ISSN: 1692-7257 - Volumen 1 – Número 37 - 2021

Revista Colombiana de Tecnologías de Avanzada

Recibido: 23 septiembre 2020
Aceptado: 12 de noviembre de 2020

IDENTIFICACIÓN DE FALLAS EN SISTEMAS DE BOMBEO MECÁNICO DE PETRÓLEO UTILIZANDO NEUROFUZZY

IDENTIFICATION OF FAULTS IN MECHANICAL PETROLEUM PUMPING SYSTEMS USING NEURO FUZZY

MEng. Jorge Enrique Meneses Flórez [*], **Ing. Fredy A. Garavito, Ing. Edxon Meneses**

[*] **Universidad Industrial de Santander (UIS)**, Escuela de Ingeniería Mecánica.
Grupo de Investigación en Conectividad y Procesado de Señales - CPS
Ciudad Universitaria, Bucaramanga, Santander, Colombia.
Tel.: (+577) 634 4000 Ext. 2483/2829.
E-mail: jmeneses@uis.edu.co

Resumen: En el bombeo mecánico de petróleo, para minimizar los costos operativos y maximizar la producción, es esencial identificar los problemas de forma rápida y precisa. El dinagrama de fondo de pozo es decisivo para analizar las condiciones de trabajo del sistema de bombeo, y normalmente el diagnóstico de fallas se ha basado en la interpretación visual de su forma, por un experto humano. Se presenta una arquitectura (NeFSuckerRod) para el diagnóstico automático del dinagrama, basada en un sistema NeuroFuzzy, que permite identificar las fallas y prescindir del experto humano. Al combinar el poder de aprendizaje de las redes neuronales artificiales y la representación explícita del conocimiento de la lógica fuzzy, y presentando un conjunto de cartas dinamométricas con diferentes fallas, el sistema NeuroFuzzy se entrena obteniendo un modelo fuzzy capaz de diagnosticar fallas en un sistema de bombeo.

Palabras clave: Dinagramas, bombeo mecánico, neurofuzzy.

Abstract: In mechanical oil pumping, to minimize operating costs and maximize production, it is essential to identify problems quickly and accurately. The downhole dynagram is decisive in analyzing the working conditions of the pumping system, and normally the fault diagnosis has been based on the visual interpretation of its shape by a human expert. An architecture (NeFSuckerRod) is presented for the automatic diagnosis of the dynagram, based on a NeuroFuzzy system, which allows to identify faults and dispense with the human expert. By combining the learning power of artificial neural networks and the explicit representation of fuzzy logic knowledge and presenting a set of dynamometric charts with different faults, the NeuroFuzzy system is trained obtaining a fuzzy model capable of diagnosing faults in a system of pumping.

Keywords: Dynagrams, sucker-rod pumping, neurofuzzy.

1. INTRODUCCIÓN

1.1 Bombeo Mecánico y Dinagrama de fondo

Se requiere de un Sistema de Levantamiento Artificial (SLA), cuando el crudo de una formación petrolera no tiene autosuficiencia para fluir a la superficie. El 50% de los pozos de petróleo del mundo tienen algún tipo de SLA, y de ellos, el 40% utilizan el denominado “bombeo mecánico” (Gabor, 2015).

El bombeo mecánico (BM) es un procedimiento de succión y transferencia casi continua del petróleo hasta la superficie, que involucra un sistema que utiliza elementos tanto en la superficie como en el fondo del pozo, donde los componentes principales del sistema son: unidad de bombeo, sartas de varillas y bomba de fondo de pozo. La unidad de bombeo es un equipo ubicado en la superficie que, utilizando un motor y un mecanismo de balancín, biela y manivela, genera un movimiento alternativo en una sarta de varillas, la cual mueve el pistón de una bomba de desplazamiento positivo (bomba de fondo), colocada en el extremo inferior de la sarta, a cierta profundidad del fondo del pozo (Fig. 1).

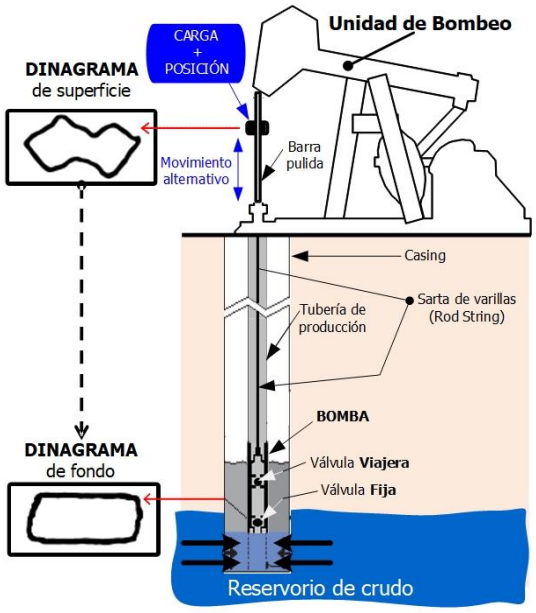

Fig. 1. Sistema de bombeo mecánico y dinagramas

La función de la bomba es levantar fluido (crudo) desde el fondo del pozo hasta la superficie, impulsándolo por la tubería de producción hasta el punto de recolección, constituyéndose en el elemento más importante de una instalación y su funcionamiento incorrecto, tiene un impacto directo en el nivel de beneficio económico del pozo y del yacimiento de petróleo (Sanchez *et al.*, 2007).

La bomba de fondo (Fig. 2) se compone de un cilindro (barril de trabajo), un pistón (émbolo) y dos válvulas de bola, denominadas "fija" y "viajera". El pistón contiene la válvula viajera y es el elemento móvil de la bomba, con un movimiento alternativo transmitido por la unidad de bombeo a través de la sarta de varillas. El barril es la cámara de la bomba donde se aloja el fluido durante la acción de bombeo; debido a la variación de su volumen y como consecuencia del movimiento alternativo del pistón que se encuentra en su interior, se produce la admisión y la descarga del fluido. Se denomina carrera o "stroke", al desplazamiento alternativo del pistón desde el punto muerto superior hasta el punto muerto inferior (movimiento descendente) y su regreso al punto muerto superior (movimiento ascendente).

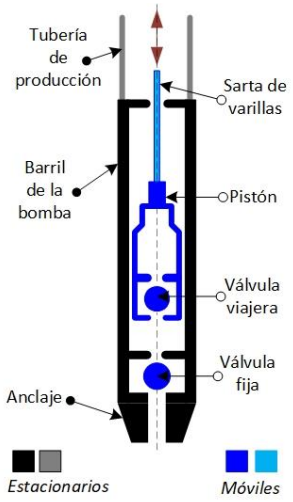

Fig. 2. Bomba de fondo

Al medir en el fondo del pozo durante una carrera, la posición del pistón y simultáneamente la fuerza instantánea requerida para para moverlo, y graficar esas dos variables, se obtendrá una curva cerrada que se denomina "carta dinagráfica" o "dinagrama de fondo" (Fig. 3). Walton Gilbert (Gilbert, 1936) evidenció que mediante el análisis del dinagrama de fondo, una persona experimentada puede diagnosticar con precisión cualquier problema que presente una bomba y por ello se desarrollaron los dinamómetros de fondo de pozo.

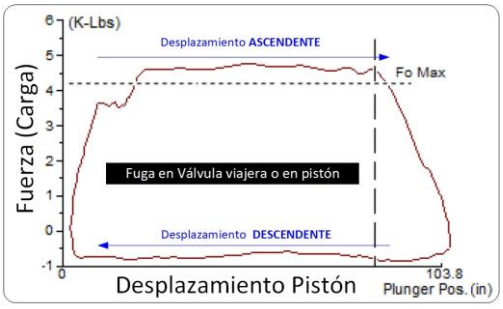

Fig. 3. Dinagrama con una falla identificada.

Es evidente la dificultad para realizar mediciones directamente sobre la bomba, en el fondo del pozo, y aunque los dinamómetros de fondo proporcionan

una representación precisa del comportamiento de la bomba, debido al costo, su uso diario no es práctico. Es más funcional colocar un dinamómetro de superficie que, unido a la barra pulida en la cabeza del pozo, mida la fuerza (carga) en función de su posición, para generar lo que se denomina "dinagrama de superficie" (McCoy y Podio, 1995).

Los dinagramas de superficie no permiten un diagnóstico completo del comportamiento del sistema, siendo valiosos solo para diagnosticar problemas en la superficie, sin embargo, a partir de ellos, se puede inferir las condiciones del fondo del pozo (Gibbs, y Nelly, 1966) resolviendo la denominada "Ecuación de Onda", que permite obtener el dinagrama de fondo. El dinagrama de fondo obtenido se constituye en un insumo fundamental para diagnosticar los problemas de la bomba (Eickmeier,1967), y su obtención a partir del dinagrama de superficie es ahora el estándar de la industria (DaCunha y Gibbs, 2007).

En conclusión, los dinagramas de fondo son esenciales para determinar el funcionamiento de un sistema de bombeo mecánico, ya que mostrarán formas disímiles que podrían corresponder a una operación normal o a situaciones de fallas (Fig. 4).

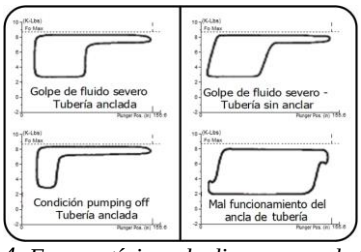

Fig. 4. Formas típicas de dinagramas de fondo

1.2 Detección de fallas a través de técnicas de IA

Tradicionalmente el diagnóstico de fallas se llevado a cabo gracias al conocimiento y habilidad de un experto humano, y su metodología se basa en hallar los problemas comparando visualmente las formas de los dinagramas de fondo obtenidos, con formas características de falla (Dickinson y Jennings, 1990).

La complejidad de los sistemas de bombeo mecánico, la diversidad de formas de los dinagramas de fondo, la dependencia de la experiencia y de la habilidad del experto humano que diagnostica, y finalmente la imposibilidad de diagnosticar en tiempo real, son factores que indudablemente tienen un impacto importante en la operación de un yacimiento petrolero. Para lograr una producción óptima y reducir los costos de operación y mantenimiento, es necesario desarrollar y aplicar metodologías que permitan identificar, de forma rápida, los problemas que puedan afectar la operación del sistema de bombeo. Lo anterior conduce inevitablemente a la necesidad de un sistema con una arquitectura "automática e inteligente" (Sletcha *et al.*, 2020; de Best, 2012), que implica no solamente capturar y procesar las variables físicas que permitan obtener los dinagramas de superficie y de fondo (Meneses *et al.*, 2015; Gómez *et al.*, 2013), sino que adicionalmente requiere de un nivel software que permita prescindir del experto humano y diagnosticar las fallas de manera automática. Es aquí donde los conceptos y las técnicas de inteligencia artificial pueden ayudar al equipo del campo petrolero a administrar no solo los pozos, sino también a "digitalizar" su propia experiencia de diagnóstico de fallas, en una base de datos de "conocimiento digitalizada", para ser usada en un sistema automático inteligente.

Un sistema de bombeo mecánico es complejo, y en el caso de enfrentar el problema del diagnóstico de sus fallas usando modelos matemáticos, la realidad mostrará que éstos no son lo suficientemente precisos. Entonces, en lugar de enfrentar el problema con un modelamiento matematico detallado donde es requerido un conocimiento a priori cuantitativo sobre el sistema, lo mejor es que a partir de la disponibilidad de una gran cantidad de datos históricos (dinagramas de fondo) con diagnóstico de las fallas ocurridas durante la operación, estas se transformen y se presenten como una base conocimiento (humana), mediante la cual *se entrene a un sistema software* que lo digitalice y con ello se pueda realizar un diagnóstico automático. Lo anterior se conoce como extracción de características, donde son especialmente aplicables las tecnologías de inteligencia artificial, las cuales aportan algoritmos que permiten materializar en software, el conocimiento que un experto humano posee al diagnosticar un sistema. La arquitectura (NeFSuckerRod), se basa en una técnica de inteligencia artificial hibrida que combina la lógica fuzzy y las redes neuronales, que utilizando dinagramas de fondo clasificados por un experto humano, entrena una red NeuroFuzzy obteniendo un *Modelo fuzzy* capaz de diagnosticar fallas en un sistema de bombeo mecánico de petróleo.

1.3 Sistemas Neurofuzy

Construir un modelo fuzzy implica digitalizar el conocimiento que posee un experto humano, mediante la obtención de reglas fuzzy y conjuntos fuzzy, lo cual conlleva tiempo y propensión al error. Lo deseable sería poseer un sistema que permitiera el aprendizaje a partir de ejemplos presentados, y que el conocimiento digitalizado estuviera disponible en el lenguaje natural utilizado por los seres humanos. Ni los modelos basados en la lógica fuzzy, ni los basados en redes neuronales artificiales, satisfacen simultáneamente esos dos requerimientos. Surge así la idea de aplicar los algoritmos de aprendizaje a los sistemas fuzzy, donde las redes neuronales con sus características computacionales de aprendizaje contribuyen en los sistemas fuzzy, recibiendo de ellos la interpretación y claridad de la representación computacional del lenguaje natural basado en predicados, utilizado por los seres humanos.

De manera general, todas las combinaciones de técnicas basadas en redes neuronales y lógica fuzzy se denominan "*Sistemas Neurofuzy*". Las diferentes combinaciones de estas técnicas poseen las siguientes características (Nauck et al., 1997): a) en los *Sistemas NeuroFuzzy Cooperativos*, las redes neuronales se utilizan solo en una fase inicial para determinar sub-bloques del sistema fuzzy, es decir, para obtener los conjuntos fuzzy y las reglas fuzzy utilizando los datos de entrenamiento. Luego se eliminan las redes neuronales y solo se ejecuta el sistema fuzzy. b) En los *Sistemas NeuroFuzzy Concurrentes*, la red neuronal ayuda al sistema fuzzy continuamente (o viceversa) a determinar los parámetros requeridos. Es la forma más débil de establecer una combinación NeuroFuzzy. c) En los *Sistemas NeuroFuzzy Híbridos*, se combinan las ventajas de los sistemas fuzzy, que se ocupan del conocimiento explícito (en lenguaje natural) que se puede explicar y comprender, y las redes neuronales que se ocupan del conocimiento implícito que se puede adquirir mediante un aprendizaje supervisado a partir de ejemplos. Tanto los sistemas cooperativos como los concurrentes, no optimizan el sistema fuzzy, sino que solo ayudan a mejorar el rendimiento del sistema en general; en contraste, los sistemas *NeuroFuzzy Híbridos* si se ocupan de la optimización y por ello fue utilizado en este trabajo.

La intención principal de un sistema *NeuroFuzzy Híbrido* es crear o mejorar un sistema fuzzy apoyado por las redes neuronales artificiales, o si se quiere ver de otra manera, mejorar el proceso de aprendizaje de una red neuronal artificial a partir de la lógica fuzzy, constituyéndose en la opción más adecuada para la clasificación de patrones, porque ofrece la interpretabilidad de las reglas generadas en el entrenamiento de la red, la posibilidad de agregar conocimiento previo al sistema para acortar los tiempos de entrenamiento y una sintonización automática de los conjuntos fuzzy. El término "NeuroFuzzy", es usado recurrentemente en la literatura, para referirse a los sistemas *NeuroFuzzy Híbridos*, y de igual manera ocurrirá de aquí en adelante en este documento.

1.4 NEFCLASS

NEFCLASS (NEuro Fuzzy CLASSification) es un sistema NeuroFuzzy hibrido (Nauck y Kruse, 1995) que permite determinar la categoría correcta de un patrón de datos (clasificar), a través de un "*Modelo Fuzzy*" obtenido mediante la propagación de los patrones de datos a través de la red, utilizando un algoritmo de "aprendizaje supervisado". Se basa en el modelo genérico de un "Perceptrón Fuzzy" (Nauck, 1994) que se puede utilizar para derivar redes neuronales fuzzy o sistemas fuzzy neuronales para dominios específicos. Las características consideradas para su utilización en este trabajo fueron:

- La interpretabilidad de la clasificación, basada en el lenguaje natural y sus predicados del tipo SI-ENTONCES, que lo convierten en un clasificador legible, con una exactitud aceptable.
- Respecto a otros sistemas, posee mayor rapidez en la clasificación.
- Permite que el usuario pueda influir en el algoritmo de aprendizaje.

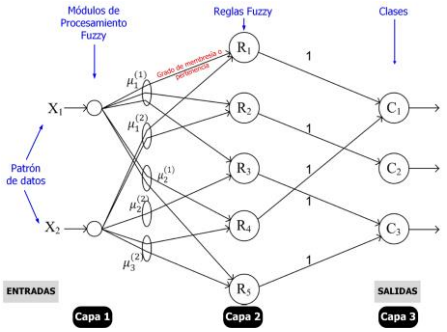

Fig. 5. NEFCLASS con dos entradas, cinco reglas y tres salidas (clases)

NEFCLASS puede ser visto como una red neuronal de 3 capas feedforward (Fig. 5), donde cada una de

las capas cumple una función. La primera capa (de entrada) representa los nodos fuente y cada uno de ellos funciona como un módulo (unidad) de procesamiento fuzzy (Fig. 6); se encarga de recibir cada uno de los patrones de datos, donde cada nodo recibe un dato numérico proveniente del patrón, y esa entrada se constituye en una variable fuzzy con su propio universo de discurso y etiquetas lingüísticas (conjuntos fuzzy). La segunda, capa oculta o capa de reglas, es la encargada de realizar las inferencias lingüísticas, almacenando las reglas mediante la conexión del antecedente con el consecuente de cada una de ellas; la activación de los nodos de esta capa es realizada por un operador fuzzy. La tercera capa o capa de clases, es la que representa las diferentes opciones a la que puede pertenecer un patrón de datos, y se encarga de hacer el proceso para el cálculo del error.

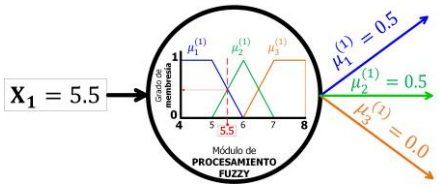

Fig. 6. Variable fuzzy con tres conjuntos fuzzy y un universo de discurso entre 4 y 8.

Como ya se ha escrito, NEFCLASS permite determinar la clase o categoría correcta de un patrón de entrada dado. Los patrones son vectores $\mathbf{x} = (x_1, \ldots, x_n) \in \mathbb{R}^n$ y una clase *C* es un subconjunto clásico o "certero" (crisp) de $\mathbb{R}^n$. Se asume que la intersección de dos diferentes clases será un conjunto vacío. El valor de las características del patrón de datos, está representado por conjuntos fuzzy y la clasificación se describe mediante un conjunto de reglas lingüísticas. Para cada característica x_i de la entrada, existen q_i conjuntos fuzzy $\mu_1^{(i)}, \ldots, \mu_{q_i}^{(i)}$ y la base de reglas contiene k *reglas* fuzzy $R_{1,\ldots,} R_k$

Remitiéndonos a la figura 5, μ_j^i representa los pesos sinápticos $w(x_i, R_j)$ donde *j* representa la variable de entrada (valor numérico) a la unidad de procesamiento *j* de la primera capa, y la *i* representa el número del conjunto fuzzy para la variable de entrada *j*. En otras palabras, cada variable de entrada es fuzificada, y para manejar su imprecisión y ambigüedad se utilizan conjuntos fuzzy (términos lingüísticos); para cada variable de entrada (variable fuzzy) se define de manera específica el número de conjuntos fuzzy y su universo de discurso. En general, un sistema NEFCLASS obedece a la siguiente configuración:

1. n unidades de entrada representadas por $U_1 = \{x_1, x_2, .., x_n\}$ U1= x1, x2,.., xn,. Donde U_1 representa la capa de entrada, n el número de variables (numéricas) de entrada y x su correspondiente neurona de la capa de entrada.

2. Un numero $k \leq k_{max}$ de reglas $U_2 = \{R_1, R_2, .., R_k\}$. Donde U_2 representa la capa escondida, k el número de reglas y R la regla o neurona de la capa oculta.

3. m unidades de salida de la capa $U_3 = \{C_1, C_2, .., C_m\}$. Donde U_3 representa la capa de salida, m el número de clases y C la neurona de salida.

4.Cada conexión entre las unidades $x_i \in U_1$ y $R_j \in U_2$ tiene un peso que es el grado de membresía de un conjunto fuzzy, el cual es etiquetado con el termino lingüístico $\mu_{jr}^{(i)} (jr \in \{1,2, \ldots., q_i\})$.

5.Los pesos de las conexiones sinápticas $W(R,C) \in \{1,0\}$ para todo $R \in U_2$ y $C \in U_3$. Es decir, los pesos de las unidades de procesamiento de la capa oculta y la capa de salida pueden ser 1 o 0.

6. Si denotamos a $L_{x,R}$ como todas las etiquetas lingüísticas de las conexiones entre las $x \in U_1$ y $R \in U_2$ para todas $R, R \in U_2$ se cumple que: $(V(x \in U_1) L_{x,R} = L_{x,R'}) \Longrightarrow R = R$ V xϵ U1 L x,R ⟹R= R´. Esto quiere decir que si todas las etiquetas lingüísticas de las conexiones que conectan a una regla R son iguales a las etiquetas de una regla R , es porque $R = R$.

7. Para todas las unidades de regla $R \in U_2$ y todas las unidades $C, C' \in U_3$ se cumple que: $(V(x \in U_1) L_{x,R} = L_{x,R'}) \Longrightarrow R = R$. Esto quiere decir que solo debe haber una conexión que salga de la unidad de regla que conecte una neurona de salida y que su peso sea 1.

8. Para todas las unidades de salida $C \in U_3$, la activación es $o_c = a_c = net_c$.

9. Para todas las unidades de salida $(V(x \in U_1) L_{x,R} = L_{x,R'}) \Longrightarrow R = R$, la entrada net_c es calculada como:

$$net_c = \left(\sum_{R \in U_1} W(R,C) \boxtimes o_R\right) \Big/ \left(\sum_{R \in U_1} W(R,C)\right)$$

10. La base de reglas puede ser definida por el usuario o de forma combinada usuario-preprocesamiento. Dado un conjunto de patrones de aprendizaje $\zeta = \{(p_1, t_1), (p_2, t_2), \dots (p_s, t_s)\}$ que consiste en un patrón de entrada $p \in \mathbb{R}^n$ y una respuesta esperada $t \in \{0,1\}^m$ el algoritmo crea k unidades de regla.

1.5 Nefclass-Q

Nefclass-Q es una herramienta software desarrollada en C++ (Caballero y Salamanca, 2011) tomando como base a NEFCLASS, que permite la creación, entrenamiento y validación de una red NeuroFuzzy, mediante la cual se obtiene un modelo fuzzy que realiza la clasificación de patrones de datos. Está compuesta por cinco librerías y una interfaz gráfica que facilita la creación, aprendizaje y entrenamiento de la red, por medio de un archivo de texto donde se encuentran los datos para el entrenamiento. La herramienta software:

- Posee una interfaz de usuario que facilita el proceso de creación de la red NeuroFuzzy
- Para un problema dado por el usuario, crea la red NeuroFuzzy con una estructura que permite el aprendizaje mediante entrenamiento supervisado.
- Genera una base de conocimiento en forma de reglas "SI-ENTONCES" expresada en términos lingüísticos, fácilmente interpretables por los humanos.
- Es flexible al permitir definir los parámetros iniciales del sistema NeuroFuzzy, tales como las ratas de aprendizaje, la base de conocimiento inicial, el número de conjuntos fuzzy por variable fuzzy de entrada, el número de épocas de entrenamiento y el máximo número de reglas.
- Puede trabajar hasta con 50 variables de entrada.
- Obtiene un *Modelo fuzzy* que permite en la etapa de operación, realizar la clasificación de patrones con las características de los usados en su entrenamiento.

2. ARQUITECTURA DESARROLLADA

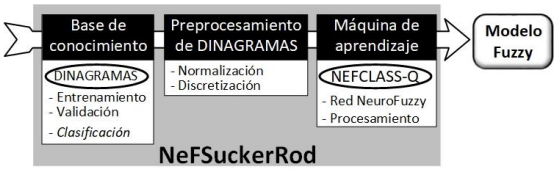

Fig. 7. Arquitectura de NeFSuckerRod

El propósito de NeFSuckerRod es obtener un "*Modelo Fuzzy*" capaz de diagnosticar cualquier falla en un pozo petrolero, utilizando un sistema ~~de~~ inferencia NeuroFuzzy que ha sido entrenado mediante un conjunto de cartas dinamométricas con diferentes fallas. Para la obtención del modelo fuzzy, la arquitectura desarrollada contempló tres fases (Fig. 7).

2.1 Base De Conocimiento

Para un sistema de bombeo mecánico, el conocimiento de los expertos humanos que se pretende digitalizar, se encuentra depositado en cada dinagrama de fondo que se ha diagnosticado identificando una falla a partir de su forma. La base de conocimiento la constituyen entonces una serie de dinagramas ya clasificados, siendo ellos fundamentales para el aprendizaje y la obtención del modelo fuzzy. La base de conocimiento utilizada en este trabajo, se basó en 220 dinagramas de fondo que contenían las 10 fallas de mayor presencia en equipos de bombeo mecánico (Fig. 8), organizados en tres conjuntos de patrones de datos almacenados en tres archivos:

- El conjunto de "*entrenamiento*" proporcionó el conocimiento humano a digitalizar. Cien (100) dinagramas conformaron este grupo.
- El conjunto de "*validación*" fue el soporte para evaluar el conocimiento digitalizado, indicando el nivel de desempeño del entrenamiento y por ende del modelo obtenido. Sesenta (60) dinagramas conformaron este grupo.
- El grupo de "*clasificación* u *operación*" sería el encargado de contener los dinagramas a clasificar cuando el modelo fuzzy obtenido se encuentre en operación. Se usaron sesenta (60) dinagramas, para emular la clasificación ya en *operación* (producción), del modelo fuzzy obtenido.

CODIFICACIÓN asignada	Tipo de FALLA
Clase 1 0 0 0 0 0 0 0 0 0	Interferencia de gas.
Clase 0 1 0 0 0 0 0 0 0 0	Tubería desanclada con golpe de fluido.
Clase 0 0 1 0 0 0 0 0 0 0	Rotura de varilla.
Clase 0 0 0 1 0 0 0 0 0 0	Fuga en la válvula fija.
Clase 0 0 0 0 1 0 0 0 0 0	Fuga en la válvula viajera.
Clase 0 0 0 0 0 1 0 0 0 0	Barril de la bomba doblado.
Clase 0 0 0 0 0 0 1 0 0 0	Buen llenado con tubería anclada.
Clase 0 0 0 0 0 0 0 1 0 0	Agujero en el barril de la bomba.
Clase 0 0 0 0 0 0 0 0 1 0	Ancla de tubería en mal funcionamiento.
Clase 0 0 0 0 0 0 0 0 0 1	Barril de la bomba gastado.

Fig. 8. Fallas (clases) codificadas

2.2 Preprocesamiento de los dinagramas

2.2.1 Normalización

Dos pozos con bombeo mecánico en el mismo campo petrolero, podrían presentar la misma falla y al observar sus dinagramas, la forma sería similar pero los valores de carga y de longitud de recorrido de la bomba podrían ser diferentes (Fig. 9), dado que dependen de factores específicos de cada pozo, tales como: la profundidad de ubicación de la bomba, el diámetro del pistón, el tipo de crudo, la cantidad de gas producido, etc. Teniendo en cuenta que el objetivo es obtener un modelo fuzzy que sea capaz de diagnosticar cualquier dinagrama sin importar el pozo del cual provenga, la normalización es un proceso que ajusta los valores de carga y recorrido de los dinagramas de la base de conocimiento a una escala común, de tal manera que cuando se le presenten a la Máquina de Aprendizaje, esta pueda reconocer la falla y digitalizar el conocimiento a partir de su forma.

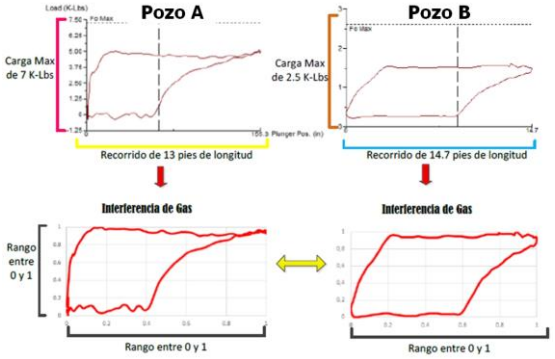

Fig. 9. Normalización de los dinagramas

2.2.2 Discretización

El proceso de convertir conjuntos de datos con atributos continuos, en conjuntos de datos de entrada con atributos discretos, se conoce como *discretización* (Chemielewski y Grzymala, 1996). La discretización (Yousefi1 y Hamilton, 2016) divide un rango numérico continuo en varios intervalos en los que los datos que caen en cada intervalo discretizado, se tratan como describibles por el mismo valor nominal. En este trabajo, cada intervalo y sus respectivos valores, se asociaron a los nodos (variables fuzzy) de la capa de entrada.

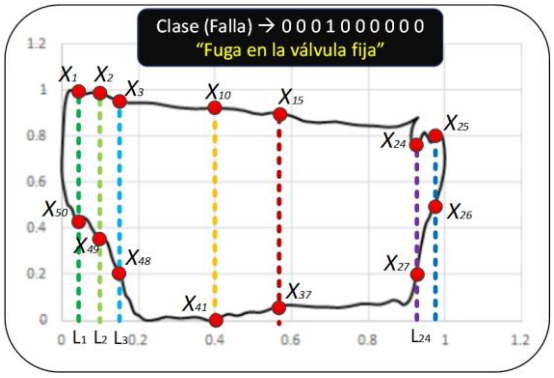

Fig. 10. Dinagrama discretizado

Un dinagrama es una gráfica que contiene una curva cerrada, donde el eje horizontal corresponde a la carrera de la bomba y el eje vertical corresponde a la carga requerida para su desplazamiento instantáneo. Para digitalizar el conocimiento y obtener el modelo fuzzy, NeFSuckerRod utiliza como base el algoritmo de NEFCLASS, el cual no reconoce el dinagrama en su forma natural, es decir no recibe gráficos como entradas. Los nodos fuente de la capa de entrada deben recibir valores numéricos provenientes de cada dinagrama, por lo que se requiere una representación numérica obtenida mediante su "Discretización", permitiendo con ello alimentar la máquina de aprendizaje en sus procesos de entrenamiento y validación.

La capa de entrada se limitó a cincuenta (50) nodos y varias estrategias de discretización fueron evaluadas, para finalmente dividir el dinagrama de fondo, en 25 partes iguales, definiendo a cada segmento de la división como $L(j)$ $(j = 1, j = 2, .., j = 25)$, generando en consecuencia 50 valores de amplitud $X(i)$ $(i = 1, i = 2, .. i = 50)$. Por cada segmento existirán dos valores de amplitud (Fig. 10), uno corresponde a la parte superior del dinagrama (carrera ascendente) y el otro corresponde a la parte inferior (carrera descendente), de forma que: la amplitud X1 y la amplitud X50 corresponden al segmento L1; la amplitud X2 y la amplitud X49 corresponden al segmento L2; ……; la amplitud X25 y la amplitud X26 corresponden al segmento L25.

2.3 Máquina de Aprendizaje

La "Máquina de Aprendizaje" de la arquitectura desarrollada se basó en la herramienta software Nefclass-Q, la cual permitió encontrar los parámetros (base de reglas y conjuntos) de un Modelo Fuzzy (Fig. 11) capaz de diagnosticar una falla en un pozo petrolero a partir de su dinagrama de fondo.

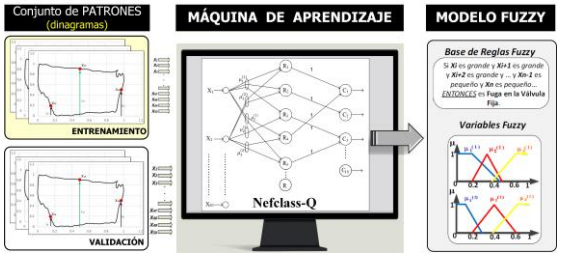

Fig. 11. Máquina de Aprendizaje / Nefclass-Q

2.3.1 red NeuroFuzzy desarrollada (descripción)

Capa de entrada o capa 1: Cada nodo es un módulo de procesamiento de una variable fuzzy, que recibe un valor numérico de amplitud $X(i)$ de los dinagramas discretizados, y entrega los valores de pertenencia (membresía) asociados a los conjuntos fuzzy de la variable fuzzy respectiva (Fig. 12). Tomando en cuenta la discretización, la capa de entrada quedo configurada con 50 módulos de procesamiento, donde cada uno de ellos es una variable fuzzy (entrada) con su propio universo de discurso y etiquetas lingüísticas (conjuntos fuzzy).

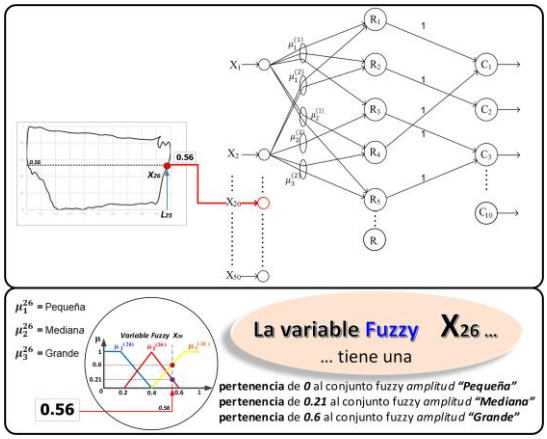

Fig.12. Capa de entrada (módulos)

Capa de Reglas Fuzzy o capa 2: Es la encargada de realizar las inferencias lingüísticas que se almacenan en forma de reglas del tipo SI-ENTONCES, conectando un antecedente con un consecuente. La regla está compuesta por dos partes fundamentales (Fig. 13): a) un antecedente formado por una serie de proposiciones relacionadas con operadores lógicos "Y" (∧); b) un consecuente que indica la CLASE a la que está asociada el antecedente (tipo de falla), y esta correlacionado con su valor de verdad booleano. El algoritmo crea las reglas a partir de cada uno de los ejemplos del conjunto de ENTRENAMIENTO de la base de Conocimiento.

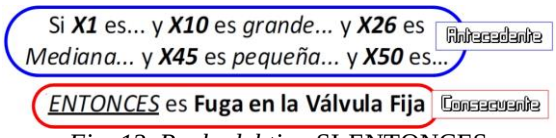

Fig. 13. Regla del tipo SI-ENTONCES

Capa de salida o capa 3: Es la capa que representa las diferentes opciones a la que puede pertenecer un dinagrama, y se encarga de hacer el proceso de cálculo del error. Cada neurona de la capa 2 (capa de reglas) es conectada solamente a una unidad de salida o clase. En este trabajo la capa de salida fue configurada y codificada con diez (10) neuronas (Fig. 8), para identificar y diagnosticar diez fallas (clases).

2.3.2 Procesamiento

El procesamiento fue realizado con Nefclass-Q. La red NeuroFuzzy configurada fue entrenada utilizando el conjunto de dinagramas de entrenamiento, almacenados en un archivo de texto, donde cada patrón de datos (dinagrama) pertenece a una clase o falla determinada. Con los dinagramas discretizados, el primer paso realizado fue la creación de los conjuntos fuzzy para cada neurona de entrada. El siguiente paso fue la creación de las Reglas Fuzzy por medio de la propagación de los datos de *entrenamiento*. Posteriormente las reglas fueron optimizadas por medio del aprendizaje de los parámetros de los conjuntos fuzzy que conforman las reglas. El entrenamiento de la red fue evaluado en la etapa de *validación* utilizando el conjunto de dinagramas de validación. Al contrastar los resultados de la validación y con el propósito de obtener los mejores resultados, se realizó un proceso de sintonización iterativa de la red, modificando parámetros de ella y sometiéndola nuevamente a su entrenamiento. Superadas las etapas de *entrenamiento* y *validación*, la red NeuroFuzzy obtenida es un *Modelo Fuzzy* apto para entrar en la

etapa de *operación* que permita identificar fallas a partir de dinagramas de fondo presentados.

Patrones de datos: para las diferentes etapas, los patrones fueron organizados en archivos, los cuales estructuralmente poseían:

a) un *encabezado* con información especificando los parámetros necesarios para cada etapa (Fig. 14).

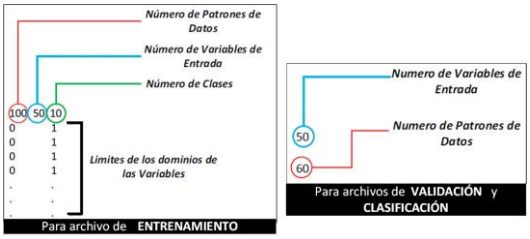

Fig. 14. Encabezados archivos de patrones

b) un *cuerpo* con los valores de amplitud de los dinagramas discretizados. Para los archivos de *entrenamiento* y *validación*, se adiciona al final de cada dinagrama discretizado, un código asociado a cada tipo de falla o "clase" (Fig. 15).

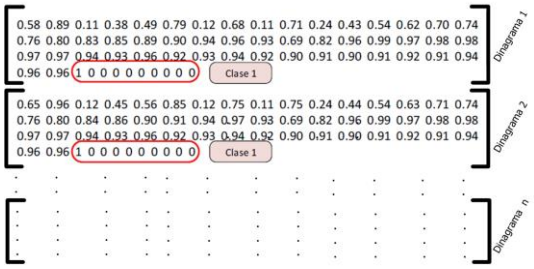

Fig. 15. Encabezados de los archivos de patrones

Cuando un archivo de entrenamiento es cargado, los datos del encabezado permiten crear la red NeuroFuzzy (Fig. 16), al establecer el número de neuronas de entrada y el número de neuronas de salida o clases (tipos de fallas). Los valores discretizados de cada dinagrama son asociados como los valores correspondientes a cada neurona de entrada.

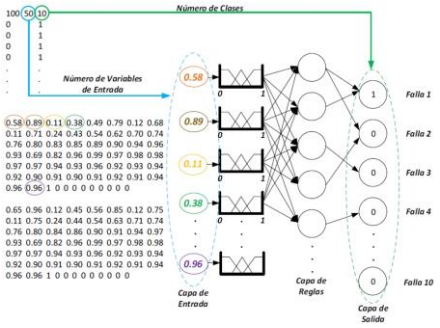

Fig. 16. Creación de la red a partir del archivo.

Creación de los conjuntos Fuzzy: Los pesos sinápticos entre la capa de entrada y la capa oculta son representados por los conjuntos Fuzzy, ellos permiten definir los valores de membresía (pertenencia) que se están propagando en términos lingüísticos. El número de conjuntos Fuzzy (términos lingüísticos) para cada unidad de entrada es uno de los parámetros a sintonizar para obtener el modelo fuzzy; en este trabajo se hicieron pruebas (corridas) definiendo para todas las entradas 3, 5 o 7 conjuntos; a partir del número de términos lingüísticos (conjuntos) definidos, estos son equitativamente distribuidos en el universo de discurso de cada variable (fuzzy) de entrada.

Creación de las Reglas: Las neuronas de la capa oculta representan las Reglas Fuzzy del modelo fuzzy a obtener, y el proceso de creación de ellas se muestra en la figura 17. El número de neuronas de la capa oculta va a ser igual al número de reglas creadas en el proceso o un valor máximo predefinido. Este proceso es muy importante, ya que las reglas permiten a la red NeuroFuzzy adquirir el conocimiento, haciendo propagar los patrones de datos del archivo de *entrenamiento*. Cuando se propaga un patrón, se calcula las pertenencias de cada una de las variables fuzzy a sus respectivos conjuntos, cuando se termina de calcular todas las pertenencias, Nefclass-Q escoge un conjunto por cada variable, teniendo en cuenta que el conjunto tenga la mayor pertenencia. Cuando ya se ha etiquetado lingüísticamente cada variable fuzzy, estas etiquetas se emplean para formar el antecedente de la regla, por último, se verifica que este antecedente exista, de no ser así, se le asigna el consecuente con la respuesta esperada y esta regla es guardada en la base de conocimiento de la red NeuroFuzzy, este proceso se realiza cuando se está propagando cada patrón de datos del archivo de entrenamiento.

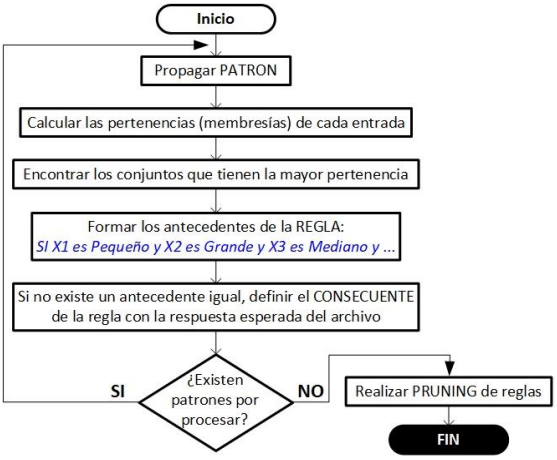

Fig. 17. Creación de las reglas

Con la finalidad de que el conocimiento sea más concreto para el usuario, se realiza un proceso de poda (pruning) de reglas, con el fin de obtener una base de reglas que contenga las más importantes, para esto es posible escoger uno de tres métodos existentes: i) "Todas las Reglas o Simple" el cual deja la base de conocimiento tal como se creó, con el mismo número de reglas; ii) "Mejores Reglas" el cual deja un determinado número de reglas a pedido del usuario, y Nefclass-Q escoge las reglas que más han tenido activación cuando se propagaron los patrones; iii) "Mejores Reglas por clase" el cual solo deja las reglas que hayan tenido más activación para cada clase, donde el número de reglas por clase es definido por el usuario.

Entrenamiento de la Red: Teniendo las reglas creadas, el entrenamiento de la red es un proceso (Fig. 18) en el cual se hace propagar cada uno de los patrones de datos del archivo de *entrenamiento*, para que la red proporcione una respuesta por cada patrón, siendo esta comparada con la respuesta deseada, generando un error, el cual es retro propagado para modificar los pesos de acuerdo con el algoritmo y las fórmulas de NEFCLASS. Una "época" consiste en presentar a la red, todos los patrones de datos entrenamiento; para obtener una red entrenada se requieren muchas "épocas", siendo su número un valor a establecer (sintonizar) por quien está al frente del entrenamiento.

Validación del entrenamiento: Terminado el entrenamiento de la red, y mediante patrones de datos de *validación* existentes en el respectivo archivo, estos se propagan de la misma forma como fue explicado en el proceso de entrenamiento. Después de haber propagado todos los patrones del archivo de validación se comparan los resultados obtenidos de clasificar cada dinagrama del archivo de validación, con la clasificación esperada, y se calcula el porcentaje de los patrones clasificados correctamente.

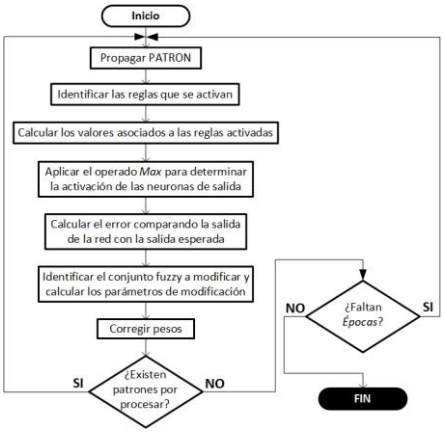

Fig. 18. Entrenamiento de la red

2.3.3 Modelo Fuzzy para diagnóstico

El objetivo de realizar cada etapa del procesamiento anteriormente descrito, persigue obtener un *Modelo Fuzzy* (Fig. 19) que identifique las fallas de un sistema de bombeo mecánico de petróleo, sintetizando y digitalizando el conocimiento del experto humano mediante: a) Reglas del tipo (SI – ENTONCES). b) Variables y conjuntos fuzzy asociados a ellas.

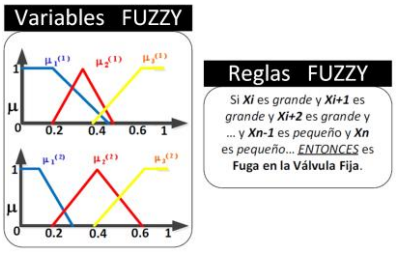

Fig. 19. Modelo fuzzy

3. RESULTADOS

En lo que concierne a los parámetros de la red, no existen referentes establecidos en cuanto a valores y sobre cuáles de ellos se debe ejercer variación, con el fin de obtener los mejores resultados al entrenar la red. Obtener una red entrenada y por ende un *Modelo Fuzzy* adecuado, fue un proceso de sintonía que implicó variar algunos de ellos y analizar su impacto en los resultados. Estos parámetros fueron:

- Número de conjuntos fuzzy (términos lingüísticos) para cada variable de entrada.
- Número máximo de reglas.
- Tipo de pruning aplicado a las reglas.
- Numero de épocas de entrenamiento.
- Ratas de aprendizaje.

Adicionalmente, y con el fin de indagar si el orden en el cual se organizaban los patrones en los conjuntos de datos (archivos) afectaba el desempeño de la red, se crearon dos (2) tipos de archivos para cada una de las etapas (entrenamiento, validación y operación):

Archivo Tipo 1: Los patrones se organizaron en orden ascendente, es decir, patrón 1 pertenece a clase 1, patrón 2 pertenece a clase 2, patrón 3 pertenece a clase 3…, etc.

Archivo Tipo 2: Los patrones se organizaron en forma grupal, es decir, patrón 1, patrón 2, patrón 3…, hasta patrón n, pertenecen a clase 1, patrón 1´, patrón 2´, patrón 3´, … hasta patrón n´, pertenecen a clase 2, y así con los demás.

3.1 Pruebas realizadas

3.1.1 Pruning Simple

En esta prueba se dejaron todas las reglas creadas en el proceso de aprendizaje, ver los resultados en la tabla 1.

Tabla 1: Pruning Simple

Resultados con Pruning simple y Archivo tipo 1				
Coef. Aprendiza	Núm. Conjuntos	Épocas	Validación	Reglas creadas
0.001	3	500	96 %	23
0.01	5	1000	98 %	32
0.001	7	500	93 %	44
0.1	3	3000	96 %	23
0.001	5	2000	95 %	32
0.01	7	1000	96 %	44
0.01	3	4000	96 %	23
0.1	5	500	95 %	32
0.1	7	500	95 %	44

Resultados con Pruning simple y Archivo tipo 2				
Coef. Aprendiza	Núm. Conjuntos	Épocas	Validación	Reglas creadas
0.001	3	500	94 %	23
0.01	5	1000	95 %	32
0.001	7	500	93 %	44
0.1	3	3000	96 %	23
0.001	5	2000	93 %	32
0.01	7	1000	95 %	44
0.01	3	4000	96 %	23
0.1	5	500	93 %	32
0.1	7	500	95 %	44

3.1.2 Pruning de las mejores reglas

En este pruning se dejaron las reglas con mayor activación, ver los resultados en la tabla 2.

Tabla 2: Pruning de las mejores reglas

Resultados con Mejores reglas y Archivo tipo 1				
Coef. Aprendiza	Núm. Conjuntos	Épocas	Validación	Reglas creadas
0.001	7	100	51 %	10
0.01	7	500	90 %	20
0.1	7	1000	96 %	25
0.001	7	2000	95 %	30
0.01	7	3000	96 %	35
0.1	7	4000	96 %	40
0.001	5	100	68 %	15
0.01	5	500	83 %	20
0.1	5	1000	81 %	25
0.01	5	2000	85 %	25
0.1	5	4000	96 %	30
0.001	3	100	76 %	10
0.01	3	500	96 %	20
0.01	3	500	76 %	10
0.1	3	1000	86 %	15

Resultados con Mejores reglas y Archivo tipo 2				
Coef. Aprendiza	Núm. Conjuntos	Épocas	Validación	Reglas creadas
0.001	7	100	36 %	10
0.01	7	500	48 %	20
0.1	7	1000	68 %	25
0.001	7	2000	78 %	30
0.01	7	3000	90 %	35
0.1	7	4000	93 %	40
0.001	5	100	68 %	15
0.01	5	500	79 %	20
0.1	5	1000	83 %	25
0.01	5	2000	83 %	25
0.1	5	4000	95 %	30
0.001	3	100	48 %	10
0.01	3	500	88 %	20
0.01	3	500	48 %	10
0.1	3	1000	66 %	15

3.1.3 Pruning de las mejore reglas por clase

Este pruning tenía como objetivo dejar las reglas con la mejor activación acumulada por clase, ver los resultados en la tabla 3.

Tabla 3: Pruning de las mejores reglas por clase

Resultados con Mejores reglas por clase y Archivo tipo 1					
Núm. Mejo. Reglas por clase	Coef. Aprendiza	Núm. Conjuntos	Épocas	Validación	Reglas creadas
10	0.001	3	500	96 %	23
20	0.01	3	1000	96 %	23
15	0.1	3	3000	96 %	23
5	0.001	5	2000	95 %	32
8	0.01	5	1000	95 %	32
13	0.1	5	2000	95 %	32
10	0.001	7	500	93 %	44
8	0.01	7	1000	96 %	44
13	0.1	7	2000	96 %	44

Resultados con Mejores reglas por clase y Archivo tipo 2					
Núm. Mejo. Reglas por clase	Coef. Aprendiza	Núm. Conjuntos	Épocas	Validación	Reglas creadas
10	0.001	3	500	92 %	23
20	0.01	3	1000	92 %	23
15	0.1	3	3000	92 %	23
5	0.001	5	2000	91 %	32
8	0.01	5	1000	91 %	32
13	0.1	5	2000	91 %	32
10	0.001	7	500	93 %	44
8	0.01	7	1000	96 %	44
13	0.1	7	2000	96 %	44

3.2 Análisis de las pruebas realizadas

A partir de los múltiples entrenamientos realizados en las diferentes pruebas, se evidenció que los mejores resultados se obtuvieron al aplicar pruning

simple, con cinco (5) conjuntos fuzzy por variable de entrada, un coeficiente de aprendizaje de 0.01, realizando el entrenamiento con los dinagramas organizados de forma ascendente (archivo tipo 1); logrando un porcentaje de validación del 98%.

El modo en que se organizan los patrones afecta el resultado de validación de la red creada, siendo los archivos organizados de forma ascendente (tipo 1) los que arrojan mejores resultados.

Cuando se utiliza un número grande de conjuntos fuzzy para cada variable de entrada, en el pruning simple se generan un gran número de reglas y el resultado de la validación no es bueno.

El valor adecuado del coeficiente de aprendizaje depende del problema (el dominio de las variables), pero cuando se utiliza un coeficiente de aprendizaje pequeño el entrenamiento de la red es menos confiable.

Al realizar el pruning con "Mejores Reglas", entre más pequeño es el coeficiente de aprendizaje utilizado, el porcentaje de validación disminuye notoriamente.

4. CONCLUSIONES

La arquitectura desarrollada en este trabajo (NeFSuckerRod), se constituye en una muy buena alternativa para el diagnóstico de sistemas de bombeo mecánico de petróleo. En paralelo con este trabajo, se desarrolló una plataforma software que tomando el modelo fuzzy obtenido, realiza su conversión a código en lenguaje C para ser incorporado a un sistema embebido (Zarate, 2016). Al integrar los resultados obtenidos en este trabajo, con la arquitectura hardware/software de pozo inteligente propuesta en (Meneses y Meneses, 2020), se puede identificar de manera automática y en tiempo real, las condiciones de fondo de pozo (fallas) en los sistemas de bombeo mecánico de petróleo, a partir de los dinagramas de fondo.

RECONOCIMIENTO

Este trabajo hizo parte del proyecto de investigación 8556 "*Desarrollo de un prototipo de pozo inteligente para Campo Escuela Colorado*", financiado por la Vicerrectoría de Investigación y Extensión de la UIS (Universidad Industrial de Santander).

REFERENCIAS

Caballero F. y Salamanca (2011). *Herramienta de aplicación software para clasificación de patrones de datos implementando una arquitectura Neuro-fuzzy.* Universidad Industrial de Santander (UIS), Bucaramanga (Colombia). Tesis de pregrado.

Chemielewski M., y Grzymal J. (1996). *Global discretization of continuous attributes as preprocessing for machine learning*. International Journal of Approximate Reasoning, pp319–331.

DaCunha, J.J. y Gibbs, S.G. (2007). *Modeling a Finite-Length Sucker Rod using the Semi-Infinite Wave Equation and a Proof to Gibbs' Conjecture*. SPE 108762.

de Best L. (2012). *Shell's Smart Fields – Sustaining and Accelerating Benefits from Intelligent Fields*. Intelligent Energy International, Utrecht, 150407-MS SPE Conference Paper.

Dickinson R. y Jennings J. (1990). *Use of pattern-recognition techniques in analyzing downhole dynamometer cards*, SPE Production Engineering, **Vol. 5**, No. 2, SPE-17313-PA.

Eickmeier, J.R. (1967). *Diagnostic Analysis of Dynamometer Cards*. SPE 1643. Journal of Petroleum Technology, January 1967, pp 97-106.

Gabor T. (2015). *Sucker-Rod Pumping Handbook*. Gulf Professional Publishing.

Gibbs, S. y Nelly, A. (1966). *Computer diagnosis of down hole conditions in sucker rod pumping wells.* Journal of Petroleum Technology, **Vol. 18**, No.1, SPE1165PA.

Gilbert, W.E. (1936*). An Oil-Well Pump Dynagraph.* Drilling and Production Practice, American Petroleum Institute. New York.

Gómez, A.E., Archila, J.F. y Meneses, J.E. (2013). *Adquisición y tratamiento de señales de un acelerómetro triaxial MEMS, para la medición del desplazamiento de una extremidad inferior*, Revista Colombiana de Tecnologías de Avanzada **Vol. 1** No. 21, pp. 113-118.

McCoy J. y Podio L. (1995). *Method and Apparatus for measuring pumping rod position and other aspects of a pumping system by use of an accelerometer.* Patent US5406482.

Meneses J.E. y Meneses D.P. (2020). *Arquitectura hardware/software para un prototipo de pozo inteligente en un campo petrolero maduro,* Revista Colombiana de Tecnologías de Avanzada **Vol. 2**, No. 36, pp. 109-121

Meneses J.E., García J.D. y Ferreira D.A. (2015). *Acelerómetros MEMS en el desarrollo de pozos y campos petroleros inteligentes*. Revista Colombiana de Tecnologías de Avanzada **Vol. 2**, No. 26, pp. 128-135.

Nauck, D., Klawon, F.; Kruse R. (1997) *Foundations of Neuro-Fuzzy Systems*. J. Wiley & Sons

Nauck, D., Kruse, R. (1995). *Nefclass - a Neuro-fuzzy Approach for the Classification of Data*. Proceedings of the 1995 ACM symposium on Applied computing.

Nauck, D. (1994) *A fuzzy perceptron as a generic model for neuro-fuzzy approaches*. Proc. Fuzzy-Systems'94, 2nd GI-Workshop, Munich, Siemens Corporation.

Sanchez J. P., Festini D. y Bel O. (2007). *Beam Pumping System Optimization Through Automation*. Latin American & Caribbean Petroleum Engineering Conference, Buenos Aires (Argentina), SPE 108112.

Sletcha B., Vivas, C., K. Saleh, F., Ghalambor A. y Salehi S. (2020). *Digital Oilfield: Review of Real-time Data-flow Architecture for Upstream Oil and Gas Rigs*, International Conference and Exhibition on Formation Damage Control, Lafayette, (Louisiana, USA), 199298-MS SPE Conference Paper.

Yousefi J., y Hamilton-Wright A. (2016). *Classification Confusion within NEFCLASS Caused by Feature Value Skewness in Multi-dimensional Datasets*. Proceedings of the 8th International Joint Conference on Computational Intelligence (IJCCI 2016) - Volume 2: FCTA, pages 21-29 ISBN: 978-989-758-201-1. Porto (Portugal) Nov 9 al 11.

Zarate C. (2016). *Preprocesador de fuzzy inference system (FIS) y motor de inferencia difusa para la plataforma de desarrollo embebida Freescale TWR-K70F120M.* Universidad Industrial de Santander (UIS), Bucaramanga (Colombia). Tesis de pregrado.