Convex Hulls and the Size of the Hidden Layer in a MLP Based Classifier

R. Majalca, and P. Acosta, *Member, IEEE*

***Abstract*—Designing a feedforward neural network has always posed many questions regarding the number of hidden layers and the number of neurons on each hidden layer. While there is no unique and global solution, it is known that the particularities of the problem to solve, especially for classification, offer some guidance about how to solve these questions. One heuristic approach, when only a hidden layer is involved, analyzes how the involved classes are separated from each other using a finite number of hyperplanes, thereby defining the size of the hidden layer on the network. On this article, using computational geometry concepts, an automated and time efficient method is presented and discussed for estimating the quantity of neurons in the hidden layer by computing the number of hyperplanes separating the classes, based on convex hulls and approximation to alpha shapes. Examples on different situations that may arise and the results on using it are illustrated. It can be seen from the results that the proposed method gives very good estimation for the number of hidden neurons.**

***Index Terms*—Convex hulls, Feedforward neural network, Hidden layer size, Hyperplanes.**

I. Introducion

Una red MLP con una sola capa de neuronas ocultas es suficiente para la mayor parte de las aplicaciones en donde una red neuronal de este tipo se puede aplicar, [1]–[3]. En [4], se muestra que una sola capa oculta con suficientes neuronas puede separar regiones arbitrarias no empalmadas de cualquier forma usando funciones de activación continuas acotadas no constantes.

Las redes MLP son populares por su gran campo de aplicación y su enorme versatilidad. En años recientes, se han estado usando como máquinas clasificadoras, con distintos niveles de éxito. Algunos ejemplos son: microestructuras de datos que contienen información genética en pacientes de cáncer [5]; detectores de información encriptada en imágenes digitales [6], señales de electroencefalogramas en pacientes con problemas visuales [7]; en conjunto con una red neuronal convolucional (CNN), para clasificar imágenes satelitales de muy alta resolución [8]; factores influyentes en diferentes ambientes de estrés en alumnos de educación básica [9]; en [10], con distintos problemas de clasificación usando conjuntos de datos de acceso público; señales provenientes de un incinerador de deshechos [11], para evitar calentamientos excesivos en la cámara de incineración; en señales de fonemas [12], deduciendo las probabilidades condicionales para cada clase en el problema.

Un problema común es la determinación del tamaño de la capa oculta de la red MLP y el tiempo requerido en su obtención. Históricamente, una variedad de metodologías han sido propuestas. Por ejemplo, están las basadas en las llamadas reglas de oro [13], que establecen el tamaño de la capa oculta, n_O, usando únicamente la cantidad de elementos en la capa de entrada, n_E y la cantidad de neuronas en la capa de salida, n_S. Por citar sólo algunas de éstas, se tiene la llamada regla piramidal, en donde $n_O = a\sqrt{n_E n_S}$ [14], con $0.5 < a < 2$. En [15], se menciona la regla del promedio, en donde $n_O = (n_E + n_S)/2$ y la regla de los dos tercios $n_o = 2n_E/3 + n_S$ en [16]. Estas reglas son fáciles de usar y sus resultados se obtienen rápidamente, pero no cuentan con un fundamento formal y sus resultados distan de ser óptimos, en general [17]. Además no contemplan otros aspectos relacionados al problema a resolver, que pueden ser de utilidad al momento de establecer el tamaño de la capa oculta [13].

Otros prefieren usar información local del problema a resolver, más allá de sólo n_E y n_S, para aproximar mejor el valor de n_O. Por ejemplo, en [18] se propone $n_o \leq 2\sqrt{(n_s + 2)n_p}$, donde n_p es la cantidad de datos de entrenamiento. En [13] se propone $n_O = n_E + \sqrt{n_P}$, para predicción de inventarios. En [19] se compara el desempeño de catorce de este tipo de reglas.

Otro enfoque muy utilizado es la búsqueda de manera exhaustiva. De esta manera, el valor óptimo para n_O es encontrado luego de un proceso de prueba y error, aunque el tiempo requerido puede ser grande. Aquí, se consideran dos grandes vertientes: los métodos constructivos o de crecimiento y los métodos de poda. Los de crecimiento inician con una capa oculta muy pequeña y paulatinamente agregan más neuronas, hasta encontrar el tamaño óptimo. Los de poda inician con una capa oculta muy grande y gradualmente eliminan neuronas de la capa oculta, hasta encontrar el tamaño óptimo.

Tratando de reducir el tiempo requerido, conservando el objetivo de llegar al número óptimo de neuronas, se han propuesto variantes de estos métodos. Entre los métodos constructivos están el Cascade Correlation Algorithm o CCA [20] y el Dynamic Node Creation o DNC [21] ,que aún siguen siendo referencia en recientes avances. Un ejemplo reciente que usa algoritmos evolutivos con fin de automatizar el proceso se muestra en [23], aunque el tiempo requerido es grande. Una buena comparativa de métodos constructivos hasta el momento de su publicación se tiene en [24]. Entre los métodos de poda de la capa oculta [25], Optimal Brain Damage y el método

R. Majalca, Universidad Autónoma de Chihuahua, Chihuahua, Mexico, rmajalca@uach.mx.

P. R. Acosta, Tecnológico Nacional de Mexico, Instituto Tecnológico de Chihuahua, Chihuahua, Mexico, pacosta@itchihuahua.edu.mx.

Optimal Brain Surgeon siguen siendo apoyo importante en el desarrollo de nuevos métodos. Además, hay combinaciones de métodos constructivos y de poda. Un ejemplo reciente que usa un término de regularización en la función de error para eliminar las neuronas redundantes obtenidas en la construcción es propuesto en [26]. Véase en [27] otro ejemplo en donde no sólo la cantidad de neuronas ocultas es variada, sino, además, la cantidad de capas ocultas, el factor de aprendizaje y hasta el factor de momento. Estas variantes al método constructivo y/o poda, al dirigir la búsqueda en vez de hacerlo exhaustivamente, logran reducir en algunos casos y problemas específicos el tiempo para la obtención del tamaño de la capa oculta.

Una idea poco explorada en su implementación, mencionada en [28] e ilustrada en la figura 1, utiliza el hecho de que en una red clasificadora MLP con clases no linealmente separables, cada una de las neuronas ocultas posiciona un hiperplano en el espacio del problema y la agregación de estos hiperplanos, por las neuronas de salida, logran separar cada clase de las otras.

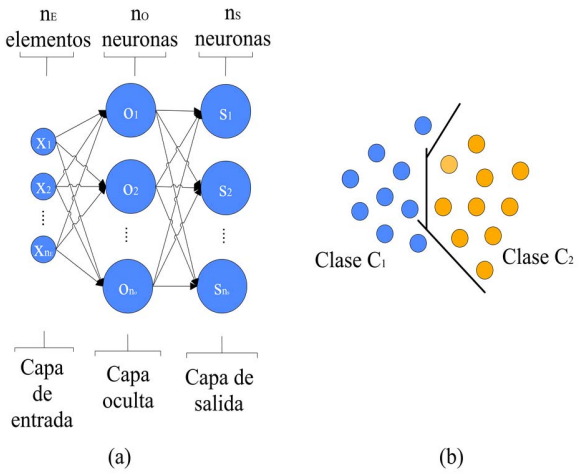

Fig. 1. (a) Red MLP clasificadora con una capa oculta. (b) Representación del problema geométrico para una red MLP con dos dimensiones, $n_E = 2$, dos clases, $n_S = 2$ y tres hiperplanos separadores, $n_O = 3$.

Así pues, utilizando la cantidad de hiperplanos, n_H, que separan las clases involucradas, se puede obtener directamente una estimación del tamaño de la capa oculta, $n_O = n_H$. En principio, se conjetura que una implementación eficiente de este método disminuiría el tiempo requerido para obtener el tamaño de la capa oculta. En [28] se puede ver una muy sencilla ilustración del uso de este método. Desde hace más de 30 años [29], ya se presenta el uso de esta idea, aplicándola en redes con dos capas de neuronas ocultas. Otro ejemplo más del uso de esta idea se puede ver en [30], aplicada en clases con forma convexa. También en [31] y más recientemente en [32] se puede ver el uso de este método. Todos los resultados mencionados son muy específicos. Un resultado formal de aplicación más general se presenta en [33], donde se calculan cotas superiores e inferiores en cuanto a la cantidad de hiperplanos separadores, para redes con sólo una salida para clasificación binaria y suponiendo ciertas características en la forma de las clases a separar, resultando en algunos casos no aplicables y en otros muy conservadoras.

Con el propósito de automatizar el proceso y reducir el tiempo requerido para la obtención del número de neuronas en la capa oculta de una red MLP, conservando el desempeño de la red clasificadora, en este artículo se presenta una metodología eficiente, no existente hasta el momento en la literatura de acuerdo al conocimiento de los autores, para determinar la cantidad de hiperplanos que separan un conjunto en una amplia gama de clases a partir de los datos de entrenamiento, estableciendo así el tamaño de la capa oculta de la red MLP clasificadora multicategoría.

El resto del artículo se organiza de la siguiente manera: la segunda sección presenta el método para calcular la cantidad de hiperplanos separadores. Después se muestran ejemplos y se analizan los resultados de aplicar esta metodología en redes clasificadoras MLP. Por último se presentan algunas conclusiones.

II. Obtención de la Cantidad de Hiperplanos Separadores

Es deseable determinar la mínima cantidad de hiperplanos que separen las clases involucradas para establecer el tamaño de la capa oculta. Un tamaño mínimo en la capa oculta asegura que el problema del sobreajuste a los datos de entrenamiento será minimizado [34].

La metodología propuesta aquí para la obtención de los hiperplanos se divide en cuatro etapas tal como se muestra en la figura 2.

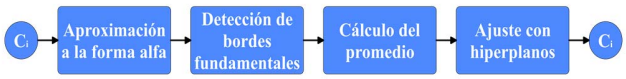

Fig. 2. Las cuatro etapas en el proceso de establecer la cantidad n_H

En la primera etapa se obtiene la aproximación de las clases a las formas alfa. La forma alfa de una nube de puntos es un subconjunto de estos, unidos por segmentos lineales, que se corresponden con los bordes de la nube de puntos completa, proporcionando una aproximación a la forma geométrica de la misma [35], [36].

La segunda etapa en el proceso consiste en seleccionar de entre todos los bordes, aquellos que establecen la posición y forma de la frontera entre clases, a estos se les denominará bordes fundamentales. La tercera etapa consiste en promediar los bordes fundamentales entre las clases, para obtener una primera aproximación a la frontera entre clases. La cuarta y última etapa consiste en ajustar la cantidad de hiperplanos sobre las muestras de la aproximación a la frontera entre clases, n_H. En las siguientes subsecciones se explica en detalle el proceso en cada una de las etapas mencionadas.

A. Aproximación a la Forma Alfa

El cálculo de la forma alfa mediante una librería de geometría computacional disponible, como la librería Qhull [37], simplifica la nube de puntos original a sólo aquellos que se localizan en los bordes de la misma. Sin embargo, la librería Qhull sólo permite este cálculo para datos en dos y tres dimensiones.

Aquí se propone, como parte de la metodología completa para establecer la cantidad de hiperplanos separadores, una técnica para aproximar la forma alfa de una nube de puntos en cualquier número de dimensiones. La técnica que se propone genera una aproximación a la forma alfa obteniendo solamente las muestras o patrones en las orillas de la nube de puntos y ninguna otra cosa normalmente asociada a la forma alfa.

Ya que, en el caso general, cada clase puede tener formas no convexas, se divide el conjunto que conforma la clase en agrupamientos, se obtienen las envolventes convexas de cada agrupamiento y después se unen estas envolventes. La envolvente convexa se define como el mínimo conjunto convexo que contiene a todos los puntos o muestras en la nube de puntos original [38]. Existen múltiples formas de calcular una envolvente convexa, véase, por ejemplo [39], donde se usa la metodología incluida en la librería Qhull. A diferencia de la forma alfa, el cálculo de la envolvente convexa en Qhull sí ocurre en cualquier número de dimensiones lo que es importante en los casos generales actuales. Así pues, en el presente trabajo, utilizando la envolvente convexa, la aproximación a la forma alfa se hace de la siguiente manera:

Algoritmo 1: aproximando la forma alfa
Entrada: muestras de clase
Salida: k envolventes convexas
1 **For** cada clase:
2 Separar las muestras de la clase en k agrupamientos G^j, para $j = 1,2, \dots, k$.
3 Para establecer los agrupamientos y su cantidad k, se utiliza el criterio Calinski-Harabasz, explicado en detalle en [40] y disponible en Matlab, que minimiza la varianza entre agrupamientos, obteniendo la cantidad óptima de agrupamientos con la menor cantidad de huecos valiéndose del algoritmo k-means.
4 **For** cada agrupamiento G^j
5 Encontrar su envolvente convexa $CH(G^j)$. Aquí se usa la librería Qhull.
6 **End for**
7 **End for**
8 La unión de las envolventes convexas es la aproximación a la forma alfa $\dot{\alpha}_{\boldsymbol{P}} = \bigcup_{i=1}^{k} CH(G^i)$.

La figura 3 proporciona una descripción gráfica de este procedimiento, para un caso específico de dos dimensiones.

B. Detección de los Bordes Fundamentales y Bordes Correspondientes

Una vez obtenidas las formas alfas aproximadas, se detectan los bordes fundamentales. Se le llamará borde fundamental al borde en una clase si la recta que lo une con al menos un borde de cualquier otra clase no intersecta la forma alfa de ninguna clase. Un borde fundamental, en una clase, puede unirse con una recta de la forma descrita a más de un borde fundamental en otras clases, pero se elige al más cercano para formar un par de bordes correspondientes. Para la obtención de los bordes correspondientes entre dos clases determinadas se toma como base a la clase con más bordes fundamentales, para buscar para cada uno de ellos sus bordes correspondientes en la otra clase. De esta forma se obtiene mayor cantidad de puntos en la frontera y por tanto mayor detalle. Esto se repite para todas las clases hasta tener todos los conjuntos de bordes correspondientes, entre cada par de clases.

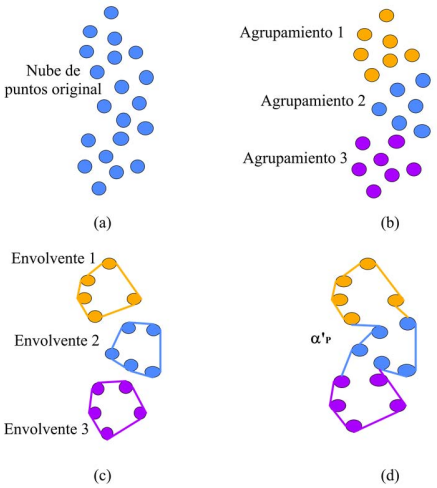

Fig. 3. La aproximación a la forma alfa. (a) Clase original. (b) Separación en agrupamientos. (c) El convex hull de cada cluster. (d) La unión de los convex hulls.

C. Promedio entre Bordes Correspondientes

Para cada par de bordes correspondientes en cada conjunto, se obtiene su promedio. Con el promedio entre cada par de bordes correspondientes se genera el conjunto de puntos que forma la aproximación a la frontera entre cada par de clases. La figura 4 muestra este proceso de aproximar la frontera entre dos clases para un problema en dos dimensiones.

D. Ajustando el Promedio Mediante Hiperplanos

Una vez disponibles los conjuntos con el promedio entre clases, lo que sigue es ajustar la mínima cantidad de hiperplanos en tales conjuntos de muestras.

De acuerdo con lo anterior, para ajustar una cantidad, n_H, de hiperplanos a las muestras de cada conjunto de promedio entre clases, avg, usando una tolerancia, $tole$, en un problema en $\Re^{n_E}$, se hace lo siguiente para cada conjunto de promedio entre clases:

Algoritmo 2: ajustando el promedio
Entrada: avg, el promedio entre clases
Salida: $C_{ajustados}$, las muestras ajustadas del promedio
1 Sea $avg = \{\boldsymbol{a}_1, \boldsymbol{a}_2, \dots, \boldsymbol{a}_{t_p}\}$ el promedio entre clases.
2 Hacer $conjunto = avg$.
3 Hacer $n_H = 0$
4 **If** $|conjunto| < n_E$ **then**
5 Terminar
5 **Else**
6 Hacer $n_H = n_H + 1$
7 Elegir n_E muestras de $conjunto$, preferentemente las más cercanas entre sí para formar el conjunto C_{Base}.
8 Hacer $C_{Resto} = conjunto \backslash C_{Base}$.
9 Hacer $C_{ajustados} = \emptyset$.

Formar el hiperplano H_{Base} que pase por los n_E puntos en C_{Base}.
For cada muestra $\boldsymbol{a}_k \in C_{Resto}$:
  Calcular distancia d_{a_k}, de $\boldsymbol{a}_k$ con H_{Base}.
  If $d_{a_k} \leq tole$ **then**
    Hacer $C_{ajustados} = C_{ajustados} \cup \{\boldsymbol{a}_k\}$
  End if.
  Hacer $conjunto = conjunto \backslash \{C_{Base} \cup C_{ajustados}\}$
End for
End if

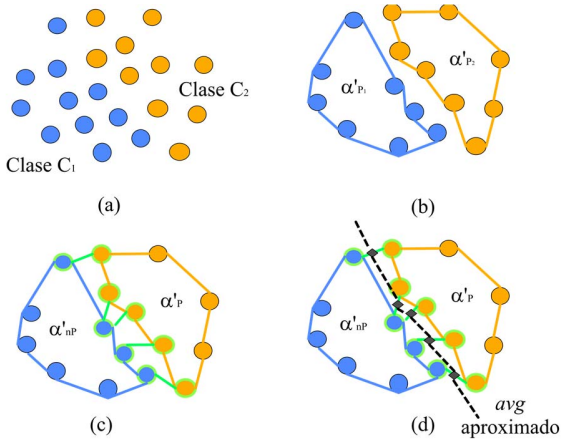

Fig. 4. Aproximando el promedio, avg, entre clases. (a) Clases originales. (b) Formas alfa aproximadas. (c) Bordes correspondientes. (d) Promedio entre clases.

Se puede observar que un valor pequeño de *tole* hace que el ajuste sea muy estricto, quizá con demasiados hiperplanos. Por otro lado, un valor mayor de $tole$ hace que el ajuste sea más relajado, con menos hiperplanos.

El valor del parámetro $tole$, utilizado en este trabajo, es la mitad de la mínima distancia entre los bordes correspondientes de las clases. Así, si la mínima distancia es grande, la tolerancia es grande y si es chica, la tolerancia es pequeña.

E. Clases no Separables

Es común que, en un problema de clasificación real, existan al menos algunas clases no separables. En estos casos, se debe buscar la forma de presentar el problema en forma de clases separables para utilizar el enfoque presentado aquí. En los ejemplos desarrollados en la siguiente sección, se utilizan los esquemas de descarte y mezclado para manipular datos empalmados mostrados y analizados en detalle en [41]. En el presente estudio, las zonas de empalme se obtienen detectando las muestras que se localizan dentro de dos o más formas alfa aproximadas de las clases. En el caso del método del descarte, estas muestras no se consideran en el proceso de generación de los hiperplanos. En el caso del esquema de mezclado, las zonas empalmadas se convierten en nuevas clases y a las clases originales involucradas se les quita la zona del empalme.

III. Ejemplos Ilustrativos

Esta sección contiene algunos ejemplos hechos con redes neuronales MLP con una sola capa oculta. Los conjuntos de datos utilizados fueron obtenidos de [42]. Los ejemplos seleccionados varían en cuanto a la cantidad y tipo de atributos utilizados para la clasificación, el tamaño de los conjuntos de datos y la cantidad de clases a ser clasificadas. 70% de los datos de entrenamiento se utilizaron para entrenamiento y 30% para prueba, lo cual es aceptado como seguro [43]. Además del método presentado aquí obteniendo la cantidad de hiperplanos separadores de las clases, el tamaño de la capa oculta se obtiene con cuatro métodos populares ya mencionados: la regla del promedio, $n_O = (n_E + n_S)/2$, Optimal Brain Surgeon (OBS), Optimal Brain Damage (OBD) y un método constructivo. El método constructivo utilizado inicia con una sola neurona oculta y aumenta una neurona a la vez hasta encontrar el mejor valor de exactitud en clasificación (razón de muestras correctamente clasificadas, con respecto al total de ellas). En los resultados de cada método se muestra n_O; TCO, tiempo de cómputo para obtención de n_O; Rec, recuerdo (razón de las muestras correctamente clasificadas positivamente, VP, entre todas las muestras que debieron ser clasificadas positivamente (VP+ falsos negativos, FN)); Presn, precisión (razón de VP entre todas las muestras clasificadas como positivas (VP+falsos positivos, FP)); F1, media armónica entre la precisión y el recuerdo, 2/(1/Rec+1/Presn), que es utilizada como capacidad de predicción, [44]; Exact, exactitud de clasificación; Dev, desviación estándar del cálculo del tamaño de la capa oculta y ECM, el error cuadrático medio obtenido del proceso de entrenamiento. Las estadísticas se obtuvieron con diez ejecuciones de cada método. Es claro que el tiempo requerido para obtener la cantidad de neuronas con el método del promedio es muy pequeño, independientemente de cualquier parámetro en todos los ejemplos, por lo que no se analizará en comparación con los demás métodos, en los ejemplos mostrados. Todas las simulaciones y cálculos fueron realizadas en una computadora personal, usando un procesador Intel i7-6970HQ de 6th generación, 3.60 Ghz y 32 Gb de RAM, usando Windows 10, versión 1809, 64 bits y utilizando Matlab 2018b, también en 64 bits.

A. Ejemplo 1

Se considera el problema de clasificación usando el conjunto de datos Iris, $\Re^4$, formado por tres clases de flores de la especie iris: Setosa, Versicolor y Virginica. Las cuatro dimensiones (atributos) están definidas por el ancho y largo del pétalo, así como el ancho y el largo de sépalo. Cada clase consta de 50 muestras. En este problema la clase Setosa se encuentra separada de las otras dos, el 25 % de la clase Versicolor se encuentra en la zona de empalme, mientras que el 15 % de Virginica se encuentra empalmada. Para el manejo del empalme se usa el método del descarte, por lo que el clasificador sólo separa a las clases iris Setosa, iris Versicolor e Iris Virginica, sin sus patrones o muestras conflictivas. En la tabla I se observa que el tamaño de la capa oculta es pequeño. Si las clases fueran linealmente separables, se esperaría que la cantidad de neuronas fuera tres. La cantidad de neuronas con los métodos promedio, constructivo e hiperplanos es la misma. TCO es significativamente menor utilizando el método de los hiperplanos comparado con OBD, OBS y el método

constructivo. OBS resultó con la menor capacidad de recuerdo, precisión y predicción.

TABLA I
COMPARATIVA DEL DESEMPEÑO DEL EJEMPLO 1

MÉTODO	n_o	TCO	REC	PRES	F1	EXACT	DEV	ECM
PROMEDIO	4	0.001 s	76 %	82 %	79 %	100 %	0	0,008
OBD	6	300 s	25 %	54 %	34 %	99 %	0,2	0,12
OBS	3	320 s	56 %	79 %	66 %	98 %	0,32	0,10
CONSTRUCTIVO	4	400 s	76 %	82 %	79 %	100 %	0	0,008
HIPERPLANOS	4	90 s	76 %	82 %	79 %	100 %	0,32	0,008

B. Ejemplo 2

Este es un problema de clasificación entre dos especies de cangrejo Leptograspus, la clase anaranjada y la clase azul. Ambas clases utilizan 6 atributos para describir cada individuo, por lo que es un problema en $\Re^6$. El conjunto de datos se compone de 100 muestras para cada clase. La clase azul tiene cerca del 20 % de sus muestras empalmadas, mientras que la clase anaranjada tiene un empalme de cerca del 17 %. Aquí también se utiliza el esquema del descarte. En este caso, con la capa oculta obtenida por todos los métodos tienen la misma exactitud (Tabla II) de clasificación, aunque el método propuesto aquí, OBD y OBS determinan una neurona más para la capa oculta y el tiempo requerido para obtener su tamaño es significativamente menor con el método de los hiperplanos. En comparación con el ejemplo anterior, el tiempo para obtener la cantidad de neuronas es muy similar con 6 atributos en vez de 4, 2 clases en vez de 3 y 200 muestras en vez de 150. Por tanto, aquí no parece que alguno de los factores considerados tenga algún peso considerablemente mayor en relación con los otros.

TABLA II
COMPARATIVA DE DESEMPEÑO. EJEMPLO II.

Método	n_o	TCO	Rec	Presn	F1	Exact	Dev	ECM
Promedio	4	0.001 s	98 %	98 %	98 %	98 %	0	0,033
OBD	5	360 s	97 %	97 %	97 %	98 %	0,32	0,025
OBS	5	370 s	97 %	97 %	97 %	98 %	0,32	0,025
Constructivo	4	200 s	98 %	98 %	98 %	98 %	0	0,033
Hiperplanos	5	98 s	97 %	97 %	97 %	98 %	0,32	0,025

C. Ejemplo 3

En este problema, hay originalmente tres clases a ser separadas: Asistentes de maestro con buen desempeño, aquellos con desempeño regular y aquellos con bajo desempeño, a partir de cinco atributos, $\Re^5$. Contiene 49 muestras para la clase de bajo desempeño, 50 muestras para la clase de regular desempeño y 52 muestras para la clase con alto desempeño. La clase de maestros con alto desempeño tiene un 20 % aproximadamente de empalme, la clase de desempeño medio presenta un empalme del 48 % y la clase de bajo desempeño un empalme de 40 %, aproximadamente. En este caso se utiliza el esquema de mezclado para manejar el empalme de clases, por tanto, la red clasificadora debe separar las clases de alto, medio y bajo desempeño sin patrones conflictivos y además a la clase conflicto. Se puede ver en la tabla III, que el desempeño de la red clasificadora cuya capa oculta ha sido calculada con el método de los hiperplanos, tiene un desempeño más cercano al de la red cuya capa oculta fue construida con el método constructivo, que fue el mejor. Las neuronas obtenidas con el método de los hiperplanos son 7 en vez de 8 obtenidas con el método constructivo. El tiempo para obtener la cantidad de neuronas es la mitad del tiempo requerido por el método constructivo y casi la mitad del requerido para OBD y OBS. En comparación con el ejemplo 1, aquí usando un atributo y una clase más, el tiempo para obtener la cantidad de neuronas ocultas con el método presentado aquí es poco más del doble, para obtener 7 en vez de 4 neuronas. En este caso, el recuerdo, la precisión y F1 son relativamente pequeños para todos los métodos, aunque mejores para los métodos constructivo e hiperplanos.

TABLA III
COMPARATIVA DE DESEMPEÑO. EJEMPLO III.

Método	n_o	TCO	Rec	Presn	F1	Exact	Dev	ECM
Promedio	4	0.01 s	45 %	24 %	31 %	82 %	0	0,23
OBD	5	360 s	45 %	24 %	31 %	82 %	0,32	0,19
OBS	9	370 s	45 %	36 %	41 %	86 %	0,32	0,16
Constructivo	8	400 s	46 %	38 %	42 %	90 %	0	0,16
Hiperplanos	7	200 s	46 %	38 %	42 %	88 %	0,64	0,16

D. Ejemplo 4

Este problema de clasificación es entre dos clases: la clase piel y la clase no piel. La clasificación ocurre sobre imágenes de texturas de piel del rostro de personas de distintas etnicidades y también de texturas que no corresponden a piel. Cada muestra tiene como atributos valores para el rojo, el azul y el verde en la imagen, $\Re^3$. La clase piel consta de 50859 muestras, mientras que la clase no piel consta de 194198 muestras. Las clases se encuentran severamente empalmadas. El 100 % de la clase piel se encuentra empalmada con la clase no piel. Para el empalme se usa el método de mezclado. El problema de clasificación en la primera etapa sigue siendo binario ya que hay que separar a la clase conflicto, que contiene por completo a la clase piel, de la clase no piel. Aquí, el tiempo para obtener la cantidad de neuronas de la capa oculta con el método de los hiperplanos es menos de la mitad del utilizado con el método constructivo y significativamente menor que OBS y OBD, Tabla IV. La exactitud en clasificación es similar, aunque la cantidad de neuronas en la capa oculta es significativamente mayor con el método de hiperplanos presentado aquí y claramente menor con el método del promedio. En este ejemplo, la cantidad de datos es grande para generar la aproximación a las formas alfa, la separación entre las clases es muy pequeña y la forma de la frontera hace que se generen más hiperplanos (y por tanto determinación de mayor cantidad de neuronas ocultas) con el algoritmo presentado aquí.

En este caso, aun cuando la cantidad de datos de entrenamiento es aproximadamente mil veces la del ejemplo 2 y con la mitad de los atributos, el tiempo requerido para la obtención de la cantidad de neuronas es sólo casi tres veces para obtener más del triple de neuronas. Los clasificadores construidos con la regla del promedio y el método OBD muestran significativamente menor capacidad de recuerdo, precisión y predicción.

TABLA IV
COMPARATIVA DE DESEMPEÑO. EJEMPLO IV.

Método	n_o	TCO	Rec	Presn	F1	Exact	Dev	ECM
Promedio	2	0.01 s	46 %	52 %	49 %	90 %	0	0,023
OBD	8	360 s	48 %	55 %	51 %	93 %	0,64	0,012
OBS	16	380 s	71 %	76 %	73 %	91 %	0,64	0,018
Constructiv -	12	600 s	72 %	78 %	74 %	93 %	0	0,012
Hiperplanos	18	280 s	71 %	76 %	73 %	91 %	0,32	0,018

E. Ejemplo 5

Aquí se debe clasificar si una balanza tiene su indicador cargado a la izquierda, cargado a la derecha, o en equilibrio, utilizando cuatro atributos, $\Re^4$. Se tiene un conjunto de entrenamiento con 288 muestras para la clase cargada a la derecha, 288 para la clase cargada a la izquierda y sólo 49 muestras para la clase en equilibrio. La clase cargada a la derecha no tiene muestras empalmadas, la clase cargada a la izquierda tiene 10 % de empalme y la clase en equilibrio tiene 15 % de empalme. Se usa el esquema del mezclado para manejar el empalme, dando un problema de clasificación de cuatro clases incluyendo la zona de empalme. En la Tabla V se muestran los resultados de separar las clases cargada a la izquierda, a la derecha, en equilibrio y la clase conflicto.

TABLA V
COMPARATIVA DE DESEMPEÑO. EJEMPLO V.

Método	n_o	TCO	Rec	Presn	F1	Exact	Dev	ECM
Promedio	3	0.01 s	46 %	64 %	50 %	76 %	0	0,23
OBD	13	420 s	75 %	82 %	77 %	80 %	0,31	0,19
OBS	16	450 s	82 %	86 %	84 %	81 %	0,42	0,19
Constructivo	23	1150 s	96 %	96 %	96 %	85 %	0	0,18
Hiperplanos	24	170 s	96 %	96 %	96 %	85 %	0,31	0,18

Este problema tiene la particularidad de que la clase en equilibrio se encuentra seriamente desfavorecida en cuanto a la cantidad de muestras de entrenamiento, se puede afirmar que es un problema seriamente desbalanceado. Utilizando el método presentado aquí, se obtiene sólo una neurona más en la capa oculta que con el método constructivo y el tiempo requerido para obtener la cantidad de neuronas es mucho menor. La exactitud de clasificación es la misma para las dos estructuras. Comparando con el ejemplo 3, aquí con un atributo menos y aún con obtención de más de tres veces la cantidad de neuronas, el tiempo de cálculo es menor con el método propuesto. Nuevamente, la red clasificadora obtenida por la regla del promedio es claramente inferior en cuanto a precisión, capacidad de recuerdo y predicción, además de exactitud con muy pocas neuronas. Los métodos constructivo e hiperplanos tuvieron el mejor desempeño.

F. Ejemplo 6

En este problema, un clasificador en $\Re^4$ debe separar pacientes de cáncer en una de dos categorías, aquellos que sobrevivieron menos de cinco años, luego de recibir tratamiento y aquellos que sobrevivieron cinco o más años después del mismo. El conjunto de datos consta de 212 muestras para la primera clase y 357 para la segunda. Aquí la primera clase tiene cerca del 75% de sus muestras en la zona de empalme. La tabla VI muestra la comparativa de desempeño.

TABLA VI
COMPARATIVA DE DESEMPEÑO. EJEMPLO VI.

Método	n_o	TCO	Rec	Presn	F1	Exact	Dev	ECM
Promedio	3	0.01 s	67 %	74 %	73 %	90 %	0	0,19
OBD	5	360 s	68 %	75 %	72 %	93 %	0	0,16
OBS	10	370 s	67 %	70 %	68 %	91 %	0	0,19
Constructivo	7	210 s	70 %	78 %	74 %	95 %	0	0,18
Hiperplanos	5	140 s	68 %	75 %	72 %	93 %	0	0,16

Este problema tiene un claro empalme que pone a la primera clase en total desventaja con la clase dos. Aun así, el método propuesto tiene un desempeño apenas por debajo del método constructivo, pero con un tiempo mucho menor para obtención de n_o . El empalme se manejó con el método de mezclado. La red del promedio tiene la menor exactitud y el menor recuerdo junto a la de OBS, que tiene también la menor precisión y predicción.

G. Ejemplo 7

Este es un problema de clasificación en $\Re^4$, entre dos clases: muestras de cheques que se sabe, son falsificaciones, y muestras de cheques que son reales. Existen, en el conjunto de datos, 600 muestras de cheques falsos, y 732 de cheques reales. En este problema, la clase de cheques falsos, que es la que mayor empalme muestra, tiene cerca del 32% de sus muestras en la zona de conflicto. Se utilizó el método de mezclado para resolver el empalme. La tabla VII muestra la comparativa en el desempeño de estos clasificadores.

TABLA VII
COMPARATIVA DE DESEMPEÑO. EJEMPLO VII.

Método	n_o	TCO	Rec	Presn	F1	Exact	Dev	ECM
Promedio	3	0.01 s	46 %	56 %	51 %	88 %	0	0,11
OBD	5	360 s	45 %	56 %	50 %	92 %	0,31	0,09
OBS	9	370 s	100 %	100 %	100 %	98 %	0,31	0,002
Constructivo	9	110 s	100 %	100 %	100 %	100 %	0	0,001
Hiperplanos	9	99 s	100 %	100 %	100 %	100 %	0,31	0,002

En este problema, el método propuesto obtiene un desempeño final igual al del método constructivo, en menos tiempo para obtención de n_o y con la misma capa oculta. Obsérvese nuevamente la baja capacidad de recuerdo, predicción y menor precisión de las redes construidas por la regla del promedio con una red muy pequeña y con el método OBD.

H. Ejemplo 8

Este es un problema de clasificación en $\Re^{13}$, entre tres clases: cada una es una región específica del norte de Italia, de donde proviene la muestra, misma que denota la composición química del vino. Existen, en el conjunto de datos, 59 muestras de la región uno, 71 para la región dos y 48 para la región tres. En este problema, la clase que describe a la región dos, que muestra el mayor empalme, tiene sólo el 5% de sus muestras en la zona de conflicto. Este empalme se manejó con el método de

descarte. La tabla VIII muestra la comparativa en el desempeño de estos clasificadores.

TABLA VIII
COMPARATIVA DE DESEMPEÑO. EJEMPLO VIII.

Método	n_O	TCO	Rec	Presn	F1	Exact	Dev	ECM
Promedio	7	0.01 s	43 %	65 %	52 %	90 %	0	0,021
OBD	2	480 s	35 %	52 %	44 %	95 %	0	0,009
OBS	4	490 s	93 %	90 %	91 %	96 %	0,31	0,008
Constructivo	3	170 s	95 %	97 %	96 %	97 %	0	0,008
Hiperplanos	3	90 s	95 %	97 %	96 %	97 %	0	0,008

En este problema, el método propuesto obtiene un desempeño idéntico al del método constructivo, en menos tiempo para obtención de n_O y con el mismo tamaño de capa oculta. De nuevo, es clara la baja capacidad de recuerdo, precisión y predicción que presentan las redes construidas usando la regla de promedio, con muchas neuronas y el método OBD con una neurona menos.

I. Ejemplo 9

Este es un problema de clasificación en $\Re^{34}$, entre dos clases: muestras de señales de radar que rebotan con la ionosfera de la tierra y regresan y aquellas señales que, al no rebotar en la ionósfera, no regresan. Existen, en el conjunto de datos, 225 muestras de señales que rebotan y 126 de las que no lo hacen. En este problema, la clase de señales que rebotan, que es la que mayor empalme muestra, tiene apenas un 5% de sus muestras en la zona de conflicto, por tanto se utilizó la técnica de descarte para resolver el empalme. La tabla IX muestra la comparativa en el desempeño de estos clasificadores.

TABLA IX
COMPARATIVA DE DESEMPEÑO. EJEMPLO IX .

Método	n_O	TCO	Rec	Presn	F1	Exact	Dev	ECM
Promedio	1	0.01 s	87 %	91 %	89 %	100 %	0	0,055
OBD	1	420 s	87 %	91 %	89 %	100 %	0	0,055
OBS	1	430 s	87 %	91 %	89%	100 %	0	0,055
Constructivo	1	140 s	87 %	91 %	89 %	100 %	0	0,055
Hiperplanos	1	100 s	87 %	91 %	89 %	100 %	0	0,055

Este problema, que tiene un evidente desbalance y un casi inexistente empalme, resultó ser un problema linealmente separable. En todos los casos, una sola neurona bastó para separar las clases. El método propuesto obtuvo más rápido n_O. Para resolver este problema de alta dimensionalidad, fue necesario recurrir a técnicas de programación lineal a fin de aproximar las envolventes convexas, sin recurrir a la librería Qhull.

J. Ejemplo 10

Este es un problema de clasificación en $\Re^{12}$, entre dos clases: aquellas personas que ganan más de 50000 dólares anuales, y aquellas que ganan menos. Existen, en el conjunto de datos, 8140 muestras de la primera clase, y 24421 muestras de la segunda clase. En este problema, la clase de personas que ganan más de 50000 dólares anuales tiene un fuerte empalme, pues cerca del 65% de sus muestras son conflictivas. La tabla X muestra la comparativa en el desempeño de estos clasificadores.

TABLA X
COMPARATIVA DE DESEMPEÑO. EJEMPLO X.

Método	n_O	TCO	Rec	Presn	F1	Exact	Dev	ECM
Promedio	7	0.01 s	84 %	90 %	87 %	91 %	0	0,016
OBD	6	300 s	82 %	89 %	86 %	90 %	0,33	0,016
OBS	10	310 s	83 %	89 %	86 %	90 %	0	0,21
Constructivo	7	180 s	84 %	90 %	87 %	91 %	0	0,015
Hiperplanos	9	120 s	83 %	90 %	87 %	90 %	0,30	0,016

Este problema tiene un claro desbalance, y un severo empalme. Sin embargo, el método propuesto funciona apenas con poco menos exactitud y capacidad de recuerdo que el método constructivo y con menos tiempo para obtención de n_O. El empalme se manejó con la técnica de mezclado. En este caso, el método del promedio obtuvo el mismo desempeño que el método constructivo.

Resumiendo lo observado en los resultados obtenidos, los ejemplos mostrados tienen desde dos atributos hasta treinta y cuatro, así como de dos a cuatro clases a clasificar. También los conjuntos de datos considerados incluyen problemas balanceados y con gran desbalance en la cantidad de muestras en las clases. Además, se utilizan conjuntos de datos sin empalme y con gran empalme entre clases. La cantidad de datos de entrenamiento varía de 150 a 245057. En los resultados, se compara el método propuesto con cuatro métodos para obtención de la cantidad de neuronas ocultas. El método de hiperplanos propuesto aquí, aventajando claramente en el tiempo requerido para obtener la cantidad de neuronas ocultas, tiene resultados de desempeño muy similares a los del método constructivo que dio los mejores resultados. El método del promedio resultó una buena suposición en cuatro de los casos, pero en los otros seis fue de los peores en los diferentes parámetros de desempeño. Los otros dos métodos tuvieron comúnmente problemas en la capacidad de recuerdo y la exactitud en general no fue ni la peor ni la mejor. Al obtener la cantidad de neuronas con cada método, la variación observada en el peor de los casos fue de una neurona de diferencia en un intento con respecto a los otros nueve intentos, lo cual se refleja en Dev. El ECM resultante del entrenamiento es pequeño en lo general, siendo los de los métodos constructivo y de hiperplanos los más pequeños en general.

IV. CONCLUSIONES

Se ha presentado una metodología eficiente y automatizada para establecer el tamaño de la única capa oculta en una red MLP que ha de funcionar como clasificador multicategoría utilizando hiperplanos. En los ejemplos considerados, el tiempo requerido para la obtención de la cantidad de neuronas en la capa oculta es significativamente menor y el desempeño muy similar al obtenido utilizando el método constructivo que tuvo el mejor desempeño en el presente estudio. La metodología obtiene la cantidad de hiperplanos en $\Re^{n_E}$, para $n_E \geq 2$, que separan a las clases involucradas, determinando así la cantidad de neuronas en la capa oculta.

Se utilizan las envolturas convexas para obtener una aproximación a las formas alfa de las clases y con la frontera entre éstas se determinan hiperplanos separadores. La cantidad obtenida de hiperplanos se utiliza como el tamaño de la capa oculta y se hace el entrenamiento de la red para obtener los pesos de las neuronas. El método propuesto para la determinación de la cantidad de hiperplanos es muy simple y de rápido cálculo.

De los resultados obtenidos se puede observar que, en la mayoría de los casos, la cantidad de neuronas determinadas con el método propuesto aquí es muy similar al del tamaño óptimo observado con el método constructivo. Esto implica que el método presentado aquí es una buena opción para la obtención de la capa oculta, reduciendo significativamente el tiempo de obtención de dicha capa y un desempeño de clasificación muy cercano al mejor observado.

En el ejemplo 4 se puede observar que, bajo ciertas condiciones en la forma de las clases, esta manera de determinar los hiperplanos puede resultar en una cantidad significativamente mayor a la óptima. En trabajos siguientes se podría utilizar más de n_E puntos para generar cada uno de los hiperplanos base, aplicando algún método de ajuste tal como la mínima suma de errores cuadrados, obteniendo una menor cantidad de hiperplanos en casos con curvas muy pronunciadas en la frontera sin violar la tolerancia permitida.

Por otro lado, la mayor parte del tiempo se invierte en la obtención de las envolturas convexas, por lo tanto, trabajos posteriores para mejoras en los algoritmos o utilización de otros enfoques para este proceso serían de utilidad. El incremento en dimensiones, clases a clasificar, cantidad de datos y neuronas resultantes, aumenta el tiempo requerido para la obtención de la cantidad de neuronas. Aunque puede verse en el ejemplo 4, que el aumento en tiempo requerido al incrementar el número de datos no es significativo. Además, se puede concluir del análisis presentado que hay otros factores, no considerados, que también afectan en el tiempo para calcular la cantidad de neuronas. Un análisis más profundo sobre otros factores determinantes y cómo afectan, podría ser objeto de un trabajo futuro.

Esta metodología aplica en problemas separables, así que en problemas con clases empalmadas, se requiere preprocesar los datos de entrenamiento usando esquemas tales como descarte o mezclado, para obtener un problema con clases separables. El esquema del descarte pudiera en algunos casos producir problemas sobre simplificados en la clasificación, ya que podría hacer muy amplia la separación de las clases, no reflejando las características del problema original.

Por otro lado, en la implementación de la metodología en lenguajes o paquetes de programación científica, como Matlab, Python, Mathematica, entre otros, se recurre a librerías de geometría computacional, como Qhull para el cálculo de envolventes convexas, distancias, intersecciones o cualquier otro geométrico imperativo. Las limitantes de estas librerías son heredadas por la metodología aquí presentada. La más notoria de tales limitantes es que los cálculos geométricos se complican demasiado para dimensiones más allá de doce. Por tanto, otro trabajo futuro sería mejorar los algoritmos de dichas librerías o prescindir de ellas, tal como se presentó introductoriamente en el caso de 34 dimensiones en el ejemplo 9, el cual se resolvió satisfactoriamente utilizando programación lineal para detectar la envolvente.

REFERENCIAS

[1] G. Cybenko, "Degree of approximation by superpositions of a sigmoidal function," *Math. Control. signals Syst.*, vol. 9, no. 3, pp. 303–314, 1989.

[2] G. Huang, L. Chen, and C. Siew, "Universal Approximation Using Incremental Constructive Feedforward Networks With Random Hidden Nodes," vol. 17, no. 4, pp. 879–892, 2006.

[3] N. J. Guliyev and V. E. Ismailov, "A single hidden layer feedforward network with only one neuron in the hidden layer can approximate any univariate function," *Neural Comput.*, vol. 28, no. 7, pp. 1289–1304, 2016.

[4] G. Bin Huang, Y. Q. Chen, and H. A. Babri, "Classification ability of single hidden layer feedforward neural networks," *IEEE Trans. Neural Networks*, vol. 11, no. 3, pp. 799–801, 2000.

[5] T. H. Hang and L. Q. Don, "Using Dimension Reduction with Feature Selection to Enhance Accuracy of Tumor Classification," *2016 Int. Conf. Biomed. Eng.*, pp. 14–17, 2016.

[6] A. Santos Brandao and D. Calhau Jorge, "Artificial Neural Networks Applied to Image Steganography," *IEEE Lat. Am. Trans.*, vol. 14, no. 3, pp. 1361–1366, 2016.

[7] K. Sabancı and M. Köklü, "The Classification of Eye State by Using kNN and MLP Classification Models According to the EEG Signals," *Int. J. Intell. Syst. Appl. Eng.*, vol. 3, no. 4, p. 127, 2015.

[8] C. Zhang *et al.*, "A hybrid MLP-CNN classifier for very fine resolution remotely sensed image classification," *ISPRS J. Photogramm. Remote Sens.*, vol. 140, pp. 133–144, 2018.

[9] J. Ruby and K. David, "Analysis of Influencing Factors in Predicting Students Performance Using MLP -A Comparative Study," *Int. J. Innov. Res. Comput. Commun. Eng. (An ISO Certif. Organ.*, vol. 3297, no. 2, pp. 1085–1092, 2015.

[10] B. Ploj, R. Harb, and M. Zorman, "Border Pairs Method-constructive MLP learning classification algorithm," *Neurocomputing*, vol. 126, pp. 180–187, 2014.

[11] H. You *et al.*, "Comparison of ANN (MLP), ANFIS, SVM, and RF models for the online classification of heating value of burning municipal solid waste in circulating fluidized bed incinerators," *Waste Manag.*, vol. 68, pp. 186–197, 2017.

[12] Y. Shekofteh, F. Almasganj, and A. Daliri, "MLP-based isolated phoneme classification using likelihood features extracted from reconstructed phase space," *Eng. Appl. Artif. Intell.*, vol. 44, pp. 1–9, 2015.

[13] J. Ke and X. Liu, "Empirical Analysis of Optimal Hidden Neurons in Neural Network Modeling for Stock Prediction," 2008.

[14] A. K. Palit and D. Popovic, *Computational Intelligence in Time Series Forecasting: Theory and Engineering Applications*. Springer London, 2006.

[15] A. J. Thomas, M. Petridis, S. D. Walters, S. M. Gheytassi, and R. E. Morgan, "On predicting the optimal number of hidden nodes," *Proc. - 2015 Int. Conf. Comput. Sci. Comput. Intell. CSCI 2015*, no. December, pp. 565–570, 2016.

[16] F. S. Panchal and M. Panchal, "Review on Methods of Selecting Number of Hidden Nodes in Artificial Neural Network," *Int. J. Comput. Sci. Mob. Comput.*, vol. 3, no. 11, pp. 455–464, 2014.

[17] J. Heaton, *Artificial Intelligence for Humans, Volume 3: Deep Learning and Neural Networks*, vol. 3. 2015.

[18] G. B. Huang, "Learning capability and storage capacity of two-hidden-layer feedforward networks," *IEEE Trans. Neural Networks*, vol. 14, no. 2, pp. 274–281, 2003.

[19] A. Lunt and S. Xu, "An empirically-sourced heuristic for predetermining the size of the hidden layer of a multi-layer perceptron for large datasets," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2016.

[20] S. E. Fahlman and C. Lebiere, "The Cascade-Correlation Learning Architecture," *Adv. Neural Inf. Process. Syst.*, pp. 524–532, 1990.

[21] T. Ash, "Dynamic Node Creation in Backpropagation Networks," *Conn. Sci.*, vol. 1, no. 4, pp. 365–375, 1989.

[22] O. Aran, O. T. Yildiz, and E. Alpaydin, "an Incremental Framework Based on Cross-Validation for Estimating the Architecture of a

Multilayer Perceptron," *Int. J. Pattern Recognit. Artif. Intell.*, vol. 23, no. 02, pp. 159–190, 2009.
[23] E. Real *et al.*, "Large-scale Evolution of Image Classifiers," in *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, 2017, pp. 2902–2911.
[24] A. Khosravi, S. Nahavandi, D. Creighton, and A. F. Atiya, "Comprehensive review of neural network-based prediction intervals and new advances," *IEEE Transactions on Neural Networks*. 2011.
[25] M. Augasta and T. Kathirvalavakumar, "Pruning algorithms of neural networks—a comparative study," *Cent. Eur. J. ...*, vol. 3, no. 3, pp. 105–115, 2013.
[26] H. Z. Alemu, W. Wu, and J. Zhao, "Feedforward neural networks with a hidden layer regularization method," *Symmetry (Basel).*, vol. 10, no. 10, 2018.
[27] R. P. Ferreira, A. Martiniano, A. Ferreira, A. Ferreira, and R. J. Sassi, "Study on Daily Demand Forecasting Orders using Artificial Neural Network," *IEEE Lat. Am. Trans.*, vol. 14, no. 3, pp. 1519–1525, 2016.
[28] M. T. Hagan, H. B. Demuth, and M. H. Beale, *Neural Network Design*. 2014.
[29] R. P. Lippmann, "An Introduction ' to Computing with Neural Nets," 1987.
[30] P. Strobach, "A neural network with Boolean output layer," *ICASSP, IEEE Int. Conf. Acoust. Speech Signal Process. - Proc.*, vol. 4, no. May 1990, 1990.
[31] M. Li, K. Mehrotra, C. Mohan, and S. Ranka, "Sunspot numbers forecasting using neural networks," in *Proceedings. 5th IEEE International Symposium on Intelligent Control 1990*, 1990, pp. 524–529 vol.1.
[32] G. Montúfar, R. Pascanu, K. Cho, and Y. Bengio, "On the Number of Linear Regions of Deep Neural Networks," in *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2*, 2014, pp. 2924–2932.
[33] W. Wenzel, N. Ay, and F. Pasemann, "Hyperplane Arrangements Separating Arbitrary Vertex Classes in n-Cubes," *Adv. Appl. Math.*, vol. 306, pp. 284–306, 2000.
[34] T. Masters, "Practical Neural Network Recipies in C++." p. 490, 1993.
[35] H. Edelsbrunner, D. Kirkpatrick, and R. Seidel, "On the shape of a set of points," *IEEE Trans. Inf. theory*, vol. 29, no. 4, pp. 551–559, 1983.
[36] W. Zhou and H. Yan, "Alpha shape and delaunay triangulation in studies of protein-related interactions," *Brief. Bioinform.*, vol. 15, no. 1, pp. 54–64, 2014.
[37] B. Barber, D. Dobkin, and H. Huhdanpaa, "The Quickhull Algorithm for Convex Hulls," *ACM Ttransactions Math. Softw.*, vol. 22, no. 4, pp. 469–483, 1995.
[38] M. De Berg, O. Cheong, M. Van Kreveld, and M. Overmars, *Computational Geometry: Algorithms and Applications*, vol. 17. 2008.
[39] A. Beltran and S. Mendoza, "SymmetricHull: A convex hull algorithm based on 2d geometry and symmetry," *IEEE Lat. Am. Trans.*, vol. 16, no. 8, pp. 2289–2295, 2018.
[40] T. Calinski and J. Harabasz, "A dendrite method for cluster analysis," *Commun. Stat. - Theory Methods*, vol. 3, no. 1, pp. 1–27, 1974.
[41] H. Xiong, J. Wu, and L. Liu, "Classification with Class Overlapping: A Systematic Study," *2010 Int. Conf. E-bus. Intell.*, pp. 491–497, 2010.
[42] D. Dheeru and E. Karra Taniskidou, "{UCI} Machine Learning Repository." 2017.
[43] S. Suthaharan, *Machine Learning Models and Algorithms for Big Data Classification: Thinking with Examples for Effective Learning*. Springer US, 2015.
[44] Z. C. Lipton, C. Elkan, and B. Naryanaswamy, "Optimal Thresholding of Classifiers to Maximize F1 Measure," in *Machine Learning and Knowledge Discovery in Databases*, 2014, pp. 225–239.

Ricardo Majalca Martínez Profesor en la Facultad de Ingeniería de la Universidad Autónoma de Chihuahua. Recibió el título de ingeniero en sistemas computacionales por la Universidad Autónoma de Chihuahua y el grado de maestro en ciencias en ingeniería electrónica por el Instituto Tecnológico de Chihuahua. (2000). Sus intereses están en el área de desarrollo de software e inteligencia artificial.

Pedro Rafael Acosta Cano de los Ríos, profesor e investigador en el grupo de Automática e Informática Industrial de la División de Posgrado e Investigación del Instituto Tecnológico de Chihuahua, México. Recibió el título de doctor por la Universidad Politécnica de Valencia, España en 2005. Es ingeniero industrial en electrónica y maestro en ciencias en ingeniería electrónica por el Instituto Tecnológico de Chihuahua. Sus intereses actuales en investigación están en el área de control automático dentro de la teoría y aplicación de control de sistemas no lineales y particularmente en control por modos deslizantes.