Extensión de una red neuronal relacional difusa incorporando distintos productos relacionales a la etapa de entrenamiento

Efraín Mendoza Castañeda, Carlos Alberto Reyes García, Hugo Jair Escalante

Department of Computer Science,
Instituto Nacional de Astrofísica, Óptica y Electrónica (INAOE),
Puebla, México

{efrain.mendoza.c,kargaxxi,hugojair}@inaoep.mx

Resumen. En este trabajo se extiende la red neuronal relacional difusa basada en el modelo de Pedrycz. Dicha ampliación consiste en dotarla de la posibilidad de cambiar el producto relacional en la fase de entrenamiento. Los productos relacionales propuestos para esto son los llamados *BK-Products*: *SubTriangle*, *SupTriangle* y *Square*, además del uso de operadores más generales (*t-norms* y *s-norms*) en sus definiciones, esto también se aplica al producto relacional Circulo usado por Pedrycz. Exploramos la efectividad de esta extensión en problemas de clasificación (incluyendo bases de datos con un esquema de ruido) y encontramos que en muchos casos la habilidad de clasificación de esta red es incrementada.

Palabras clave: Redes neuronales relacionales difusas, computo suave, sistemas neuro-difusos, productos relacionales.

1. Introducción

Los sistemas neuronales y difusos se combinan naturalmente asemejando a un sistema adaptativo con componentes sensoriales y cognitivos [15]; a los métodos que generan esta combinación se les conoce como métodos de hibridación neuro-fuzzy, estos pueden ser agrupados en dos grandes grupos [13]: (I) como redes neuro-difusas (*Fuzzy Neural Networks, FNN*), donde una red neuronal es equipada con la capacidad para manejar información difusa; y (II) como sistemas difuso-neurales (*Neuro-fuzzy systems, NFS*) que involucran un sistema de inferencia difuso (*Fuzzy Inference System, FIS*) combinado con una red neuronal para proporcionarle capacidad de aprendizaje. El presente trabajo se centra en el primer método de hibridación, donde las neuronas son diseñadas para ejecutar varias operaciones usadas en la teoría de conjuntos difusos en lugar de las operaciones comunes de multiplicación y adición, concretamente sobre las llamadas redes neuronales relacionales difusas (*Fuzzy Relational Neural Network, FRNN*).

La estructura del presente trabajo queda como sigue: en la Sección 2 se da detalle de la forma y funcionamiento de la FRNN usada, la extensión propuesta

se muestra en Sección 4, las pruebas para examinar el comportamiento de la red son establecidas en la Sección 5 (la FRNN es enfrentada a problemas de clasificación), finalmente, las secciones 6 y 7 muestran un análisis de los resultados obtenidos.

2. Red neuronal relacional difusa (FRNN)

Una FRNN usa productos relacionales (*Relationals Products*, RP), también llamados composiciones, para su funcionamiento. Un RP es un operador binario que opera sobre una relación R, entre un conjunto X y un conjunto Y, y una relación S, de Y a Z, cuyo resultado es una relación del conjunto X al conjunto Z, esto es:

$$[\Re(X \to Y) \times \Re(Y \to Z)] \to \Re(X \to Z) \tag{1}$$

estos RPs están compuestos por operaciones difusas: intersección, unión e implicación. Las operaciones de intersección y unión sobre conjuntos difusos son generalizadas por medio de las llamadas *t-norms* $\wedge$ y *s-norms* $\vee$ [8], respectivamente.

En las siguientes subsecciones se explica con más detalle el funcionamiento del modelo de FRNN utilizado.

2.1. Red de Pedrycz

Un perceptrón *feed-forward* de una capa esta formado por una colección de nodos de entrada $X = \{x_1, x_2, ..., x_m\}$, una colección de nodos de salida $Y = \{y_1, y_2, ..., y_l\}$, y una matriz de pesos $W = \{w_{ij} | i \in m, j \in l\}$, donde l son las clases a las que puede pertenecer el patrón de entrada. La salida del nodo $y_j = f(\sum_i x_i w_{ij})$ es una función de la suma ponderada por los pesos de las entradas. Un nodo de sesgo (*bias*) x_0 puede ser incluido.

Pedrycz [14] reemplaza estos componentes por relaciones difusas. La matriz de pesos del perceptrón es reemplazada por una matriz relacional difusa que representa una relación difusa de X a Y, $R = \{xRy | x \in X, y \in Y\}$, de tal forma que la conexión entre x_i y y_j tiene un valor relacional $R(x_i, y_j)$. Los valores de salida Y son generados por el producto relacional del conjunto de entradas X y las relaciones R, es decir $Y = X \circ R$. Para esta red Pedrycz utiliza la composición *max-min* [17]. Esta composición se representa como $y_j = max(x_i, min(x_i, R(x_i, y_j)))$.

En [14] se propone también un índice de igualdad difuso basado en la implicación de Łukasiewicz, esto provee de una medida de desempeño para la evaluación del error en el entrenamiento. Esta métrica puede ser usada en técnicas de gradiente descendente para actualizar los pesos de la matriz en una forma similar al algoritmo estándar de retropropagación (*Backpropagation*, BP). Puesto que las técnicas de gradiente descendente requieren que la función usada sea derivable,

Pedrycz calcula un aproximado de la derivada de la composición *max-min*. Muestra también, que el problema XOR, un problema irresoluble por las redes estándar de una capa tiene solución y converge con la red propuesta.

2.2. Modificaciones de Reyes-García

Reyes-García propone (ver [15]) un marco de trabajo para explotar el uso de la FRNN de Pedrycz, con la particularidad de que esta FRNN usa diferentes productos relacionales en la etapa de procesamiento (descrita en breve). En esta arquitectura la capa de entrada esta formada por $N \times n$ neuronas, cada una de las cuales corresponde a una de los N términos lingüisticos asignados a cada una de las n entradas. La capa de salida esta compuesta por l neuronas, cada una de las cuales pertenece a una de las l clases. Existe una conexión entre cada nodo en la capa de entrada a cada nodo en la capa de salida. La Figura 1 muestra el marco de trabajo y las fases de la FRNN. Cada fase y cada módulo serán descritos posteriormente. La operación de la FRNN es dividida en dos fases; la primera es para aprendizaje y la segunda para el procesamiento.

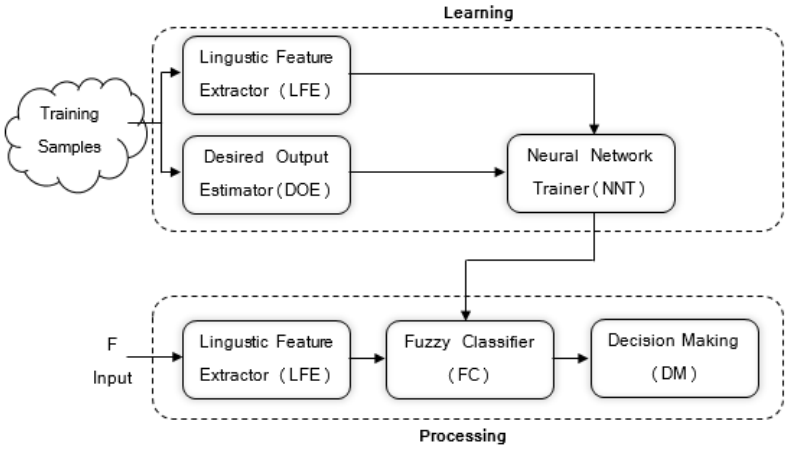

Fig. 1. Diagrama de bloques del marco de trabajo de un sistema basado en una Red Neuronal Relacional Difusa (FRNN).

2.2.1. Fase de aprendizaje

La fase de aprendizaje esta dividida en tres módulos: el Extractor de Características Lingüisticas (*Linguistic Feature Extractor*, LFE), Estimador de Salidas Deseadas (*Desired Output Estimator*, DOE) y el Entrenador de la Red Neuronal (*Neural Network Trainer*, NNT).

módulo LFE. Toma las muestras de entrenamiento F y cada característica de F es transformada en valores de membresía a cada uno de las propiedades lingüisticas asignadas. De este modo un vector conteniendo n características es transformado en un 3n-dimensional (describiendo pertenencia baja, media y alta), 5n-dimensional (pertenencia muy baja, baja, media, alta y muy alta), o 7n-dimensional (pertenencia muy baja, baja, mas o menos baja, media, más o menos alta, alta, muy alta). El vector resultante es llamado Vector de Propiedades Lingüisticas (*Linguistic Properties Vector*, LPV). Para calcular los valores de membresía pueden ser usadas diferentes tipos de funciones tales como: *Gaussian*, Trapezoidal, Triangular y *Bell* (ver [16]).

módulo DOE. Dado que, una FRNN es un método de aprendizaje supervisado, el segundo módulo, DOE, se encarga de calcular los valores de membresía para cada ejemplo en cada clase de salida, el arreglo resultante es llamado Vector de Salida Deseado (*Desired Output Vector*, DOV), que es posteriormente utilizado, en el calculo del error de la red después de cada iteración. Para obtener los valores de membresía deseados es necesario calcular la distancia del patrón de entrenamiento F_i a la $k - ésima$ clase como en la ecuación:

$$z_{ik} = \sqrt{\sum_{j=0}^{n} \left[\frac{F_{ij} - \mu_{kj}}{\sigma_{kj}} \right]^2} | k = 1...l \tag{2}$$

donde F_{ij} es la $j - ésima$ característica del patrón i, l el número de clases, μ_{kj} y σ_{kj} denotan, respectivamente, la media y la desviación estándar de la característica j y la clase k. El valor de membresía de F_i^{th} es definido como sigue:

$$\mu_l(F_i) = \frac{1}{1 + \left(\frac{z_{ik}}{f_d} \right)^{f_e}} \tag{3}$$

donde f_e es el generador exponencial difuso, y f_d es el generador denominacional difuso, estos parámetros controlan la cantidad de difusión en esta clase. En este caso, mientras mayor sea la distancia del patrón a una clase, menor será su membresía a esta. Dado que los datos de entrenamiento tienen límites de clase difusos, un patrón puede pertenecer a una o varias clases al mismo tiempo en diferentes grados.

módulo NNT. Toma los vectores LPV y DOV como la base para entrenar la red. El LPV se pasa a la capa de entrada y el DOV es utilizado para el entrenamiento de la red. Las salidas de la red son calculadas para obtener el error de la capa de salida. El error esta representado por la distancia entre la salida actual y la salida objetivo. Minimizar este error es el objetivo del proceso de entrenamiento. Durante cada paso del aprendizaje, una vez que el error ha sido calculado, el entrenador ajusta los valores de la matriz de relaciones de las conexiones correspondientes, por medio el algoritmo BP, hasta que se obtiene un error mínimo o se ha completado un número dado de iteraciones. La salida

del NNT es una matriz relacional que contiene el conocimiento necesario para posteriormente mapear un vector desconocido de entrada a su correspondiente clase durante la fase de procesamiento. El proceso de aprendizaje es explicado en detalle en [14].

2.2.2. Fase de procesamiento

La fase de procesamiento comienza usando el mismo LFE definido en la fase de aprendizaje, en esta fase el LFE opera en una forma similar para *fuzzificar* los datos de prueba. El clasificador difuso (FC, *Fuzzy Classifier*) entonces usa la matriz relacional desarrollada durante la fase de aprendizaje y el LPV para procesar los patrones de entrada. Los patrones de entrada son procesados usando diferentes RPs. El último módulo, DMM, es el encargado de interpretar la salida difusa de la FRNN y depende del problema bajo análisis, para fines de clasificación toma la membresía más alta del vector de clases y la asigna a la muestra de entrada.

3. Trabajo relacionado

La composición *max-min* es una de las más usadas ([5–7, 12, 14, 15]). W. Pedrycz, como se mencionó anteriormente, [14] propone la estructura básica de la FRNN. Blanco et al. [5] integra el concepto de "derivación suave" a la aproximación de la derivada; Valente de Oliveira [12] expande la composición a *max-t*. Reyes-García [15] extiende la red desarrollada por Pedrycz [14] incorporando los productos relacionales *BK* [2]- [9]; Davis & Kohout [6] incorporan generalizaciones de los RPs *BK*; Barajas y Reyes (ver [4]) usan un algoritmo genético (para seleccionar número de términos lingüísticos, tipo de funciones de membresía, método de clasificación y tasa de aprendizaje) en combinación con el método BP. Otros trabajos ([1]) se centran en el uso de normas difusas suaves (*smooth fuzzy norms*), es decir, normas que son derivables.

En el presente trabajo se extienden los métodos presentados por [15] y [6]. La extensión introducida consiste en la integración de otros productos relacionales en la etapa de entrenamiento.

4. Extensión propuesta

Este trabajo propone utilizar diferentes RPs en la etapa de entrenamiento, así como el uso de *t-norms* y *s-norms* en las definiciones de estos.

La incorporación de más RPs a la FRNN se hace con base en las observaciones hechas por Bandler & Kohout [3]:

- Existen muchas maneras significativas distintas de definir la inclusión de una estructura difusa en otra; esto depende de la elección de un operador de implicación en particular.

- La elección de las propiedades semánticas de un operador de implicación particular depende de algunas consideraciones pragmáticas, determinadas por las preguntas metodológicas de la aplicación en particular.

De lo anterior se desprende que, si solo entrenamos a la FRNN con una composición (*max-min* en este caso), limitamos la habilidad de esta para representar relaciones entre estructuras, por lo que se hace necesario proveerla de la capacidad de sustituir el RP de entrenamiento por otro que resulte más conveniente, dependiendo del problema bajo análisis.

En las siguientes subsecciones se muestra cual es el camino tomado para incorporar otros productos relacionales al entrenamiento de la FRNN.

4.1. Fase de entrenamiento

El cambio de un RP por otro en el entrenamiento de la FRNN no puede hacerse de forma transparente; puesto que el algoritmo de aprendizaje (BP) usado por la FRNN es un método de gradiente descendente y requiere que los operadores utilizados para el entrenamiento sean derivables, condición que la mayoría de los operadores difusos no cumple. En este punto se tienen dos opciones: (1) limitarse al uso de operadores que sean derivables (2) calcular para los operadores - en caso de que no sean derivables - un aproximado de su derivada. Nosotros nos apegaremos a la segunda opción, puesto que existen muchos operadores que han probado ser de gran valor al representar las relaciones existentes entre patrones de datos.

Pedrycz incorpora un índice de igualdad Q expresado por implicaciones difusas (ver [14]), este índice sirve como medida para evaluar el error de la red, al calcular la derivada de este error respecto a los pesos, obtiene lo siguiente:

$$\Delta w = \frac{\partial Q}{\partial w} = -\sum_{i:y_i>t_i} \frac{\partial f(x_i; w, \vartheta)}{\partial w} + \sum_{i:y_i<t_i} \frac{\partial f(x_i; w, \vartheta)}{\partial w} \tag{4}$$

$$\Delta \delta = \frac{\partial Q}{\partial \delta} = -\sum_{i:y_i>t_i} \frac{\partial f(x_i; w, \vartheta)}{\partial \delta} + \sum_{i:y_i<t_i} \frac{\partial f(x_i; w, \vartheta)}{\partial \delta} \tag{5}$$

donde x_i es el patrón de entrada, w es la matriz de relaciones y ϑ es el vector de sesgo. El resto de la derivada (Δw o $\Delta \delta$) puede ser calculado cuando se especifica la forma de la función f. Por ejemplo, si tomamos en cuenta la definición del producto *circlet* ($\circ$) con un sesgo, $y_k = \vee\left(\vee\left(x_{ij} \wedge w_{kj}\right), \vartheta_k\right)$ donde $k = 1, .., l$, tenemos (los índices son omitidos para mayor claridad):

$$\begin{aligned} \frac{\partial(\vee(w \wedge x) \vee \vartheta)}{\partial w} &= \frac{\partial(\vee(w \wedge x) \vee \vartheta)}{\partial \vee (a \wedge x)} * \frac{\partial(\vee(w \wedge x))}{\partial w} \\ &= \frac{\partial(\vee(w \wedge x) \vee \vartheta)}{\partial \vee (w \wedge x)} * \frac{\partial(\vee(w \wedge x))}{\partial(w \wedge x)} * \frac{\partial(w \wedge x)}{\partial w} \end{aligned} \tag{6}$$

esta derivada esta en términos de *t-norms* y *s-norms*, estos operadores tienen que ser sustituidos por implementaciones que cumplan con las restricciones inherentes a estos. De este modo si reemplazamos las *t-norms* por el operador *min* y las *s-norms* por *max* (considerando que conocemos las aproximaciones de sus derivadas [14] [5]), la derivada queda como sigue

$$\begin{aligned}
\frac{\partial(\vee(w \wedge x) \vee \vartheta)}{\partial \vee (w \wedge x)} &= \begin{cases} 1 & \vee(w \wedge x) \geq \vartheta, \\ 0 & otro\ caso. \end{cases} \\
\frac{\partial(\vee(w \wedge x))}{\partial(w \wedge x)} &= \begin{cases} 1 & (w \wedge x) \geq (\vee(w \wedge x), \\ 0 & otro\ caso. \end{cases} \\
\frac{\partial(w \wedge x)}{\partial w} &= \begin{cases} 1 & a \leq x, \\ 0 & otro\ caso. \end{cases}
\end{aligned} \tag{7}$$

como se puede observar la derivada del producto $max - min$ se reduce a una serie de casos, que son iguales a los mostrados por Pedrycz [14]. Adoptando este enfoque podemos agregar más RPs para el entrenamiento definiendo la derivada del producto relacional hasta el punto que se muestra en la ecuación (6) y posteriormente, el resto del proceso es transparente.

Los RPs que en este trabajo se integrarán a la fase de entrenamiento son denominados *BK products* (por Bandler-Kohout): SupTriangle, SubTriangle y Square; por las cualidades de estos para representar relaciones entre estructuras [10]. Las derivadas y definiciones de estos productos se muestran en la Tabla 1.

Tabla 1. Productos relacionales que se incorporarán a la etapa de entrenamiento.

Producto Relacional.	Definicion	Derivada
Circlet ($\circ$)	$\vee_j(R_{ij} \wedge S_{ij})$	$\frac{\partial(\vee(w\wedge x)\vee\vartheta)}{\partial\vee(w\wedge x)} * \frac{\partial(\vee(w\wedge x))}{\partial(w\wedge x)} * \frac{\partial(w\wedge x)}{\partial w}$
SubTriangle ($\triangleright$)	$\wedge_j(R_{ij} \rightarrow S_{jk})$	$\frac{\partial(\wedge(x\rightarrow w)\vee\vartheta)}{\partial(\wedge(x\rightarrow w))} * \frac{\partial(\wedge(x\rightarrow w))}{\partial(x\rightarrow w)} * \frac{\partial(x\rightarrow w)}{\partial w}$
SupTriangle ($\triangleleft$)	$\wedge_j(R_{ij} \leftarrow S_{jk})$	$\frac{\partial(\wedge(x\leftarrow w)\vee\vartheta)}{\partial(\wedge(x\leftarrow w))} * \frac{\partial(\wedge(x\leftarrow w))}{\partial(x\leftarrow w)} * \frac{\partial(x\leftarrow w)}{\partial w}$
Square ($\square$)	$\wedge_j(R_{ij} \leftrightarrow S_{jk})$	$\frac{\partial(\wedge(x\leftrightarrow w)\vee\vartheta)}{\partial(\wedge(x\leftrightarrow w))} * \frac{\partial(\wedge(x\leftrightarrow w))}{\partial(x\leftrightarrow w)} * \frac{\partial(x\leftrightarrow w)}{\partial w}$

Para completar el proceso de entrenamiento necesitamos definir las *t-norms*, *s-norms* e implicaciones, con sus respectivas derivadas, que los RPs utilizarán. Los operadores fueron elegidos por su amplio uso en la literatura, los detalles de estos son mostrados en la Tabla 2.

Tabla 2. Operadores difusos usados por los RPs en la etapa de entrenamiento.

Operador	Implementación		Derivada
$\rightarrow$	Łukasewicz	$min(1, 1-a+b)$	$-\frac{\partial min(1,1-b+a)}{\partial a}$
$\leftrightarrow$		$min(a \rightarrow b, b \rightarrow a)$	$\frac{\partial min(a\rightarrow b, b\rightarrow a)}{\partial a} = \frac{\partial min(a\rightarrow b, b\rightarrow a)}{\partial a \rightarrow b} * \frac{\partial a \rightarrow b}{\partial a} + \frac{\partial min(a\rightarrow b, b\rightarrow a)}{\partial b \rightarrow} * \frac{\partial b \rightarrow a}{\partial a}$
$\vee$	Máximo	$max(a, b)$	$\frac{\partial max(a,b)}{\partial a} = \begin{cases} 1, & a >= b \\ 0, & otros \end{cases}$
$\wedge$	Mínimo	$min(a, b)$	$\frac{\partial min(a,b)}{\partial a} = \begin{cases} 1, & a <= b \\ 0, & otros \end{cases}$

4.2. Fase de procesamiento

Como se describió anteriormente en los trabajos de Reyes-García y Davis & Kohout, la FRNN ha sido extendida usando distintos RPs en la etapa de procesamiento de la red. En el presente trabajo nos apegaremos a este enfoque: usaremos como base para procesamiento los RPs *Circlet*, *Suptriangle*, *Subtriangle* y *Square* (ver Tabla 3). Los operadores difusos usados como instancias particulares de las *t-norm*, *s-norm* e implicación se muestran en Tabla 3.

5. Experimentos

Para probar la validez de la extensión propuesta la FRNN fue sometida a problemas de clasificación, esto es, asignar etiquetas de clase l, de un conjunto de etiquetas L, a los ejemplos de prueba, donde los valores de las características son conocidos, pero la etiqueta de clase no [11]. Una FRNN puede utilizarse para afrontar este problema gracias a la interpretación que puede darse a las salidas de la red. Estas salidas son grados de pertenencia a cada clase de $l \in L$, para problemas de clasificación el módulo DM asignará la etiqueta con mayor grado de pertenencia al patrón de entrada.

La evaluación fue realizada usando una validación cruzada de 10 pliegues y el criterio para la evaluación fue el porcentaje de patrones clasificados correctamente (exactitud).

5.1. Conjuntos de datos

Los conjuntos de datos utilizados se obtuvieron del *UCI Machine Learning Repository*: *iris*, *car*, *ecoli*, *pima*, *nursery*, *glass*, *wine*, *heart*, *ionosphere* (un resumen de las propiedades de los conjuntos de datos es dado en la Tabla 4). Una de las fortalezas de los algoritmos que utilizan lógica difusa es la tolerancia al ruido, por lo que en las pruebas de clasificación se toman en cuenta algunas

Tabla 3. Productos relacionales que se incorporarán a la etapa de procesamiento.

	Operador	Definición
Implicación $a \rightarrow b$	Łukasewicz	$min(1, 1 - a + b)$
	Kleene-Diens	$\vee(1 - a, b)$
	Gaines	$\wedge(1, b/a)$
Equidad $a \leftrightarrow b$		$\wedge(a \rightarrow b, b \rightarrow a)$
t-norm $\wedge(a, b)$	Mínimo	$min(a, b)$
	Producto	$a * b$
	Einsten product	$\frac{a*b}{2-a+b-a*b}$
	Hamacher product	$\frac{a*b}{a+b-a*b}$
	Drastic *t-norm*	$\begin{cases} b \text{ if } a == 1 \\ a \text{ if } b == 1 \\ 0 \text{ otros casos} \end{cases}$
	Nilpotent	$\begin{cases} min(a, b) \text{ if } (a + b) > 1 \\ 0 \text{ otros casos} \end{cases}$
s-norm $\vee(a, b)$	Máximo	$max(a, b)$
	Probabilistic sum	$a + b - a * b$
	Einsten product	$\frac{a+b}{1+a*b}$
	Hamacher sum	$\frac{a+b-2*(a*b)}{1-a*b}$
	Drastic *s-norm*	$\begin{cases} b \text{ if } a == 0 \\ a \text{ if } b == 0 \\ 1 \text{ otros casos} \end{cases}$
	Nilpotent	$\begin{cases} max(a, b) \text{ if } (a + b) < 1 \\ 1 \text{ otros casos} \end{cases}$

bases de datos con ruido. Las bases de datos fueron tomadas del *Keel-dataset repository* y el ruido fue introducido con el patrón propuesto por Wu et al [18] con un esquema *Noisy Train - Noisy Test* al 20 % (información más detallada de las bases de datos tomadas se muestran en la Tabla 4, marcadas con un '*').

Tabla 4. Estadísticas de las bases de datos.

Bd	#Atributos (R/I/N)	# Ejemplos	# Clases
*iris	4 (4/0/0)	150	3
car	6 (0/0/6)	1728	4
ecoli	7 (7/0/0)	336	8
*pima	8 (8/0/0)	768	2
glass	9 (9/0/0)	214	7
*wine	13 (13/0/0)	178	3
*heart	13 (1/12/0)	270	2
ionosphere	33 (32/1/0)	351	2

5.2. Configuración de la FRNN

La FRNN requiere de los siguientes parámetros para funcionar: RP para procesamiento, RP para entrenamiento, número de términos lingüísticos, tipo

de función de membresía y número de épocas. Los RPs para el procesamiento son el resultado de todas las posibles combinaciones que se originan de sustituir los operadores (Tabla 3) en la definición general de los RPs (Tabla 1). Para el entrenamiento de la red se tomaron los RPs listados en Tabla 1 combinados con los operadores Tabla 2. El número de términos lingüísticos puede establecerse en 3, 5 y 7 (ver detalle en [15]). Las funciones de membresía son usadas para calcular el nivel de pertenencia de un patrón de entrada a una clase, para este trabajo las elecciones posibles de funciones de membresía son *Pi*, *Triangular* y *Trapezoidal*. El número de épocas fue establecido en 5 ya que entrenamiento adicional no produjo efectos significativos en el rendimiento sobre los conjuntos de pruebas.

Todas las configuraciones de FRNN posibles, con los parámetros fijados anteriormente, fueron probadas sobre cada una de las bases de datos.

6. Resultados

Como se puede ver en la Tabla 5 los resultados obtenidos con la extensión propuesta en este trabajo mejoran a los alcanzados por la red de Pedrycz y la extensión de Reyes-García & Bandler.

Tabla 5. Comparación de desempeño de la red FRNN original y la extensión propuesta en este artículo. La segunda sección muestra los resultados obtenidos sobre las bases de datos con ruido.

Conjunto de datos	FRNN	E. FRNN
iris	96.66 ± 3.51	97.33 ± 3.44
car	78.47 ± 1.57	79.39 ± 2.73
ecoli	65.79 ± 4.03	73.25 ± 8.24
pima	74.35 ± 3.16	74.35 ± 3.16
glass	59.87 ± 6.09	62.34 ± 6.04
wine	87.09 ± 5.92	97.22 ± 3.92
heart	76.29 ± 8.76	84.81 ± 8.63
ionosphere	88.04 ± 4.56	88.04 ± 4.00
iris	82.66 ± 5.62	90.66 ± 7.82
pima	71.22 ± 3.29	72.13 ± 4.56
wine	80.35 ± 8.10	89.86 ± 81.44
heart	72.96 ± 7.20	81.85 ± 74.55

Es importante resaltar que el mejor RP de entrenamiento presentó variación dependiendo del conjunto de datos procesado, pues este era el resultado esperado; esto puede observarse en la Tabla 6, que muestra las mejores configuraciones de FRNN encontradas.

Tabla 6. Configuraciones de FRNN con mejores resultados (exactitud). La segunda sección de la tabla muestra las mejores configuraciones para las bases de datos con ruido.

Dataset	Función de membresía	# térm. ling.	RP Entrenamiento | RP procesamiento	Precisión
iris	Triangular	3	∘{∧ :Min, ∨ :Max} ◁{∧ :Min, → {KleeneDiens{∧ :HamacherSum}}}	97.33 ± 3.44
car	Trapezoidal	7	∘{∧ :Min, ∨ :Max} ◁{∧ :Min, → {KleeneDiens{∧ :HamacherSum}}}	9.39 ± 2.73
ecoli	Trapezoidal	7	□{↔ {Lukasewicz{∧ :Min}}, ∧ :Min} ∘{∧ :Ham.Prod.}, ∨ :ProbSum}	73.25 ± 8.24
pima	Triangular	3	∘{∧ :Min, ∨ :Max} ∘{∧ :Min, ∨ :Max}	74.35 ± 3.16
glass	Triangular	3	∘{∧ :Min, ∨ :Max} ∘{∧ :Nilp, ∨ :Max}	62.34 ± 6.04
wine	Pi	5	□{↔ {Lukasewicz{∧ :Min}}, ∧ :Min} ∘{∧ :HamacherProd, ∨ :EinstenSum}	97.22 ± 3.92
heart	Pi	5	◁{∧ :Min, → {Lukasewicz{∧ :Min}}} ∘{∧ :HamacherProd, ∨ :HamacherSum}	84.81 ± 8.63
ionosphere	Trapezoidal	5	□{↔ {Lukasewicz{∧ :Min}}, ∧ :Min} ∘{∧ :Min, ∨ :EinstenProduct}	88.04 ± 4.00
iris	Pi	3	▷{∧ :Min, → {Lukasewicz{∧ :Min}}} ∘{∧ :Min, ∨ :Prob.Sum}	90.66 ± 7.82
pima	Trapezoidal	7	∘{∧ :Min, ∨ :Max} ∘{∧ :Min, ∨ :SNilp.}	72.13 ± 4.56
wine	Trapezoidal	5	□{↔ {Lukasewicz{∧ :Min}}, ∧ :Min} □{↔ {→ {KD{∨ :Max}}, ∧ :Min}, ∧ :Product}	89.86 ± 81.44
heart	Triangular	7	◁{∧ :Min, → {Lukasewicz{∧ :Min}}} ∘{∧ :Ham.Prod.}, ∨ :ProbSum}	81.85 ± 74.55

[1] La descripción del RP seleccionado (para entrenamiento o procesamiento) se da en formato *JSON*, este formato esta formado de un conjunto desordenado de pares nombre/valor. Un objeto comienza con '{' (llave de apertura) y termine con '}' (llave de cierre). Cada nombre es seguido por ':' (dos puntos) y los pares nombre/valor están separados por ',' (coma).

7. Conclusiones y trabajo futuro

En este trabajo se exploró la incorporación de otros RPs a la etapa de entrenamiento de una FRNN. Se plantea una forma sencilla de agregar, a conveniencia, otros RPs más adecuados a la aplicación. Encontramos que, para algunas bases de datos, entrenando la FRNN con los RPs propuestos (*Circlet* generalizado), *SubTriangle*, *SupTriangle* y *Square*) se consiguen resultados que superan a la composición usada por Pedrycz (*max-min*).

La principal desventaja de este enfoque, es que, al agregar un mayor número de operadores puede ser prohibitivo evaluar todas las posibles combinaciones, por lo que, como trabajo futuro se plantea la adición de un método de selección de modelo que encuentre la mejor configuración de FRNN para la base de datos bajo análisis; esto con la finalidad de mitigar el inconveniente del rápido crecimiento de combinaciones posibles de FRNN. También, teniendo en cuenta que las salidas de la FRNN representan niveles de membresía del patrón procesado a cada una de las clases, se usará esta aproximación a problemas que den un mayor uso a esta información, como *Label ranking*.

Referencias

1. Ashtiani, A.A., Menhaj, M.B.: Numerical solution of fuzzy relational equations based on smooth fuzzy norms. Soft Computing 14(6), 545–557 (2010)
2. Bandler, W., Kohout, L.J.: Mathematical relations, their products and generalized morphisms. Techn Report Man-Machine Systems Lab Dept Electrical Engin Univ Essex, Colchester, Essex, UK, EES-MMS-REL pp. 77–3 (1977)
3. Bandler, W., Kohout, L.J.: Semantics of implication operators and fuzzy relational products. International Journal of Man-Machine Studies 12(1), 89–116 (1980)
4. Barajas, S.E., Reyes, C.A.: Your fuzzy relational neural network parameters optimization with a genetic algorithm. In: Fuzzy Systems, 2005. FUZZ'05. The 14th IEEE International Conference on. pp. 684–689. IEEE (2005)
5. Blanco, A., Delgado, M., Requena, I.: Identification of fuzzy relational equations by fuzzy neural networks. Fuzzy Sets and Systems 71(2), 215–226 (1995)
6. Davis, I., Warren, L.: Enhancing pattern classification with relational fuzzy neural networks and square bk-products (2006)
7. Garcia, C.A.R., Bandler, W.: Implementing a fuzzy relational neural network for phonetic automatic speech recognition. In: Fuzzy Modelling, pp. 115–139. Springer (1996)
8. Klement, E.P., Mesiar, R., Pap, E.: Triangular norms. position paper i: basic analytical and algebraic properties. Fuzzy Sets and Systems 143(1), 5–26 (2004)
9. Kohout, L.J.: Boolean and fuzzy relationsboolean and fuzzy relations. In: Encyclopedia of Optimization, pp. 189–202. Springer (2001)
10. Kohout, L.J., Kim, E.: The role of bk-products of relations in soft computing. Soft Computing 6(2), 92–115 (2002)
11. Kotsiantis, S.B.: Supervised machine learning: a review of classification techniques. Informatica (03505596) 31(3) (2007)
12. de Oliveira, J.: Neuron inspired learning rules for fuzzy relational structures. Fuzzy sets and systems 57(1), 41–53 (1993)
13. Pal, S.K., Mitra, S.: Neuro-fuzzy pattern recognition: methods in soft computing. John Wiley & Sons, Inc. (1999)
14. Pedrycz, W.: Neurocomputations in relational systems. IEEE Transactions on Pattern Analysis and Machine Intelligence 13(3), 289–297 (1991)
15. Reyes, C.A.: On the design of a fuzzy relational neural network for automatic speech recognition. Ph.D. thesis, Doctoral Dissertation, The Florida State University, Tallahassee, Fl (1994)
16. Rosales-Pérez, A., Reyes-García, C.A., Gómez-Gil, P.: Genetic fuzzy relational neural network for infant cry classification. In: Pattern Recognition, pp. 288–296. Springer (2011)
17. Zadeh, L.A.: Similarity relations and fuzzy orderings. Information sciences 3(2), 177–200 (1971)
18. Zhu, X., Wu, X., Yang, Y.: Error detection and impact-sensitive instance ranking in noisy datasets. In: AAAI. pp. 378–384 (2004)