Algoritmo AASSBL tiene mejor desempeño"*	

Para la base de datos de Pima Indians Diabetes se realizó la comparativa entre las variantes del algoritmo. Los resultados se presentan en la *Tabla 8*. Podemos observar que existe una diferencia significativa entre el algoritmo AASC sobre sus variantes AASSBL y ASAS, así se puede decir que el algoritmo AASC tiene un mejor desempeño que sus variantes AASSBL y ASAS para la instancia Pima-Indian Diabetes.

De la misma manera, el algoritmo AASSBL tiene una diferencia significativa sobre el algoritmo ASAS con lo cual su desempeño es estadísticamente superior en la instancia Pima Indians Diabetes.

Tabla 9. Resultados de la validación estadística entre AASC, AASSBL y ASAS para el conjunto Cancer

Prueba de Wilcoxon para el conjunto Breast Cancer			
	AASC	**AASSBL**	**ASAS**
AASC		P value: **0.000018** *"Se rechaza H_0, existe diferencia significativa"* SPDR: **441** SNDR: **24** *"Algoritmo AASC tiene mejor desempeño"*	P value: **0.000002** *"Se rechaza H_0, existe diferencia significativa"* SPDR: **465** SNDR: **0** *"Algoritmo AASC tiene mejor desempeño"*
AASSBL	P value: **0.000018** *"Se rechaza H_0, existe diferencia significativa"* SPDR: **24** SNDR: **441** *"Algoritmo AASC tiene mejor desempeño"*		P value: **0.000002** *"Se rechaza H_0, existe diferencia significativa"* SPDR: **465** SNDR: **0** *"Algoritmo AASSBL tiene mejor desempeño"*
ASAS	P value: **0.000002** *"Se rechaza H_0, existe diferencia significativa"* SPDR: **0** SNDR: **465** *"Algoritmo AASC tiene mejor desempeño"*	P value: **0.000002** *"Se rechaza H_0, existe diferencia significativa"* SPDR: **0** SNDR: **465** *"Algoritmo AASSBL tiene mejor desempeño"*	

De la misma manera, para la base de datos Breast Cancer se realizó la comparativa entre las variantes del algoritmo. Los resultados se presentan en la *Tabla 9*, podemos observar que existe una diferencia significativa del algoritmo AASC sobre sus variantes AASSBL y ASAS. Así se puede decir que el algoritmo AASC tiene un mejor desempeño que sus variantes AASSBL y ASAS para la instancia Breast Cancer.

De la misma manera, el algoritmo AASSBL tiene una diferencia significativa sobre el algoritmo ASAS con lo cual su desempeño es estadísticamente superior en la instancia Breast Cancer.

Tabla 10. Resultados de la validación estadística entre AASC, AASSBL y ASAS para el conjunto Heart

Prueba de Wilcoxon para el conjunto Heart Statlog			
	AASC	**AASSBL**	**ASAS**
AASC		P value: **0.000002** *"Se rechaza H_0, existe diferencia significativa"* SPDR: **465** SNDR: **0** *"Algoritmo AASC tiene mejor desempeño"*	P value: **0.000004** *"Se rechaza H_0, existe diferencia significativa"* SPDR: **458** SNDR: **7** *"Algoritmo AASC tiene mejor desempeño"*
AASSBL	P value: **0.000002** *"Se rechaza H_0, existe diferencia significativa"* SPDR: **0** SNDR: **465** *"Algoritmo AASC tiene mejor desempeño"*		P value: **0.159884** *"Se acepta H_0, no existe diferencia significativa y ambos algoritmos tienen un desempeño equivalente"*
ASAS	P value: **0.000004** *"Se rechaza H_0, existe diferencia significativa"* SPDR: **7** SNDR: **458** *"Algoritmo AASC tiene mejor desempeño"*	P value: **0.159884** *"Se acepta H_0, no existe diferencia significativa y ambos algoritmos tienen un desempeño equivalente"*	

Se realizó también la evaluación de los algoritmos usando la base de datos Heart Statlog y los resultados se muestran en la *Tabla 10*. Se puede observar que el algoritmo AASC tiene una diferencia significativa de desempeño frente a sus variantes AASSBL y ASAS.

Sin embargo, también podemos aseverar que el algoritmo AASSBL y el algoritmo ASAS son equivalentes entre si ya que su valor de p-value es superior al nivel de significancia establecido.

Observando las *Gráficos 7 y 8* de la sección anterior podemos observar que el algoritmo ASAS tiene inclusive mejores resultados que su variante AASSBL, sin

embargo, estadísticamente no es posible concluir que es mejor, si no se concluye que son equivalentes para el nivel de significancia definido.

5.5 Conclusiones del análisis estadístico

Gracias a los resultados del análisis estadístico podemos concluir que el algoritmo con ajuste sináptico completo AASC es estadísticamente mejor que sus variantes para las bases de datos analizadas.

La mejora de este algoritmo está dada por el procedimiento de ajuste sináptico, ya que los resultados fueron mejores al del algoritmo que no incorporaba un procedimiento de ajuste sináptico y solo incorporaba el ajuste topológico.

De la misma manera en las pruebas realizadas para este algoritmo se obtuvo un mejor desempeño que el algoritmo AASSBL el cual incorpora una técnica sináptica que explora en el espacio de soluciones pero que carece de una técnica de explotación como lo es el procedimiento de búsqueda local.

Se observa entonces que el procedimiento de ajuste sináptico cumple por si solo su función de escape de valores óptimos locales, lo cual permite una mejor exploración en el espacio de soluciones y por lo tanto obtiene mejores resultados. Sin embargo, incorporar un procedimiento de búsqueda local aumenta la posibilidad de que los resultados obtenidos mejoren aún más buscando óptimos locales en el espacio de soluciones y encontrando soluciones iguales o mejores a las ya encontradas.

La comparativa sin embargo entre la técnica sin ajuste sináptico y aquella que cuenta con un ajuste sináptico sin búsqueda local presenta una ventaja en el algoritmo AASSBL para las bases de datos diabetes y cáncer, sin embargo, para la base de datos Heart Statlog es equivalente al procedimiento sin ajuste sináptico.

Así, se puede concluir que un procedimiento de ajuste sináptico en la neuroevolución mejora la calidad de los resultados y permite obtener valores de

desempeño superiores sin comprometer los mecanismos de ajuste y evolución topológica.

5.6 Comparativa con algoritmos del estado del arte

A continuación, se presentan los resultados de una comparativa del mejor algoritmo propuesto en este trabajo **AASC** contra los trabajos analizados en el Capítulo 3 los cuales pertenecen al estado del arte. La comparativa se realiza con el conjunto de datos diabetes.

Los algoritmos analizados en el estado del arte reportan el mejor valor obtenido, el valor medio obtenido y la desviación estándar, gracias a eso es posible calcular una muestra aleatoria que se ajuste a estos valores y así, poder obtener histogramas y hacer una validación estadística.

En esta sección se comparan los algoritmos del estado del arte: una red neuronal con aprendizaje *Back Propagation*, una red neuronal con aprendizaje híbrido mediante un algoritmo genético y *Back Propagation*, el enfoque de neuroevolución por topologías incrementales(*NEAT*) y el enfoque de neuroevolución por cruza basada en similitudes(*SimBa*) (Gowda, Manjunath, & Jayaram, 2011) (Dragoni, Azzini, & Tettamanzi, 2014).

En la siguiente *Tabla 11* se muestran los resultados de los algoritmos del estado del arte y el algoritmo AASC propuesto en este trabajo. Se puede observar la clara superioridad en la media y mediana de los resultados del algoritmo desarrollado en este trabajo frente al enfoque clásico de redes neuronales con aprendizaje *Back Propagation*, además se observa también la superioridad frente al enfoque de red neuronal clásica con el método híbrido de aprendizaje de *Back Propagation* con un algoritmo genético. Por otro lado, también se puede observar que existe una similitud en cuanto a resultados con el enfoque de topologías incrementales y una superioridad del enfoque de cruza basada en similitudes frente a todos los otros algoritmos.

Los aparentes resultados de similitud y superioridad que se observan en la tabla no son suficientes aun para determinar si el algoritmo propuesto tiene un desempeño superior o equivalente frente a los algoritmos del estado del arte, por lo que más a delante se presentará un análisis estadístico para determinar esto.

Tabla 11. Comparativa de resultados del algoritmo AASC contra los algoritmos del estado del arte

Pima Indians Diabetes					
	Mejor valor	Peor valor	Valor medio	Mediana	Desviación estándar
Red neuronal con aprendizaje *Back Propagation* **(Dragoni, Azzini, & Tettamanzi, 2014)**	75.2%	71.7%	73%	73.6%	0.132
Red neuronal con hibridación *Back Propagation* y Algoritmo Genético **(Gowda, Manjunath, & Jayaram, 2011)**	77.7%	77.7%	77.7%	77.7%	0
Neuroevolución por topologías incrementales **(Dragoni, Azzini, & Tettamanzi, 2014)**	82.9%	76.1%	79.5%	80.4%	0.0221
Neuroevolución con cruza basada en similitudes **(Dragoni, Azzini, & Tettamanzi, 2014)**	87.1%	78.6%	83.8%	83.7%	0.0219
Algoritmo con ajuste sináptico completo ***AASC***	82.1%	77.2%	79.1%	78.8%	0.0149

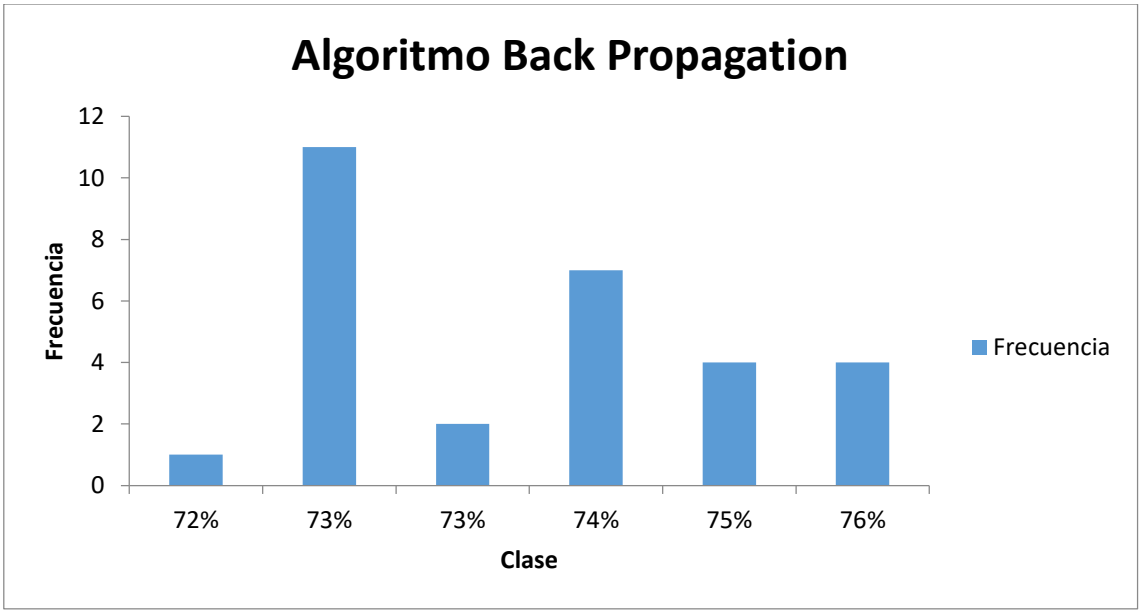

Gráfico 10. Histograma de frecuencias de Back Propagation para el conjunto Diabetes.

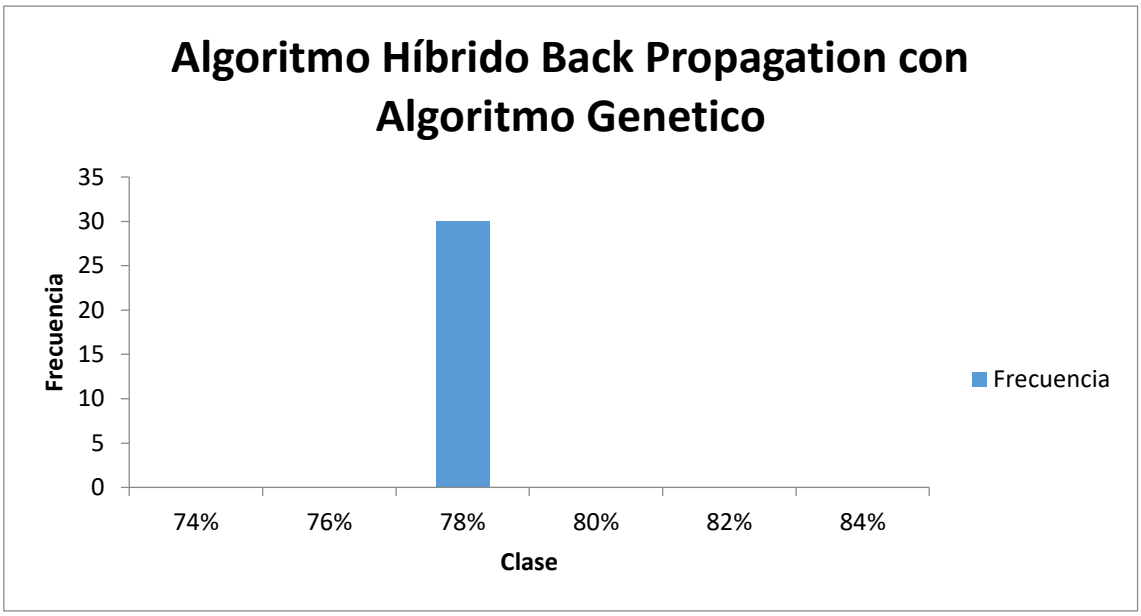

Gráfico 11. Histograma de frecuencias del Back Propagation con Algoritmo Genético para el conjunto Diabetes.

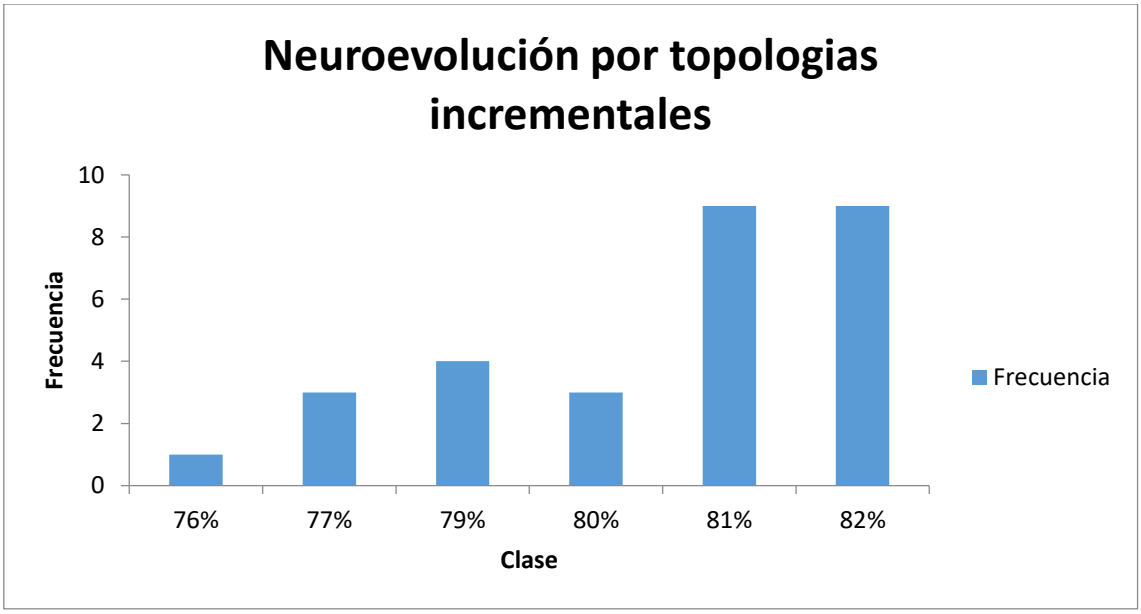

Gráfico 12. Histograma de frecuencias de NEAT para el conjunto Diabetes.

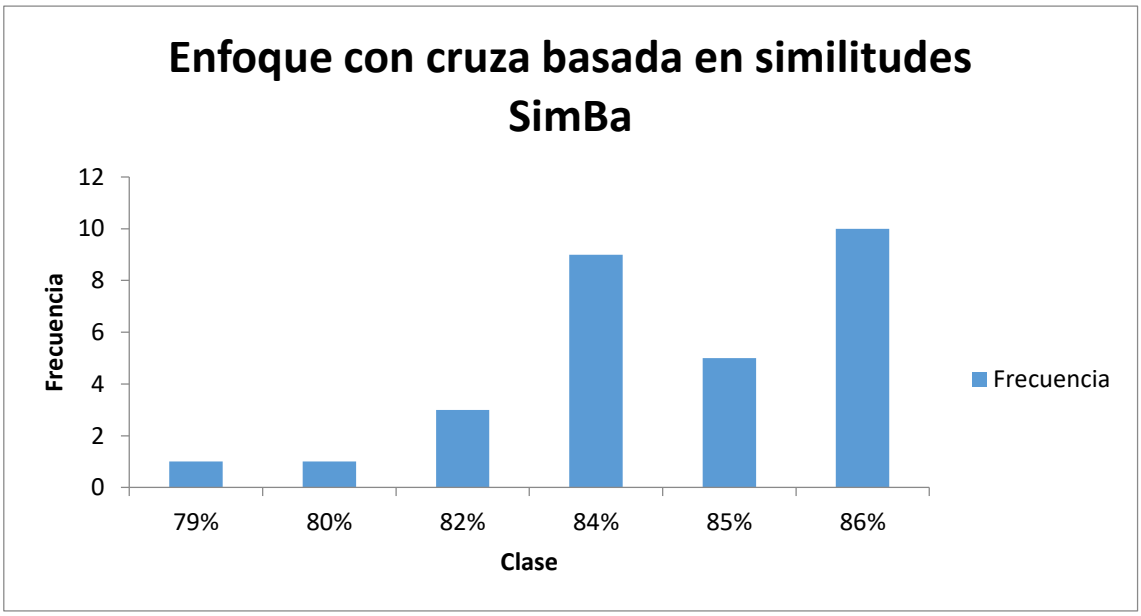

Gráfico 13. Histograma de frecuencias de SimBa para el conjunto Diabetes.

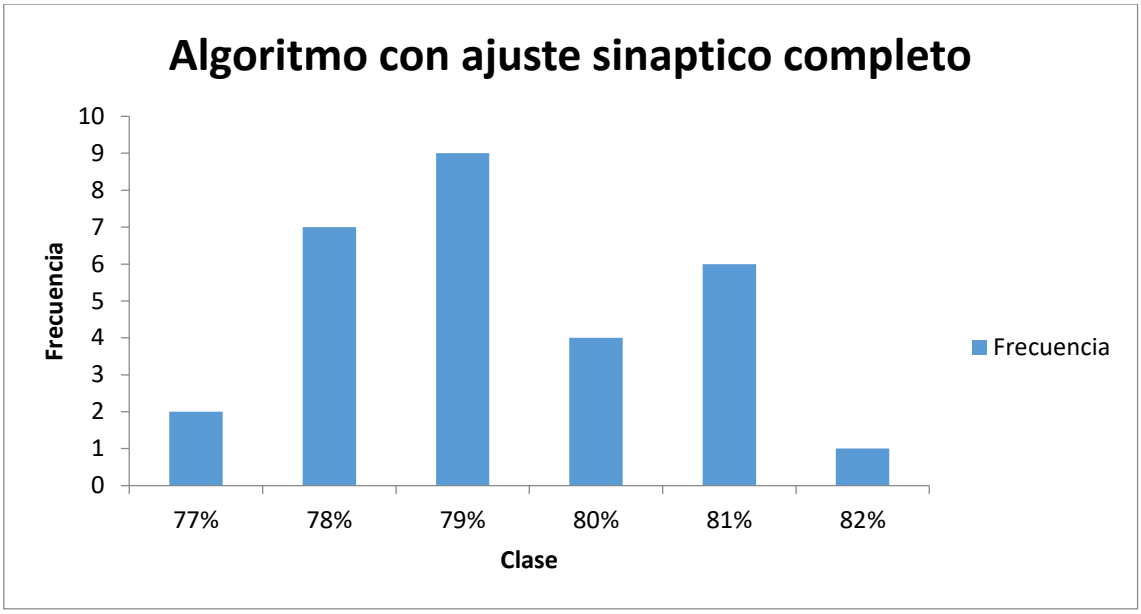

Gráfico 14. Histograma de frecuencias de ASSC para el conjunto Diabetes.

En las *Gráficas 10, 11, 12, 13 y 14* se muestran los histogramas de frecuencias de los algoritmos analizados. El enfoque de hibridación genética y de *Back Propagation* solo nos presenta un valor promedio (Gowda, Manjunath, & Jayaram, 2011), por lo que se tiene la limitación de considerar sus resultados como el valor presentado. Cabe destacar las similitudes que existen entre los histogramas del enfoque de topologías incrementales y el algoritmo propuesto, aunque las frecuencias y clases no son iguales los rangos y frecuencias son muy similares.

Tabla 12. Comparativa estadística de AASC frente a los algoritmos del estado del arte.

Comparativa estadística	
	AASC
Red neuronal con aprendizaje *Back Propagation* **(Dragoni, Azzini, & Tettamanzi, 2014)**	P value: **0.000002** *"Se rechaza H_0, existe diferencia significativa"* SPDR: **0** SNDR: **465** *"Algoritmo AASC tiene mejor desempeño"*
Red neuronal con hibridación *Back Propagation* y Algoritmo Genético **(Gowda, Manjunath, & Jayaram, 2011)**	P value: ***0.000002*** *"Se rechaza H_0, existe diferencia significativa"* SPDR: **0** SNDR: **465** *"Algoritmo AASC tiene mejor desempeño"*
Neuroevolución por topologías incrementales **(Dragoni, Azzini, & Tettamanzi, 2014)**	P-value**: 0.308615** *"Se acepta H_0, no existe diferencia significativa y ambos algoritmos tienen un desempeño equivalente"*
Neuroevolución con cruza basada en similitudes **(Dragoni, Azzini, & Tettamanzi, 2014)**	P value: **0.000002** *"Se rechaza H_0, existe diferencia significativa"* SPDR: **465** SNDR: **0** *"Algoritmo Simba tiene mejor desempeño"*

En la *Tabla 12* se presentan los resultados de la comparativa estadística, donde el algoritmo AASC tiene una superioridad frente al aprendizaje *Back Propagation* y *Back Propagation* evolutivo, además es equivalente al enfoque de neuroevolución por topologies incrementales; frente al enfoque SimBa, existe una diferencia significativa en el desempeño a favor de la neuroevolución basada en similitudes.

5.7 Conclusiones de la comparativa con algoritmos del estado del arte

Gracias a la evaluación estadística y los resultados mostrados se puede concluir que los enfoques que incorporan evolución topológica obtienen un mejor resultado frente a los enfoques cuya topología es definida manualmente. Esto puede deberse a la dificultad en encontrar manualmente una topología que segmente el espacio de soluciones de manera eficiente, así la automatización de la exploración de diversas topologías otorga una mayor exploración y una mejor elección de la topología de la red.

Dentro de los enfoques que incorporan neuroevolución, el enfoque más utilizado es la neuroevolución por topologías incrementales; el algoritmo propuesto resulta equivalente estadísticamente a este algoritmo, con lo cual resulta en una opción competitiva en la implementación de redes neuroevolutivas.

Con respecto al enfoque SimBa, se obtienen resultados menores, aunque competitivos, lo cual representa un reto para proponer algunas estrategias de mejora en la sección de conclusiones y trabajos futuros.

Capítulo 6. Conclusiones y trabajos futuros

6.1 Conclusiones

En este trabajo, se propuso una hibridación entre redes neuronales y algoritmos evolutivos; esta hibridación dio como producto una red neuronal con evolución topológica y evolución sináptica.

A partir de estos se obtuvieron resultados que fueron evaluados en una prueba de estadística no paramétrica de Wilcoxon, mediante la cual se determinó que el algoritmo es competitivo frente a los algoritmos reportados en el estado del arte y superior frente a la misma estrategia, pero con diferentes niveles de evolución.

Por parte de la evolución topológica se concluye que las estrategias presentadas de perturbación y combinación facilitan la exploración y el crecimiento de las topologías; y esto se ve reflejado en un aumento de desempeño frente a estructuras estáticas.

La incorporación de un mecanismo de ajuste sináptico por cada topología generada convierte a una nueva topología en una estructura lo suficientemente fuerte para poder competir frente a las demás. Este mecanismo representó una innovación exitosa frente a los enfoques existentes. Así, se concluye que un ajuste sináptico mejora la calidad de las topologías generadas en una red neuroevolutiva.

Dentro del mecanismo de ajuste sináptico se observa que un algoritmo meta-heurístico combinado con una estrategia de búsqueda local permite escapar y llegar a un valor óptimo local. Esta combinación de estas estrategias nos permite una mejor exploración del espacio de soluciones y escapar de los óptimos locales, con lo cual se evita un problema típico en el aprendizaje de las redes neuronales que es el estancamiento en óptimos locales. Así, se concluye que la implementación de estas estrategias apoya la obtención de un buen ajuste sináptico de la red neuronal.

En general, se puede concluir que la hibridación e incorporación de estrategias evolutivas y de búsqueda local como métodos de ajuste y aprendizaje en redes neuronales aumentan la calidad de clasificación y el desempeño de estas.

6.2 Publicaciones

Se escribió un artículo científico y una presentación para el “XII Congreso Internacional de Cómputo y Software” organizado por la Universidad Autónoma del Estado de Morelos (UAEM).

También en el marco del evento el artículo fue aceptado para publicarse en la revista Programación Matemática y Software la cual es editada por la UAEM, con ISSN (e): 2007-3283 y se encuentra indexada en Latindex (Latindex, s.f.). Este fue aceptado para su publicación en el volumen III del año 2018.

El artículo lleva por título “Enfoque de aprendizaje híbrido evolutivo para redes neuronales en la clasificación de casos médicos” y evalúa el algoritmo para las bases de datos Pima Indians Diabetes y Breast Cancer realizando una comparativa estadística contra enfoques clásicos y evolutivos reportados en el estado del arte.

6.3 Trabajos futuros

De acuerdo a lo propuesto y a lo explorado en el estado del arte, existen diversas técnicas o enfoques para trabajar con redes neuronales evolutivas; sin embargo, aún existe la posibilidad de seguir explorando con otras técnicas para lograr obtener un mejor desempeño o simplificar el proceso de aprendizaje.

Gracias al trabajo realizado y a los resultados obtenidos con los algoritmos algoritmo presentados, se pueden proponer diversas modificaciones o estrategias que pueden llevar a obtener una mejora futura en los resultados.

Es importante realizar una propuesta de nuevas técnicas de exploración topológica, ya que como se concluyó, estas nos permiten hacer una exploración

del espacio de soluciones. Utilizar enfoques de evolución topológica por capas o segmentos de capas podría resultar en una opción viable para obtener mejores resultados.

Por otro lado, con relación al ajuste sináptico, se demostró que este proceso puede mejorar el desempeño del mecanismo de aprendizaje; sin embargo, debido al enfoque incremental que no mantiene la estructura de capas propuesto en este trabajo no fue posible incorporar estrategias basadas en *Back Propagation*. Se plantea la necesidad de realizar una exploración de nuevas estrategias de ajuste sináptico tomando como punto de partida un enfoque de evolución topológica por capas, compatible con estrategias de ajuste sináptico como *Back Propagation*, incluida dentro de un algoritmo meta-heurístico que permita escapar de óptimos locales, solucionando así el problema clásico del aprendizaje por *Back Propagation*.

Asimismo, las estrategias de hibridación propuestas pueden conjugarse con otros algoritmos de clasificación como máquinas de soporte vectorial, lo que representa un enfoque prometedor en la obtención de resultados competitivos o superiores a los reportados en el estado del arte.

Referencias

Alatorre, A. (2004). *¿Qué es el cáncer?* Ciudad de México: Editorial Selector.

Bustamante, F., & Chavarría, E. (1992). El desarrollo de la computación y su influencia en la medicina. *Revista costarricense de ciencias médicas*, 59-70.

Castillo, P., Castellano, J., Merelo, J., & Prieto, A. (2001). Diseño de redes neuronales artificiales mediante algoritmos evolutivos. *Revista Iberoamericana de Inteligencia Artificial*, 2-32.

Center, I. K. (s.f.). *SPSS Statistics: Nivel de medición de variable*. (IBM Knowledge Center) Recuperado el 02 de Noviembre de 2017, de https://www.ibm.com/support/knowledgecenter/es/SSLVMB_23.0.0/spss/base/dataedit_define_variable_measurement.html

Cerian, J. (2015). Errores de diagnóstico en la práctica médica. *Arch Argent Pediatr: scielo, 113*(3), 194-195.

Coello, C., Gary, L., & Van Veldhuizen, D. (2007). *Evolutionary algorithms for solving multi-objective problems vol. 5.* New York: Springer.

Delgado, A. (1999). Aplicación de las Redes Neuronales en Medicina. *Revista de la Facultad de Medicina*, 221-223.

Díaz, J., Fernandez, T., & Parede, F. (1997). *Aspectos básicos de bioquímica clínica.* Madrid: Ediciones Díaz de Santos.

Dragoni, M., Azzini, A., & Tettamanzi, A. (2014). SimBa: A novel similarity-based crossover for neuro-evolution. *Neurocomputing; Elsevier*, 130,108-122.

Durán, E., & Costaguta, R. (2007). Minería de datos para descubrir estilos de aprendizaje. *Revista iberoamericana de educación; OEI*, 42(2),1-10.

EducarChile. (24 de 11 de 2016). Obtenido de Neurona y conducción nerviosa: http://www.educarchile.cl/ech/pro/app/detalle?ID=137486

Flórez, R., & Miguel, F. J. (2008). *Las Redes Neuronales Artificiales, Metodología y Análisis de Datos en Ciencias Sociales.* Oleiros: NetBiblo.

Fogel, L. (1964). On the Evolution of Artificial Intelligence. *Proceedings of the Fifth National Symposium on Human Factors in Electronics*, 63-76.

Fraser, A., & Burnell, D. (1970). *Computer models in genetics.*

Glover, F., & Laguna, M. (2000). Fundamentals of Scatter Search and Path Relinkig. *Control and Cybernetics*.

Gowda, A., Manjunath, A., & Jayaram, M. (2011). Application of genetic algorithm optimized neural network connection on weights for medical diagnosis of pima indians diabetes. *International journal on soft computing; AIRCC*, 2(2),15-23.

Gurney, K. (2003). *An Introduction to Neural Networks.* New York: CRC Press.

Hebb, D. (1949). *The Organization of Behavior A Neuropsychological Theory.* New York: John Wiley & Sons, Inc.

Hernández, J. (2004). *Introducción a la minería de datos.* Madrid: Pearson Prentice Hall.

Hopfield, J., & Tank, D. (1985). “Neural” computation of decisions in optimization problems. En *Biological cybernetics* (págs. 141-152). Berlin: Springer Berlin Heidelberg.

Houghton, A. R. (2011). *Chamberlain: sintomas y signos en la medicina clinica: una introduccion al diagnostico médico.* Reino Unido: McGraw Hill.

Inteligencia artificial : fundamentos, práctica y aplicaciones. (2012). RC Libros.

Lafarja, C. M. (1994). *Biología celular de la neurona y de la sinapsis.* Cantabria: Ed. Universidad de Cantabria.

Laguna, M., & Cunquero, R. (2003). *Scatter Search: Methodology and Implementations in C.* Massachusetts: Kluwer Academic Publishers.

Latindex. (s.f.). *Sistema Regional de Información en Línea para Revistas Científicas de América Latina, el Caribe, España y Portugal*. Recuperado el 2017 de 12 de 01, de http://www.latindex.org/latindex/ficha?folio=22701

Macarulla, T., Ramos, F., & Tabernero, J. (2011). *Comprender el cáncer.* Barcelona: Editorial AMAT.

Madero, I. T. (s.f.). *LANTI*. (Tecnológico Nacional de México) Recuperado el 27 de 11 de 2017, de http://lanti.org.mx/index.php/infraestructura#equipo

Mathivet, V. (2017). *Inteligencia artificial para desarrolladores: conceptos e implementación en Java.* Barcelona: Ediciones Eni.

Maureira, F. (2017). *¿Qué es la inteligencia?* España: Bubok publishing.

McCulloch, W., & Pitts, W. (1943). Logical calculus of the ideas immanent in nervous activity. *Bulletin of mathematical biophysics*, 115-133.

Minsky, M., & Papert, S. (1972). *Perceptrons.* Massachusetts: MIT Press.

NCI. (9 de Marzo de 2015). *National Cancer Institute*. (Physician Data Query) Recuperado el 31 de Octubre de 2017, de https://www.cancer.gov/espanol/cancer/diagnostico-estadificacion/estadificacion

Oracle. (s.f.). *Oracle The Java Documentation*. (Oracle) Recuperado el 2017 de 11 de 21, de https://docs.oracle.com/javase/tutorial/essential/concurrency/procthread.html

Ortiz, D., & Villa, F. (2007). A Comparison between Evolutionary Strategies. *Avances en sistemas e informatica*, 135-144.

Pino, D. R., Gomez, G. A., & De Abajo, M. N. (2001). *Introducción a la inteligencia artificial: sistemas expertos, redes neuronales artificiales y algoritmos evolutivos.* Oviedo: Universidad de Oviedo.

Rechenberg, I. (1978). Evolutionsstrategien. En *Simulationsmethoden in der Medizin und Biologie* (págs. 83-114). Berlin: Springer Berlin Heidelberg.

Reeves, C. (17 de 11 de 1993). Using Genetic Algorithms With Small Populations. *ICGA*, 92. Obtenido de ResearchGate: https://www.researchgate.net/profile/Colin_Reeves/publication/2264198_Using_Genetic_Algorithms_With_Small_Populations/links/00b495205750f212f9000000.pdf

Reeves, C. R. (2010). Genetic Algorithms. En M. Gendreau, & J.-Y. Potvin, *Handbook of Metaheuristics* (págs. 109-141). New York: Springer.

Rosenblatt, F. (1958). The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, 386-408.

Rumelhart, D., Hinton, G., & Williams, R. (1986). Learning representations by back-propagating errors. *Nature*, 533-536 .

Sabán, J. (2012). *La Diabetes Mellitus como enfermedad sistémica.* Madrid: Ediciones Díaz de Santos.

SciStatCalc. (s.f.). *Wilcoxon Signed Rank Test Calculator*. Recuperado el 2017 de 11 de 30, de http://scistatcalc.blogspot.mx/2013/10/wilcoxon-signed-rank-test-calculator.html

Sempere, M., Gallego, F., Llorens, F., Pujol, M., & Rizo, R. (2005). Un Entorno de Aprendizaje Neuroevolutivo:. *Department of Computer Science and Artificial Intelligence University of Alicante*.

Tébar, F., & Escobar, F. (2009). *La diabetes en la practica clínica.* Madrid: Editorial Médica Panamericana.

UC Irvine Machine Learning Repository. (s.f.). (Center of Machine Learning and Intelligent Systems) Recuperado el 2017 de 11 de 28, de https://archive.ics.uci.edu/ml/index.php

Valencia, M., & Márquez, C. (2006). *Instituto Politecnico Nacional.* Obtenido de Algoritmo Backpropagation para redes neuronales: conceptos y aplicaciones: http://www.repositoriodigital.ipn.mx/bitstream/handle/123456789/8628/Archivo%20que%20incluye%20portada,%20%EDndice%20y%20texto.pdf?sequence=1

Waikato, T. U. (01 de Abril de 2002). *Attribute-Relation File Format (ARFF)*. Recuperado el 01 de Noviembre de 2017, de https://www.cs.waikato.ac.nz/ml/weka/arff.html