Polibits

ISSN: 1870-9044

polibits@nlp.cic.ipn.mx

Instituto Politécnico Nacional

México

Garro, Beatriz A. Garro; Sossa, Humberto; Vazquez, Roberto A.

Diseño Automático de Redes Neuronales Artificiales mediante el uso del Algoritmo de Evolución Diferencial (ED)

Polibits, vol. 46, 2012, pp. 13-27

Instituto Politécnico Nacional

Distrito Federal, México

# Diseño Automático de Redes Neuronales Artificiales mediante el uso del Algoritmo de Evolución Diferencial (ED)

Beatriz A. Garro, Humberto Sossa, Roberto A. Vazquez

***Resumen*—En el área de la Inteligencia Artificial, las Redes Neuronales Artificiales (RNA) han sido aplicadas para la solución de múltiples tareas. A pesar de su declive y del resurgimiento de su desarrollo y aplicación, su diseño se ha caracterizado por un mecanismo de prueba y error, el cual puede originar un desempeño bajo. Por otro lado, los algoritmos de aprendizaje que se utilizan como el algoritmo de retropropagación y otros basados en el gradiente descenciente, presentan una desventaja: no pueden resolver problemas no continuos ni problemas multimodales. Por esta razón surge la idea de aplicar algoritmos evolutivos para diseñar de manera automática una RNA. En esta investigación, el algoritmo de Evolución Diferencial (ED) encuentra los mejores elementos principales de una RNA: la arquitectura, los pesos sinápticos y las funciones de transferencia. Por otro lado, dos funciones de aptitud son propuestas: el error cuadrático medio (MSE por sus siglas en inglés) y el error de clasificación (CER) las cuales, involucran la etapa de validación para garantizar un buen desempeño de la RNA. Primero se realizó un estudio de las diferentes configuraciones del algoritmo de ED, y al determinar cuál fue la mejor configuración se realizó una experimentación exhaustiva para medir el desempeño de la metodología propuesta al resolver problemas de clasificación de patrones. También, se presenta una comparativa contra dos algoritmos clásicos de entrenamiento: Gradiente descendiente y Levenberg-Marquardt.**

***Palabras clave*—Evolución diferencial, evolución de redes neuronales artificiales, clasificación de patrones.**

## Automatic Design of Artificial Neural Networks by means of Differential Evolution (DE) Algorithm

***Abstract*—Artificial Neural Networks (ANN) have been applied in several tasks in the field of Artificial Intelligence. Despite their decline and then resurgence, the ANN design is currently a trial-and-error process, which can stay trapped in bad solutions. In addition, the learning algorithms used, such as back-propagation and other algorithms based in the gradient descent, present a disadvantage: they cannot be used to solve non-continuous and multimodal problems. For this reason, the application of evolutionary algorithms to automatically designing ANNs is proposed. In this research, the Differential Evolution (DE) algorithm finds the best design for the main elements of ANN: the architecture, the set of synaptic weights, and the set of transfer functions. Also two fitness functions are used (the mean square error—MSE and the classification error—CER) which involve the validation stage to guarantee a good ANN performance. First, a study of the best parameter configuration for DE algorithm is conducted. The experimental results show the performance of the proposed methodology to solve pattern classification problems. Next, a comparison with two classic learning algorithms—gradiant descent and Levenberg-Marquardt—are presented.**

***Index Terms*—Differential evolution, evolutionary neural networks, pattern classification.**

Manuscript received April 22, 2012. Manuscript accepted for publication July 20, 2012.

Beatriz A. Garro y Humberto Sossa pertenecen al Centro de Investigación en Computación del Instituto Politécnico Nacional, CIC-IPN, Av. Juan de Dios Bátiz s/n, esquina con Miguel de Othón de Mendizábal, 07738, Ciudad de México, México (Email: bgarrol@ipn.mx, hsossa@cic.ipn.mx).

Roberto A. Vazquez pertenece al Grupo de Sistemas Inteligentes, Facultad de Ingeniería, Universidad la Salle, Benjamín Franklin 47, Col. Hipódromo Condesa, 06140, Ciudad de México, México (Email: ravem@lasallistas.org.mx).

## I. Introducción

LAS REDES neuronales artificiales han sido por muchos años una herramienta indispensable en el área de la Inteligencia Artificial debido a su aplicación satisfactoria en lo concerniente a la clasificación de patrones y la predicción de series de tiempo, entre otras problemáticas. Estos sistemas se basan en el comportamiento que tiene la red neuronal biológica del cerebro. Ramón y Cajal [1], fue el primer científico en demostrar que el sistema nervioso se compone de células individuales llamadas neuronas, las cuales se conectan entre ellas, creando un sistema complejo de comunicación y cuyo procesamiento de información es hasta el día de hoy, un misterio científico.

Una RNA emplea un mecanismo de aprendizaje en la etapa de entrenamiento, donde se optimiza una función que evalúa la salida de la red; con ello se determina la eficiencia del aprendizaje. Después de ser entrenada, la RNA puede ser utilizada para resolver algún problema con información totalmente desconocida pero, que puede dar un veredicto correcto conforme lo aprendido. Esta etapa recibe el nombre de generalización.

Los pesos sinápticos, las funciones de transferencia, el número de neuronas y el tipo de arquitectura o topología (determinado por las conexiones entre neuronas) son escenciales y determinantes en el desempeño de una RNA. Por este motivo, es importante seleccionar los parámetros del diseño de manera adecuada para obtener la mejor

eficiencia en la red neuronal. Sin embargo, expertos en el área generan arquitecturas en un procedimiento de prueba y error, seleccionando de entre ellas, aquella que otorga el mejor desempeño. Esto puede provocar que no se explore adecuadamente otras formas de diseñar que pueden otorgar mejores resultados. Por otra parte, para entrenar la red neuronal se selecciona un tipo de algoritmo que ajusta los pesos sinápticos hasta otorgar el comportamiento deseado de la red. El más utilizado es el algoritmo de retropropagación (back-propagation BP) [2], el cuál se basa en la técnica del gradiente descendiente. Las técnicas de ajuste que se basan en el cálculo de derivadas como BP, presentan un problema: no pueden ajustar información proveniente de problemas que son no continuos. Por otro lado, sólo pueden trabajar con problemas donde sólo existe un valor óptimo (o en el peor caso, el mejor) esto quiere decir que al presentarse problemas multimodales (donde se presentan varios valores óptimos) el ajuste podría no ser el mejor ya que podría hacercarse a una falsa solución.

Por tal motivo, el deseo de construir un diseño adecuado que resuelva problemas del mundo real giró hacia otras técnicas inspiradas en la naturaleza como los llamados algoritmos evolutivos [3]. Estos algoritrmos heurísticos se utilizan para resolver problemas no lineales que no pueden ser resueltos por técnicas clásicas de optimización, donde el espacio de búsqueda es muy grande, combinatorial y/o multimodal. Estas técnicas se basan en conceptos biológicos como lo es el Neo-Darwinismo, teoría formada por el pensamiento de Charles Darwin sobre evolución de las especies [4], el pensamiento de August Weismann sobre el plasma germinal responsable de la herencia [5] y el pensamiento de Gregor Mendel sobre la teoría de las leyes de la herencia [6].

Los algoritmos evolutivos son procesos que mezclan un concepto de individuos distribuidos en un espacio de búsqueda determinado. La población de individuos no es más que un conjunto de posibles soluciones que, en un determinado tiempo, deben converger a la solución óptima (si se conoce) o simplemente converger a la mejor solución (se espera que sea la más cercana a la óptima). Estos individuos cambian (evolucionan) al realizar operaciones de cruza o mutación para mejorar su desempeño mientras se incrementa el tiempo. Para indicar cuál individuo representa la mejor solución se deben evaluar cada una de las soluciones o individuos en una función de aptitud o función objetivo. La función de aptitud es diseñada de tal manera que se realice una optimización en el desempeño de los individuos y se indique que tan bueno es cada uno para resolver un determinado problema. El proceso termina al guardar el mejor individuo que genera la mejor evaluación en la función de aptitud.

Estas técnicas de optimización han sido empleadas para ajustar los pesos sinápticos de una RNA debido a su eficiencia para resolver problemas complejos. Algunas investigaciones como [7], [8] proponen una modificación de Evolución diferencial para ajustar los pesos sinápticos de una red neuronal multicapa. En [9] tres arquitecturas con diferentes técnicas de entrenamiento, entre ellas ED, son aplicados a la predicción del clima. En [10], los autores utilizan ED para el ajuste de los pesos sinápticos de una RNA, así mismo utilizan la técnica llamada optimización por enjambre de partículas (PSO por sus siglas en inglés), proporcionando como métrica el error cuadrático medio (MSE por sus siglas en inglés). La incursión del algoritmo de Evolución diferencial no ha sido muy explorado para el diseño de RNA. Sin embargo, otros algoritmos evolutivos y algoritmos bioinspirados han sido aplicados en el entrenamiento de las RNA y la selección del número de neuronas en un número de capas especificado por el diseñador [11]. En [12] los autores presentan el diseño de una RNA generada por el algoritmo ED, el cuál evoluciona al mismo tiempo la arquitectura (topología), pesos sinápticos y funciones de transferencia utilizando como métrica el error cuadrático medio.

En esta investigación se decribe la metodología que permite diseñar de manera automática la arquitectura, el conjunto de pesos sinápticos y las funciones de transferencia por cada neurona que componene a una RNA. El diseño será generado al aplicar el algoritmo evolutivo llamado Evolución diferencial, el cuál evaluará las soluciones mediante dos funciones de aptitud. La primera función toma en cuenta el error cuadrático medio (mean square error-MSE) y la etapa de validación la cuál impide generar redes neuronales con el problema de sobreaprendizaje. La segunda utiliza el error de clasificación (CER) también considerando la validación. Estas funciones de aptitud, son muy adecuadas ya que los problemas que se quieren resolver son problemas de clasificación de patrones.

La estructura del escrito está dividida en las siguientes secciones: la sección 2 describe los conceptos básicos de una RNA; en la sección 3 se introduce a la técnica de Evolución diferencial; la descripción del diseño automático se detalla en la sección 4; la sección 5 describe cómo se obtiene la salida general de la RNA; los resultados experimentales se encuentran en la sección 6 y finalmente en la sección 7 se pueden encontrar las conclusiones que cierran esta investigación.

## II. Redes neuronales artificiales

La red neuronal artificial (RNA) basada en el sistema neuronal biológico, es un sistema computacional que permite realizar un mapeo de un conjunto de datos o patrones de entrada a un conjunto de salida. En [13], Kohonen describe: “Las redes neuronales artificiales son redes interconectadas masivamente en paralelo de elementos simples (usualmente adaptativos) y con organización jerárquica, las cuales intentan interactuar con los objetos del mundo real del mismo modo que lo hace el sistema nervioso biológico”. Las RNA se componen por unidades básicas llamadas neuronas, las cuales estan conectadas entre sí y dependiendo de la capa a la que pertenezcan pueden modificar la información que reciben o simplemente la envían tal y como la recibieron a otras neuronas con las que tienen conexión sináptica.

Las neuronas se encuentran organizadas por capas. Algunas neuronas componen la capa de entrada, la cuál, se encarga de recibir la información del entorno o problema a resolver. Otras neuronas forman la capa de salida las cuales, entregan un patrón asociado al patrón de la entrada. Las neuronas restantes constituyen la llamada capa oculta donde la información es procesada, enviada a otras neuronas y evaluada en funciones de transferencia que entregarán una salida por cada neurona. Cada conexión indica un peso sináptico y está representado por un valor numérico determinado. Dependiendo del tipo de conexión que se tenga, es el tipo de flujo de información. Puede ser hacia adelante, cuando la información fluye desde la capa de entrada hacia la de salida (flujo unidireccional) [14] o puede ser recurrente [15] es decir, cuando la información fluye en ambos sentidos con presencia de posibles retroalimentaciones. La salida de la red neuronal está dada por las neuronas de la capa de salida, las cuales conjuntan toda la información procedente de capas anteriores. Dicha salida permite evaluar el diseño de la red neuronal.

Para que una RNA resuelva un determinado problema, su diseño necesita ser evaluado mediante una etapa de entrenameinto en la cuál se lleva a cabo el aprendizaje. Esta etapa consiste en alimentar la red con patrones que codifican un problema específico. Esta información pasa por cada capa de la red en donde es procesada por los pesos sinápticos y después se transforma por medio de funciones de transferencia. Este hecho se da hasta alcanzar la capa de salida. Si la métrica que se utiliza para medir la salida no es la deseada, los pesos sinápticos cambian con ayuda de una regla de aprendizaje con el fin de volver al paso anterior y así generar una mejor salida que la anterior.

Existen varios tipos de aprendizaje de una RNA [16]. El aprendizaje supervisado será utilizado en esta investigación. El aprendizaje supervisado consite en asociar un conjunto de patrones de entrada con un correspondiente patrón deseado el cuál, es conocido. De tal manera que se puede supervisar si la salida de la RNA es la deseada o no.

Cuando la red neuronal ya ha aprendido, es de sumo cuidado conocer si no aprendió de más, es decir que se haya convertido en un sistema experto en la resolución del problema con el que se entrenó. Al aprender de manera experta cada entrada, el sistema neuronal será incapás de reconocer alguna entrada contaminada con algún error y no podrá reconocer nueva información que determina el mismo problema a resolver. Este problema se resuelve utilizando una etapa de validación [17] la cuál consite en tomar un conjunto de patrones diferentes al conjunto de entrenamiento, y probarlos con la red entrenada. Si el error que se genera es menor al error en el aprendizaje, la RNA continúa ajustando sus pesos sinápticos, pero, si el error que se genera con el conjunto de validación es mayor al error generado por el entrenamiento, la etapa de aprendizaje debe ser suspendida para evitar el sobreaprendizaje.

Después de ser entrenada y probada con el conjunto de validación, la RNA estará lista para recibir patrones de datos diferentes a los utilizados durante el entrenamiento y así realizar una generalización eficiente, la cuál determina qué tan bueno fue el aprendizaje de la red y que tan robusta para resolver el problema, en este caso de clasificación de patrones.

Para entender más a detalle cómo opera una RNA a continuacíon se explica el funcionamiento de sus elementos esenciales [18].

### *A. Entradas de la RNA*

El conjunto de entradas $x_j(t)$ de la RNA es un conjunto de patrones los cuales codifican la información de un problema que se quiere resolver.

Las variables de entrada y de salida pueden ser binarias (digitales) o continuas (analógicas), dependiendo del modelo y la aplicación. Por ejemplo un perceptrón multicapa (MLP Multilayer Perceptron, por sus siglas en inglés) puede trabajar con ambos tipos de señales. En el caso de salidas digitales las señales se pueden representar por $0, +1$, en el caso de las salidas analgógicas la señal se da en un cierto intervalo.

### *B. Pesos sinápticos*

Los pesos sinápticos $w_{ij}$ de la neurona $i$ son variables relacionadas a la sinapsis o conexión entre neuronas, los cuales representan la intensidad de iteracción entre la neurona presináptica $j$ y la postsináptica $i$. Dada una entrada positiva (puede ser del conjunto de datos de entrada o de la salida de otra neurona), si el peso es positivo tenderá a exitar a la neurona postsináptica, si el peso es negativo tenderá a inhibirla.

### *C. Regla de propagación*

La regla de propagación permite obtener, a partir de las entradas y los pesos sinápticos, el valor del potencial postsináptico $h_i$ de la neurona $i$ en función de sus pesos y entradas.

$$h_i(t) = \sigma_i(w_{ij}, x_j(t)) \tag{1}$$

La función más habitual es de tipo lineal, y se basa en la suma ponderada de las entradas con los pesos sinápticos

$$h_i(t) = \sum_j w_{ij} x_j \tag{2}$$

que también puede interpretarse como el producto escalar de los vectores de entrada y pesos

$$h_i(t) = \sum_j w_{ij} x_j = \mathbf{w}_i^T \mathbf{x} \tag{3}$$

### D. Función de transferencia

La función de transferencia $f$ de la neurona $i$ proporciona el estado de activación actual $a_i(t)$ a partir del potencial postsináptico $h_i(t)$ y del propio estado de activación anterior $a_i(t-1)$

$$a_i(t) = f_i(a_i(t-1), h_i(t)) \tag{4}$$

Sin embargo, en muchos modelos de RNA se considera que el estado actual de la neurona no depende de su estado anterior, si no únicamente del actual

$$a_i(t) = f_i(h_i(t)) \tag{5}$$

### E. Función de salida

La función de salida proporciona la salida global de la neurona $y_i(t)$ en función de su estado de activación actual $a_i(t)$. Muy frecuentemente la función de salida es simplemente la identidad $F(x) = x$ de tal modo que el estado de activación de la neurona se considera como la propia salida de la red neuronal

$$y_i(t) = F_i(a_i(t)) = a_i(t) \tag{6}$$

## III. Evolución diferencial

En 1994 surgió un adaptativo y eficiente esquema: el algoritmo de Evolución diferencial, propuesto por Kenneth Price y Rainer Storn. Este algoritmo se utilizó para la optimización global sobre espacios continuos [19]. Debido a su capacidad de exploración sobre un espacio de búsqueda, dado un problema, el algoritmo de Evolución diferencial (DE por sus siglas en inglés) evita quedar atrapado en mínimos locales. Este algoritmo tiene pocos parámetros y converge más rápido al óptimo en comparación con otras técnicas evolutivas. Todas estas características convierten a este algoritmo en una excelente técnica para la optimización de problemas no diferenciables. La idea detrás de esta técnica de optimización es generar vectores de prueba.

Dado una población de vectores $\vec{\mathbf{x}_i} \in \mathbb{R}^D, i = 1, ..., M$ en un espacio multidimensional $D$, el algoritmo consiste, en eligir de manera aleatoria un vector objetivo $\mathbf{x}_i$ y un vector base $\mathbf{x}_{r_3}$, donde $r$ es un número aleatorio entre $[1, M]$. Por otro lado, se deben elegir aleatoriamente dos miembros de la población $\mathbf{x}_{r_1}$ y $\mathbf{x}_{r_2}$, y se realiza una diferencia entre ellos. A continuación, el resultado es operado por un factor constante, denotado por $F$, así se obtendrá un vector ponderado. Inmediatamente después, el vector ponderado y el vector base son sumados. El nuevo vector que surge, se le llamará vector mutante $\mathbf{u}_i$.

Finalmente se realiza la operación de cruza, la cuál involucra una comparación (variable por variable) entre el vector mutante y el vector objetivo. De la operación de cruza se genera un nuevo vector llamado vector de prueba. La comparación consiste en una simple regla: si un número aleatorio es menor que el factor de cruza $CR$ entonces, la variable del vector que se elige es la del vector mutante; si no, entonces se elige la variable del vector objetivo, así el vector prueba será una mezcla de variables del vector mutante y el vector objetivo. Finalmente el último paso es la selección del mejor vector (aquél con la mejor aptitud, según sea el tipo de optimización). Esta selección involucra comparar la aptitud del vector objetivo y la del vector prueba.

Existen varias estrategias de ED [20]. La estrategia descrita en esta sección es la estrategia técnicamente llamada "DE/rand/1/bin" cuyo pseudocódigo se muestra en el Algoritmo 1 tomado de [21]. Esta nomenclatura cambia dependiendo del tipo de estrategia que se esté implementando. Las diferentes estrategias varían al cambiar la siguiente nomenclatura, donde DE/*x*/*y*/*z* toma las siguientes variables: *x* se refiere a cómo será elegido el vector objetivo, puede ser aleatorio o el mejor de la población. *y* se refiere a cuántos pares de vectores se tomarán para realizar la diferencia; puede ser un par y sumarle un tercer vector (vector base) o puede ser dos pares cuya respectiva diferencia se sume a un quinto vector (vector base).

El tipo de cruza se representa por *z*. Ésta puede ser del tipo *bin* (cruza binomial) en donde para cada variable dada una probabilidad, se hereda la información de uno u del otro vector o puede ser del tipo *exp* (cruza exponencial) en donde dado un número aleatorio entre $(0, 1)$, si dicho número es mayor a $CR$ entonces se suspende la cruza [22].

## IV. Diseño automático de una RNA mediante ED

Para aplicar el algoritmo de Evolución diferencial es necesario codificar el individuo o la solución con la información que se requiere. El algoritmo ED generará una población de dichas soluciones que evolucionarán en un número de generaciones (iteraciones) y se evaluará cada una de ellas en una función de aptitud. Dicha función de aptitud nos indicará que tan buena es la solución, guardando la mejor al finalizar la ejecución del mismo.

La descripción de cada elemento en el algoritmo evolutivo se explica a continuación.

### A. Codificación del individuo

Un individuo representa una solución. La codificación de ese individuo consiste en contar con la información necesaria para el diseño de una red neuronal artificial. Como se mencionó en las restricciones de esta investigación, la metodología será aplicada por el momento a problemas de clasificación de patrones.

En general el problema a resolver se puede describir de la siguiente manera.

Dado un conjunto $X$ con $p$ patrones que definen un problema de clasificación definido por $X = \left\{\mathbf{x}^1, ..., \mathbf{x}^p\right\}, \mathbf{x} \in \mathbb{R}^n$, y dado un conjunto $D$ con $p$ patrones deseados que definen la clase a la que pertenece cada patrón definido por $D = \left\{\mathbf{d}^1, ..., \mathbf{d}^p\right\}, \mathbf{d} \in \mathbb{R}^m$; encontrar una RNA, cuyo diseño está representado por una matriz $\mathbf{W}$ donde $W \in \mathbb{R}^{q\times(q+3)}$ tal

**Algorithm 1** Pseodocódigo de Evolución diferencial al aplicar la estrategia "DE/rand/1/bin". $CR$ es un número entre $(0, 1)$, $MAXITER$ es el número máximo de iteraciones, $G$ es una iteración específica, $M$ es el número total de individuos en la población , $randint(1, D)$ es una función que regresa un número entero entre 0 y $D$. $rand_j[0, 1)$ es una función que regresa un número real entre $(0, 1)$. Ambas funciones basadas en una probabilidad de distribución uniforme.

$G = 0$
Crear una población inicial aleatoria $\vec{\mathbf{x}}_{i,G} \forall i, i = 1, \ldots, M$
Evaluar $f(\vec{\mathbf{x}}_{i,G}) \forall i, i = 1, \ldots, M$
**for** $G = 1$ hasta $MAXITER$ **do**
  **for** $i = 1$ hasta $M$ **do**
    Seleccionar aleatoriamente $r_1 \neq r_2 \neq r_3$
    $j_{rand} = randint(1, D)$
    **for** $i = 1$ hasta $M$ **do**
      **if** $rand_j[0, 1) < CR$ o $j = j_{rand}$ **then**
        $\mathbf{u}_{i,j,G+1} = \mathbf{x}_{r_3,j,G} + F \cdot (\mathbf{x}_{r_1,j,G} - \mathbf{x}_{r_2,j,G})$
      **else**
        $\mathbf{u}_{i,j,G+1} = \mathbf{x}_{i,j,G}$
      **end if**
    **end for**
    **if** $f(\vec{\mathbf{u}}_{i,G+1}) \leq f(\vec{\mathbf{x}}_{i,G})$ **then**
      $\vec{\mathbf{x}}_{i,G+1} = \vec{\mathbf{u}}_{i,G+1}$
    **else**
      $\vec{\mathbf{x}}_{i,G+1} = \vec{\mathbf{x}}_{i,G}$
    **end if**
  **end for**
  $G = G + 1$
**end for**

que una función $f$ sea optimizada $(min)f(D, X, W)$, donde $q$ es el número de neuronas.

### B. Individuo

El individuo que representa el diseño de una RNA, está dado por una matriz **W**. Esta matriz está compuesta por tres partes: la topología (*T*), los pesos sinápticos (*SW*) y las funciones de transferencia (*TF*), tal como se muestra en la Figura 1.

El tamaño del individuo depende de un número máximo de neuronas ($MNN$), el cuál está definido por $q$. En esta investigación se desarrolló una ecuación para obtener el $MNN$, la cuál depende del problema a resolver. Ésta se encuentra definida a continuación:

$$q = (m + n) + \left(\frac{m + n}{2}\right) \tag{7}$$

donde $n$ es la dimensión del vector de los patrones de entrada y $m$ es la dimensión del vector de los patrones deseados.

Debido a que la matriz está compuesta por tres diferentes informaciones, se consideró tres rangos diferentes para cada una. En el caso de la topología, el rango se encuentra

$$\underbrace{\begin{bmatrix} w_{1,1} \\ \vdots \\ w_{1,1} \end{bmatrix}}_{T} \underbrace{\begin{bmatrix} w_{1,2} & \cdots & w_{1,MNN+1} \\ \vdots & \ddots & \vdots \\ w_{MNN,2} & \cdots & w_{MNN,MNN+1} \end{bmatrix}}_{SW} \underbrace{\begin{bmatrix} w_{1,MNN+2} \\ \vdots \\ w_{MNN,MNN+2} \end{bmatrix}}_{TF}$$

Fig. 1. Representación del individuo que codifica la topología (T), los pesos sinápticos (SW) y las funciones de transferencia TF).

entre $\left[1, 2^{MNN} - 1\right]$, los pesos sinápticos tienen un rango de $[-4, 4]$ y para las funciones de transferencia el rango es $[1, nF]$, donde $nF$ es el número total de funciones de transferencia a utilizar. Al generar los individuos, todas las matrices **W** están compuestas de valores reales en sus respectivos rangos y al momento de decodificar la información para ser evaluados en la función de aptitud, tanto la arquitectura y la función de transferencia sufren un redondeo del tipo $\lfloor x \rfloor$.

### C. Arquitectura y pesos sinápticos

Para poder evaluar los diseños de las RNA, es necesario decodificar la información del individuo. La primera información a decodificar es la topología o arquitectura (*T*), la cual se evalúa con los pesos sinápticos (*SW*) y las funciones de transferencia (*TF*) codificados en la misma matriz.

Para hacer válida una topología de RNA para la metodología propuesta, se deben de seguir ciertas reglas. Las RNA generadas están compuestas de tres capas: la capa de entrada, la capa oculta y la capa de salida y las reglas de conexión entre las neuronas de cada capa siguen las siguientes condiciones.

Donde $ILN$ es el conjunto de $I$ neuronas que componen la capa de entrada, $HLN$ es el conjunto de $J$ neuronas pertenecientes a la capa oculta y $OLN$ es el conjunto de $K$ neuronas, las cuales pertenecen a la capa de salida.

*Primera regla (para las neuronas de la capa de entrada-ILN)*: La neurona $ILN_i$, i = 1,..., I sólo puede enviar información a las neuronas de la capa oculta $HLN_j$ y a las neuronas de la capa de salida $OLN_k$.

*Segunda regla (para las neuronas de la capa oculta-HLN)*: La neurona $HLN_j$, j = 1,..., J sólo puede enviar información a las neuronas de la capa de salida $OLN_k$ y a las neuronas de la capa oculta $HLN_j$ pero, para ésta última con una restricción. Para la neurona $HLN_j$ sólo puede haber conexión con las neuronas del tipo $HLN_{j+1}, ..., HLN_J$.

*Tercera regla (para las neuronas de la capa de salida-OLN)*: La neurona $OLN_k$, k = 1,..., K sólo puede enviar información a otras neuronas de su misma capa pero, con una restricción. Para la neurona $OLN_k$ sólo puede existir conexión con neuronas del tipo $OLN_{k+1}, ..., OLN_K$.

Para decodificar la arquitectura siguiendo las tres reglas de conexión y recordando que la información en $\mathbf{W}_{ij}$ con $i = 1, ..., MNN$ y $j = 1$ está en base decimal, sufre un redondeo como se explicó anteriormente y se codifica a

binario en una matriz **Z**. Esta matriz representará un grafo (arquitectura de la RNA), donde cada componente de la matriz $z_{ij}$ indica las aristas o conexiones entre la neurona (vértices del grafo) $i$ y la neurona $j$ cuando $z_{ij} = 1$. Por ejemplo: supongamos que $\mathbf{W}_{i,1}$ tiene un número entero "57" el cuál será transformado en su correspondiente número binario "0111001". Este número binario indica las conexiones de la neurona $i$-ésima con siete neuronas (número de bits en la cadena binaria). En este caso, solo las neuronas en la posición de la cadena binaria de izquierda a derecha donde exista un 1 como la neurona dos, tres, cuatro y siete, tendrán una conexión a la neurona $i$.

Al extraer de la matriz **W** la arquitectura se evaluará con los correspondientes pesos sinápticos de la componente $\mathbf{W}_{ij}$ con $i = 1, ..., MNN$ y $j = 2, ..., MNN + 1$. Finalmente cada neurona de la arquitectura calculará su salida con su correspondiente función de transferencia indicada en la misma matriz.

### *D. Funciones de transferencia*

Las funciones de transferencia (*TF*) se encuentran representadas en la componente $\mathbf{W}_{ij}$ con $i = 1, ..., MNN$ y $j = MNN + 3$. Dependiendo del valor entero en la componente, se eligirá una de las funciones propuestas en esta investigación.

Aunque existen otras funciones de transferencia que pueden ser implementadas en el contexto de las RNA, en esta investigación se utilizarán sólo las más utilizadas en el área. Estas TF con su respectiva nomenclatura son: la función sigmoide (LS), la función hipertangencial (HT), la función seno (SN), la función gausiana (GS), la función lineal (LN) y la función límite duro (HL).

Hasta este momento se ha explicado la codificación del individuo y la forma de decodificar la información cuando se hace la evaluación de la solución en la función de aptitud. En la siguiente sección se explican las diferentes funciones de aptitud desarrolladas en esta investigación.

### *E. Funciones de Aptitud*

La función de aptitud permite saber que tan buena es la solución dependiendo del problema de optimización que se quiere resolver. En este caso, el problema de optimización es del tipo $minf(x)|x \in A \subseteq \mathbb{R}^n$ donde $\mathbf{x} = (x_1, ..., x_n)$ y $n$ es la dimensiónalidad.

En esta investigación dos funciones de aptitud fueron aplicadas. La primera consiste en evaluar a las diferentes soluciones que se generen utilizando el error cuadrático medio (MSE) sobre el conjunto de entrenamiento $MSE_T$ y sobre el conjunto de validación $MSE_V$ ver Ec. 8. La segunda función consiste en considerar al mismo tiempo el error de clasificación (CER) sobre el conjunto de entrenamiento $CER_T$ y el de validación $CER_V$, dándole más peso al error de validación, ver Ec. 9.

$$F_1 = 0.4 \times (MSE_T) + 0.6 \times (MSE_V) \quad (8)$$

$$F_2 = 0.4 \times (CER_T) + 0.6 \times (CER_V) \quad (9)$$

El desempeño de estas funciones de aptitud serán presentadas más adelante.

## V. Salida de la red neuronal artificial

Una vez decodificada la información del individuo, se calcula la salida de la RNA, de tal forma que, es posible determinar la eficiencia de la red mediante la función de aptitud. Dicha salida se calcula aplicando el Algoritmo V.

**Algorithm 2** Pseudocódigo de la salida de la RNA. $o_i$ es la salida de la neurona $i$, $a_j$ es el patrón de entrada a la RNA, $n$ es la dimensionalidad del patrón de entrada, $m$ es la dimensionalidad del patrón deseado y $y_i$ es la salida de la RNA.

1: **for** $i = 1$ hasta $n$ **do**
2: Calcular $o_i = a_i$
3: **end for**
4: **for** $i = n + 1$ hasta $MNN$ **do**
5: Obtener el vector de conexiones **z** de la neurona $i$ a partir del individuo $W_i$.
6: Obtener los pesos sinápticos **s** de la neurona $i$ a partir del individuo $W_i$.
7: Obtener el bias $b$ de la neurona $i$ a partir del individuo $W_i$.
8: Obtener el índice $t$ de la función de transferencia de la neurona $i$ a partir del individuo $W_i$.
9: Calcular la salida de la de la neurona $i$ como $o_i = f_t\left(\sum_{j=1}^{i} s_j \cdot z_j \cdot a_j + b_j\right)$.
10: **end for**
11: **for** $i = MNN - m$ hasta $MNN$ **do**
12: Calcular la salida de la RNA con, $y_{i-MNN-m+1} = o_i$.
13: **end for**

Para el caso de la función de aptitud $F_2$, la salida de la red será modificada mediante la técnica del *ganador toma todo*, es decir la neurona que genere el valor mas alto en su salida se le asignará el valor binario de uno y a las restantes se les asignara el valor de cero. Esta nueva salida binaria será comparada con el conjunto de patrones deseados asignados para cada problema de clasificación.

## VI. Resultados experimentales

La metodologá propuesta se evaluó al resolver problemas de clasificación de patrones. Como se mencionó anteriormente, se utilizará un aprendizaje supervisado, lo que indica que será utilizado un conjunto de patrones deseados.

Debido que el algoritmo de Evolución diferencial presenta algunos parámetros, la configuración de los mismos puede repercutir en el desempeño de los resultados. Por tal motivo,

se realizó un estudio de sensibilidad para encontrar la mejor configuración de los parámetros de ED y así obtener los mejores resultados en la experimentación. Para encontrar dicha configuración se propusieron diferentes valores para cada parámetro y se evaluaron los diseños de RNA con las funciones de aptitud $F_1$ y $F_2$.

Al obtener la mejor configuración del algoritmo se procedió a realizar la experimentación que proporcionará datos estadísticamente válidos sobre el reconocimiento, los mejores resultados y la evaluación de los errores que se generaron al aplicar la metodología propuesta.

### A. Descripción de la experimentación

Para evaluar el desempeño de la metodología, se seleccionaron diez problemas de clasificación de patrones de diferente complejidad. El problema de la planta del Iris, el del vino, el cáncer de mama, el problema de diabetes, el de desórdenes del hígado y el problema del vidrio son problemas que se encuentran en el repositorio de UCI machine learning [23]. El problema de reconocimiento de objetos se obtubo de [24], y los problemas como la espiral y las dos sintéticas se desarrollaron en nuestro laboratorio. La Figura 2 muestra los patrones dispersos de éstos últimos problemas.

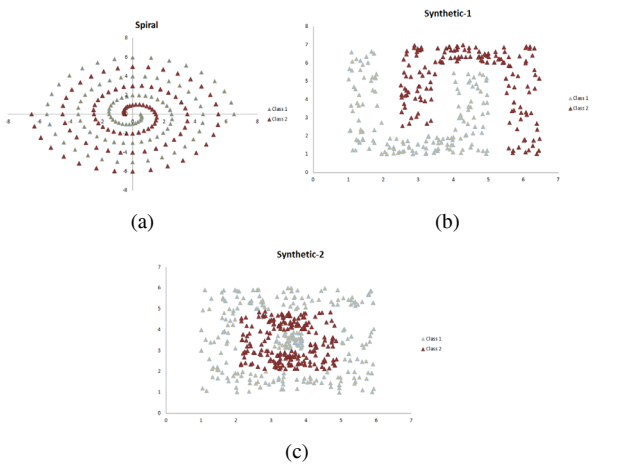



Fig. 2. Dispersión de datos para los problemas sintéticos. (a) Datos del problema de espiral. (b) Datos del problema sintético 1. (c) Datos del problema sintético 2.

En la Tabla I se encuentra la descripción de los patrones de cada problema de clasificación.

Para obtener los tres conjuntos de datos para entrenar y validar la RNA, se dividió el número de patrones totales de cada base de datos en tres conjuntos: el de entrenamiento, el de validación y el de generalización. La selección de los patrones que componene cada conjunto se realizó de manera aleatoria con el fin de validar estadísticamente los resultados obtenidos. Esta selección tiene una distribución del 33% de los patrones totales para el entrenamiento, 33% para la validación y 34% para la generalización para cada problema de clasificación.

TABLA I
DESCRIPCIÓN DE LOS PROBLEMAS DE CLASIFICACIÓN DE PATRONES.

| Problemas de clasificación | Descripción de los patrones | Patrones totales |
|---|---|---|
| Rec. objetos. | 7 caractéristicas que describen 5 clases | 100 |
| Planta Iris | 4 caractéristicas que describen 3 clases | 150 |
| Vino | 13 caractéristicas que describen 3 clases | 178 |
| Cáncer de mama | 7 caractéristicas que describen 2 clases | 683 |
| Diabetes | 8 caractéristicas que describen 2 clases | 768 |
| Desórdenes del hígado | 6 caractéristicas que describen 2 clases | 345 |
| Vidrio | 9 caractéristicas que describen 6 clases | 214 |
| Espiral | 2 caractéristicas que describen 2 clases | 194 |
| Sintética 1 | 2 caractéristicas que describen 2 clases | 300 |
| Sintética 2 | 2 caractéristicas que describen 2 clases | 450 |

### B. Análisis de sensibilidad

El análisis de sensibilidad consiste en evaluar los resultados obtenidos con diferentes valores asignados a los parámetros del algoritmo de Evolución deferencial. De este modo, se puede determinar cómo las diferentes configuraciones proporcionan desempeños variados, de los cuales podemos seleccionar la mejor configuración con el que el algoritmo se desempeña mejor.

La configuración para detrerminar cuál es el valor para cada parámetro está determinada por la siguiente secuencia: $v-w-x-y-z$ donde cada variable representa un parámetro. Para el caso del número de individuos en la población la variable $v = \{50, 100\}$ donde el elemento 1 corresponde a 50 individuos y el elemento 2 corresponde a 100. En el caso de el tamaño del espacio de búsqueda $w = \{2, 4\}$ el primer elemento indica que el rango se establece en $[-2, 2]$ y el caso del segundo elemento indica que el rango está determinado entre $[-4, 4]$. Para determinar el tipo de función de aptitud, la variable $x = \{3, 4\}$ indica con el primer elemento, que será seleccionada la función $F_1$ y el segundo elemento define la selección de la función $F_2$. El algoritmo ED, tiene dos parámetros propios: el factor de cruza $CR$ y una constante $F$, las cuales se representan por las variables $y$ y $z$ respectivamente. Ambas variables toman los valores $y = z = \{1, 2, 3\}$ donde dichos elementos estan asociados a los siguientes valores $\{0.1, 0.5, 0.9\}$. El número de generaciones o iteraciones del algoritmo se fijó en 2000 y el número de experimentos para cada combinación de los parametros, para cada problema de clasificación y cada función de aptitud se fijó en cinco corridas del algoritmo. De tal manera que la secuencia 2-1-3-2-3 significa que la configuración correspondiente es: $100 - [-2, 2] - F_1 - 0.5 - 0.9$

Se debe considerar que para cada experimento se obtienen dos valores: el error de entrenamiento y el error de generalización. Por ese motivo, se obtuvo una ponderación de estos dos valores y así se determinó quien presenta mejores resultados. Con ayuda de la evaluación ponderada de la suma del entrenameinto y la generalización se decidió asignar mayor peso a la etapa de generalización al ser multiplicada por un factor de $0.6$ y en el caso del entrenamiento el factor es $0.4$. La

ecuación que determina la ponderación es la que se describe en Ec. 10.

$$rp = 0.4 \times (Training) + 0.6 \times (Test) \quad (10)$$

donde $Training$ representa el porcentaje de reconocimiento durante la etapa de entrenamiento y $Test$ representa el porcentaje de reconocimineto obtenido durante la etapa de generalización.

Para cada configuración de los parámetros y cada problema se realizaron 5 experimentos, de los cuales se obtubo un promedio ponderado $rp$. La Tabla II muestra el mejor promedio ponderado para las dos funciones de aptitud y cada problema de clasificación de patrones, así como también muestra las configuraciones de los parámetros que generaron dichos valores.

TABLA II
PROMEDIO PONDERADO PARA CADA PROBLEMA DE CLASIFICACIÓN Y CADA FUNCIÓN DE APTITUD.

| Problemas de clasificación | Función aptitud $F_1$ | | Función aptitud $F_2$ | |
|---|---|---|---|---|
| | Promedio P. | Config. | Promedio P. | Config. |
| Espiral | 0.3753 | 2,2,3,2,2 | 0.3062 | 2,2,4,1,1 |
| Sintética 1 | 0.0128 | 1,2,3,1,2 | 0.0060 | 2,1,4,2,1 |
| Sintética 2 | 0.1400 | 2,2,3,2,2 | 0.0997 | 2,2,4,2,1 |
| Planta de Iris | 0.0256 | 2,2,3,3,3 | 0.0216 | 2,2,4,1,3 |
| Cáncer de mama | 0.0209 | 2,1,3,3,3 | 0.0204 | 2,1,4,2,2 |
| Diabetes | 0.2181 | 1,1,3,3,3 | 0.2170 | 2,1,4,3,1 |
| Desórdenes del hígado | 0.2845 | 1,1,3,2,1 | 0.2793 | 2,2,4,3,2 |
| Rec. Objetos | 0.0 | 2,2,3,3,2 | 0.0 | 1,1,4,1,1 |
| Vino | 0.0183 | 2,1,3,3,2 | 0.0237 | 2,1,4,1,3 |
| Vidrio | 0.3346 | 1,1,3,3,3 | 0.3425 | 2,1,4,1,1 |

En la Tabla II se puede observar que los mejores valores fueron obtenidos con la función $CER$, mostrando valores mínimos para ocho de diez problemas; la función de aptitud $F_1$ presentó mejores resultados aislados para el caso de los problemas del Vino y del Vidrio.

Los resultados anteriores muestran para cada problema de clasificación y cada función de aptitud, una configuración específica que genera el mejor desempeño del algoritmo evolutivo para el diseño de RNA. Sin embargo, debido a que se desea crear una metodologá no específica para cada problema a resolver, se buscó aquella configuación que en promedio resolviera mejor todas los bases de datos (una solución general), por lo que se obtubo por cada función de aptitud, el promedio que involucra el promedio ponderado $rp$ de todos los problemas de clasificación. De lo anterior, se obtubo que para el caso de la función de aptitud $F_1$ el porcentaje de error ponderado fue de $15.37\%$ y en el caso de la función de aptitud $F_2$ el porcentaje del error ponderado fue de $14.83\%$.

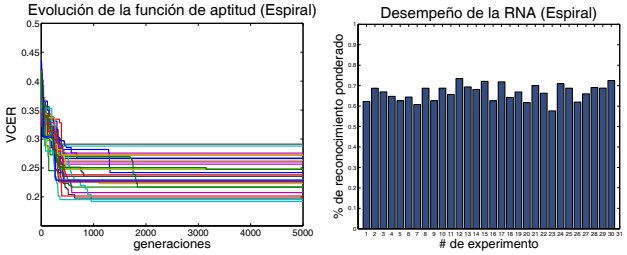
Como se mencionó anteriormente, el análisis de sensibilidad permite conocer cómo los valores de los diferentes parámetros afectan el desempeño del algoritmo ED y por consiguiente el diseño de las RNA. Por este motivo, se requiere de la minimización del error que se genere con las diferentes configuraciones. En este caso, el error mínimo se generó con la función de aptitud $F_2$ con la configuración 2-2-4-3-2. Esta configuración representa la mejor de entre todas las que se generaron al realizar la experimentación con todas las posibles configuraciones. Esta mejor configuración, será utilizada para realizar el análisis experimental que involucra todos los problemas de clasificación de patrones.

### C. Análisis experimental

Para realizar una experimentación estadísticamente válida, se generaron 30 experimentos para cada problema de clasificación utilizando la mejor configuración para el algoritmo de ED, la cuál fue encontrada previamente en el análisis de sensibilidad. Los resultados para cada problema al utilizar la mejor configuración: una población de $100$ individuos, en un espacio de búsqueda de $[-4, 4]$ con la función de aptitud $F_2$ y los valores para $CR = 0.9$ y para $F = 0.5$ se describen a continuación. Estos resultados se evaluaron durante un número de generaciones de $5000$. Los resultados presentan la evolución del error, el porcentaje de reconocimiento ponderado que conjunta la etapa de entrenamiento y generalización así como las arquitecturas con las que se generó el mejor y el peor error.

A continuación se describen los resultados obtenidos con la mejor configuración para cada uno de los diez problemas de clasificación.

*1) Espiral:* La evolución del error obtenida de la función de aptitud $F_2$ se muestra en la Figura 3, en la cual el error de clasificación y la validación se conjuntan para obtener una red neuronal con el mejor desempeño, es decir con el mínimo valor encontrado por la función de aptitud. Como se puede observar, para la mayoría de las experimentaciones el error desciende casi en su totalidad antes de las $1000$ generaciones, después el error se mantiene constante.

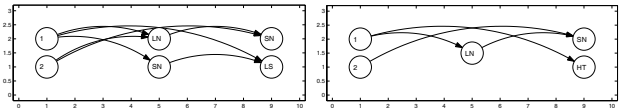



(a) (b)

Fig. 3. Resultados experimentales en los 30 experimentos para el problema de Espiral. (a) Evolución del error. (b) Porcentaje de reconocimiento ponderado.

En la Figura 4 se muestran dos de las 30 arquitecturas que se generaron para el problema de la Espiral. La Figura 4(a) es el diseño con el cuál se generó el mejor error, en donde se puede apreciar que, para las neuronas de la capa intermedia se utilizaron las funciones lineal y seno como funciones de transferencia y en la capa de salida las funciones seno y

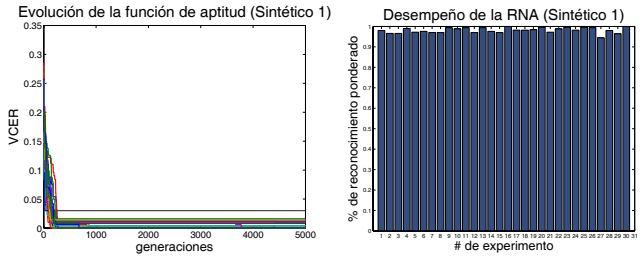
sigmiode. En la Figura 4(b) se muestra la arquitectura que generó el peor error durante la experimentación. A diferencia de la mejor, esta arquitectura presenta menos conexiones. Cabe señalar que las funciones de transferencia son las mismas utilizadas por la mejor configuración.



(a) (b)

Fig. 4. Diseño de arquitecturas de RNA generadas para el problema de Espiral en los 30 experimentos. (a) Arquitectura con el mejor desempeño. (b) Arquitectura con el peor desempeño.

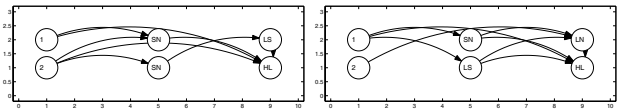
*2) Sintética 1:* Como se puede ver en la Figura 5(a), la evolución del error converge rápidamente a un error mínimo. La Figura 5(b) muestra el porcentaje de reconocimiento para la base de datos Sintética 1. En ella podemos observar que para los 30 experimentos, el desempeño de las RNA diseñadas presentan un entrenamiento y generalización muy altos, por arriba del 95% y en algunos casos alcanza el reconocimiento máximo (100%) para ambas etapas.




(a) (b)

Fig. 5. Resultados experimentales en los 30 experimentos para el problema Sintético 1. (a) Evolución del error. (b) Porcentaje de reconocimiento ponderado.

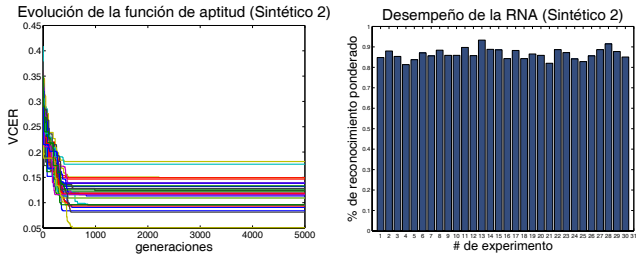
En la Figura 6 se presentan la mejor y la peor arquitectura encontradas para el problema Sintético 1. En ella podemos apreciar que la mejor arquitectura o topología presenta conexiones directas desde la capa de entrada hasta la capa de salida. Esto se debe a las reglas de conexiones propuestas anteriormente. Las funciones de transferencia utilizadas para dicha arquitectura fue la función seno, la sigmoide y la límite duro.



(a) (b)

Fig. 6. Diseño de arquitecturas de RNA generadas para el problema Sintético 1 en los 30 experimentos. (a) Arquitectura con el mejor desempeño. (b) Arquitectura con el peor desempeño.

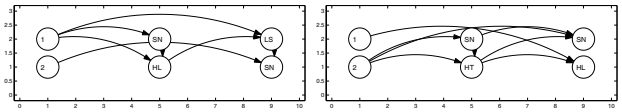
*3) Sintética 2:* La evolución del error para este problema de clasificación se muestra en la Figura 7(a), en donde se puede observar que el mínimo error generado por la función de aptitud no mejora después de las 1000 iteraciones. Por otro lado, la Figura 7(b) presenta el porcentaje de reconocimiento ponderado, en ella podemos ver que en más de la mitad de los experimentos totales, se obtubo un porcentaje por arriba del 85%.




(a) (b)

Fig. 7. Resultados experimentales en los 30 experimentos para el problema de Sintético 2. (a) Evolución del error. (b) Porcentaje de reconocimiento ponderado.

La Figura 8(a) muestra la arquitectura que alcanza el mínimo error (el mejor desempeño). Su arquitectura describe algunas conexiones directas desde la capa de entrada hasta la de salida. El conjunto de funciones de transferencia que utiliza cada neurona son: las funciones seno, límite duro y sigmoide. Para el caso de la peor arquitectura encontrada durante la experimentación, la Figura 8(b) muestra dicha topología diseñada con las funciones de transferencia seno, hipertangencial y la límite duro.



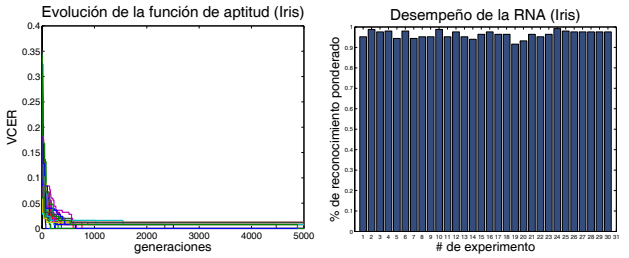
(a) (b)

Fig. 8. Diseño de arquitecturas de RNA generadas para el problema de Sintético 2 en los 30 experimentos. (a) Arquitectura con el mejor desempeño. (b) Arquitectura con el peor desempeño.

*4) Planta de Iris:* La Figura 9(a), muestra para las 5000 generaciones la evolución del error encontrado por la función de aptitud $F_2$. En ella podemos observar que al incrementar el número de generaciones el error disminuye drásticamente en algunos experimentos. Sin embargo, la mayoría de los experimentos alcanza su mejor valor antes de las 1000 generaciones.

En la Figura 9(b) se muestra el porcentaje de reconocimiento ponderado para cada experimento. Como se puede apreciar el desempeño de las RNA generadas alcanza un reconocimiento mayor al 90%.

En la Figura 10(a) se muestra la mejor arquitectura encontrada durante la experimentación. Este interesante y peculiar diseño, tiene una neurona que carece de conexión con otras en la capa de salida. Esta neurona trabaja sólo con




(a) (b)

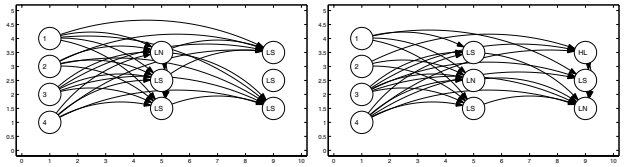
Fig. 9. Resultados experimentales en los 30 experimentos para el problema de la Planta de Iris. (a) Evolución del error. (b) Porcentaje de reconocimiento ponderado.

el bias correspondiente, que al ser evaluado en la respectiva función de transferencia sigmoide, se transforma a un valor que será exitosamente utilizado al momento de evaluar la salida de la RNA con la técnica el *ganador toma todo*. Al generarse las correspondientes salidas para cada patrón de entrada las neuronas restantes en la capa de salida son las que detallan la clase a la que pertenece cada patrón, mientras que la neurona que no tiene conexión funciona como un umbral fijo. Por ejemplo, suponga que la segunda neurona de salida genera en su salida un valor de 0.45, sin importar el patrón de entrada; por otro lado la primera y tercera neurona generan en su salida los valores de 0.65 y 0.55 respectivamente, al ser estimuladas con un patrón de entrada que pertenece a la clase 1. Al ser evaluada la salida de la RNA por la técnica del ganador toma todo, se obtendría la salida 1,0,0 la cuál indica que el patrón de entrada pertenece a la clase 1. Ahora suponga que la salida de la primera y tercera neurona generan en su salida los valores de 0.35 y 0.25 respectivamente, al ser estimuladas con un patrón de entrada que pertenece a la clase 2. En este caso, al ser evaluada la salida de la RNA por la técnica del *ganador toma todo*, se obtendría la salida 0,1,0 la cuál indica que el patrón de entrada pertenece a la clase 2, recuerde que la salida de la segunda neurona no cambia, es decir, permanece en 0.45 porque no está conectada con otras neuronas. Finalmente, en un tercer caso suponga que la salida de la primera y tercera neurona generan en su salida los valores de 0.35 y 0.65 respectivamente, al ser estimuladas con un patrón de entrada que pertenece a la clase 3. En este caso, al ser evaluada la salida de la RNA por la técnica del *ganador toma todo*, se obtendría la salida 0,0,1 la cuál indica que el patrón de entrada pertenece a la clase 3.

Las funciones de transferencia que este diseño necesita son las funciones lineal y sigmoide.

La arquitectura que genera el peor error durante las 30 experimentaciones se muestra en la Figura 10(b), la cual utiliza las funciones sigmide, lineal, sigmoide e hipertangencial como funciones de trasnferencia para cada neurona.

*5) Cáncer de mama:* Para el caso del problema de cáncer de mama, la evolución del error generado mediante la función de aptitud $F_2$ es presentado en la Figura 11(a), donde se puede observar que el error se mantiene casi constante para las 30

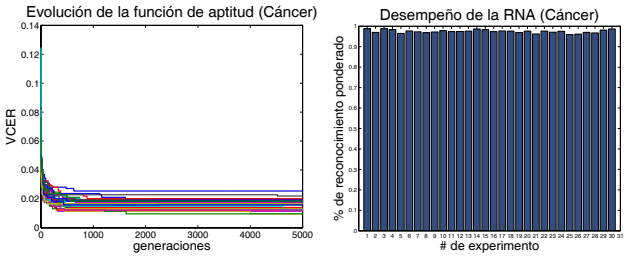


(a) (b)

Fig. 10. Diseño de arquitecturas de RNA generadas para el problema de la Planta de Iris en los 30 experimentos. (a) Arquitectura con el mejor desempeño. (b) Arquitectura con el peor desempeño.

experimentaciones durante el tiempo límite especificado en 5000 generaciones.

Por otro lado, el porcentaje de reconocimento ponderado para las etapas de entrenamiento y generalización se presentan en la Figura 11(b). El porcentaje se mantiene exitosamente, para todas las experimentaciones, por arriba del 95%, lo que indica que se encontraron los mejores diseños de las redes para resolver este problema de clasificación.




(a) (b)

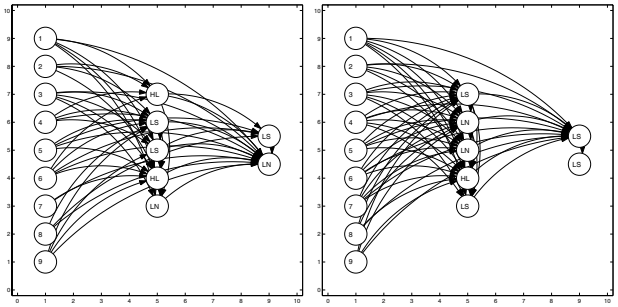
Fig. 11. Resultados experimentales en los 30 experimentos para el problema de Cáncer de mama. (a) Evolución del error. (b) Porcentaje de reconocimiento ponderado.

El mejor y el peor error de entrenamiento fue alcanzado con las arquitecturas mostradas en la Figura 12, donde también se presentan las funciones de transferencia utilizadas para cada neurona. Para el caso del mejor diseño, las funciones de transferencia seleccionadas por la metodología fueron el límite duro, sigmoide y lineal. Para el caso del peor diseño se obtubo el conjunto de funciones compuesto por la función sigmoide, la lineal y el límite duro.

*6) Diabetes:* Para el problema de la diabetes, la evolución del error a diferencia de las Figuras anteriores, muestra que el proceso de convergencia tarda más generaciones, ver Figura 13(a).

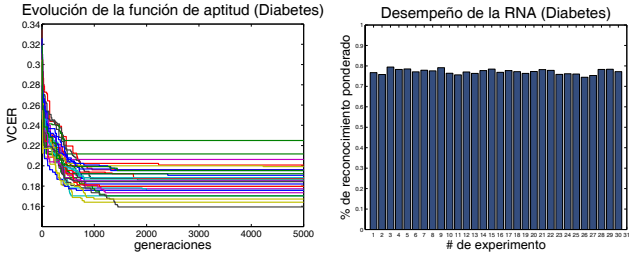
En el caso del reconocimento ponderado, el porcentaje alcanzado para toda la experimentación no fue mayor del 80%, ver Figura 13(b).

La arquitectura que genera el mejor desempeño es la que se muestra en la Figura 14(a), donde hay cuatro neuronas que componenen la capa intermedia con las siguientes funciones de transferencia: sigmoide, gaussiana y lineal. El peor caso se muestra en la Figura 14(b) donde se generó una arquitectura con una sola neurona. Esta arquitectura tiene en la capa de entrada, una neurona que no presenta conexión.



(a) (b)

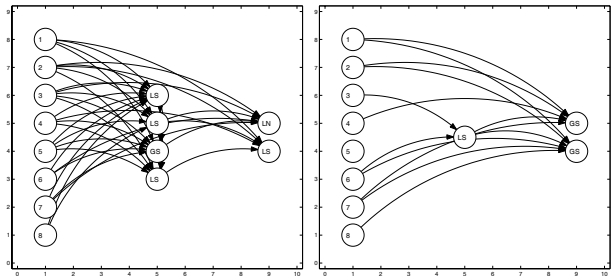
Fig. 12. Diseño de arquitecturas de RNA generadas para el problema de Cáncer de mama en los 30 experimentos. (a) Arquitectura con el mejor desempeño. (b) Arquitectura con el peor desempeño.




(a) (b)

Fig. 13. Resultados experimentales en los 30 experimentos para el problema de Diabetes. (a) Evolución del error. (b) Porcentaje de reconocimiento ponderado.

Este hecho se puede presentar al diseñar las RNA y no significa que el desempeño se reduzca; al contrario significa que la dimensionalidad del patrón de entrada se puede reducir evitando tener información redundante.



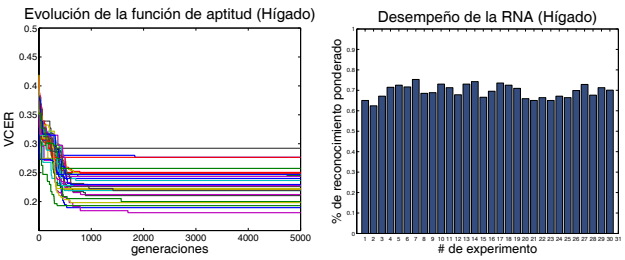
(a) (b)

Fig. 14. Diseño de arquitecturas de RNA generadas para el problema de Diabetes en los 30 experimentos. (a) Arquitectura con el mejor desempeño. (b) Arquitectura con el peor desempeño.

*7) Desórdenes del hígado:* La Figura 15 muestra la evolución del error para los 30 experimentos. Antes de las 2000 generaciones el algoritmo de ED encuentra el mejor error para cada experimento, después se mantiene constante.

Para el caso del porcentaje de reconocimiento, éste se encuentra arriba del 60% para algunos casos y para otros arriba del 70%.

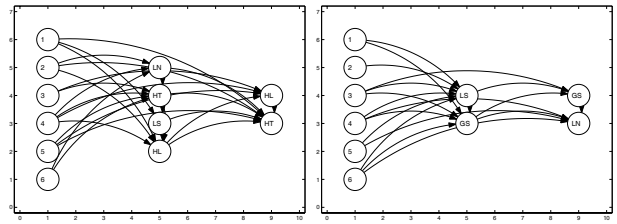
La mejor y la peor arquitectura para el problema de desórdenes del hígado se muestran en la Figura 16. El




(a) (b)

Fig. 15. Resultados experimentales en los 30 experimentos para el problema de Desórdenes del hígado. (a) Evolución del error. (b) Porcentaje de reconocimiento ponderado.

diseño con el mejor desempeño utiliza las funciones de transferencia lineal, hipertangencial y sigmoide. En el caso del peor desempeño, el conjunto de funciones de transferencia está compuesto por: la sigmoide, gausiana y la lineal.

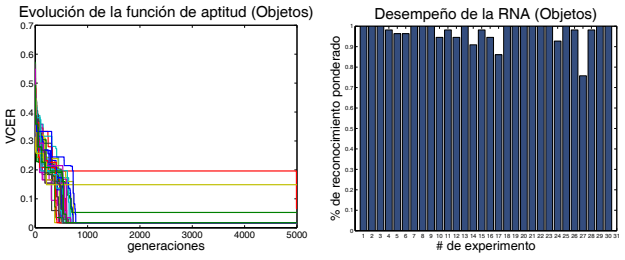


(a) (b)

Fig. 16. Diseño de arquitecturas de RNA generadas para el problema de Desórdenes del hígado en los 30 experimentos. (a) Arquitectura con el mejor desempeño. (b) Arquitectura con el peor desempeño.

*8) Reconocimiento de objetos:* La evolución del error para el problema de reconocimiento de objetos se muestra en la Figura 17(a). En ella se observa que la evaluación de la función de aptitud alcanza el error mínimo en menos de 1000 generaciones.

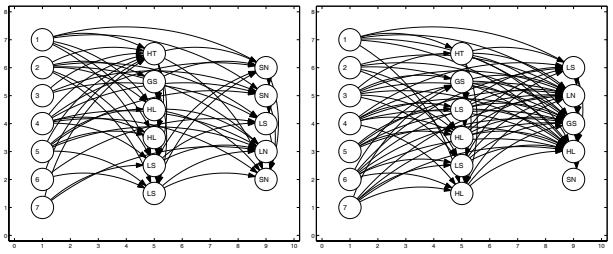
El resultado de esa evolución se ve reflejada en el reconocimiento ponderado para cada experimentación. De las 30 arquitecturas diseñadas, 16 alcanzaron un reconocimiento del 100%, ver Figura 17(b). El desempeño de la mayoría de las arquitecturas restantes se encuentran por arriba del 90%.




(a) (b)

Fig. 17. Resultados experimentales en los 30 experimentos para el problema de Reconocimiento de objetos. (a) Evolución del error. (b) Porcentaje de reconocimiento ponderado.

La Figura 18(a) muestra la arquitectura con el mejor desempeño y en Figura 18(b) aquella con el peor.

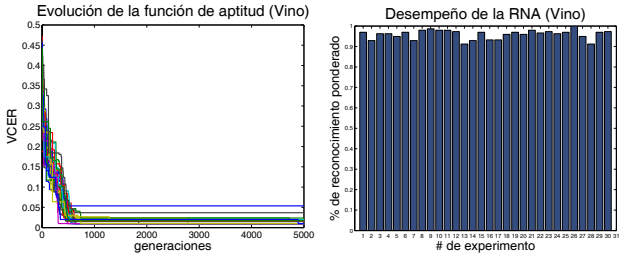


(a) (b)

Fig. 18. Diseño de arquitecturas de RNA generadas para el problema de Reconocimiento de objetos en los 30 experimentos. (a) Arquitectura con el mejor desempeño. (b) Arquitectura con el peor desempeño.

*9) Vino:* La Figura 19(a) muestra que la evolución del error tiende a un valor mínimo antes de las 1000 generaciones. En este caso se puede observar que el error de la mayoría de las experimentaciones convergen a valores cercanos.

En el caso del porcentaje de reconocimiento ponderado para el problema del vino, el cuál presenta los patrones con mayor número de características, se generó un porcentaje por arriba del 90% y en un experimento se logró generar el diseño con el mejor desempeño, es decir con el 100% de reconocimiento, ver Figura fig19(b).




(a) (b)

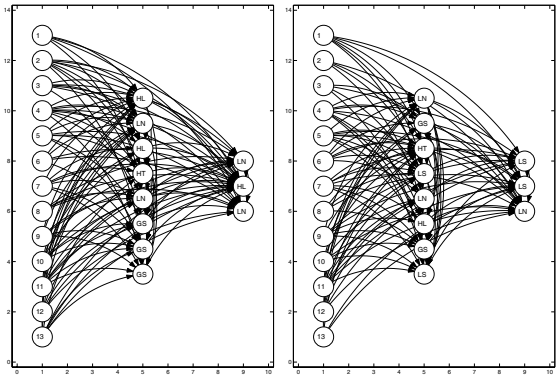
Fig. 19. Resultados experimentales en los 30 experimentos para el problema del Vino. (a) Evolución del error. (b) Porcentaje de reconocimiento ponderado.

Dos diseños generados por la metodología propuesta se presentan en la Figura 20, estos diseños son aquellos que generaron el mejor y el peor desempeño para el problema del vino.

*10) Vidrio:* El problema del vidrio es aquél problema de clasificación que presenta el mayor número de características por cada patrón en la salida. La evolución del error generado continúa descendiendo durante las primeras 3500 generaciones, ver Figura 21(a).

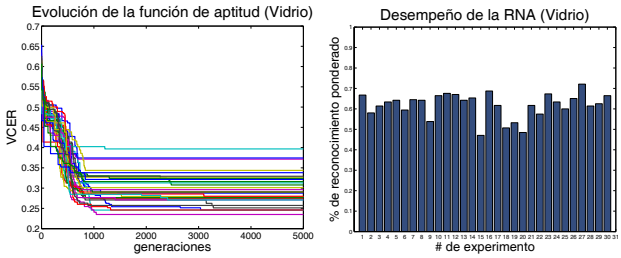
En el caso del porcentaje de reconocimiento para el problema del vidrio, se tiene que la mayoría de la experimentación se encuentra por arriba del 60% con algunos ejemplos por debajo del mismo, ver Figura 21(b).

La Figura 22 muestra las arquitecturas con el mejor y el peor desempeño durante la experimentación.



(a) (b)

Fig. 20. Diseño de arquitecturas de RNA generadas para el problema del Vino en los 30 experimentos. (a) Arquitectura con el mejor desempeño. (b) Arquitectura con el peor desempeño.



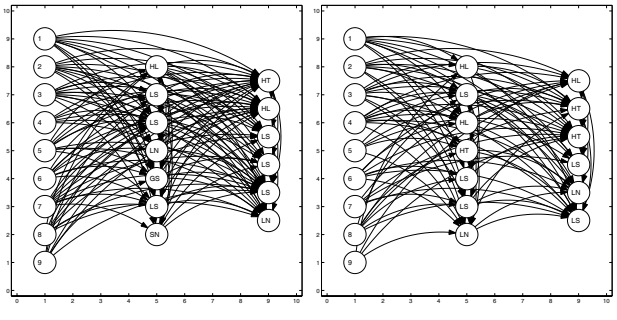

(a) (b)

Fig. 21. Resultados experimentales en los 30 experimentos para el problema del Vidrio. (a) Evolución del error. (b) Porcentaje de reconocimiento ponderado.

## *D. Discusión general*

A continuación se muestran los desempeños promedio de los 30 experimentos para cada uno de los problemas de clasificación.

La Figura 23(a) muestra el error promedio para cada problema de clasificación. En dicha Figura se muestra que la función de aptitud con la que se realizaron las experiementaciones presentó en general un error bajo. Sin

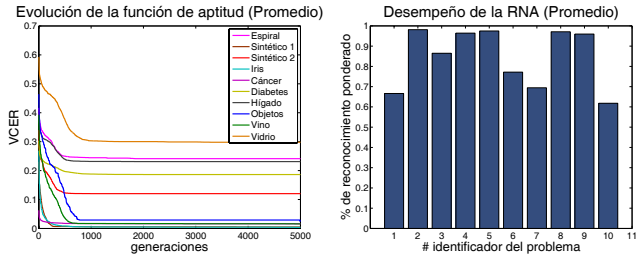


(a) (b)

Fig. 22. Diseño de arquitecturas de RNA generadas para el problema del Vidrio en los 30 experimentos. (a) Arquitectura con el mejor desempeño. (b) Arquitectura con el peor desempeño.

embargo, las bases de datos que presentan un mínimo en el error durante las 5000 generaciones fueron: reconocimiento de objetos, vino, planta de Iris, cáncer de mama y el problema sintético 1.

En la Figura 23(b) muestra el porcentaje de reconocimiento ponderado promedio. Las bases de datos alcanzaron el siguiente porcentaje: espiral de 66.62%, sintética 1 de 98.12%, sintética 2 de 86.49%, planta de Iris 96.41%, cáncer de mama un 97.47%, diabetes 77.17%, desórdenes del hígado un 69.45%, reconocimieno de objetos 97.09%, para el caso del Vino 95.95% y por último para el caso del problema del vidrio, éste alcanzó un reconocimiento del 61.79%.




(a) (b)

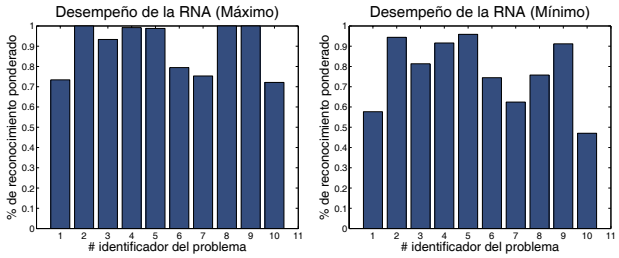
Fig. 23. Diseño de arquitecturas de RNA generadas para el problema del Vidrio en los 30 experimentos. (a) Arquitectura con el mejor desempeño. (b) Arquitectura con el peor desempeño.

La Figura 24 que a continuación se muestra, presenta los porcentajes de reconocimiento ponderado máximo y mínimo, con el fin de sintetizar cuál fue el mejor desempeño alcanzado para cada problema de clasificación al utilizar la metodología propuesta.

En la Figura 24(a) se muestra que para el caso del problema Sintético 1, del reconocimieto de objetos y el vino, se alcanzó el 100% de reconocimento. Para el caso de los problemas restantes de (izquierda a derecha) el máximo porcentaje obtenido fue: Espiral 73.41%, problema sintético 2 con 93.33%, para la planta de Iris de 99.20%, cáncer de mama %98.77, para el problema de diabetes se alcanzó un 79.45%, para el problema de desórdenes del hígado el máximo porcentaje de reconocimiento fue 75.30% y para el problema del vidrio de 72.11%.

Al contrario, la Figura 24(b) muestra el desempeño promedio de las RNA en términos del porcentaje de reconocimiento ponderado mínimo para cada problema. Para el caso de espiral, el mínimo porcentaje fue 57.66%, para sintético 1 fue de 94.40%, para problema sintético 2 de 81.33%, para la planta de Iris de 91.60%, cáncer de mama 95.87%, para el problema de diabetes se alcanzó un 74.45%, para el problema de desórdenes del hígado el mínimo porcentaje de reconocimiento fue 62.43%, para el problema de reconocimiento de objetos de 75.76%, para el vino de 91.19% y por último para el vidrio fue de 47.04%.

Por otro lado, el número de veces que fueron seleccionadas las diferentes funciones de transferencia para cada problema está descrita en la Tabla III. En ella, se puede apreciar que para




(a) (b)

Fig. 24. Diseño de arquitecturas de RNA generadas para el problema del Vidrio en los 30 experimentos. (a) Arquitectura con el mejor desempeño. (b) Arquitectura con el peor desempeño.

el caso del problema de la espiral la función de transferencia que fue seleccionada con mayor frecuencia es la función sigmoide, para el caso del problema sintético 1 fue también la función sigmoide, para el caso del problema sintético 2 fue la función seno, para el problema de la planta de Iris fue la función sigmoide, para el problema del cáncer de mama la función con una mayor selección fue la sigmoide, para el problema de diabetes también la función sigmoide fue seleccionada con mayor frecuencia, para el problema de desórdenes del hígado las funciones sigmoide y lineal tuvieron el mismo número de frecuencia con la que fueron seleccionadas; el problema de reconocimiento de objetos, el vino y el vidrio utilizaron también con mayor frecuencia la función sigmoide.

TABLA III
NÚMERO DE VECES EN QUE CADA FUNCIÓN DE TRANSFERENCIA FUE SELECCIONADA PARA CADA PROBLEMA DE CLASIFICACIÓN. FUNCIONES: LS: SIGMOIDE, HT: HIPERTANG, SN: SENO, GS: GAUSIANA, LN: LINEAL, HL: LÍM. DURO.

| Problemas de clasificación | LS | HT | SN | GS | LN | HL |
|---|---|---|---|---|---|---|
| Espiral | 14 | 11 | 41 | 8 | 35 | 5 |
| Sintética 1 | 39 | 3 | 31 | 7 | 11 | 23 |
| Sintética 2 | 37 | 7 | 40 | 13 | 9 | 6 |
| Planta de Iris | 78 | 12 | 5 | 18 | 39 | 24 |
| Cáncer de mama | 99 | 19 | 11 | 10 | 26 | 40 |
| Diabetes | 75 | 19 | 13 | 23 | 32 | 24 |
| Desórdenes del hígado | 37 | 27 | 17 | 23 | 37 | 23 |
| Rec. Objetos | 75 | 49 | 56 | 60 | 52 | 36 |
| Vino | 141 | 44 | 4 | 28 | 54 | 50 |
| Vidrio | 133 | 50 | 26 | 43 | 71 | 65 |
| **Total** | 728 | 241 | 244 | 233 | 366 | 296 |

Finalmente, se realizó una comparación con resultados generados por el método del gradiente descendiente (algoritmo de retropropagación) y el algoritmo de Levenberg-Marquardt; algoritmos clásicos de entrenamiento para las RNA. El porcentaje promedio ponderado de cada algoritmo es mostrado en la Tabla IV. Se realizaron dos configuraciones diferentes para las arquitecturas de las RNA entrenadas con cada algoritmo clásico. Estas arquitecturas consiste en generar una capa oculta y otra con dos capas ocultas.

El número máximo de neuronas totales $MNN$ de las RNA entrenadas con los algoritmos clásicos, se generaron mediante la misma ecuación en nuestra metodología propuesta para el caso de una capa oculta, ver Ec. 7 pero, para RNA con dos capas, la distribución se hace a traves de la Ec. 11.

$$DN = 0.6 \times (MNN) + 0.4 \times (MNN) \tag{11}$$

en donde la primera capa tiene el 60% de las neuronas ocultas y el 40% de las neuronas de la capa oculta está en la segunda capa, es decir una arquitectura piramidal.

Los parámetros para el algoritmo del gradiente descendiente y Levenberg-Marquardt tuvieron dos criterios de paro: al alcanzar las $5000$ generaciones o alcanzar un error de $0.000001$. Los problemas de clasificación fueron divididos en tres partes: el 40% de los patrones totales fue utilizado para el entrenamiento, el 50% fue utilizado para la generalización y el 10% fue utilizado para la validación. Se utilizó un tasa de aprendizaje del $0.1$.

Estas redes neuronales, generadas para la aplicación de los algoritmos clásicos, fueron diseñadas con el fin de obtener los mejores desempeños y así poder compararlas contra las RNA generadas por la metodología propuesta.

En la Tabla IV, podemos observar que el porcentaje promedio ponderado generado por la metodología propuesta es mejor en seis problemas de clasificación: en la espiral, sintético 1, sintético 2, cáncer de mama, diabetes y desórdenes del hígado. En el caso de la planta de Iris y del problema de reconocimiento de objetos, el mejor promedio ponderado se alcanzó con el algoritmo de Lebenberg-Marquardt de una capa. Para los problemas del vino y del vidrio, el algoritmo de Levenberg-Marquardt de dos capas obtubo el mejor porcentaje de reconocimiento ponderado.

A pesar de que el algoritmo clásico Levenberg-Marquardt, obtuvo en algunos casos mejores resultados que la metodología propuesta, el promedio general de todos los problemas de clasificación fue mayor con la metodología propuesta.

## VII. Conclusiones

Aunque la metodología propuesta ya fue presentada en [25] y [26], los autores no habían contemplado el incluir el conjunto de validación durante la etapa de entrenamiento, ni se había realizado un estudio de sensibilidad previa a la experimentación, aunado a ésto, se agragaron más problemas de clasificación.

En esta investigación, se presentó la metodología que permite diseñar de manera automática una red neuronal. Este diseño incluye, la arquitectura (cómo se conectan las neuronas y cuántas neuronas son suficientes), el valor del conjunto de pesos sinápticos y el tipo de funciones de transferencia dado un conjunto. El algoritmo evolutivo que se aplicó fue el llamado Evolución diferencial.

En una primera etapa de la experimentación, se realizó un estudio de sensibilidad de los parámetros de dicho algoritmo evolutivo. En esta experimentación se probaron diferentes valores de los parámetros y se seleccionó la mejor configuración. Esta configuración (que generó el mejor desempeño en las RNA generadas para cada problema) se dió al utilizar un número de individuos de 100, en un espacio de búsqueda entre $[-4, 4]$.

De dos funciones de aptitud seleccionadas para el anális de sensibilidad (las funciones de MSE y CER), las cuales incluyeron la etapa de validación, se encontró que, la mejor fue $F_2 = 0.4 \times (CER_T) + 0.6 \times (CER_V)$. Para el caso del factor de cruza $CR$ el mejor valor fue $0.9$ y para la constante $F$ un valor de $0.5$.

Como es bien sabido, la etapa de validación juega un papel indispensable en la etapa de entrenamiento de una red neuronal artificial, ya que impide el sobreaprendizaje. Por ese motivo, se decidió implementar la etapa de validación en la función de aptitud, con el fin de encontrar una solución con error mínimo de clasificación y al mismo tiempo que no generara un sobreaprendizaje.

Para validar estadísticamente los resultados, dicha configuración se aplicó a 30 corridas del algoritmo en cada problema de clasificación. Se encontró que el desempeño de las RNA diseñadas por la metodología bajo las condiciones dadas en el párrafo anterior, presentan un porcentaje de reconocimiento alto: en el 50% de los problemas el reconocimeinto es mayor al 95% siendo el más bajo de 61.79%.

En el caso específico para cada problema, durante las 30 experimentaciones se alcanzaron porcentajes de reconocimiento del 100% tanto en la etapa de entrenameitno como en la generalización.

Con esto, podemos decir que la etapa de validación y la mejor configuración del algoritmo de Evolución diferenial generaron resultados exitosos. Recordemos que los tres conjuntos, a saber, los conjuntos de entrenamiento, validación y generalización en los que se dividió cada problema se eligieron de manera aleatoria para cada experimento, lo que hace aún más valiosos los resultados obtenidos, pues esto indica que los resultados se validan estadísticamente y experimental.

La metodología propuesta presenta un desempeño general (en todos los problemas de clasificación) mejor que el generado con los algoritmos clásicos de entrenamiento. A pesar que hubo algunos casos donde el mejor promedio de reconocimiento ponderado se alcanzó con Levenberg-Marquardt, la metodología propuesta presenta varias ventajas: la primera es que el diseño se realiza de manera automática. En segundo lugar, la metodología no depende de un algoritmo basado en el cálculo de derivadas, lo que la hace robusto ante problemas más complejos.

Con esto podemos concluir que es posible generar diseños de redes neuronales artificiales con desempeños hasta con el 100% de reconocimineto en la etapa de entrenamiento y generalización, que se puede encontrar varios diseños que resuelven el mismo problema con diferente configuración y mismos resultados y que algunos diseños presentan

TABLA IV
PROMEDIOS DEL PORCENTAJE DE RECONOCIMIENTO PONDERADO PARA LOS ALGORITMOS CIÁSICOS Y LA METODOLOGÍA PROPUESTA.

| Problemas de clasificación | Gradiente Descendiente (1 capa ) | Gradiente Descendiente (2 capas) | Levenberg Marquardt (1 capa) | Levenberg Marquardt (2 capas) | Metodología propuesta (ED) |
|---|---|---|---|---|---|
| Espiral | 0.500824742 | 0.50137457 | 0.509209622 | 0.50137457 | **0.666185897** |
| Sintética 1 | 0.749911111 | 0.770444444 | 0.790088889 | 0.777288889 | **0.9812** |
| Sintética 2 | 0.544859259 | 0.51442963 | 0.69997037 | 0.562488889 | **0.864888889** |
| Planta de Iris | 0.932266667 | 0.652266667 | **0.979111111** | 0.756266667 | 0.964133333 |
| Cáncer de mama | 0.967696547 | 0.944751762 | 0.969269149 | 0.957415353 | **0.974685447** |
| Diabetes | 0.757864583 | 0.727604167 | 0.765260417 | 0.760902778 | **0.771692708** |
| Desórdenes del hígado | 0.604435184 | 0.576515437 | 0.675610073 | 0.662586369 | **0.694492754** |
| Rec. Objetos | 0.744533333 | 0.694133333 | **0.982133333** | 0.727466667 | 0.970909091 |
| Vino | 0.982921348 | 0.933782772 | 0.968614232 | **0.979101124** | 0.959548023 |
| Vidrio | 0.707040498 | 0.685358255 | 0.789034268 | **0.798380062** | 0.617934272 |
| **Total** | 0.749235327 | 0.700066104 | 0.812830146 | 0.748327137 | **0.846567041** |

características como la reducción de la dimensionalidad de las características de los patrones de entada.

## AGRADECIMIENTOS

H. Sossa agradece a la SIP-IPN y al DAAD, por el apoyo económico bajo el número 20111016 y al DAAD-PROALMEX J000.426/2009. B. A. Garro agradece al CONACYT por la beca otorgada durante sus estudios doctorales. H. Sossa también agradece a la Unión Europea y el CONACYT por el apoyo económico FONCICYT 93829. El contenido de este artículo es responsabilidad exclusiva del CIC-IPN y no puede ser considerado como posición de la Unión Europea.

## REFERENCES

[1] S. R. y Cajal, Ed., *Elementos de histología normal y de técnica micrográfica para uso de estudiantes*, 3rd ed. Madrid, España: Imprenta y librería de Nicolas Moya, 1901.

[2] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, *Learning internal representations by error propagation*. Cambridge, MA, USA: MIT Press, 1988, pp. 673–695. [Online]. Available: http://dl.acm.org/citation.cfm?id=65669.104449

[3] T. Back, D. B. Fogel, and Z. Michalewicz, Eds., *Handbook of Evolutionary Computation*, 1st ed. Bristol, UK, UK: IOP Publishing Ltd., 1997.

[4] C. Darwin, *The origin of species /*. New York :P.F. Collier,, 1859, http://www.biodiversitylibrary.org/bibliography/24252. [Online]. Available: http://www.biodiversitylibrary.org/item/65514

[5] A. Weismann, *Die Kontinuität des Keimplasmas als Grundlage einer Theorie der Vererbung /*. Jena, Germany: Gustav Fischer, 1885.

[6] G. Mendel and W. Bateson, *Experiments in plant-hybridisation /*. Cambridge, Mass. :Harvard University Press,, 1925, http://www.biodiversitylibrary.org/bibliography/4532. [Online]. Available: http://www.biodiversitylibrary.org/item/23469

[7] J. Ilonen, J.-K. Kamarainen, and J. Lampinen, "Differential evolution training algorithm for feed-forward neural networks," *Neural Process. Lett.*, vol. 17, no. 1, pp. 93–105, Mar. 2003. [Online]. Available: http://dx.doi.org/10.1023/A:1022995128597

[8] N. Guiying and Z. Yongquan, "A modified differential evolution algorithm for opti-mization neural network," in *Proceedings of the International Conference on Intelligent Systems and Knowledge Engineering*, ser. Advances in Intelligent Systems Research, 2007.

[9] H. M. Abdul-Kader, "Neural networks training based on differential evolution al-gorithm compared with other architectures for weather forecasting34," *International Journal of Computer Science and Network Security*, vol. 9, no. 3, pp. 92–99, march 2009.

[10] B. Garro, H. Sossa, and R. Vazquez, "Evolving neural networks: A comparison between differential evolution and particle swarm optimization," in *Advances in Swarm Intelligence*, ser. Lecture Notes in Computer Science, Y. Tan, Y. Shi, Y. Chai, and G. Wang, Eds. Springer Berlin / Heidelberg, 2011, vol. 6728, pp. 447–454.

[11] X. Yao, "Evolving artificial neural networks," 1999.

[12] B. A. Garro, H. Sossa, and R. A. Vázquez, "Design of artificial neural networks using differential evolution algorithm," in *Proceedings of the 17th international conference on Neural information processing: models and applications - Volume Part II*, ser. ICONIP'10. Berlin, Heidelberg: Springer-Verlag, 2010, pp. 201–208. [Online]. Available: http://dl.acm.org/citation.cfm?id=1939751.1939779

[13] T. Kohonen, "An introduction to neural computing," *Neural Networks*, vol. 1, no. 1, pp. 3–16, 1988.

[14] F. Rosenblatt, *Principles of Neurodynamics*. Spartan Book, 1962.

[15] J. J. Hopfield, "Neural networks and physical systems with emergent collective computational abilities," *Proceedings of the National Academy of Sciences*, vol. 79, no. 8, pp. 2554–2558, Apr. 1982.

[16] S. Haykin, *Neural Networks: A Comprehensive Foundation*. New York: Macmillan, 1994.

[17] P. Isasi Viñuela and I. M. Galván León, *Redes de neuronas artificiales: Un enfoque práctico*. Madrid, España: Pearson Educación, 2004.

[18] B. Martín del Brío and A. Saenz Molina, *Redes Neuronales y Sistemas Borrosos*. Madrid, España: Alfaomega, 2007.

[19] R. Storn and K. Price, "Differential evolution — a simple and efficient adaptive scheme for global optimization over continuous spaces," International Computer Science Institute, Berkeley, Tech. Rep., 1995. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.1.9696

[20] S. Das and P. N. Suganthan, "Differential evolution: A survey of the state-of-the-art," *IEEE Trans. Evolutionary Computation*, vol. 15, no. 1, pp. 4–31, 2011.

[21] E. Mezura-Montes, J. Velázquez-Reyes, and C. A. C. Coello, "A comparative study of differential evolution variants for global optimization," in *GECCO*, M. Cattolico, Ed. ACM, 2006, pp. 485–492.

[22] R. Storn and K. Price. (2012, april) Official web site this is a test entry of type @ONLINE. [Online]. Available: http://www.icsi.berkeley.edu/storn/code.html,

[23] D. N. A. Asuncion, "UCI machine learning repository," 2007. [Online]. Available: http://www.ics.uci.edu/∼mlearn/MLRepository.html

[24] R. A. Vázquez and H. Sossa, "A new associative model with dynamical synapses," *Neural Processing Letters*, vol. 28, no. 3, pp. 189–207, 2008.

[25] B. A. G. Licon, J. H. S. Azuela, and R. A. Vázquez, "Design of artificial neural networks using a modified particle swarm optimization algorithm," in *IJCNN*. IEEE, 2009, pp. 938–945.

[26] B. A. Garro, H. Sossa, and R. A. Vázquez, "Artificial neural network synthesis by means of artificial bee colony (abc) algorithm," in *IEEE Congress on Evolutionary Computation*. IEEE, 2011, pp. 331–338.