ALGORITMO PARA LA INFERENCIA DE LA PROPIEDAD FISICOQUÍMICA GRAVEDAD API DE CRUDOS A PARTIR DE LA LECTURA DE RMN UTILIZANDO REDES NEURONALES ARTIFICIALES CON ENTRENAMIENTO BASADO EN OPTIMIZACIÓN BIOINSPIRADA

SONIA VIVIANA LUNA CARRILLO
YULY PAOLA LIPEZ PINZON

UNIVERSIDAD INDUSTRIAL DE SANTANDER

FACULTAD DE INGENIERAS FSICO MECNICAS

ESCUELA DE INGENIERA DE SISTEMAS E INFORMTICA

BUCARAMANGA

2011

ALGORITMO PARA LA INFERENCIA DE LA PROPIEDAD FISICOQUÍMICA GRAVEDAD API DE CRUDOS A PARTIR DE LA LECTURA DE RMN UTILIZANDO REDES NEURONALES ARTIFICIALES CON ENTRENAMIENTO BASADO EN OPTIMIZACIÓN BIOINSPIRADA

SONIA VIVIANA LUNA CARRILLO
YULY PAOLA LIPEZ PINZON

Trabajo presentado como requisito para optar el título de
Ingeniero de Sistemas.

DIRECTOR:
ALFONSO MENDOZA CASTELLANOS

CODIRECTOR:
GUILLERMO LUQUE Y GUZMAN SAENZ

UNIVERSIDAD INDUSTRIAL DE SANTANDER
FACULTAD DE INGENIERAS FSICO MECNICAS
ESCUELA DE INGENIERA DE SISTEMAS E INFORMTICA
BUCARAMANGA
2011

DEDICATORIA

Dedico este trabajo a
A Dios por permitir y ayudarme a cumplir mis sueños
A mis padres, María Eugenia y Germán, por hacerme la persona que hoy soy, por ayudarme apoyarme y siempre confiar en mí y en mis capacidades.
A mi hermana Kelly, por ser más que una hermana, una amiga, una cómplice, por sus consejos y aguantarme en los momentos más difíciles .
A mi hermano Jefferson por ser el motor de mi vida.
A Sonia por ser una compañera increíble, sus consejos y dedicación hicieron esto posible.
A mi amiga Rossi Basto por darme ánimos cuando sentía que ya no podía.
A mis Amigos porque hicieron que mi paso por la universidad sea una de las mejores etapas de mi vida.
A toda mi familia, abuelas, tíos y primos por creer siempre en mí, sin ustedes jamás habría podido cumplir este sueño.
Gracias

Yuly Paola Lípez Pinzón

Dedico este trabajo a
A Dios, por permitirme recorrer este camino y finalizarlo con éxito.
A mis padres Misaelito y Carmencita a quienes amo con todo el corazón. A pesar de no estar conmigo físicamente sus palabras me llenaron de fuerza para continuar.
A mis hermanos Alba, Naydú y Nelson quienes vivieron conmigo este proceso, No hubiesie sido lo mismo sin ustedes. Los quiero mucho.
A mi familia quienes siempre me han apoyado. Mi tía Lilia y primo Luis por estar pendientes de mí. A mis abuelitos y tíos que con sus oraciones me ayudaron siempre
A mi compañera Yuly de quien aprendí mucho.
A todos mis compañeros y amigos en este camino.
Al GIIB por darme la oportunidad de investigar y crecer no solo en la parte profesional, sino también en lo personal.

Sonia Viviana Luna Carrillo

AGRADECIMIENTOS

Las autoras del proyecto expresan sus agradecimientos:

A Dios, por Darnos la sabiduría necesaria para lograr este proyecto.

A Fabián Cardozo por su dedicación, consejos y aportes al proyecto, más que un asesor, un amigo.

Al grupo de Investigación en ingeniería Biomédica (GIIB) por el apoyo incondicional, por permitirnos ser parte de este grupo que nos hizo crecer no solo en la parte profesional sino en la personal.

Al profesor Alfonso Mendoza Director del proyecto, por creer en nosotras.

Al profesor Guillermo Luque y Guzmán Codirector, por sus explicaciones y dedicación sin esto no habríamos podido sacar este proyecto adelante.

A la Universidad Industrial de Santander por darnos las herramientas necesarias para formarnos como profesionales

Al Instituto Colombiano de Petróleos (ICP) por permitirnos desarrollar este proyecto.

A nuestras familias por darnos todo su amor y apoyo incondicional.

Índice general

Índice de figuras

Índice de cuadros

Anexos

RESÚMEN

Título: ALGORITMO PARA LA INFERENCIA DE LA PROPIEDAD FISICOQUÍMICA GRAVEDAD API DE CRUDOS A PARTIR DE LA LECTURA DE RMN UTILIZANDO REDES NEURONALES ARTIFICIALES CON ENTRENAMIENTO BASADO EN OPTIMIZACIÓN BIOINSPIRADA*

Autor(es) : SONIA VIVIANA LUNA CARRILLO**
YULY PAOLA LIPEZ PINZON**

Palabras Clave : REDES NEURONALES, COLONIA DE HORMIGAS, GRAVEDAD API, ALGORTIMO BACKPROPAGATION.

En la industria petrolera, es de suma importancia el estudio de las propiedades fisicoquímicas del petróleo, ya que estas proporcionan información que ayuda al experto a decidir la calidad del crudo. Una de estas propiedades es la Gravedad API, que suministra información sobre que tan pesado o liviano es un crudo en comparación al agua.
Para obtener estas propiedades fisicoquímicas es necesario realizar pruebas de laboratorio que muchas veces son costosas y el tiempo en el laboratorio es considerable. Se requieren entonces métodos que agilicen este proceso.
En este trabajo se presenta el desarrollo de un algoritmo basado en optimización por colonia de hormigas que, dada la arquitectura de una red neuronal artificial encuentre los pesos iniciales adecuados que harán que la red neuronal obtenga resultados confiables. La red neuronal es entrenada con los datos de la RMN y como datos de salida los valores de la propiedad fisicoquímica gravedad API. El algoritmo de entrenamiento utilizado es el Backpropagation, este tiene una característica y es que algunas veces se queda atrapado en un minimo local haciendo que la red neuronal no de resultados satisfactorios. Con la ayuda del algortimo de optimización por colonia de hormigas se evita este problema, haciendo que la red neuronal tenga una rápida convergencia

*Proyecto de grado

**Facultad de Ingeniería Físico-Mecánicas. Escuela de Ingeniería de sistemas e informática. Director Alfonso Mendoza Castellanos. Codirector Gullermo Luque y Guzmán

ABSTRACT

Title: Algorithm for inference of physicochemical property API gravity of crude from reading NMR Using artificial neural networks based on optimization with training Bioinspired*
Authors : Sonia Viviana Luna Carrillo**
Yuly Paola Lípez pinzón**

Key Words : Neural networks, ant colonies, API gravity, backpropagation algorithm.

In the petroleum industry, it is very important to study the physicochemical properties of oil, as these provide information that helps the expert to decide the quality of oil. One of these properties is the API gravity, which provides information about how heavy or light oil is compared to water. For these physicochemical properties It's necessary laboratory tests that are often expensive and the time in the lab is considerable. Then methods are needed to expedite this process.

This paper presents the development of an algorithm based on ant colony optimization, when the architecture of an artificial neural network is given, ants find the appropriate initial weights that will make the neural network to obtain reliable results. The neural network is trained with NMR data and as output the values of the physicochemical property API gravity. The training algorithm used is backpropagation, it has a feature and it sometimes gets stuck in a local minimum by the neural network results are not satisfactory. With the help of the algorithm of ant colony optimization for this problem is avoided by making the neural network has a fast convergence.

** Physics and Mechanics Engineering College. Systems and Informatics School. Director Alfonso Mendoza Castellanos. Codirector Gullermo Luque y Guzmán

INTRODUCCIÓN

Uno de los principales objetivos de la industria petrolera es obtener productos de calidad, por esto, a cada uno de sus productos, son realizados pruebas para medir la calidad y especificaciones. Para la realización de estas pruebas, muchas veces, es necesario realizar estudios a nivel de laboratorio. Dentro de estas pruebas se encuentran la medición de propiedades fisicoquímicas, las cuales pueden llegar a ser costosas en tiempo y dinero. Se ha recurrido entonces a métodos más rápidos y económicos para conocer las especificaciones de los productos. Uno de estos métodos es la resonancia magnética nuclear [1] Con los resultados de la resonancia, es posible caracterizar los hidrocarburos. Teniendo en cuenta el área bajo la curva de los datos de la resonancia se obtienen los grupos funcionales presentes en el crudo que están relacionados con las propiedades fisicoquímicas, es decir, conociendo los grupos funcionales presentes en el crudo se conocen de manera implícita las propiedades fisicoquímicas, sin embargo, es necesaria una herramienta que permita estimar y clasificar los datos arrojados por las resonancias, relacionándolos con la propiedad API. Teniendo en cuenta que las redes neuronales artificiales han demostrado ser una alternativa eficiente en la solución de problemas donde es necesario

[1] La espectroscopia de RMN fue desarrollada a finales de los años cuarenta para estudiar los núcleos atómicos. Los químicos descubrieron que la espectroscopia de resonancia magnética nuclear podía ser utilizada para determinar las estructuras de los compuestos orgánicos. Esta técnica espectroscópica puede utilizarse sólo para estudiar núcleos atómicos con un número impar de protones o neutrones (o de ambos). Esta situación se da en los átomos de H^1,C^{13},F^{19} y P^{31}.

correlacionar parámetros, la industria de los hidrocarburos las ha utilizado en diferentes áreas. Algo fundamental en una red neuronal artificial es su algoritmo de entrenamiento, cuyo objetivo es ajustar los pesos para obtener el mínimo error de salida. El algoritmo más utilizado es el backpropagation, pero este algoritmo tiene lenta convergencia y fácilmente puede quedar atrapado en un mínimo local. [1]

El algoritmo de optimización por colonia de hormigas propuesto por Dorigo a principios de los noventa es un algoritmo meta heurístico, que está inspirado en el comportamiento social de las hormigas. [2] Fue diseñado para resolver problemas de optimización combinatoria. Trabajos investigativos han demostrado que la optimización por colonia de hormigas es un buen método para encontrar la mejor configuración de una red neuronal [1], ya que en el espacio de búsqueda de dicha configuración hay múltiples combinaciones de pesos y conexiones entre neuronas, por lo tanto se convierte en un problema combinatorio, siendo las conexiones las más importantes para obtener una respuesta adecuada de la red .

Capítulo 1

DESCRIPCIÓN DEL PROYECTO

1.1. Objetivos

1.1.1. Objetivo General

Diseñar un algoritmo para la inferencia de la propiedad fisicoquímica Gravedad API de crudos a partir de la lectura de RMN utilizando redes neuronales artificiales con entrenamiento basado en optimización por colonia de hormigas.

1.1.2. Objetivos Específicos

1. Diseñar e implementar un algoritmo hibrido backpropagation-colonia de hormigas para el entrenamiento de la red neuronal artificial perceptrón multicapa con el objetivo de:

    - Encontrar las conexiones aptas entre neuronas para la red neuronal artificial.
    - Encontrar los pesos adecuados para el entrenamiento de la red neuronal artificial.

2. Generalizar la red neuronal para inferir la propiedad fisicoquímica Gravedad API de los crudos según datos proporcionados por el ICP

3. Medir el rendimiento y la eficiencia del algoritmo desarrollado.

1.2. Justificación

Diversas áreas del conocimiento, están haciendo uso de la inteligencia artificial para llegar a soluciones aproximadas cuya solución exacta es costosa o no existe un método determinístico para hacerlo. Por ejemplo, para conocer las propiedades fisicoquimicas de los hidrocarburos, se debe acudir a pruebas de laboratorio que son costosas y demoradas.

Se requiere entonces, una técnica que ayude a encontrar la solución y además reduzca los costos. Debido a la efectividad de las redes neuronales en la solución de este tipo de problemas, se propone la creación de una red neuronal para la inferencia de la propiedad API.

Para que la red neuronal efectúe una buena aproximación, es necesario encontrar una configuracion óptima de manera que encuentre soluciones de manera precisa y exacta.

La metodología aplicada pretende optimizar los pesos iniciales de una red neuronal artificial con una arquitectura definida, pues no existen métodos para seleccionar los pesos iniciales que garanticen encontrar el mínimo global, en general, dichos pesos se generan aleatoriamente, por lo que es conveniente realizar varias sesiones de entrenamiento sobre la misma red con diferentes pesos, en forma tal de comenzar

el aprendizaje desde diferentes puntos de la función objetivo y así tener más posibilidad de alcanzar el mínimo global.

ACO tiene la capacidad de buscar la solución óptima global, y el algoritmo back-propagation tiene la característica de rápida convergencia de los óptimos locales. El híbrido adecuado de los dos algoritmos (ACO-BP) puede acelerar la velocidad de evolución de las redes neuronales y mejorar la precisión de la red.

Capítulo 2

ESTADO DEL ARTE

La optimización de la estructura de una red neuronal puede ser catalogada de acuerdo a la meta a alcanzar; al respecto se han realizado distintas investigaciones, en las cuales, la optimización puede verse desde el punto de vista de la topología o arquitectura, para la cual se busca la mejor manera de configurar la cantidad de capas y número de neuronas por capa, o la optimización de los pesos, en la cual se busca la mejor manera de combinar los pesos en una arquitectura de red definida [3].

Para la optimización de la topología de la red, se han empleado diversas técnicas, entre ellos los métodos evolutivos. Los mas empleados han sido los algoritmos genéticos [4]. Al respecto, en la Universidad Industrial de Santander, se han relizado proyectos de grado, en los cuales se generan topologías óptimas de redes neuronales ,encontradas mediante algoritmos genéticos [5].

No existen métodos para seleccionar los pesos iniciales que garanticen encontrar el mínimo global, en general, dichos pesos se generan aleatoriamente, por lo que es conveniente realizar varias sesiones de entrenamiento sobre la misma red con diferentes pesos, en forma tal de comenzar el aprendizaje desde diferentes puntos de

la función objetivo y así tener más posibilidad de alcanzar el mínimo global. Por este motivo, para la optimización de los pesos, los investigadores han estudiado la evolución de los pesos, otras usan la capacidad de la búsqueda global para determinar el espacio de búsqueda de los pesos para el problema y luego usan backpropagation como una búsqueda local para refinar los pesos [5].

Específicamente, el híbrido entre redes neuronales con colonia de hormigas, ha sido utilizado, pués ACO tiene la capacidad de buscar la solución óptima global, y el algoritmo backpropagation tiene la característica de rápida convergencia de los óptimos locales. El híbrido adecuado de los dos algoritmos (ACO-BP) puede acelerar la velocidad de evolución de las redes neuronales y mejorar la precisión de las redes bien entrenadas [6].

ACO originalemente es aplicado a problemas de optimización discreta, por ejemplo los problemas combinatorios. Blum en su artículo [7], propone la aplicación ACO a problemas de optmización continua. Para lograrlo, el lugar de resolver un problema combinatorio, se asume que se quiere minimizar un función continua.

El algoritmo modificado es llamado ACOR y es aplicado para hallar la mejor configuración de pesos de una red neuronal feedforward [1]. Como los pesos pertenecen al espacio de los reales, este problema se convierte en un problema de optmización continua.

El espacio de búsqueda puede ser modelado por n variables de decisión $X_i(i = 1, ..., n)$ con dominios continuos. ACOR usa un archivo solución T para deducir la distribución de probabilidad sobre el espacio de búsqueda.El archivo solución man-

[1] Retroalimentada

tiene un historial de su proceso de búsqueda por medio de almacenamiento de soluciones. En primer lugar, el archivo de solución debe ser inicializado. Luego, en cada iteración una serie de soluciones son probabilísticamente construidas. Estas soluciones pueden ser mejoradas por cualquier mecanismo de mejora (por ejemplo, la búsqueda local o una técnica de gradiente). Por último, el archivo se actualiza con la solución de las soluciones generadas.

Como resultado a la investigación hecha por Blum en [7], se demostró que el rendimiento de ACOR independiente fue en general malo respecto al rendimiento de los algoritmos especializados para la formación de redes neuronales[2], sin embargo, el híbrido entre ACOR y el algoritmo de Levenberg-Marquardt o el algoritmo de backpropagation, da buenos resultados en un problema de clasificación. Respecto a otros algoritmos de optimización como algoritmos genéticos, ACOR proporciona ampliamente mejores resultados.

A diferencia del enfoque de Blum, autores como Liu [8] o Pokudom [1] , prefieren transformar el problema de optimización de configuración de los pesos en un problema discreto con el fin de aplicar el algortimo ACO original.

ACO guía el proceso de búsqueda, es decir con ACO se obtienen los pesos iniciales, y bakpropagation refina solución. Las soluciones candidatas se construyen de una manera probabilística utilizando una distribución de probabilidad sobre el espacio de búsqueda, el cuál es dividido en puntos discretos. Se selecciona un rango de pesos Wmax y Wmin y dentro de ese rango están los puntos candidatos.

Las soluciones candidatas se utilizan para modificar la distribución de probabilidad

[2]Como backpropagation y Levenberg-Marquardt

de manera que se guía hacia soluciones de alta calidad.

Los algoritmos ACO para problemas de optimización combinatoria hacen uso de un modelo de feromonas con el fin de construir soluciones probabilísticamente. Un modelo de feromonas es un conjunto de los llamados parámetros de camino de feromonas. Los valores numéricos de estos parámetros de feromona (es decir, los valores de feromonas) reflejan la experiencia de búsqueda del algoritmo y se utilizan para la construcción de la solución, con el tiempo las regiones del espacio de búsqueda que contienen soluciones de alta calidad.

Capítulo 3

MARCO TEÓRICO

3.0.1. Marco Teórico

Pruebas de Laboratorio Espectroscopía por Resonancia Magnética Nuclear (RMN)

Uno de los objetivos de la industria de los hidrocarburos es entregar productos refinados óptimos. Usualmente para conocer el rendimiento y la calidad de las fracciones resultantes es necesario evaluar y estudiar cada crudo a nivel de laboratorio. El tiempo de laboratorio requerido para estas pruebas es de 3-4 semanas y podría costar alrededor de U$ 15.000 por crudo, dependiendo de la cantidad de fracciones y el análisis fisicoquímico requerido. Por lo tanto, para las refinerías, es de gran interés encontrar métodos mas rápidos y menos costosos [9].

La Espectroscopía por Resonancia Magnética Nuclear ha sido ampliamente aplicada para caracterizar las diferentes fracciones de los crudos mediante la determinación de promedios de parámetros moleculares o por la cuantificación de otras familias de hidrocarburos obteniendo muy buenos resultados. A continuación se dirá en qué consiste esta prueba de laboratorio.

Resonancia Magnética Nuclear(RMN)

La espectroscopia por RMN fue desarrollada a finales de los años cuarenta para estudiar los núcleos atómicos. En 1951 los químicos descubrieron que la espectroscopia de resonancia magnética nuclear podía ser utilizada para determinar las estructuras de los compuestos orgánicos. Esta técnica espectroscópica puede utilizarse sólo para estudiar núcleos atómicos con un número impar de protones o neutrones (o de ambos) [10].

La RMN estudia los núcleos atómicos al alinearlos a un campo magnético constante para posteriormente perturbar este alineamiento con el uso de un campo magnético alterno, de orientación ortogonal.

Para obtener un espectro de RMN, se coloca una pequeña cantidad del compuesto orgánico disuelto en medio mililitro de disolvente en un tubo de vidrio largo que se sitúa dentro del campo magnético del aparato. El tubo con la muestra se hace girar alrededor de su eje vertical. Un breve pulso de radiación excita a todos los núcleos simultáneamente. A medida que dichos núcleos vuelven a su posición inicial estos emiten una radiación de frecuencia, un ordenador recoge la intensidad respecto al tiempo y convierte dichos datos en intensidad respecto a frecuencia, esto es lo que se conoce con el nombre de transformada de Fourier.

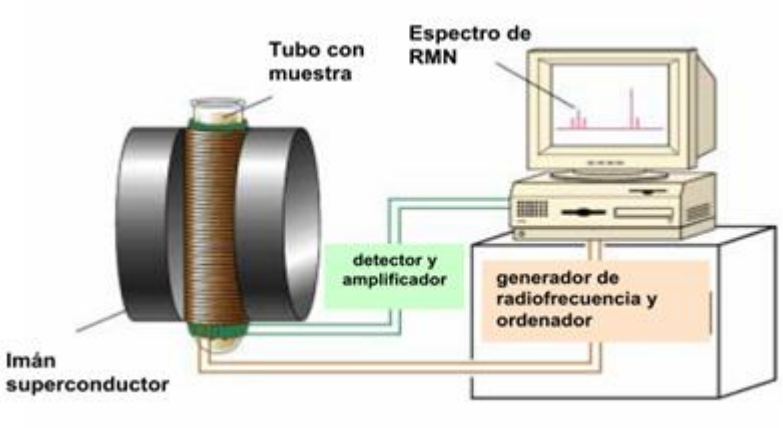


Figura 3.1 Equipo Utilizado para la toma de Resonancia Magnética Nuclear

Fuente:**http://www.uv.es/jcastell/Espectroscopia.pdf**

Gravedad API

La Gravedad API, es una medida de densidad que describe que tan ligero o pesado es el petróleo en comparación con el agua. Si los grados API son mayores que 10, es más liviano que el agua. La gravedad API es usada para comparar densidades de fracciones extraídas del petróleo.

Generalmente, un mayor valor de gravedad API en un producto de refinería representa que este tiene un mayor valor comercial, esto es debido a la facilidad de producir destilados valiosos como gasolina, jet fuel y gasóleo con alimentaciones de crudos livianos y a los altos rendimientos de los mismos. Esta regla es válida hasta los 45 grados API, mas allá de este valor hacen que los productos tengan menor valor comercial.

El petróleo es clasificado en liviano, mediano, pesado y extrapesado, de acuerdo a su medición de gravedad API.

- **Crudo Liviano:** Es aquel que tiene gravedades API mayores a 31.1 grados API

- **Crudo Mediano:** Gravedad API entre 22.3 y 31.1 grados API
- **Crudo Pesado:** Definido como aquel que tiene gravedades API entre 10 y 22.3 grados API
- **Crudo Extrapesado:**Son aquellos que tienen gravedades API menores que 10 grados API.

3.0.2. Redes Neuronales

Inspiracion Biológica y Modelado de una Neurona

Las neuronas reciben señales de otras neuronas vía conexiones sinápticas que pueden ser excitantes o inhibidoras. En función de las señales recibidas, una neurona envía a su vez una señal a otras neuronas por medio del axón. Una neurona contiene un potencial interno continuo llamado potencial de membrana. Cuando este excede un cierto valor umbral, la neurona puede transmitir todo su potencial por medio del axón [11].

Se estima que el cerebro humano contiene más de cien mil millones 10^{11} de neuronas y que hay más de 1000 sinapsis a la entrada y a la salida de cada neurona.
Una neurona artificial, o unidad procesadora que modela matemáticamente una neurona biológica, sobre un conjunto de nodos N, es una tripleta X, f, Y donde X es un subconjunto de N, Y es un único nodo de N y $f :\rightarrow XXXXX$ es una función neuronal (también llamada función activación) que calcula un valor de salida para Y basado en una combinación lineal de los valores de las componentes de X, es decir,

$$Y = f(\sum_{x_i \in X} w_i x_i). \tag{3.1}$$

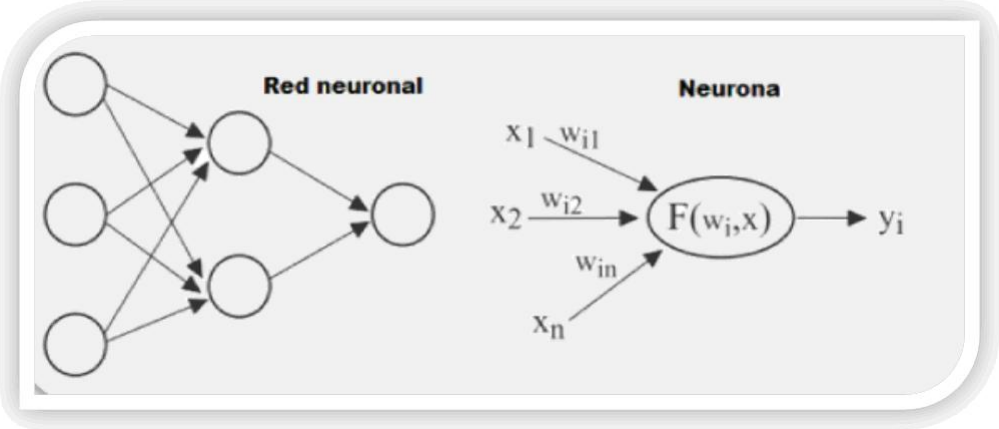


Figura 3.2 Estructura de una Red Neuronal

Fuente: Autores del proyecto

Los elementos X, Y y f se denominan conjunto de nodos de entrada, conjunto de nodos de salida, y función neuronal de la unidad neuronal, correspondientemente; x_i se relaciona con cada una de las entradas y w_i su respectivo peso asociado.(Ver figura 3.2).

Red Neuronal artificial

Una red neuronal artificial es un procesador masivamente paralelo que tiende naturalmente a almacenar conocimiento empírico y a hacerlo aprovechable. Se asemeja al cerebro en dos aspectos:

1. El conocimiento es adquirido del ambiente por la red a través de un proceso de aprendizaje.

2. Las fuerzas de conexión entre neuronas, conocidos como pesos sinápticos, son usados para almacenar el conocimiento [12].

Una red neuronal artificial (RNA) es un par (N, U), donde N es un conjunto de nodos y U es un conjunto de unidades procesadoras sobre N que satisface la siguiente condición:

Cada nodo $x_i \in N$ tiene que ser un nodo de entrada o de salida de al menos una unidad procesadora de U. El comportamiento de la red está determinado por su topología, los pesos de las conexiones y la función característica de las neuronas. (Ver figura 3.2). Dependiendo del problema para el cual la red neuronal es diseñada,

esta puede tener aprendizaje supervisado o no supervisado.

Aprendizaje Supervisado: El aprendizaje supervisado esta basado en comparaciones directas entre la salida actual de la RNA y la salida deseada, también conocidas como salidas objetivo.

Aprendizaje No Supervisado: Este tipo de aprendizaje no requiere la influencia externa para ajustar los pesos de las conexiones entre sus neuronas. La red no recibe ninguna información por parte del entorno que le indique si la salida generada en respuesta a una determinada entrada es correcta. Estas redes deben encontrar las correlaciones o categorías que se pueden establecer entre los datos de entrada, la salida puede ser interpretada de varias maneras:

- La salida representa el grado de familiaridad o similitud entre la información de entrada y las informaciones mostradas con anterioridad.
- Clusterización o establecimiento de categorías, la salida indica a que categoría pertenece la información de entrada, siendo la propia red la que establece las correlaciones oportunas.
- Mapeo de características, obteniéndose una disposición geométrica que representa un mapa topográfico de las características de los datos de entrada.

Construcción de la Topología de Redes Neuronales artificiales

Se puede nombrar topología a la arquitectura o estructura de la red neuronal, la cuál consiste en la manera en cómo se conectan cada uno de los nodos o neuronas. Estas conexiones determinan el comportamiento de la red. Generalmente las neuronas se agrupan en unidades estructurales denominadas capas. Dependiendo del número de capas se constituirá la red neuronal.

La elección de la mejor configuración de una red neuronal no es una tarea fácil debido a la gran cantidad de configuraciones posibles en cuanto al número de capas y el número de neuronas por capa. Como no existe una metodología clara para este proceso, se ha recurrido a la técnica de prueba y error, que es tediosa y además no garantiza el mejor resultado. Una manera de elegir la mejor configuración de una red neuronal es mediante el teorema de Cybenko.

Teorema de Cybenko

Sea φ cualquier función sigmoidea (por ejemplo, la función logística). Dada cualquier función continua en $[01]^n$(o en cualquier conjunto compacto de R^n) y un $\epsilon > 0$, existen unos vectores $W_1, W_2, ..., W_N, \alpha y \theta$, y una función parametrizada $G(., W, \alpha, \theta) de [0, 1]^n$en R tal que :

$$|G(x, W, \alpha, \theta) - f(x)| < \epsilon, \forall x \in [0, 1]^n$$

, donde

$$G(x, W, \alpha, \theta) = \sum_{j=1}^{N} \alpha_j \varphi(W_j^T x + \theta_j)$$

y

$$W_j \in R^n, \alpha_j, \theta_j \in R$$

Dicho resultado establece que un perceptrón multicapa con una única capa intermedia de unidades ocultas es capaz de aproximar uniformemente cualquier función multivariante con el grado de precisión deseado. Por lo tanto, cuando se aplica el perceptrón multicapa a un problema real para aproximar una función continua y no se consigue la precisión deseada es porque no se ha conseguido una determinación adecuada de los pesos sinápticos de la red o no se ha utilizado el número apropiado de unidades de proceso en la capa oculta.

Algoritmo de Entrenamiento Backpropagation(Propagación hacia atrás)

Este es el algoritmo de entrenamiento más utilizado, y fue propuesto por primera vez en el año 1974 por Paul Werbos, este algoritmo se desarrollo en un contexto muy general, para cualquier tipo de redes, siendo la red neuronal artificial un caso especial, razón por la cual no fue aceptado por la comunidad de desarrolladores de redes neuronales. A mediados del los 80 cuando el algoritmo backpropagation fue redescubierto por varios investigadores David Rumelhart, Geoffrey Hinton y Ronal Williams y fue incluido por los sicólogos David Rumelhart y James McClelland en el libro “Parallel Distributed Processing Group” cuando tomó gran fuerza en el campo de la investigación con redes neuronales. [13]

El algoritmo Backpropagation está basado en la evaluación del error o función de calidad (FW), que depende de los pesos. Esta función evalúa la calidad de la salida generada por la neurona comparada con la salida deseada (esperada).

$$F(W) = \frac{1}{2}\sum_{p=1}^{P}(t_p - y_p)^2 \tag{3.2}$$

$\boldsymbol{p}$: Entrada o patrón que se está evaluando

$\boldsymbol{P}$: Número total de Patrones

$\boldsymbol{t_p}$: Target o salida esperada

$\boldsymbol{y_p}$: Salida generada por la red

$\boldsymbol{W}$: Vector de pesos.

Estos errores calculados con la ecuación (3.2) se retro propagan desde la capa de salida a todas las neuronas en la capa oculta o capas ocultas que interfieran directamente en la salida, este proceso se repite capa por capa, hasta que todas las neuronas de la red hayan recibido un error que describa su aporte al error total. Basándose en el error recibido, se reajustan los pesos de conexión de cada neurona de manera que la próxima vez que se entrene la red con el mismo patrón, la salida obtenida esté más cerca de la salida deseada.

Para el Reajuste o actualización de los pesos, es necesario el cálculo del gradiente que define la superficie del error.

$$g = \frac{\partial E}{\partial W_j} = -f' * (\sum_{j=1}^{n} w_j * x_j) * x_j \tag{3.3}$$

$\boldsymbol{f}$: Función de activación del nodo.

$\boldsymbol{w_j}$: Peso de la conexión j.

$\boldsymbol{x_j}$: Entrada del nodo j.

El Gradiente tendrá la dirección de incremento del error, por lo tanto el negativo del gradiente dará la dirección para minimizar el error. Después de obtener el

gradiente se modifican los pesos de la siguiente manera:

$$w_{j+1} = w_j + \alpha * g * x_j \tag{3.4}$$

$\boldsymbol{\alpha}$: Tasa de aprendizaje.

Una variante al algoritmo backpropagation es el algoritmo Levenberg-Marquardt (LM), se explicará en qué consiste este algoritmo.

Algortimo Levenberg-Marquardt(LM)

El algoritmo Backpropagation ha demostrado converger muy lentamente en varias aplicaciones, en especial cuando se tienen gran cantidad de patrones de entrada y hacen que el error cuadrático medio sea demasiado grande y es por esto que muchas veces este algoritmo se queda atrapado en un mínimo local. Levenberg-Marquardt es un algoritmo iterativo basado en el método de aproximación de Newton y trata de minimizar la distancia entre dos puntos.

El algoritmo de Levenberg-Marquardt tiene convergencia rápida a pesar de que su complejidad en cálculos es mayor. Este algoritmo para cálculo del gradiente utiliza la matriz jacobiana que se define como la primera derivada de los errores de la red con respecto a los pesos.

$$g = J^T * E \tag{3.5}$$

Donde J^T es la matriz jacobiana traspuesta y E es el vector de errores entre la salida esperada y la obtenida.

La actualización de los pesos se hace de la siguiente manera:

$$W_{k+1} = W_k + \Delta * W_k \tag{3.6}$$

Debido a la gran cantidad de arquitecturas posibles, se debe buscar la que mejor se ajuste a los resultados esperados, para esto se utilizan técnicas de inteligencia artificial como optimización por colonia de hormigas, en la que se enfoca este proyecto. Si bien es poca la literatura al respecto se ha demostrado que la optimización por colonia de hormigas es un buen método para encontrar la mejor configuración de una red neuronal [1], ya que la especialidad de éste son los problemas combinatorios.

Otra técnica de inteligencia artificial a la que se ha recurrido para resolver este problema son los algoritmos genéticos. Estos utilizan el error de la red entrenada como medida de desempeño para guiar la evolución. De esta forma la búsqueda de la mejor estructura R de una red neuronal se puede ver como la maximización de la función F dada por la ecuación mostrada a continuación:

$$R = F(M_i, N_i, f_i, T_i, x_i) \tag{3.7}$$

donde M_i representa el número de capas de la red, N_i es el número de neuronas por cada capa, f_i es la función de cada una de las capas, T_i es el algoritmo de entrenamiento y x_i son los ejemplos con que se cuenta para entrenar la red neuronal. Además cada una de estas variables está limitada a un conjunto finito de valores para acotar el espacio de búsqueda, de esta forma $M_i \in \{1, m\}$, donde m es el máximo número de capas y $N_i \in \{1, n\}$, donde n es el máximo número de neuronas por capa [11].

En varios trabajos investigativos se ha propuesto una manera de optimizar el diseño de la arquitectura de la red por medio del uso de algoritmos genéticos [14]. Aunque esta técnica de optimización es efectiva, puede surgir el problema de la permutación haciendo que el número de redes probables sea $n!$ y esto influye en la convergencia

del algoritmo genético [4]. El algoritmo de colonia de hormigas, supera este problema por ser diseñado para problemas combinatorios como anteriormente se había dicho [7].

3.0.3. Colonia de Hormigas

Colonia de Hormigas es una técnica meta-heuristica. La meta-heuristica consiste en una serie de conceptos de algoritmos que pueden ser usados para definir métodos heuristicos aplicables a una amplia gama de problemas diferentes, es decir estos algoritmos pueden aplicarse a diversos problemas de optimización con pocos cambios significativos si ya existe algún método heurístico que solucione el problema; de hecho la Meta-heurística es reconocida como una de las mejores aproximaciones para atacar problemas de optimización combinatoria.
Antes de explicar como funcionan las colonias de hormigas artificiales es necesario entender el comportamiento de las hormigas naturales, la cual se explica a continuación.

Las Colonias de Hormigas Naturales

Las hormigas son insectos sociales que viven en colonias, debido a su colaboración mutua, son capaces de mostrar comportamientos complejos y realizar tareas que una sola hormiga no podría realizar. Una habilidad interesante que tienen las hormigas, es la capacidad de encontrar los caminos mas cortos desde el hormiguero hasta la fuente de comida. Este hecho es muy importante si se tiene en cuenta que las hormigas son casi ciegas y no se guian por pistas visuales.

Cada hormiga cuando sale de su hormiguero a la fuente de comida y viceversa, va dejando un “rastro” llamado feromona(esta sustancia se puede “oler”).

Si las hormigas no encuentran ningún rastro de feromona las hormigas se mueven de manera aleatoria, pero si llegaran a encontrar esta sustancia las hormigas siguen este rastro. De hecho los experimentos hechos por Biologos han demostrado que las hormigas prefieren los caminos marcados con altas concentraciones de feromonas.

En la realidad las hormigas deben decidir por varios caminos que se cruzan, en este caso, ellas lo eligen de una manera probabilistica que esta sesgada por la cantidad de feromona: cuanto mas grande sea la cantidad de feromona depositada en el camino, mayor probabilidad tendrá de ser escogido. Puesto que las hormigas depositan feromona en el camino que siguen este comportamiento ayuda a reforzar la cantidad de feromona haciendola cada vez mas elevada. Gracias a este comportamiento las hormigas encuentran los caminos mas cortos desde su hormiguero, hasta la fuente de alimento [15].

En la figura 3.3 se muestra como las hormigas encuentran el camino mas corto. En un principio no hay ningún rastro de feromona en el medio y cuando una hormiga llega a una intersección de caminos elige de manera aleatoria una opción. Según transcurre el tiempo y mientras las hormigas recorren los caminos mas prometedores estos reciben una cantidad superior de feromona. Esto ocurre porque como son los caminos mas cortos, las hormigas consiguen alimento mas rapidamente, por lo que comienzan su viaje de retorno al hormiguero antes, es por esto que en el camino mas corto habrá un rastro de feromona mayor, y por lo tanto las decisiones de las demás hormigas estarán dirigidas en mayor medida a dicho camino. Este proceso finaliza, haciendo que la probabilidad de que la hormiga escoja el camino mas corto aumente paulatinamente y que al finalizar el recorrido la colonia converja al

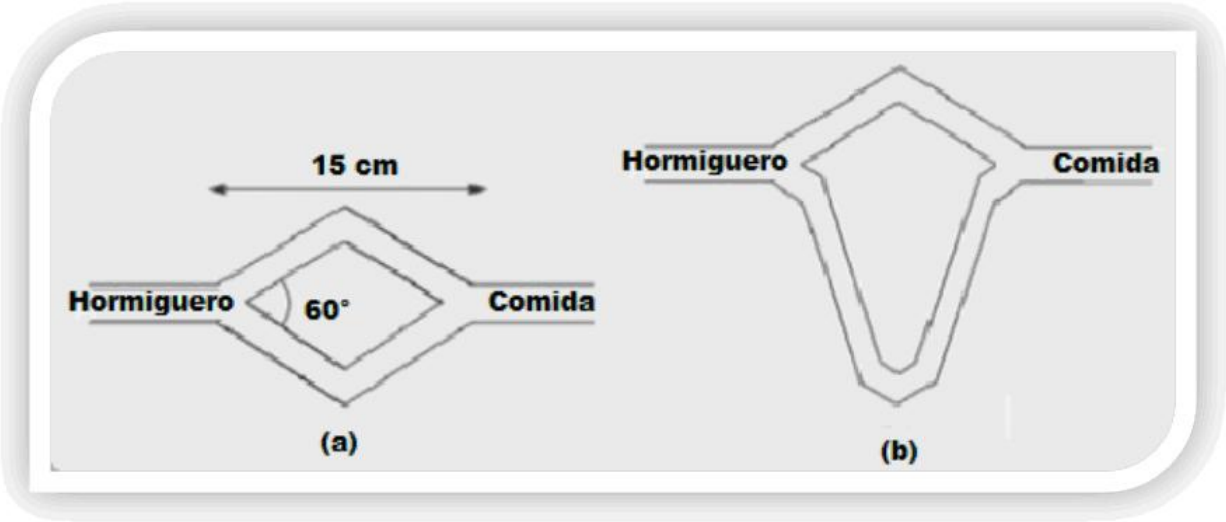


Figura 3.3 Experimento del doble puente.(a) caminos equidistantes. (b) Caminos de distinta longitud.

Fuente:Autores de Proyecto

camino mas corto de todos los caminos posibles.

Deneubourg realizó este experimento conocido como “el doble puente”, para observar el comportamiento de las hormigas y su relación con la feromona. En este experimento, una colonia de hormigas argentinas podía acceder a una fuente de comida por medio de 2 caminos, ambos de la misma longitud (Figura 3.3 (a)). El resultado fue que las hormigas iniciaban aleatoriamente siguiendo cualquier camino hasta llegar al alimento, luego el camino que tenía más rastro de feromona era más transitado. Deneubourg repitió el experimento varias veces y pudo observar que cada uno de los caminos era seleccionado por las hormigas en aproximadamente un 50 % de los casos.

En la segunda parte del experimento, uno de los caminos era más largo que el otro (figura 3.3 (b)). Al Igual que en el caso anterior, las hormigas en primera instancia selecionaban aleatoriamente el camino, pero luego de un rato la mayoria de

hormigas transitaban por el camino más corto. Esto se debia a que el rastro de feromona que habia en dicho camino era mas fuerte pues durante el mismo tiempo que demoraba una hormiga en recorrer el camino largo, (depositando feromona) se habian realizado mas recorridos en el camino corto, por tanto había mas feromona [16].

La convergencia de las hormigas se complementa con el ambiente que provoca que la feromona se evapore transcurrido cierto tiempo. Es así como los caminos mas largos pierden progresivamente feromona porque son visitados cada vez por menos hormigas.

De las Hormigas Naturales a la Optimizacion por Colonia de Hormigas

Los algoritmos de Optimizacion por Colonia de Hormigas (OCH) se inspiran indirectamente en el comportamiento de colonias de hormigas reales para solucionar problemas de optimización combinatoria. Se basan en colonias de hormigas artificiales, es decir simulan este comportamiento haciendo agentes compatacionales que trabajan de manera cooperativa y se comunican mediante rastros de feromona artificiales.

Los algoritmos de colonia de hormigas son procedimientos de búsqueda estocástica en los cuales, cada hormiga construye una solución al problema recorriendo un grafo, cada arista del grafo representa los pasos que la hormiga puede dar dependiendo de la información obtenida del ambiente que guian a la hormiga a una solucion. Esta información se divide en *Información Heurística e Información por rastro de feromona*.

- **Información Heurística:** Mide la preferencia que hay de moverse de un nodo r(posición actual) a un nodo s. La informacion heurística se denota por η_{rs}

- **Información por rastro de Feromona:** Mide la probabilidad de seguir un camino en particular. En cada camino, la información por rastro de feromona representa una relación de cuantas veces una hormiga artificial ha transitado dicho camino. Imita la feromona real que depositan las hormigas naturales. El modelo de feromona puede ser derivado del problema combinatorio, el cuál tiene un número finito de soluciones, pero ese número es tan grande que lleva un tiempo enorme llegar a la solución óptima.

El problema combinatorio p en colonia de hormigas está definido como sigue:

$$p = \langle s, \Omega, f \rangle \tag{3.8}$$

Donde s representa el espacio de busqueda, Ω las restricciones y f la función objetivo. El algoritmo básico de colonia de hormigas se describe de la siguiente manera:

Inicializa feromona: Los valores de feromona son inicializados con una constante de valor $c > 0$.

Construye solución: La construcción de la solucion comienza con una solucion parcial vacía $s^p = \langle \square \rangle$, luego en cada paso se va construyendo la solucion, añadiendo a la solucion actual s^p componentes factibles. Estos son escogidos probabilisticamente teniendo en cuenta la información heuristica η y la informacion por feromona τ.

Evaporación de la Feromona: La evaporación de la feromona la lleva a cabo el entorno y se usa como un mecanísmo que evita el estancamiento en la busqueda y permite que las hormigas exploren nuevas regiones en el espacio.

La optimización por colonia de hormigas ha hecho grandes aportes a la solución de problemas combinatorios, sin embargo existen problemas mas complejos en donde

las colonias de hormigas no llegan a una solución óptima; es por esto que se han hecho modificaciones obteniendo mejores algoritmos basados en colonia de hormigas.

Modelos de Optimización Basados en Colonia de Hormigas

Ant Colony Optimization (ACO)

El primer Algoritmo de Optimización por colonia de hormigas fue propuesto por Dorigo en 1992 en su tesis doctoral. La finalidad de este algoritmo es encontrar la distancia mas corta entre 2 nodos sobre un grafo, $G = (V, E)$ donde V son los vertices del grafo y E es la matriz que representa las conexiones entre nodos. Un ejemplo de esto se puede observar en la figura 3.4, La longitud del camino esta dado por el peso de las aristas, y la concentración de la feromona τ_{ij} asociada a cada arista del grafo.

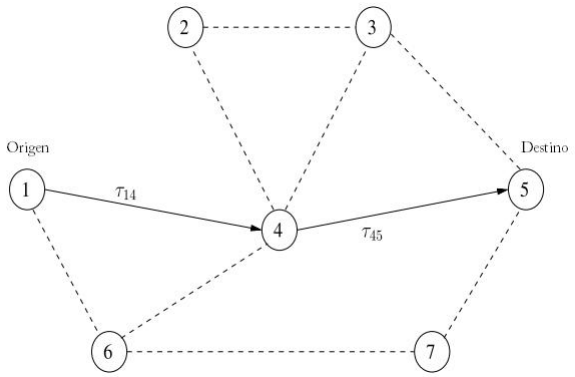


Figura 3.4 Grafo para encontrar camino mas corto.

Fuente:*Andries P. Engelbrecht*,Computational Intelligence 2007

Un número definido de hormigas, $k = 1, \ldots, n_k$ son situadas en el nodo origen. Para cada iteración de ACO, cada hormiga construye un camino(solución) para el nodo destino. En cada nodo, cada hormiga ejecuta una politica de desición para

determinar cual es el siguiente nodo a visitar. Si la hormiga k esta situada en el nodo i, esta selecciona al siguiente nodo $j \in N_i^k$ basado en la regla ge probabilidad

$$P_{ij} = \begin{cases} \frac{\tau_{ij}^{\alpha}}{\sum_{j \in N_i^k} \tau_{ij}^{\alpha}} & si \text{j} \in N_i^k \\ 0 & si \text{j} \notin N_i^k \end{cases} \tag{3.9}$$

Donde N_i^k es el conjunto de nodos faltantes por visitar conectados al nodo i, con respecto a la hormiga k. Si para algún nodo de la hormiga k $N_i^k = 0$ significa que el predecesor ya esta incluido en N_i^k, esto puede causar bucles dentro de los caminos construidos. Estos bucles son removidos una vez el nodo destino se ha alcanzado. En la Ecuación anterior, α es una constante positiva usada para ampliar la influencia de la feromona. Grandes valores de α dan una excesiva importancia a la concentración de la feromona [17].

*A*ctualización de la feromona: Este paso es requerido para evitar la rápida convergencia del algoritmo en regiones no óptimas y favorece la exploración de nuevas áreas en el espacio de búsqueda. Equivalentemente a los factores ambientales y circunstanciales, ρ es el factor de evaporación.

$$\tau_{ij} = \tau_{ij} - \rho * \tau_{ij} + \Delta\tau_{ij} \tag{3.10}$$

Donde $\Delta\tau_{ij}$ representa la suma de las contribuciones de todas las hormigas que se mueven sobre ij, dejando su rastro de feromona al construir su camino. Las contribuciones de las hormigas son proporcionales a la calidad de las soluciones logradas, es decir es mejor solución el camino que tenga más contribuciones de feromona.

Ant System (AS)

Ant System (AS) fue desarrollado por Dorigo, Maniezzo y Colorni en 1996 [18], basicamente es el mismo de ACO con la diferencia de que en este algoritmo no

sólo se tiene en cuenta la informacion de la feromona sino que también la información heurística. La regla de escogencia del nodo a visitar se hace con la siguiente ecuación de probabilidad

$$P_{ij} = \begin{cases} \frac{\tau_{ij}^{\alpha}\eta_{ij}^{\beta}}{\sum_{j \in N_i^k}\tau_{ij}^{\alpha}\eta_{ij}^{\beta}} & si \text{j} \in N_i^k \\ 0 si \text{j} \notin N_i^k \end{cases} \tag{3.11}$$

Los parámetros α y β son respectivamente la importancia que se le da a la feromona y a la información heurística. Si α=0 se le dara mas importancia a la información heurística por lo cual los caminos mas cortos tendrán preferencias de ser elegidos, si $\beta = 0$ entonces quedaría reducido este algoritmo a la importancia de la feromona lo cual lo convertiría en ACO. Lo recomendable es establecer una adecuada proporción entre la información heurística y la información de la feromona. Una diferencia entre ACO y AS es en la actualización de la feromona. En AS la feromona se evapora antes de actualizar de la siguiente manera

$$\tau_{ij} = (1 - \rho) * \tau_{ij} \tag{3.12}$$

Despues de evaporar la feromona se actualiza de la misma manera que en ACO. (ecuación 3.10).

Ant Colony System (ACS)

Este Algoritmo fue desarrollado por Dorigo y Gambardella [19], ACS usa una regla de transición distinta llamada regla de transición pseudo-aleatoria. Sea una hormiga k posicionada en el nodo r selecciona el siguiente nodo s para moverse aplicando la siguiente regla:

$$s = \begin{cases} \arg \max_{u \in J_k(r)}\{[\tau(r,u)] \cdot [\eta(r,u)]^{\beta}\}, \\ \quad \text{if } q \leq q_0 \quad \text{(exploitation)} \\ S, \quad \text{otherwise} \quad \text{(biased exploration)} \end{cases}$$

Donde q es un número ramdómico uniformemente distribuido $[0 \dots 1]$, q_0 es un parámetro $(0 \leq 1)$ y S pertenece a N_i^k que es un nodo ramdomicamente seleccionado acorde a la ecuación 3.11. N_i^k es el conjunto de nodos faltantes por visitar.

La regla de transición mostrada anteriormente, favorece la transición hacia nodos conectados por ejes cortos con una gran cantidad de feromona, el parámetro q_0 determina la importancia de la exploración contra la explotación. Si $q \leq q_0$ la hormiga escoge el camino con ejes mas cortos(explotación) si $q > q_0$ la hormiga explora y halla el siguiente nodo por medio de la ecuación(3.11). El valor de α =1.

ACS utiliza dos actualizaciones de feromona. Regla de actualización global y Regla de actualización local. En la regla de actualización Global sólo a la mejor hormiga (desde el principio del experimento) se le permite depositar feromona. Esta elección junto con la regla de transición pseudoaleatoria intenta hacer la búsqueda mas directa. La actualización global se hace cuando todas las hormigas han terminado de construir su camino.

Capítulo 4

DESARROLLO DEL PROYECTO

4.1. Metodología Utilizada

Para el desarrollo de este proyecto se utilizó la metodología de investigación - acción.
La investigación-acción realiza simultáneamente la expansión del conocimiento científico y la solución de un problema, mientras aumenta, igualmente, la competencia de sus respectivos participantes (sujetos coinvestigadores) al ser llevada a cabo en colaboración, en una situación concreta y usando la realimentación de la información en un proceso cíclico. [20]

Con dicha metodología, se va investigando y poniendo en práctica lo que se aprende, para cumplir con el objetivo de optimizar los pesos iniciales de una red neuronal artificial con una arquitectura definida, debido a que no existen métodos para seleccionar los pesos iniciales que garanticen encontrar el mínimo global, en general, dichos pesos se generan aleatoriamente, por lo que es conveniente realizar varias sesiones de entrenamiento sobre la misma red con diferentes pesos, en forma tal de comenzar el aprendizaje desde diferentes puntos de la función objetivo y así tener

más posibilidad de alcanzar el mínimo global. ACO tiene la capacidad de buscar la solución óptima global, y el algoritmo backpropagation tiene la característica de rápida convergencia de los óptimos locales. El híbrido adecuado de los dos algoritmos (ACO-BP) puede acelerar la velocidad de evolución de las redes neuronales y mejorar la precisión de la red.

Para la utilización de ACO, se tuvo que observar la forma de aplicarlo en el contexto del problema, para lo cual se hizó un estado del arte. Dentro de la investigación que se realizó, surgieron dos posibilidades de aplicar la optimización por colonia de hormigas.

La primera, aplicar ACO a la los pesos de la red neuronal, viendo ese problema como un problema combinatorio el cual es discreto, la segunda, la aplicación ACO a problemas de optmización continua, en la cual el lugar de resolver un problema combinatorio, se asume que se quiere minimizar un función continua.

Se optó por la primera opción pues se contaban con más conocimientos para su aplicación, además la segunda opción tenía muchas mas variables, muchas de esas desconocidas para las autoras.

Para cumplir con los objetivos del proyecto, el desarrollo se dividió en fases. Primero la fase de construcción de la red neuronal, luego la fase de construcción del algoritmo híbrido ACO-BP, finalmente la fase de generalización y pruebas.
Para cada una de las fases se realizó un prototipo, el cuál fué evolucionando a medida que se agregaban y modificaban funcionalidades. A continuación se mostrará mas a fondo cada una de las fases.

4.1.1. Herramienta de desarrollo

El objetivo principal de este proyecto de investigación fué la creación de un algortimo híbrido de retropropagación del error y colonia de hormigas para encontrar los pesos adecuados de una red neuronal artificial. Se tenía la opción de construir la red neuronal artificial y a esta aplicarle el algoritmo de colonia de hormigas.

Esta opción no fué tenida en cuenta, pues esto requería de un inmenso esfuerzo y tiempo de desarrollo para implementar un entorno completo de creación y entrenamiento de redes neuronales, además se quería centrar la atención en la construcción del algoritmo híbrido ACO-BP.

Para tal fin, para seleccionar la herramienta de desarrollo se tuvo en cuenta que esta contara con un toolbox especializado en el manejo de redes neuronales artificiales. La mejor opción para este propósito fué usar Matlab, pues esta herramienta cuenta con un toolbox especializado en redes neuronales el *Neural network* , el cuál permitió que la atención se centrara en la construcción del algoritmo.

Para el propósito de encontrar los mejores pesos, se creó una red neuronal con el toolbox de matlab con una arquitectura de red definida y en el ambiente de matlab se desarrolló el algoritmo de colonia de hormigas.

4.2. Construcción de la Red Neuronal Artificial

Por ser una red neuronal con aprendizaje supervisado, requiere que se ingresen pares de entradas y salidas.

La entrada a la red neuronal es una matriz que corresponde a 12 espectros de resonancias magnéticas nucleares realizadas por el ICP a 24 crudos Colombianos y 6 mezclas de crudos.(Cuadro 4.1) La salida es el valor experimental de la propiedad API para cada uno de los crudos (Cuadro 4.2).

En la figura 4.1 se observa el modelo de entradas y salidas para la red neuronal artificial.

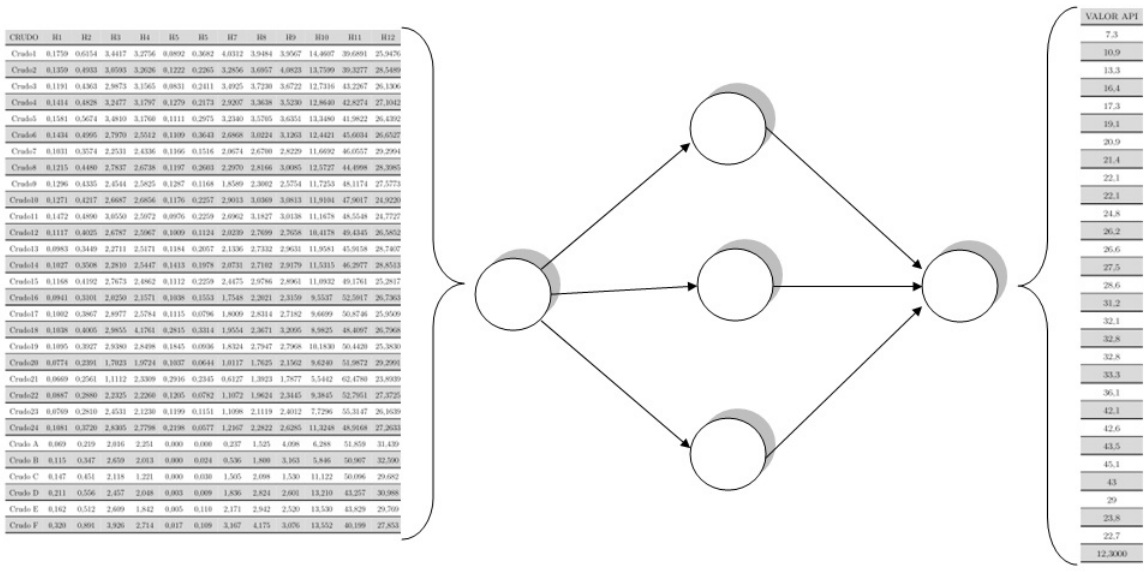


Figura 4.1 Modelo de entradas y salidas de la red

Fuente: Autores del proyecto

CRUDO	H1	H2	H3	H4	H5	H5	H7	H8	H9	H10	H11	H12
Crudo1	0,1759	0,6154	3,4417	3,2756	0,0892	0,3682	4,0312	3,9484	3,9567	14,4607	39,6891	25,9476
Crudo2	0,1359	0,4933	3,0593	3,2626	0,1222	0,2265	3,2856	3,6957	4,0823	13,7599	39,3277	28,5489
Crudo3	0,1191	0,4363	2,9873	3,1565	0,0831	0,2411	3,4925	3,7230	3,6722	12,7316	43,2267	26,1306
Crudo4	0,1414	0,4828	3,2477	3,1797	0,1279	0,2173	2,9207	3,3638	3,5230	12,8640	42,8274	27,1042
Crudo5	0,1581	0,5674	3,4810	3,1760	0,1111	0,2975	3,2340	3,5705	3,6351	13,3480	41,9822	26,4392
Crudo6	0,1434	0,4995	2,7970	2,5512	0,1109	0,3643	2,6868	3,0224	3,1263	12,4421	45,6034	26,6527
Crudo7	0,1031	0,3574	2,2531	2,4336	0,1166	0,1516	2,0674	2,6700	2,8229	11,6692	46,0557	29,2994
Crudo8	0,1215	0,4480	2,7837	2,6738	0,1197	0,2603	2,2970	2,8166	3,0085	12,5727	44,4998	28,3985
Crudo9	0,1296	0,4335	2,4544	2,5825	0,1287	0,1168	1,8589	2,3002	2,5754	11,7253	48,1174	27,5773
Crudo10	0,1271	0,4217	2,6687	2,6856	0,1176	0,2257	2,9013	3,0369	3,0813	11,9104	47,9017	24,9220
Crudo11	0,1472	0,4890	3,0550	2,5972	0,0976	0,2259	2,6962	3,1827	3,0138	11,1678	48,5548	24,7727
Crudo12	0,1117	0,4025	2,6787	2,5967	0,1009	0,1124	2,0239	2,7699	2,7658	10,4178	49,4345	26,5852
Crudo13	0,0983	0,3449	2,2711	2,5171	0,1184	0,2057	2,1336	2,7332	2,9631	11,9581	45,9158	28,7407
Crudo14	0,1027	0,3508	2,2810	2,5447	0,1413	0,1978	2,0731	2,7102	2,9179	11,5315	46,2977	28,8513
Crudo15	0,1168	0,4192	2,7673	2,4862	0,1112	0,2259	2,4475	2,9786	2,8961	11,0932	49,1761	25,2817
Crudo16	0,0941	0,3101	2,0250	2,1571	0,1038	0,1553	1,7548	2,2021	2,3159	9,5537	52,5917	26,7363
Crudo17	0,1002	0,3867	2,8977	2,5784	0,1115	0,0796	1,8009	2,8314	2,7182	9,6699	50,8746	25,9509
Crudo18	0,1038	0,4005	2,9855	4,1761	0,2815	0,3314	1,9554	2,3671	3,2095	8,9825	48,4097	26,7968
Crudo19	0,1095	0,3927	2,9380	2,8498	0,1845	0,0936	1,8324	2,7947	2,7968	10,1830	50,4420	25,3830
Crudo20	0,0774	0,2391	1,7023	1,9724	0,1037	0,0644	1,0117	1,7625	2,1562	9,6240	51,9872	29,2991
Crudo21	0,0669	0,2561	1,1112	2,3309	0,2916	0,2345	0,6127	1,3923	1,7877	5,5442	62,4780	23,8939
Crudo22	0,0887	0,2880	2,2325	2,2260	0,1205	0,0782	1,1072	1,9624	2,3445	9,3845	52,7951	27,3725
Crudo23	0,0769	0,2810	2,4531	2,1230	0,1199	0,1151	1,1098	2,1119	2,4012	7,7296	55,3147	26,1639
Crudo24	0,1081	0,3720	2,8305	2,7798	0,2198	0,0577	1,2167	2,2822	2,6285	11,3248	48,9168	27,2633
Crudo A	0,069	0,219	2,016	2,251	0,000	0,000	0,237	1,525	4,098	6,288	51,859	31,439
Crudo B	0,115	0,347	2,659	2,013	0,000	0,024	0,536	1,800	3,163	5,846	50,907	32,590
Crudo C	0,147	0,451	2,118	1,221	0,000	0,030	1,505	2,098	1,530	11,122	50,096	29,682
Crudo D	0,211	0,556	2,457	2,048	0,003	0,009	1,836	2,824	2,601	13,210	43,257	30,988
Crudo E	0,162	0,512	2,609	1,842	0,005	0,110	2,171	2,942	2,520	13,530	43,829	29,769
Crudo F	0,320	0,891	3,926	2,714	0,017	0,109	3,167	4,175	3,076	13,552	40,199	27,853

Cuadro 4.1 Datos de Resonancia Magnética Nuclear

7,3	10,9	13,3	16,4	17,3	19,1	20,9	21,4	22,1	22,1	24,8	26,2	26,6	27,5	28,6	31,2	32,1	32,8	32,8	33,3	36,1	42,1	42,6	43,5	45,1	43	29	23,8	22,7	12,3

Cuadro 4.2 Valores de API para cada uno de los crudos y mezclas

Para la creación de la red neuronal artificial, en un principio se pensó en un perceptrón sencillo debido al desconocimiento del problema.

Se relizó una prueba con los pares de datos, obteniendo unos resultados malos (Figura: 4.2).

A partir de este primer prototipo se tuvo un conocimiento mas amplio del problema y se comprendió que el problema no es linealmente separable, razón por la cuál un perceptrón no podía aportar mucho a la solución.

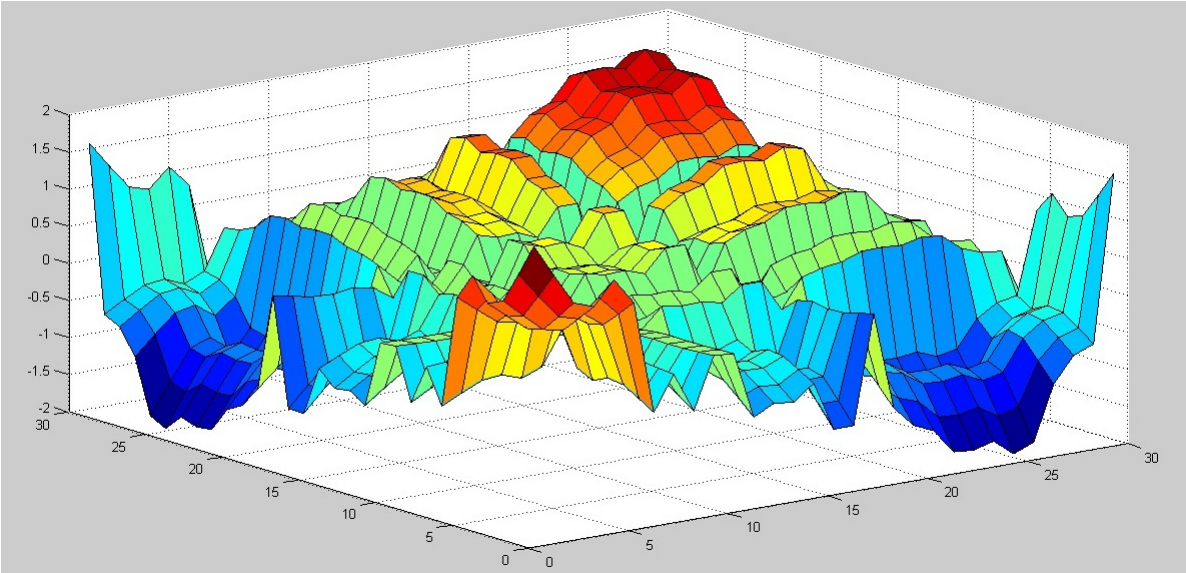


Figura 4.2 Superficie del error obtenido utilizando un perceptrón

Fuente: Autores del proyecto

Al indagar e investigar mas acerca del problema, como segundo prototipo de red se diseñó un perceptrón multicapa.
La arquitectura de red seleccionada fué de dos capas. Tres neuronas en la capa oculta con función de transferencia tansigmoidal[1]. y una neurona en la capa de salida con función de transferencia lineal [2].

El criterio de selección de esta arquitectura se basó en el teorema de Cybenko, donde también demostró que el perceptrón multicapa con una sola capa oculta, en la cual cada nodo tendría como función de transferencia una función sigmoidea y una unidad de salida con función de transferencia lineal, es un clasificador universal, es decir que puede aproximar con la precisión deseada cualquier función [21].

Al hacer pruebas con este prototipo se tuvo un acercamiento a la solución, sin embargo, todavía no se tenía el resultado esperado. La razón de este comportamiento era la falta de ejemplos a la red neuronal y los malos pesos iniciales.
Como la cantidad de datos para el entrenamiento era insificiente, se procedió a crear mas datos agregandole ruido a los existentes y con estos datos entrenar la red.
Con este cambio mejoraron los resultados arrojados por la red neuronal.(Figura: 4.2).

Al obtener un prototipo aceptable de arquitectura de red neuronal, se prosiguió a implementar el algoritmo ACO-BP, teniendo en cuenta que se deían seleccionar de la mejor manera los parámetros del algoritmo para alcanzar un buen resultado.

[1] $f(x) = \frac{1}{1+e^{-2n}} - 1$
[2] $f(x) = x$

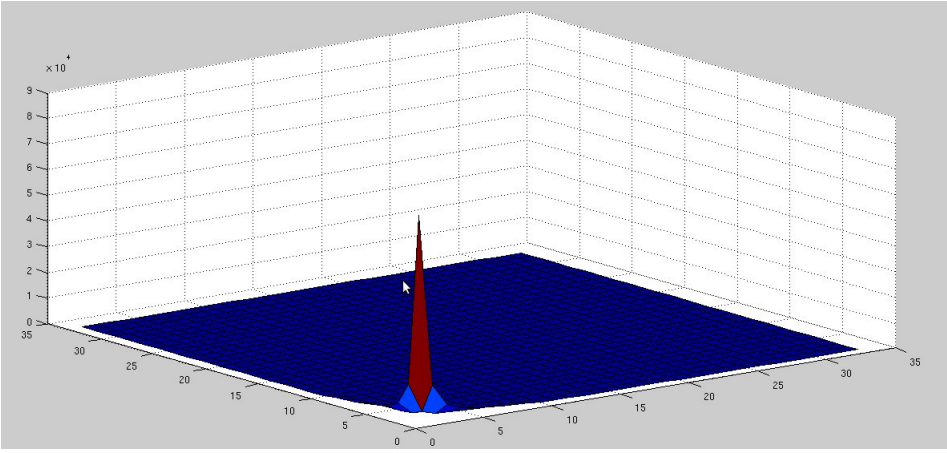


Figura 4.3 Superficie del error obtenido utilizando un perceptrón multicapa

Fuente: Autores del proyecto

4.3. Híbrido entre colonia de hormigas y backpropagation

De la misma forma que en el problema del agente viajero TSP [3], en el cuál se consigue una ruta con el mínimo costo, para el caso de la red neuronal, la hormiga debe seleccionar para cada conexión, el peso que mejor se ajuste, es decir que arroje el mínimo error.

Como la red neuronal utilizada tiene aprendizaje supervisado, se dispone de un conjunto de aprendizaje con p patrones, el cual tiene salidas dadas por:

$$tp = tp_1, tp_2, tp_3, tp_4 \ldots tp_m$$

,siendo m la última neurona de salida.

El error cuadrático para un patrón determinado tiene la siguiente expresión (Ec.4.2) Donde tp es la salida esperada de la red correspondientes al patron p y yp es la salida obtenida por la red.

Dada la arquitectura de la red, deben hallarse los pesos iniciales para que esta sea entrenada. El criterio de selección de esta arquitectura se basó en el teorema de Cybenko como se dijo anteriormente.

Para lograr hallar los mejores pesos iniciales, la hormiga debe encontrar la manera de combinar estos pesos para obtener el mínimo error.

Con el fin de aplicar ACO (Procedimiento.1), el espacio de búsqueda de solución debe volverse discreto.

[3]TSP (travel salesman problem): Sean N ciudades de un territorio. El objetivo es encontrar una ruta que, comenzando y terminando en una ciudad concreta, pase una sola vez por cada una de las ciudades y minimice la distancia recorrida por el viajante.

Para conseguirlo, para cada una de las conexiones,$w_1, w_2, ..., w_n$, se crea una tabla (Cuadro 4.3) con *m*+1 parámetros, correspondientes a los pesos en un intervalo $[W_{max}W_{min}]$, en donde cada uno de los parámetros es un punto o posible peso para esa conexión y tiene asociado un valor de probabilidad $P(i)$ y feromona $\tau(i)$, los cuales, son inversamente proporcionales al error que produce al ser implementado en la red (Figura 4.4).

Una vez construidas las tablas, la hormiga selecciona por probabilidad (Ec. 4.1), uno de los parámetros de la tabla para cada conexión y construye un tour, el cuál contiene todos los valores de los pesos de la red. Los valores de los pesos seleccionados por la hormiga son implementados en la red y posteriormente se calcula el error producido (Ec. 4.2), el cuál es utilizado para la actualización de la feromona (Ec. 4.3).

	w_i	
Feromona	$\rho_1, \rho_2, ..., \rho_{m+1}$	[0,1]
Probabilidad	$p_i, p_2, ..., p_{m+1}$	[0,1]
Peso	$w_1, w_2, ..., w_{m+1}$	[-2,2]

Cuadro 4.3 Tabla de feromona para cada conexión

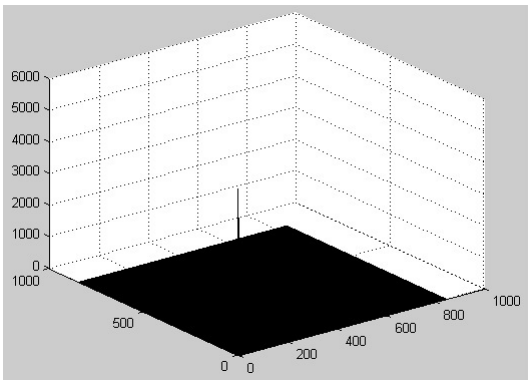


Figura 4.4 Superficie del error obtenido utilizando ACO-BP

Fuente: Autores del proyecto

Procedimiento 1 ACO

Entrada: $max.ite, \rho, p$

Salida: $pesos$

mientras $ite = max - ite$ **hacer**

1. Inicializa

2. Tour

para $hormiga = 1$ **to** $num.hormigas$ **hacer**

para $i = 1$ **to** $num.conexiones$ **hacer**

Selecciona un peso de acuerdo a la probabilidad:

$$P(i) = \frac{\tau(i)}{\sum \tau(i)} \tag{4.1}$$

$tour(i) \leftarrow peso$

fin para

3. Simula *Asigna pesos seleccionados por la hormiga y produce salida* yp

4. Error *con la salida* yp *y los patrones* tp *calcula:*

$$E = \frac{1}{2} \sum (tp - yp)^2 \tag{4.2}$$

5. Actualiza *Siendo* $\rho \in (0, 1)$ *es el factor de decaimiento de la feromona.*

$$\tau(i+1) = \rho * \tau(i) + \triangle\tau(i) \tag{4.3a}$$

$$\triangle \tau(i) = \begin{cases} \frac{1}{E} & \textit{si} \quad \tau(i) \in tour \\ \\ 0 & \textit{si} \quad \tau(i) \notin tour \end{cases} \tag{4.3b}$$

fin para

fin mientras

Diagrama de flujo del algoritmo híbrido ACO-BP

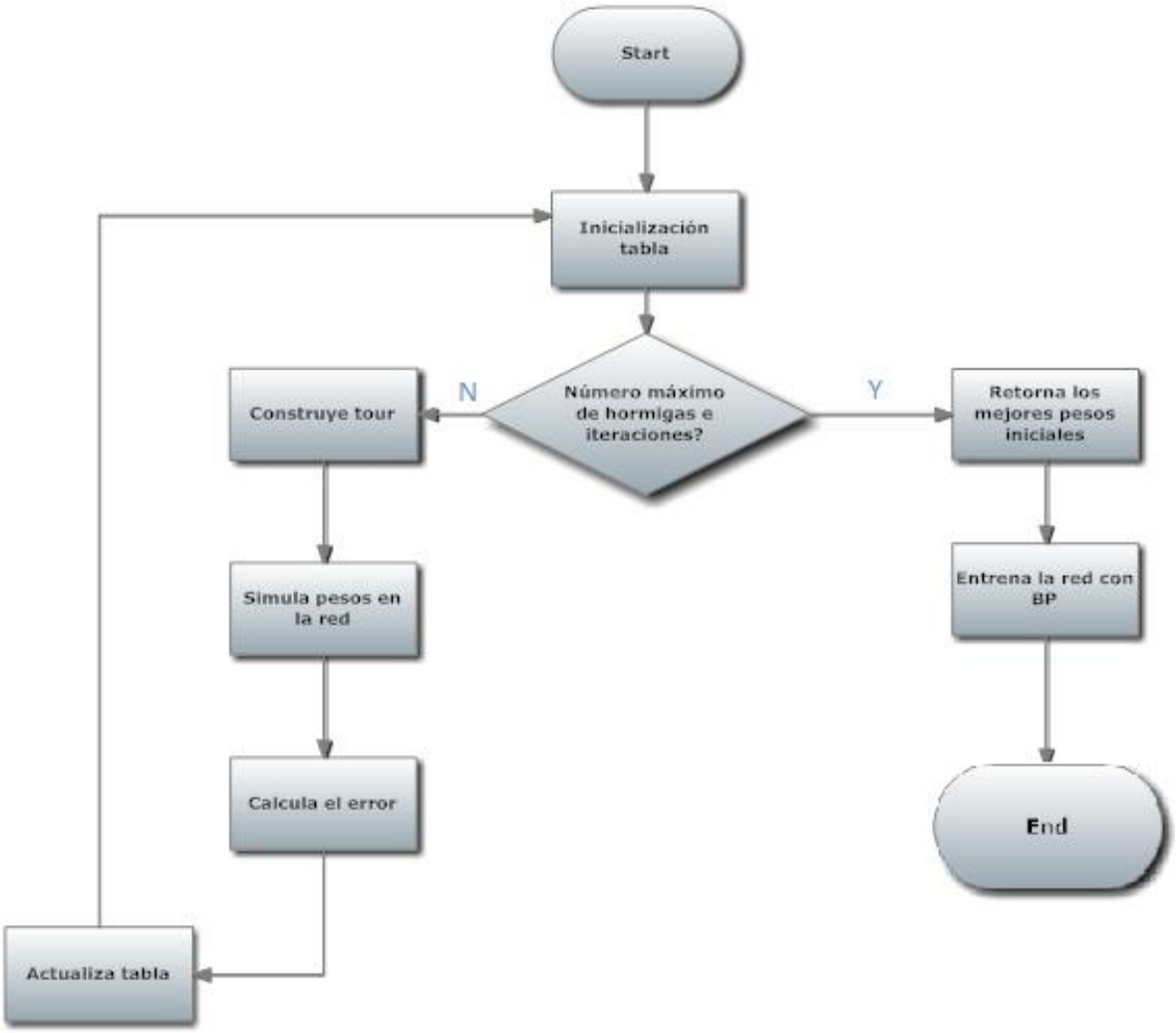


Figura 4.5 Diagrama de flujo del algoritmo híbrido ACO-BP

Fuente: Autores del proyecto

4.4. Generalización de la red neuronal artificial

En una RN entrenada, si las salidas calculadas por la red con nuevos ejemplos están próximas a los valores deseados, hay generalización [12].
Hay que tener en cuenta en este punto que la RNA requiere alcanzar un grado de generalización, por lo tanto no debe aprender de memoria los casos, sino que teniendo algunos ejemplos de cada uno, sea capaz de determinar correctamente otro que no haya visto antes en base a los que le fueron mostrados durante su entrenamiento.

Entendiendo por generalización la habilidad de una red neuronal de almacenar en sus pesos sinápticos características que le son comunes a todos los patrones de entrenamiento que fueron usados durante la fase de entrenamiento.
El número de casos utilizados para el entrenamiento deben ser suficientes y además representativos del conjunto de casos que se desea generalizar.

En nuestro caso, se tenia la resonancia magnética nuclear de 30 crudos, con estos datos se debia entrenar, validar y probar la red neuronal. La cantidad de datos era insuficiente, razón por la cual se ve la necesidad de agregar ruido[4] y de esta manera aumentar la cantidad de datos. Para la validación o generalización se utilizaron 10 crudos que no se habían presentado en la etapa de entrenamiento.

Si una red no tiene suficientes conexiones entre nodos, el algoritmo de entrenamiento puede no converger; la red neuronal no es capaz de aproximar la función. Por el otro lado, en una red densamente conectada, puede ocurrir el sobre-ajuste (overfitting). El sobre-ajuste es un problema de los modelos estadísticos donde se

[4]se generó un número aleatorio y este se dividió en 100, a cada crudo se le sumo un número diferente

presentan demasiados parámetros. Esto es desfavorable porque en lugar de aprender a aproximar la función presente en los datos, la red simplemente puede memorizar cada ejemplo de entrenamiento.

Para que la red tenga una buena capacidad de generalización, es necesario que las conexiones y los pesos sinápticos sean los adecuados, para lo cuál el algortimo ACO debe presentar los mejores pesos iniciales. Para esto, es indispensable que los parámetros del algoritmo sean elegidos de la mejor forma.

4.4.1. Selección de parámetros

Los parámetros para la construcción de ACO son el número de hormigas, número de elementos de la tabla y máximo número de iteraciones. Para desarrollar el algoritmo, es necesario que los parámetros sean escogidos de la mejor manera, pues estos influyen en gran medida en su rendimiento.

Para este fin, se realizaron algunas pruebas de sensibilidad, verificando que el rendimiento de la red y la estimación que esta arroja dieran un buen resultado.

4.4.2. Prueba de sensibilidad para seleccionar el Número de hormigas

Dentro del algoritmo de colonia de hormigas, el número de hormigas influye en proporción al rastro de feromona. Cuanto más hormigas pasen por cierto camino, se vuelve mas probable que dicho camino se convierta en la solución seleccionada porque tiene un rastro de feromona mayor, además existe mayor exploración del espacio de soluciones.

Otro factor que afecta la feromona es el valor de ρ, el factor de decaimiento de la feromona, el cual debe estar entre $(0, 1)$. Este factor permite que las hormigas olviden soluciones viejas y exploren nuevas soluciones. El valor seleccionado para ρ

fué 0.9.

Se debe tener en cuenta igualmente como influye el número de hormigas con el tiempo de cómputo. A mayor número de hormigas, el tiempo de cómputo se incrementa para aplicaciones no paralelas.

Para la realización de las pruebas de sensibilidad para seleccionar el número de hormigas, se utilizó un equipo con Ubuntu 10.04 con memoria de 2.9 GB y 2.2Ghz, procesador Intel Dual-core, con la versión de Matlab 7.8 R2009a .

Se realizaron 10 simulaciones para cada número de hormigas (Figuras: 4.6 - 4.7), luego se calculó el promedio de los resultados.

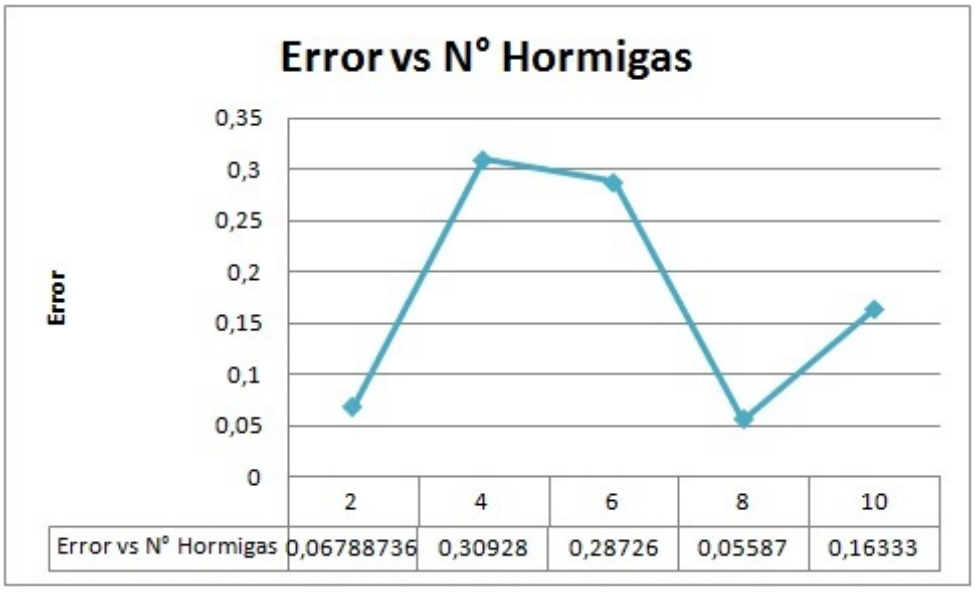


Figura 4.6 Prueba de sensibilidad para seleccionar el número de hormigas

Fuente: Autores del proyecto

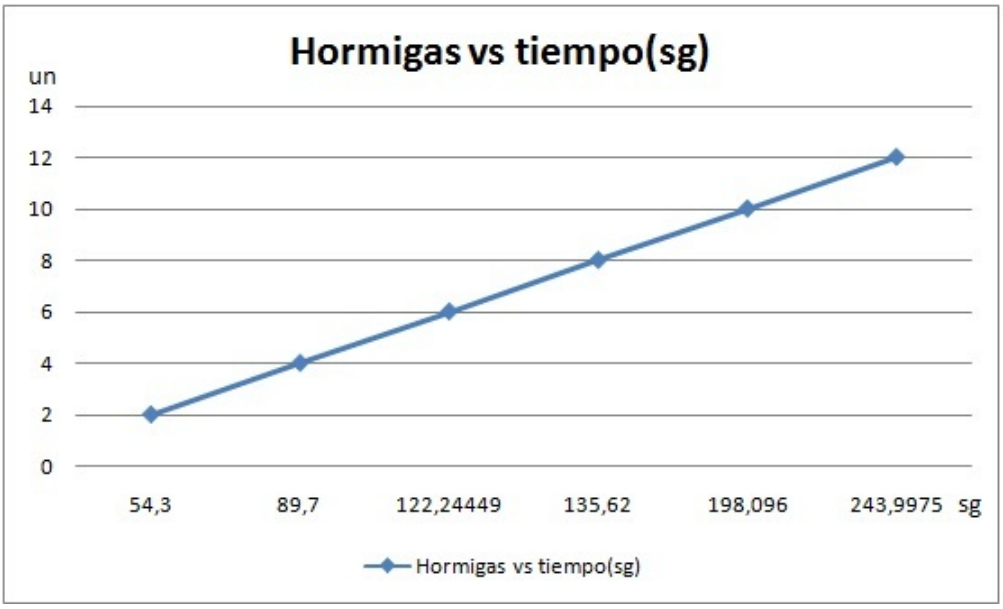


Figura 4.7 Tiempo de simulación con distintos números de hormigas

Fuente: Autores del proyecto

De acuerdo a las pruebas realizadas se seleccionaron 8 hormigas, pues arrojaba el error mas bajo y no se sacrificaba mucho tiempo de cómputo.

4.4.3. Prueba de sensibilidad para seleccionar el Número de elementos de la tabla

Dentro de la tabla, se deben tener en cuenta el rango de pesos W_{max}-W_{min} y el número de elementos de la tabla.

El rango de pesos debe incluir soluciones potenciales de pesos y su intervalo debe ser el adecuado. Si el intervalo es muy pequeño, el espacio de búsqueda se reduce limitando la búsqueda y perdiendo soluciones potenciales. Si el intervalo es muy grande, el espacio de búsqueda es amplio, pero al tener tantas opciones el algoritmo pierde precisión dificultando llegar a la solución óptima y aumentando el tiempo de cómputo.

Teniendo en cuenta lo anterior, se tomó un intervalo de W_{min}=-2 y W_{max}=2. Estos

valores fueron tomados siguiendo la sugerencia de Liu y Wu [8].

Para seleccionar el número de elementos de la tabla se requería que la aproximación que arrojara el algoritmo y el tiempo empleado para ello fueran adecuados, así que se realizaron 10 simulaciones para cada número de p parámetros de la tabla (Figuras: 4.8 - 4.9), luego se sacó el promedio de los resultados.

Para la realización de esta prueba,se utilizó un equipo con Windows 7 con memoria de 2.9 GB y 2.2 Ghz, procesador Intel Dualcore, con la versión de Matlab 7.8 m R2009a .

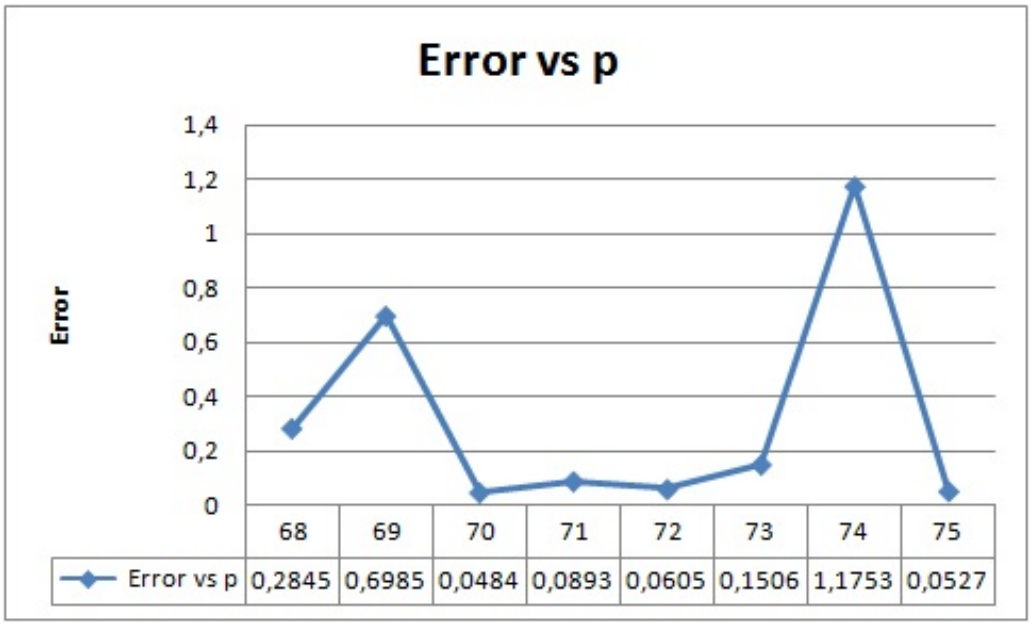


Figura 4.8 Prueba de sensibilidad para seleccionar el número de elementos de la tabla

Fuente: Autores del proyecto

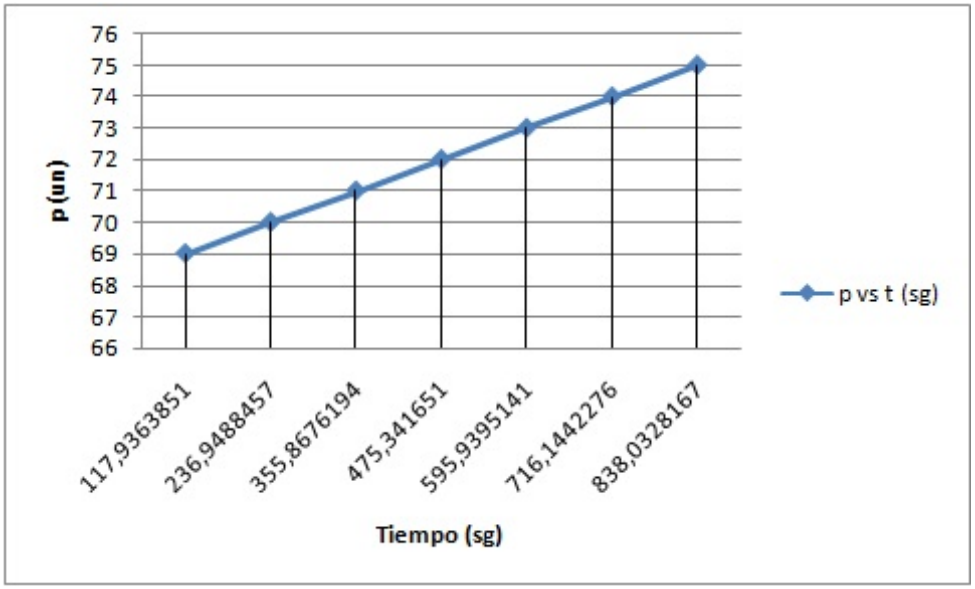


Figura 4.9 Tiempo de simulación para diferentes números de elementos de la tabla

Fuente: Autores del proyecto

Teniendo en cuenta los resultados arrojados por las pruebas, se tomo un número de parámetros p 75.

Capítulo 5

RESULTADOS

La red neuronal artificial, fué creada con ayuda del toolbox de redes neuronales *Neural network* de Matlab, al cual se incorporó el algoritmo de hormigas con todas sus funciones.

Con el fin de probar el algoritmo desarrollado, se realizaron pruebas al algoritmo con los datos de resonancia magnética nuclear suministrados por el ICP [1].

Estos datos, fueron insuficientes para el entrenamiento de la red neuronal, razón por la cual se agregó ruido[2], obteniendo así datos suficientes para el entrenamiento de la red.

Para su consecución, se utilizaron un equipo con Windows XP con memoria de 1.93 GB y 2.91Ghz, procesador AMD Athlom de 64 dual core, con la versión de Matlab 7.8 R2009a y un equipo un equipo con Windows XP con memoria de 3.23 GB y 2.33Ghz, procesador Intel Quad, con la versión de Matlab 7.9 R2009b.

Ambos algoritmos fueron simulados 50 veces. Estas pruebas se hicieron del toolbox

[1] Instituto Colombiano de Petróleo

[2] se generó un número aleatorio para cada resonancia, luego este número fue sumado a todo el espectro

de redes neuronales de matlab y del algoritmo de colonia de hormigas(ACO-BP) implementado. Para el algortimo de hormigas se utilizaron 8 hormigas, p=70 y ro= 0.9. Tanto la red del algoritmo de hormigas como la del toolbox trabajaron con los mismos parámetros: 1000 epocas, tasa de aprendizaje de 0.005 y los mismos ejemplos de entrenamiento(70 %), validación(15 %) y prueba(15 %).

Las graficas son del $Emse$ vs $performance$.
$Emse$ es el error obtenido después de presentarle a la red los ejemplos de entrenamiento y el performance es el rendimiento que la red obtuvo, es decir el error al que llegó después de ajustar sus pesos. Se hicieron pruebas, la primera es la red neuronal que matlab ofrece en su toolbox, los resultados se pueden ver en la grafica. Como se puede ver el $Emse$ que matlab obtiene al iniciar el entrenamiento es de 448.81 con un rendimiento de 0.2557 en promedio.

La segunda grafica muestra el resultado del algortimo de colonia de hormigas, en donde se utilizaron 8 hormigas, p=75, y ρ=0.9 obteniendo un Emse de 99.44 y un rendimiento de 0.0579 en promedio. El algoritmo de colonia de hormigas obtiene Emse mas pequeño ya que la hormiga permite encontrar los mejores pesos que hacen que este error disminuya, esto facilita y ayuda a encontrar el minimo global y no permite que el algoritmo se quede atrapado en un mínimo local. La tercera grafica es una comparación entre hormigas y el toolbox de matlab, en ella se ve que la hormiga obtiene un Emse más pequeño y un mejor rendimiento que el toolbox de matlab.

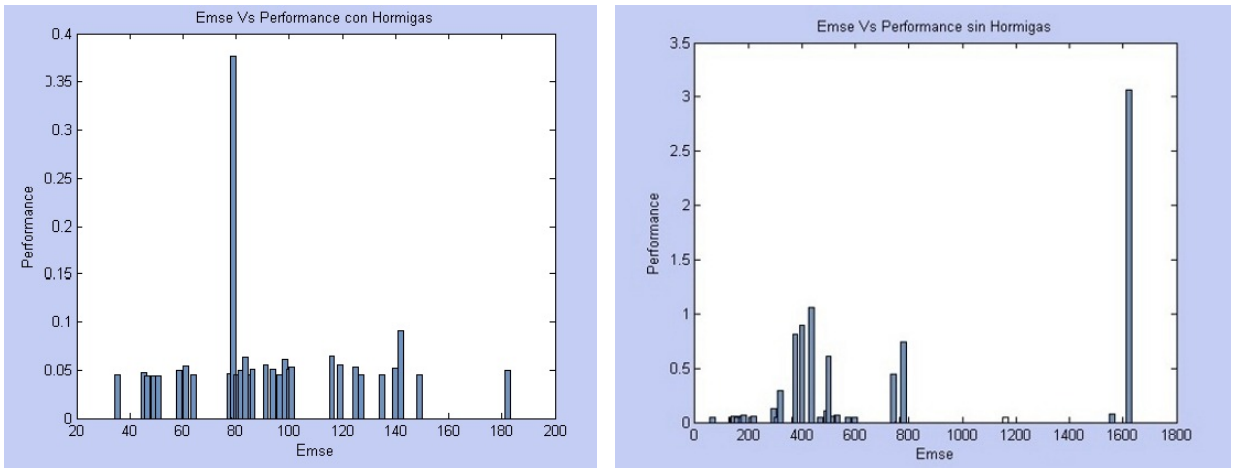


Figura 5.1 Comparación Error vs peformance con hormigas sin hormigas

Fuente: Autores del proyecto

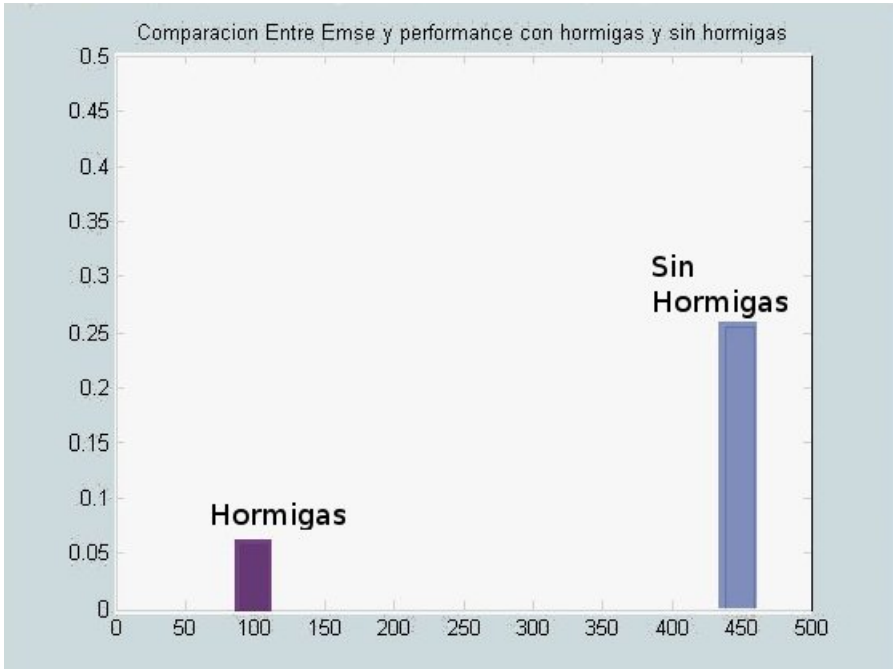


Figura 5.2 Comparación global del Error vs peformance con hormigas sin hormigas

Fuente: Autores del proyecto

Las tablas mostradas son los resultados obtenidos después de la simulación del ACO-BP, asi como el del BP.

Para las simulaciones se utilizaron 10 crudos que no fueron mostrados durante el entrenamiento.

En la primera columna se tienen los crudos usados para la simulación. Columnas 2-3 muestran los datos experimentales proporcionados por el ICP y los resultados arrojados por la red, en la cuarta columna el error entre los datos experimentales y los resultados de la RNA. El error se calculó de la siguiente manera

$$error = (t_p - y_p)$$

t_p: datos experimentales

y_p: resultado de la red

Otro error calculado es el error cuadrático medio que muestra el rendimiento de los algoritmos(tomado de Liu y Wu [8]) se calcula asi

$$Emse = \frac{1}{2}\sum_{i=1}^{n}(tp_i - yp_i)^2$$

Donde n es número total de datos usados durante el entrenamiento.

Crudos	Datos Experimentales	Resultados arrojados por la red	error
1	36.1	36.3974	0.29743
2	42.1	42.3788	0.27875
3	42.6	42.8965	0.29650
4	43.5	43.6444	0.14441
5	45.1	45.0813	0.01868
6	43.0	43.0958	0.09575
7	29.0	29.6766	0.67658
8	23.8	24.0299	0.22989
9	22.7	22.8348	0.13482
10	12.3	12.2950	0.00498

Cuadro 5.1 Resultados obtenidos por BP

Error cuadrático = 0.0271101583333333

Crudos	Datos Experimentales	Resultados arrojados por la red	error
1	36.1	36.0989	0.001099
2	42.1	42.0987	0.001264
3	42.6	42.5984	0.001621
4	43.5	43.4993	0.000743
5	45.1	45.1003	0.000289
6	43.0	43.0031	0.003127
7	29.0	28.9987	0.001347
8	23.8	23.7994	0.000536
9	22.7	22.7072	0.007152
10	12.3	12.3790	0.079032

Cuadro 5.2 Resultados obtenidos por ACO-BP

Error cuadrático: 0.000210508121790253

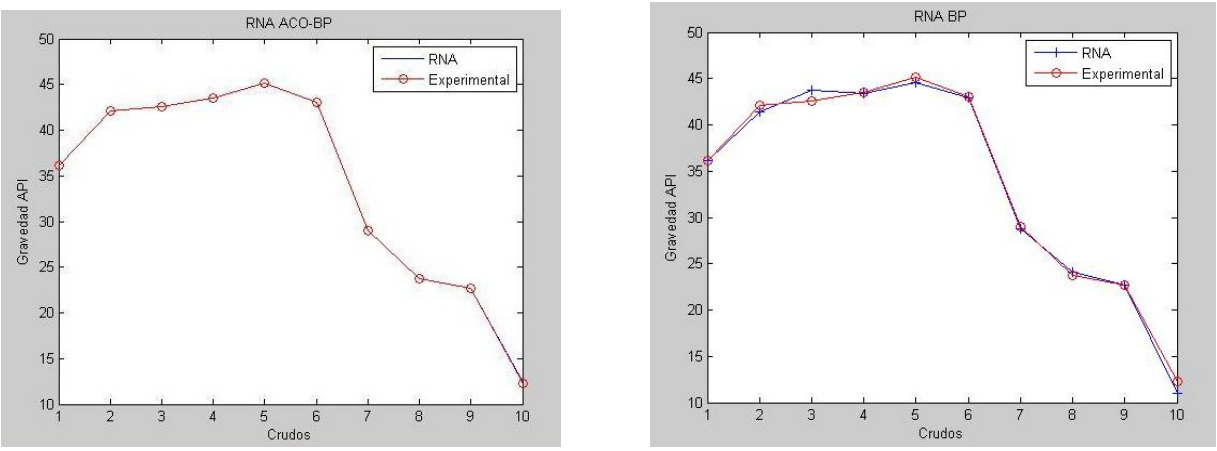


Figura 5.3 Curvas obtenidas de ACO-BP Y BP

Fuente: Autores del proyecto

Se puede observar que el algoritmo de colonia de hormigas obtiene Emse mas pequeño y un mejor rendimiento que el toolbox de matlab, ya que la hormiga permite encontrar los mejores pesos que hacen que este error disminuya, esto facilita y ayuda a encontrar el mínimo global y no permite que el algortimo quede atrapado en un mínimo local.

Para la validación del algoritmo se tomaron 10 crudos, los cules no fueron presentados a la red, con el fin de comprobar la capacidad de generalización.
Los resultados arrojados (5.1)mostraron que el algoritmo ACO-BP tiene una capacidad superior de generalización que el BP.

Resultados obtenidos simulación ACO-BP Vs BP, hecho en el toolbox de matlab. Las figuras muestran el comportamiento de cada uno de los algoritmos, desde su entrenamiento hasta la validación y entrega de resultados.

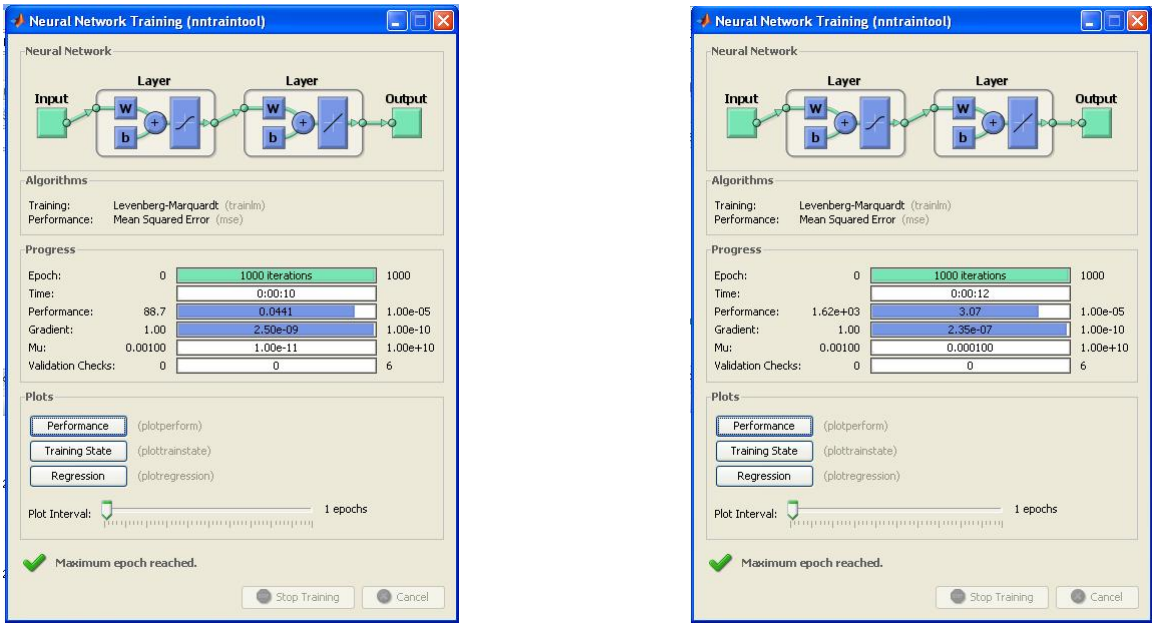


Figura 5.4 Asistente Toolbox de matlab para el entrenamiento.

A la izquierda entrenamiento con el error de los pesos iniciales arrojado por ACO. A la derecha el error arrojado por BP

Fuente: Autores del proyecto

Esta es la ventana mostrada por matlab al iniciar el entrenamiento. En ella se muestra el algoritmo de entrenamiento y el error de entrada para reducirlo. En el caso de ACO se puede observar que el error de entrada es 88.7 en comparación con el error de entrada de BP 1,62e03. La explicación de esta diferencia es la elección de los pesos iniciales, que para el primiero fueron seleccionados por las hormigas y en el segundo elegidos aleatoriamente.

El aisitente tiene además unos campos llamados épocas, gradiente y validation checks que son los criterios de parada. Cuando la red alcanza alguno de ellos significa que el entrenamiento ha terminado. En este caso, el entrenamiento terminó cuando el número de épocas fue alcanzado.

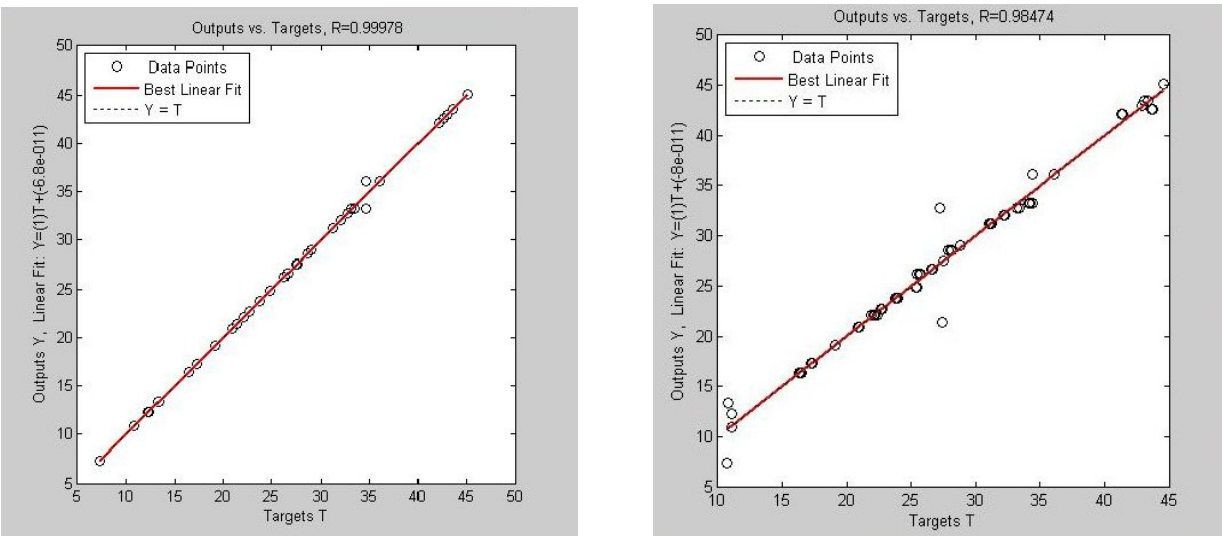


Figura 5.5 Entrenamiento de la red neuronal con el toolbox de matlab.

A la derecha ACO-BP, izquierda BP

Fuente: Autores del proyecto

La gráfica 5.5 pertenece al entrenamiento de la red y corresponde al 70 % de los datos presentados. En ambos casos es similar ya que el entrenamiento se hizo con el mismo número de datos y el mismo algortimo de entrenamiento (Levenberg Marquard).

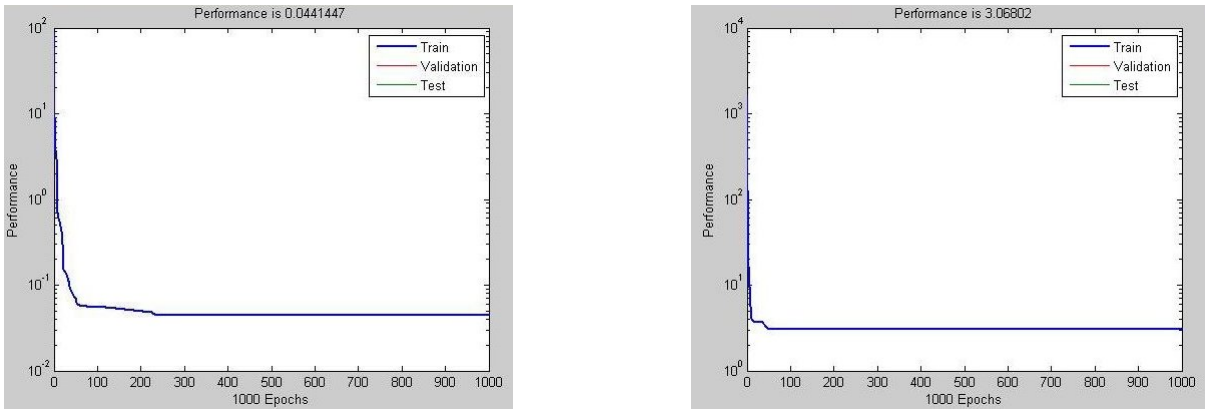
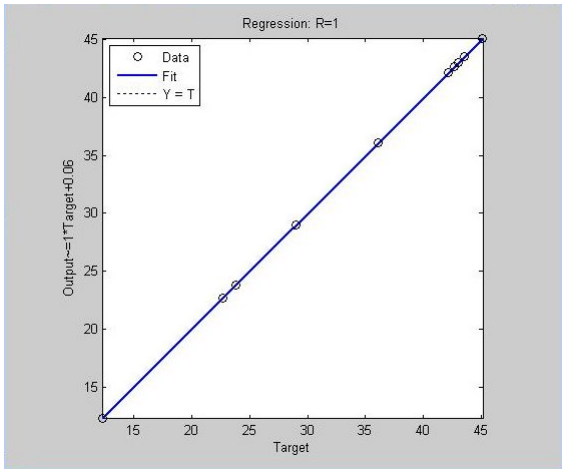


Figura 5.6 Error alcanzado por la red después de finalizadas las 1000 épocas

A la izquierda error alcanzado en el entrenamiento utilizando los pesos iniciales hallados por ACO-BP. A la derecha el error arrojado por BP

Fuente: Autores del proyecto

En la gráfica 5.6 se puede observar como evoluciona la red, en la grafica de BP se ha quedado atrapado en un mínimo local, ya que su performance no llego a un error aceptable mientras que en ACO-BP este llego hasta el mínimo global. Esto se vé reflejado en los valores de error alcanzado, el cual para ACO fué de 0,0441447 y para BP 3.06802



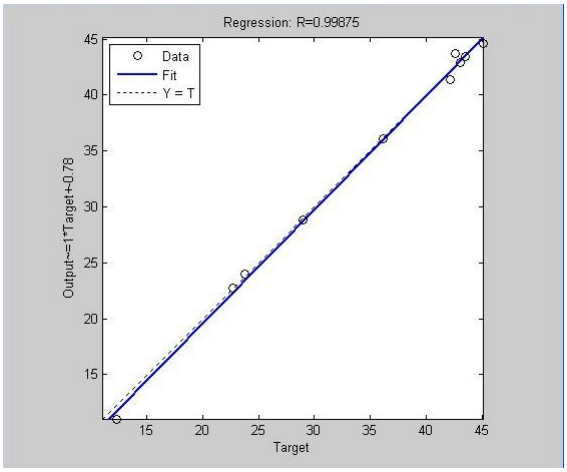


Figura 5.7 Regresión alcanzada por la red

Izquierda ACO-BP BP

Fuente: Autores del proyecto

Regresión hecha a los resultados obtenidos por la red, y muestra la correlación que hay entre los datos obtenidos por la red y los datos experimentales, la línea punteada es la regresión de los datos experimentales, la línea regresión datos RNA, los círculos son los datos de salida (targets) a los cuales se les hizo regresión. En este caso los datos de la red ACO-BP están linealmente correlacionados, eso significa que la red convergió y se obtuvieron los resultados esperados con una alta confiabilidad.

Capítulo 6

CONCLUSIONES

- Mediante las Redes neuronales y la optimización de sus pesos iniciales por medio de ACO se pudo realizar el modelo que se ajustaba a los datos de la resonancia magnetica nuclear y asi realizar el análisis de estos datos para inferir la propiedad Gravedad API.

- A pesar de que el algoritmo Backpropagation (BP) ha demostrado ser un buen método de entrenamiento de una red neuronal, algunas veces se queda atrapado en mínimo local debido aleatoriedad de los pesos iniciales.
  En el presente trabajo, se muestra un método para superar este problema basado en la optimización de los pesos iniciales con Ant Colony Optimization (ACO).

- Los resultados muestran que el híbrido de los dos algoritmos (ACO-BP), disminuyen los errores de aproximación de la red. ACO encuentra en el espacio de soluciones un punto cercano al mínimo global y BP refina la solución hasta alcanzar el mínimo.

- La limitación que tiene ACO-BP es su espacio de búsqueda, ya que está reducido a un número discreto de puntos.

- Con el desarrollo de este proyecto de investigación se profundizaron, complementaron y aplicaron algunos temas de la materia inteligencia artificial como redes neuronales y colonia de hormigas en un campo interdisciplinario. Este proyecto representa un aporte al ICP ya que es novedoso en su aplicación y es el precedente para posteriores aplicaciones.

Capítulo 7

RECOMENDACIONES

- Se propone a partir de este proyecto explorar otras técnicas de Inteligencia Artificial como optimización por enjambre de partículas (PSO) o los variantes en la optimización por colonia de hormigas, no solo para comparar el rendimiento del algoritmo ACO-BP frente a esta nueva implementación, sino también para fomentar la investigación en los estudiantes.

- Dado que la literatura existente de colonia de hormigas es escaza, sería útil que la universidad adquiriera convenios con otras universidades expertas en el área de inteligencia Artificial para que brinden ayuda y soporte a estudiantes interesados en esta área.

- Se han mostrado que los resultados dados por ACO-BP son buenos a pesar de su limitado espacio de búsqueda, sin embargo seria de gran utilidad invertir en hardware que permita ampliar este espacio sin verse afectado el tiempo de cómputo.

- Se recomienda estudiar variantes al ACO tradicional como ACO para optimización contínua o ACO implementado de manera distribuida.

Bibliografía

[1] N. Pokudom, “Determine of appropriate neural networks structure using ant colony system,” in *ICCAS-SICE, 2009*, pp. 4522 –4525, aug. 2009.

[2] M. D. T. Stutzle, *Ant colony Optimization*. Universite Libre de Bruxelles, 2006.

[3] A. S. Juan Peralta, German Gutierrez, “Adann: Automatic design of artificial neural networks,” in *, Computer Science Department University Carlos III,*, 2008.

[4] R. Fiszelew, A. & García-Martínez, “Generación automática de redes neuronales con ajuste de parámetros basado en algoritmos genéticos,” in *Buenos Aires, Argentina*, 2007.

[5] J. D. S. D. W. L. J. Eshelman, “Combinations of genetic algorithms and neural networks: A survey of the state of the art,” *IEEE*, vol. 1, pp. 1–37, 1992.

[6] G. Wei, “Study on evolutionary neural network based on ant colony optimization,” in *International Conference on Computational Intelligence and Security Workshops*, 2001.

[7] K. Socha and C. Blum, “An ant colony optimization algorithm for continuous optimization: application to feed-forward neural network training,” *Neural Computing & Applications*, vol. 16, pp. 235–247, 2007. 10.1007/s00521-007-0084-z.

[8] M.-G. W. Yan-Peng Liu and J.-X. Qian, "Evolving neural networks using the hybrid of ant colony optimization and bp algorithms," *Springer-Verlag Berlin Heidelberg*, vol. LNCS 3971, p. 714 – 722, 2006.

[9] M. J. MOLINA V DANIEL, AVARRO URIBE URIEL N, "Partial least-squares (pls) correlation between refined product yields and physicochemical properties with the 1h nuclear magnetic resonance (nmr) spectra of colombian crude oils.," *ENERGY & FUEL*, vol. 1, p. 1, 2007.

[10] "Fundamentos de química orgánica."

[11] E. Méndez and J. S. Mariño, "Sistema automático de entrenamiento de redes neuronales artificiales basado en el ajuste genético de parámetros y variación de arquitectura," in *Congreso internacional de Ingeniería Electrónica y Mecatrónica*, 2008.

[12] S. Haykin, *Neural Networks: A comprehensive foundation*. Editorial Pearson, 1999.

[13] "Historia backpropagation."

[14] D. Dasgupta and D. R. McGregor, "Designing application-specific neural networks using the structured genetic algorithm.," *IEEE*, vol. 0-8186-2787, pp. 51–92, 1992.

[15] M. Dorigo, "Ant colony optimization," *Evolutionary Computation, IEEE Transactions on*, vol. 8, pp. 422 – 423, aug. 2004.

[16] M. Dorigo and L. M. Gambardella, "Ant colony system: A cooperative learninig approach to the traveling salesman problem," *IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION*, vol. 1 No 1, pp. 1–14, 1997.

[17] M. Dorigo and C. Blum, "Ant colony optimization theory: A survey," 2005.

[18] M. Dorigo, Maniezzo, and C. V. Alberto, “The ant system: Optimization by a colony of cooperating agents,” *IEEE Transactions on Systems, Man, and Cybernetics–Part B,*, vol. Vol.26, No.1, pp. pp.1–13, 1996.

[19] M. Dorigo and L. Gambardella, “Ant colony system: A cooperative learning approach to the traveling salesman problem,” *IEEE Transactions on Evolutionary Computation,*, vol. Vol.1, No.1, pp. 1–24, 1997.

[20] *La investigación-acción en el aula Miguel Martínez Miguélez Universidad Simón Bolívar*, 2006.

[21] G. Cybenko, “Approximation by superpositions of a sigmoidal function,” *Mathematics of Control, Signals, and Systems (MCSS)*, vol. 2, pp. 303–314, 1989. 10.1007/BF02551274.

Apéndice A

Artículo realizado

OPTIMIZACIÓN DE LOS PESOS INICIALES DE UNA RED NEURONAL ARTIFICIAL USANDO ANT COLONY OPTIMIZATION

Sonia Luna Yuly Lípez

July 15, 2011

Abstract

La metodología aplicada pretende optimizar los pesos iniciales de una red neuronal artificial con una arquitectura definida a priori. En general, dichos pesos se generan aleatoriamente, por lo que es conveniente realizar varias sesiones de entrenamiento sobre la misma red con diferentes pesos, en forma tal de comenzar el aprendizaje desde diferentes puntos de la función objetivo y así tener más posibilidad de alcanzar el mínimo global. ACO tiene la capacidad de buscar la solución óptima global, y el algoritmo backpropagation tiene la característica de rápida convergencia de los óptimos locales. El híbrido adecuado de los dos algoritmos (ACO-BP) puede acelerar la velocidad de evolución de las redes neuronales y mejorar la precisión de la red.

1 INTRODUCCIÓN

El entrenamiento de una red neuronal artificial es el proceso más importante en el aprendizaje supervisado, en cual pares de entradas y salidas esperadas son usadas con objetivo de ajustar los pesos para obtener el mínimo error de salida y lograr que la red aprenda. El algoritmo de entrenamiento más utilizado es el backpropagation que trabaja usualmente con el método de optimización Levenberg Marquard, el cual tiene una rápida convergencia, pero el inconveniente de que fácilmente puede quedar atrapado en un mínimo local .

El algoritmo de optimización por colonia de hormigas (ACO) fue propuesto por Marco Dorigo [1] a principios de los noventa. Es un algoritmo meta heurístico que está inspirado en el comportamiento social de las hormigas. Fue diseñado para resolver problemas de optimización combinatoria. Trabajos investigativos han demostrado que la optimización por colonia de hormigas es un buen método para encontrar los pesos iniciales en la red [2], ya que en el espacio de búsqueda de dicha configuración hay múltiples combinaciones de pesos y conexiones entre neuronas, por lo tanto se convierte en un problema combinatorio además el algoritmo supera el problema del optimo local gracias a parámetros del algoritmo como el factor de evaporación ρ, evitando estancamientos en la búsqueda y permitiendo que las hormigas exploren diferentes regiones del espacio [3]. En el presente trabajo se muestra un híbrido entre el algoritmo backpropagation y colonia de hormigas. ACO se utiliza para encontrar un punto cercano a la solución óptima, para luego ser evaluada con backpropagation encontrando la solución global. Para probar el algoritmo, se usan datos de RMN [1]de crudos, para estimar el valor de la propiedad fisicoquímica API proporcionados por el ICP [2].

2 MARCO TEÓRICO

2.1 Colonia de Hormigas

Las hormigas son insectos que tienen la capacidad de encontrar el camino más corto a una fuente de alimento desde su colonia. Esto lo hacen con un mecanismo de comunicación basado en una sustancia química llamada feromona.

Cada hormiga cuando sale de su hormiguero a la fuente de comida y viceversa, va dejando un rastro (feromona) de la sustancia por el camino que pasa. Cuando otra hormiga lo percibe, tiende a pasar por esa marca, que se hace más intensa cuanto mayor número de hormigas pase por ahí [1].

Los algoritmos de colonia de hormigas son procedimientos de búsqueda estocástica en los cuales la probabilidad de seguir un camino particular es dependiente de la información de la fermonona. En cada camino, la información de la feromona representa una relación de cuantas veces una hormiga artificial ha transitado dicho camino. El modelo de feromona puede ser derivado del problema combinatorio, el cual tiene un número finito de soluciones, pero ese número es tan grande que lleva un tiempo enorme llegar a la solución óptima.

El problema combinatorio p es definido como sigue:

$$p = s, \Omega, f$$

Donde s representa el espacio de busqueda, Ω las re-

[1]Resonacia Magnética Nuclear

[2]Instituto Colombiano de Pétroleo

stricciones y f la función objetivo. El algoritmo básico de colonia de hormigas se describe de la siguiente manera:

1. Inicializa feromona: Los valores de feromona τ son inicializados con una constante de valor $c > 0$.

2. Construye solucion: La construccion de la solucion comienza con una solucion parcial vacía sp=0, luego en cada paso se va construyendo la solucion, añadiendo a la solucion actual sp componentes factibles.

Estos son escogidos probabilisticamente teniendo en cuenta la información heuristica η y la feromona τ.

$$P_{ij} = \begin{cases} \frac{\tau_{ij}^{\alpha}}{\sum_{j \in N_i^k} \tau_{ij}^{\alpha}} si..\text{j} \in N_i^k \\ 0 si\text{j} \notin N_i^k \end{cases} \quad (1)$$

Donde N_i^k es el conjunto de nodos faltantes por visitar conectados al nodo i, con respecto a la hormiga k. Si para algún nodo de la hormiga k $N_i^k = 0$ significa que el predecesor ya esta incluido en N_i^k, esto puede causar bucles dentro de los caminos construidos. Estos bucles son removidos una vez el nodo destino se ha alcanzado.

En la Ecuación anterior, α es una constante positiva usada para ampliar la influencia de la feromona. Grandes valores de α dan una excesiva importancia a la concentración de la feromona.

3. Actualización de la feromona: Este paso es requerido para evitar la rápida convergencia del algoritmo en regiones no óptimas y favorece la exploración de nuevas áreas en el espacio de búsqueda. Equivalentemente a los factores ambientales y circunstanciales, ρ es el factor de evaporación.

$$\tau_{ij} = \tau_{ij} - \rho * \tau_{ij} + \Delta\tau_{ij} \quad (2)$$

Donde $\Delta\tau_{ij}$ representa la suma de las contribuciones de todas las hormigas que se mueven sobre ij, dejando su rastro de feromona al construir su camino. Las contribuciones de las hormigas son proporcionales a la calidad de las soluciones logradas, es decir es mejor solución el camino que tenga más contribuciones de feromona.

2.2 Redes Neuronales Artificiales

Una neurona artificial, o unidad procesadora que modela matemáticamente una neurona biológica, sobre un conjunto de nodos N, es una tripleta X, f, Y donde X es un subconjunto de N, Y es un único nodo de N y $f :\rightarrow$ es una función neuronal (también llamada función activación) que calcula un valor de salida para Y basado en una combinación lineal de los valores de las componentes de X, es decir [4],

$$Y = f(\sum_{x_i \in X} w_i x_i). \quad (3)$$

Los elementos X,Y y f se denominan conjunto de nodos de entrada, conjunto de nodos de salida, y función neuronal de la unidad neuronal, correspondientemente ; x_i se relaciona con cada una de las entradas y w_i su respectivo peso asociado.

Una red neuronal está conformada por capas neuronas que interactúan para producir un estímulo de salida.

Se tiene una red neuronal artificial con m neuronas en la capa de salida y se dispone de un conjunto de aprendizaje con P patrones tiene salidas dadas por:

$$tp = tp_1, tp_2, tp_3, tp_4 \ldots tp_m$$

,siendo m la última neurona de salida. ,la salida de la red está dada por yp

El error cuadrático para ese patrón tiene la siguiente expresión:

$$E = \frac{1}{2} \sum (tp - yp)^2 \quad (4)$$

El algoritmo Backpropagation ha demostrado converger muy lentamente en varias aplicaciones, en especial cuando se tienen gran cantidad de patrones de entrada y hacen que el error cuadrático medio sea demasiado grande y es por esto que muchas veces este algoritmo se queda atrapado en un mínimo local [5].

Levenberg-Marquardt es un algoritmo iterativo basado en el método de aproximación de Newton y trata de minimizar la distancia entre dos puntos.

El algoritmo de Levenberg-Marquardt tiene convergencia rápida a pesar de que su complejidad en cálculos es mayor. Este algoritmo para cálculo del gradiente utiliza la matriz jacobiana que se define como la primera derivada de los errores de la red con respecto a los pesos.

$$g = J^T * E \quad (5)$$

Donde J^T es la matriz jacobiana traspuesta y E es el vector de errores entre la salida esperada y la obtenida.

3 METODOLOGÍA

3.1 Híbrido entre colonia de hormigas y Backpropagation

Dada la arquitectura de la red, deben hallarse los pesos iniciales para que esta sea entrenada. Para lograrlo, la hormiga debe encontrar la mejor manera de combinar estos pesos para obtener el mínimo error.

Como el espacio de búsqueda es infinito, se debe volver discreto. Para conseguirlo, para cada una de las conexiones, se crea una tabla (**??**) con m+1 parámetros, en donde cada uno de los parámetros es un punto o posible peso para esa conexión. Una vez construidas las tablas, la hormiga selecciona uno de los parámetros de la tabla para cada conexión y construye una ruta. Los valores de los pesos seleccionados por la hormiga son implementados en la red y posteriormente se calcula el error producido, y este es utilizado para la actualización de la feromona.

El algoritmo backpropagation usualmente inicializa todos los pesos de la red con números con valores aleato-

rios, así que corre el riesgo de quedar atrapado en un mínimo local.
ACO provee mejores valores iniciales al algoritmo backpropagation, logrando un mejor entrenamiento y por lo tanto mejores resultados.
La idea base del algoritmo es sencillamente que el algortimo ACO explore el espacio de soluciones y encuentre un lugar cercano a la solución óptima global, para que luego el algoritmo de backpropagation refine la solución hacia el mínimo global.
A continuación se muestra un bosquejo del algoritmo ACO para encontrar los pesos iniciales y ser implementados en el entrenamiento de la red. (Procedimiento 1)

Procedimiento 1 ACO

Entrada: $max.ite, \rho, p$
Salida: *pesos*
mientras $ite = max - ite$ **hacer**
1. Inicializa
2. Tour
para $hormiga = 1$ **to** $num.hormigas$ **hacer**
para $i = 1$ **to** $num.conexiones$ **hacer**
Selecciona un peso de acuerdo a la probabilidad:

$$P(i) = \frac{\tau(i)}{\sum \tau(i)} \tag{6}$$

$tour(i) \leftarrow peso$
fin para
3. Simula *Asigna pesos seleccionados por la hormiga y produce salida yp*
4. Error *con la salida yp y los patrones tp calcula:*

$$E = \frac{1}{2} \sum (tp - yp)^2 \tag{7}$$

5. Actualiza *Siendo* $\rho \in (0, 1)$ *es el factor de decaimiento de la feromona.:*

$$\tau(i+1) = \rho * \tau(i) + \triangle\tau(i) \tag{8a}$$

$$\triangle\tau(i) = \begin{cases} \frac{1}{E} & si \quad \tau(i) \in tour \\ 0 & si \quad \tau(i) \notin tour \end{cases} \tag{8b}$$

fin para
fin mientras

4 SELECCIÓN DE PARÁMETROS

La red neuronal artificial, fué creada con ayuda del toolbox de redes neuronales *Neural network* de Matlab.
Para su elaboración se utlizaron 1000 epocas, tasa de aprendizaje de 0.005 y los mismos ejemplos de entrenamiento(70%), validación(15%) y prueba(15%).
La arquitectura de red seleccionada fué de dos capas. Tres neuronas en la capa oculta con función de transferencia tansigmoidal[3]. y una neurona en la capa de salida con función de transferencia lineal [4].

A ésta se incorporó el algoritmo de hormigas con todas sus funciones. Los parámetros para la construcción de ACO son el número de hormigas, número de elementos de la tabla y máximo número de iteraciones. Para desarrollar el algoritmo, es necesario que los parámetros sean escogidos de la mejor manera, pues estos influyen en gran medida en su rendimiento.
Luego de la realización de pruebas de sensibilidad, se tomaron 8 hormigas, 75 elementos en la tabla y 100 iteraciones.

[3] $f(x) = \frac{1}{1+e^{-2n}} - 1$
[4] $f(x) = x$

5 RESULTADOS

Para el entrenamiento y prueba de la red neuronal artificial, se utilizaron datos de resonancia magnética nuclear suministrados por el ICP, de los cuales el 70% fueron para entrenamiento, 15% para testeo y 15% para validación.
Para su consecusión, se utilizaron un equipo con Windows XP con memoria de 1.93 GB y 2.91Ghz, procesador AMD Athlom de 64 dual core, con la versión de Matlab 7.8 R2009a y un equipo un equipo con Windows XP con memoria de 3.23 GB y 2.33Ghz, procesador Intel core Quad, con la versión de Matlab 7.9 R2009b .
Tanto el algoritmo BP, como el ACO-BP fueron simulados 50 veces, utilizando los mismos ejemplos para el entrenamiento y la prueba.. La gráfica (Figura 1) es una comparación entre hormigas y el toolbox de matlab. En ella se observa el *Emse* (El error obtenido después de presentarle a la red los ejemplos de entrenamiento) vs $performance$ (El error calculado después de ajustar los pesos).

Figure 1: Comparación del error con ACO-BP, BP

Se puede obsrevar que el algoritmo de colonia de hormigas obtiene Emse mas pequeño y un mejor rendimiento que el toolbox de matlab, ya que la hormiga permite encontrar los mejores pesos que hacen que este error disminuya, esto facilita y ayuda a encontrar el mínimo global y no permite que el algortimo se quede atrapado en un mínimo local.
Para la validación del algoritmo se tomaron 10 crudos, los cules no fueron presentados a la red, con el fin de comprobar la capacidad de generalización.
Los resultados arrojados (5)mostraron que el algoritmo ACO-BP tiene una capacidad superior de generalización que el BP.

Crudos	Datos Experimentales	Resultados arrojados por la red	error
1	36.1	36.3974	0.29743
2	42.1	42.3788	0.27875
3	42.6	42.8965	0.29650
4	43.5	43.6444	0.14441
5	45.1	45.0813	0.01868
6	43.0	43.0958	0.09575
7	29.0	29.6766	0.67658
8	23.8	24.0299	0.22989
9	22.7	22.8348	0.13482
10	12.3	12.2950	0.00498

Table 2: Resultados obtenidos por BP

Error cuadrático = 0.0271101583333333

Crudos	Datos Experimentales	Resultados arrojados por la red	error
1	36.1	36.0989	0.001099
2	42.1	42.0987	0.001264
3	42.6	42.5984	0.001621
4	43.5	43.4993	0.000743
5	45.1	45.1003	0.000289
6	43.0	43.0031	0.003127
7	29.0	28.9987	0.001347
8	23.8	23.7994	0.000536
9	22.7	22.7072	0.007152
10	12.3	12.3790	0.079032

Table 3: Resultados obtenidos por ACO-BP

Error cuadrático: 0.000210508121790253

6 CONCLUSION

A pesar de que el algoritmo Backpropagation (BP) ha demostrado ser un buen método de entrenamiento de una red neuronal, algunas veces se queda atrapado en mínimo local debido aleatoriedad de los pesos iniciales. En el presente trabajo, se muestra un método para superar este problema basado en la optimización de los pesos iniciales con Ant Colony Optimization (ACO). Los resultados muestran que el híbrido de los dos algoritmos (ACO-BP), disminuyen los errores de aproximación de la red. ACO encuentra en el espacio de soluciones un punto cercano al mínimo global y BP refina la solución hasta alcanzar el mínimo.

Los autores expresan un agradadecimiento especial al Instituto Colombiano de Petróleo, quién patrocinó esta investigación.

References

[1] "Ant colony optimization," *Evolutionary Computation, IEEE Transactions on*, vol. 8, pp. 422 – 423, aug. 2004.

[2] Y.-P. Liu, M.-G. Wu, and J.-X. Qian, "Evolving neural networks using the hybrid of ant colony optimization and bp algorithms," vol. 3971, pp. 714–722, 2006.

[3] M. D. T. Stutzle, *Ant colony Optimization*. Universite Libre de Bruxelles, 2006.

[4] S. Haykin, *Neural Networks - A Comprehensive Foundation*. Pearson Education, 1999.

[5] J.-B. Li and Y.-K. Chung, "A novel backpropagation neural network training algorithm designed by an ant colony optimization," in *IEEE/PES Transmission and Distribution Conference & Exhibition: Asia and Pacific Dalian, China*, 2005.

Apéndice B

Simulaciones hechas al algoritmo

68	69	70	71	72	73	74	75
0,0447	4,34	0,0467	0,0561	0,0491	0,0611	0,0587	0,0532
0,0641	0,128	0,0565	0,0848	0,0639	0,193	0,052	0,0525
0,352	0,0649	0,046	0,0442	0,0637	0,0522	0,0525	0,0521
0,0514	0,055	0,049	0,274	0,12	0,0517	10,5	0,0619
0,642	1,64	0,0497	0,0538	0,0448	0,0482	0,499	0,0521
0,0508	0,0465	0,0492	0,0578	0,0598	0,0507	0,0487	0,0527
0,0527	0,056	0,0451	0,0478	0,0495	0,0455	0,0501	0,0474
1,42	0,104	0,0458	0,0444	0,0609	0,0469	0,383	0,0489
0,117	0,503	0,049	0,113	0,0492	0,0464	0,0555	0,0456
0,051	0,0485	0,0471	0,118	0,0445	0,911	0,0536	0,0612

Cuadro B.1 Simulaciones para Número de parametros de la tabla

Número de parametros/Error

Error inicial 50 ite	Error final 50 ite	Error inicial 100 ite	Error final 100 ite
131	0,06	45,4	0,0477
122	0,344	58,6	0,0502
125	0,0551	80,9	0,0453
100	0,0557	96,2	0,0447
175	0,492	135	0,0447
99,9	0,0442	142	0,0909
58,6	0,105	48,8	0,0442
79,8	0,0551	85,3	0,0455
70,8	0,0511	119	0,0556
111	0,0544	50,6	0,0442

Cuadro B.2 Simulaciones para Número iteraciones

Error inicial-final 50 iteraciones/Error inicial-final 100 iteraciones

Error inicial Ant	Error final Ant	Error inicial Bp	Error final Bp
165	0,058	195	0,0491
67,6	0,059	410	0,0524
203	0,0508	76,4	0,0481
231	2,66	1,35E+003	0,283
109	0,0469	126	0,044
277	0,0487	121	0,0523
211	0,0409	300	4,68
128	0,298	82,4	0,0478
187	1,74	85,4	0,636
47,7	0,051	108	0,0489
90,8	0,0563	103	0,0552
84,6	0,307	324	1,61
124	0,051	252	0,0506
202	0,0461	480	0,0567
114	0,0456	182	0,0646

Cuadro B.3 Simulaciones comparativas Hormigas sin Hormigas

Se utilizaron p=75, ρ=0.9 y 8 hormigas

69	70	71	72	73	74	75
120,4	239,8	361,1	482,1	603,3	724,3	853,2
116,1	232,7	349,2	465,9	582,02	698,7	815,8
117,6	235,7	352,8	470,0	588,8	707,2	825,7
119,1	239,5	360,1	480,5	599,0	718,8	848,9
117,4	235,0	353,2	472,6	592,2	712,8	832,5
118,4	236,4	355,2	474,3	594,0	714,9	839,0
120,4	239,0	357,6	477,5	600,3	723,5	847,0
119,6	235,7	351,6	469,2	593,9	714,8	835,6
120,6	242,7	363,2	485,7	608,8	728,0	845,2
109,2	232,5	354,2	475,3	596,7	717,8	837,1

Cuadro B.4 Tiempo de simulación en sg para diferentes valores de p

Se utilizaron p=75, ρ=0.9 y 8 hormigas

4	6	8	10	12
81	120	134,9	178,49	245,7
84	121,9	156,6	207,6	244,6
83	118,8	134,6	173,4	238,9
83	133,4	135,5	175,9	246,5
89	127,8	132,8	194,2	245,7
90	121,9	136,2	217,9	244,6
95	118,8	135,3	205,8	238,5
95	123,9	135,6	211,2	246,5
96	119,6	134,6	204,8	245,7
101	115,9	135,5	211,3	240,7

Cuadro B.5 Tiempo de simulación en sg para diferente cantidad de hormigas

Se utilizaron p=75, ρ=0.9 y 8 hormigas

59	60	61	62	63	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79	80
102,5	124,8	161,2	300,4	214,6	141,2	162,7	239,1	102,5	132,2	202,3	263,8	265,2	110,7	117,7	202,0	196,9	166,2	162,8	118,4	228,9	138,2
173,4	197,4	354,6	106,0	44,90	168,0	91,72	87,83	188,8	134,3	126,8	123,9	121,7	62,16	112,4	106,6	114,3	88,37	170,1	86,98	128,5	121,1
278,4	100,7	135,3	201,1	166,1	89,93	178,5	120,9	131,2	106,4	172,8	138,1	123,6	73,5	73,6	145,6	206,9	142,2	116,7	171,7	240,1	98,70
103,9	145,7	231,2	276,2	141,4	151,1	218,2	137,2	102,4	116,5	146,7	242,7	226,1	224,4	124,3	185,6	139,3	181,9	127,3	241,3	102,0	105,3
91,74	187,9	255,5	211,1	250,4	221,5	194,5	262,4	271,6	106,9	219,6	277,7	207,2	96,54	233,5	155,6	148,4	310,5	135,0	110,3	154,4	234,9
68,4	45,54	209,5	154,9	146,5	146,4	68,58	289,9	125,6	89,0	101,6	271,8	186,2	140,6	112,3	74,3	78,9	93,4	104,3	155,2	73,7	170,3
107,8	155,2	168,7	115,6	88,78	67,06	274,4	343,7	212,1	178,6	169,3	72,1	180,1	236,1	181,5	218,4	192,8	230,8	80,19	85,10	152,9	124,5
264,8	199,8	136,0	300,6	158,0	381,0	153,5	101,7	143,8	212,1	82,9	113,2	220,3	59,3	223,1	261,9	112,4	141,1	198,8	181,0	221,7	178,9
92,04	147,1	83,14	166,7	206,9	117,4	116,5	122,4	132,1	144,9	132,7	155,4	131,7	137,6	135,9	66,45	182,9	141,4	86,31	199,8	85,82	333,9
189,8	130,0	90,16	126,1	67,84	99,51	86,04	55,03	50,61	93,5	131,1	104,1	81,9	97,5	156,1	170,4	156,3	117,8	97,5	148,5	143,5	114,1
149,0	104,4	104,3	100,6	56,87	320,7	61,61	106,9	109,1	81,08	128,1	223,0	194,9	141,0	123,0	238,5	66,06	77,82	173,5	215,2	82,40	122,4
264,8	199,8	136,0	300,6	158,0	381,0	153,5	101,7	143,8	212,1	82,9	113,2	220,3	59,3	223,1	261,9	112,4	141,1	198,8	181,0	221,7	178,9
144,6	133,4	104,4	110,8	134,6	183,4	105,1	108,0	137,8	153,3	215,5	205,5	167,9	176,5	127,1	147,9	165,0	92,02	61,76	118,4	157,1	48,61
189,8	130,0	90,1	126,1	67,84	99,51	86,04	55,03	50,61	93,5	131,1	104,1	81,9	97,5	156,1	170,4	156,3	117,8	97,5	148,5	143,5	114,1
134,7	197,2	59,94	177,2	88,42	151,4	79,26	93,39	151,4	159,4	121,0	108,8	190,9	204,0	63,76	129,5	66,00	289,8	149,2	101,7	124,9	81,08
105,6	184,5	71,15	116,0	87,92	82,39	80,22	110,6	149,4	62,4	190,0	62,5	169,5	98,6	86,7	117,7	93,3	48,4	131,3	182,3	46,3	174,0
165,4	220,6	115,1	192,0	83,34	110,3	63,34	150,7	130,3	130,6	124,2	180,2	206,3	160,5	157,4	129,6	126,5	98,14	344,3	168,3	256,8	75,49
236,0	367,5	316,4	254,3	183,2	128,7	241,2	170,8	264,5	106,3	223,2	53,63	87,53	137,8	297,2	122,1	199,6	218,5	135,6	192,8	134,3	186,4
247,6	102,1	156,6	127,5	300,8	205,7	170,3	121,1	226,8	70,56	81,58	296,1	182,3	240,7	109,7	267,4	145,7	132,0	228,1	157,8	102,7	108,7
104,6	254,7	249,3	237,9	304,2	147,0	181,3	79,34	73,52	202,8	185,2	107,5	266,7	171,2	276,7	213,1	188,6	147,2	128,4	275,7	314,7	117,6

Cuadro B.6 Error inicial arrojado por ACO para diferentes valores de p