Una revisión de redes MLP como clasificadores de múltiples clases

A survey on MLP neural networks as multi-class classifiers

Ricardo Majalca Martínez[1,2] y Pedro Rafael Acosta Cano de los Ríos[1]

Recibido: Enero 18, 2016 *Aceptado: Febrero 8, 2016*

Resumen

Se presenta el estado actual de clasificadores de múltiples clases implementados con redes *Multi Layer Perceptron*, MLP. Los clasificadores de múltiples clases basados en redes MLP han sido utilizados en muchos casos con éxito. Se presentan, primero, los aspectos generales y las diferentes formas de implementar clasificadores de múltiples clases, incluyendo las redes MLP. Después se presentan aspectos de arquitectura de las redes MLP clasificadoras incluyendo consideraciones de diseño y organización tales como: capas de entrada, ocultas y de salida, así como la cantidad de neuronas en cada capa. Luego viene una revisión acerca de las metodologías existentes para su entrenamiento, y cómo es que la organización de la red afecta las condiciones de entrenamiento. A continuación, se presentan casos de uso de las redes MLP como clasificadores, sus características y detalles acerca de los parámetros referentes al diseño de la red y también se revisan los resultados de su aplicación. En el material revisado, pareciera ser que el desempeño depende en gran medida de su aplicación específica, aunque no existe trabajo que demuestre esto en forma determinante.

Palabras clave: clasificador múltiples clases MLP, red neuronal, entrenamiento MLP, aplicación clasificadores.

Abstract

The current state of classifiers multiple classes implemented Multi Layer Perceptron networks, MLP, is presented. Multi-class classifiers based on MLP neural network have been successfully used in many cases. First, general aspects and existing approaches of implementing multi-class classifiers are introduced, including MLP neural networks. Afterwards, aspects on MLP network architecture are described, including the design and organization considerations such as input layers, hidden layers and output layers, as well as amount of neurons in each layer. Then comes a review on existing methodologies for training, and how the network organization affects the training conditions. Afterwards, some cases of MLP networks used as classifiers are revised, considering their characteristics and details about network design along with its results in the particular application. Although it seems from the review of literature that the performance of this kind of classifiers largery depends on the specific application, there exist no concluding results on it.

Keywords: MLP multi-class classifier, neural network, MLP training, classifiers application.

Introducción

Las redes MLP, como métodos para implementar un clasificador de múltiples clases han mostrado ser efectivas en aplicaciones diversas. Se han utilizado eficazmente para determinación de patrones de comportamiento, detección de condiciones específicas, fallas y otros aspectos.

[1] Instituto Tecnológico de Chihuahua. Ave. Tecnológico 2909, Col. Magisterial, Chihuahua, Chih., México. 31200. Tel: (614) 417-4353.
[2] Dirección electrónica del autor de correspondencia: rmajalca@itchihuahua.edu.mx.

En general, una máquina clasificadora, recibe como información de entrada un patrón $X \in R^n$ siendo R^n el espacio de los números reales de dimensión *n* a donde pertenecen los patrones de entrada y produciendo como respuesta una etiqueta $Y_i \in \{0,1\}$, donde $Y_i = 1$ indica que el patrón de entrada X pertenece a la categoría i. Los clasificadores, independientemente de la técnica utilizada, se dividen en dos tipos (Allwein, Schapire, & Singer, 2001; Galar, Fernández, Barrenechea, Bustince, & Herrera, 2011), dependiendo del total de categorías que particionan el espacio de entrada. Si *se particiona en sólo dos clases o categorías*, se tiene un *clasificador binario*. Este es el tipo de clasificador más estudiado. El otro tipo de clasificador particiona en más de dos categorías, conocido como *clasificador* de múltiples clases, y es el caso más general posible. En relación con las redes neuronales MLP, no es simple, en lo general, implementar una clasificación de múltiples clases en una red neuronal. Existe bastante información sobre el caso binario (G. Bin Huang, Chen, & Babri, 2000), pero extender esto al caso de múltiples clases no es un proceso trivial, como se muestra en (Jin-Seon Lee & Il-Seok Oh, 2003).

Los clasificadores de múltiples clases ocurren de dos maneras distintas (Allwein *et al.*, 2001; Ou & Murphey, 2007): Una forma consiste en diseñar tantos clasificadores binarios como categorías existan. Cada uno de ellos entrenado para distinguir a una de las categorías del resto. Así, el i-ésimo clasificador separa a la i-ésima clase. Para este i-ésimo clasificador, si $X \in C_i$, entonces $Y_i = 1$, en caso contrario $Y_i = 0$, (Lorena, De Carvalho, & Gama, 2008). De este modo, un patrón X es alimentado a cada clasificador y, posteriormente, éste es asignado a la clase para la cual el clasificador respectivo respondió con la unidad, (Galar *et al.*, 2011; Lorena *et al.*, 2008; Tax & Duin, 2002). El otro esquema consiste en un solo clasificador que deberá ser capaz de distinguir entre todas las categorías. Normalmente en este tipo de clasificador, su respuesta Y es una codificación acerca de la única categoría entre todas las posibles a quien queda asignado el patrón de entrada X, (Windeatt & Ghaderi, 2003). En cuanto a estos dos esquemas de implementación, se puede ver en (Ou & Murphey, 2007) que el desempeño varía en cuanto al criterio de comparación. Por ejemplo, en cuanto al tiempo de entrenamiento, es notorio cómo el esquema de múltiples clasificadores binarios es más rápido que una sola red neuronal. Por otro lado, algunos clasificadores de múltiples clases se ven afectados enormemente cuando en algunas clases existen más muestras que en otras (Mazurowski *et al.*, 2008).

Los clasificadores de múltiples clases suelen implementarse partiendo de diferentes técnicas, además de las redes MLP. Los más utilizados actualmente son las *máquinas de vectores soporte* o SVM (Mayoraz & Alpaydin, 1999; Angulo, Parra, & Català, 2003). Se argumenta que éstas tienen una estructura más sencilla en comparación con las redes neuronales, además, su tiempo de entrenamiento es más corto pues contemplan un único óptimo global como meta (Martínez, Iglesias, Matías, Taboada, & Araújo, 2014), aunque requieren un *kernel* de pre procesamiento. Aunque el uso de múltiples clasificadores binarios continúa siendo la forma más común de implementar clasificadores de múltiples clases con SVM, desde hace más de quince años se tiene en (Mayoraz & Alpaydin, 1999) un detallado informe sobre cómo incorporar características de clasificador de múltiples clases a una SVM, que en principio es del estilo binario. En este mismo artículo, los autores subrayan el evitar el diseño basado en múltiples clasificadores binarios, pues, aunque son más fáciles de implementar con otros esquemas como las redes neuronales, en lo referente a máquinas SVM, indican que no es difícil adaptarse al uso de una sola máquina SVM.

Otra forma de implementar clasificadores de múltiples clases, si puede caracterizar al clasificador como *función o conjunto de funciones de discriminación* estadística, es usando clasificadores bayesianos (Wu, Lin, & Weng, 2004). Es también común que cuando

estas caracterizaciones estadísticas sobre las categorías no permiten parametrizaciones, entonces se recurra a clasificadores de distancia (Lange, Mosler, & Mozharovskyi, 2014) y los clasificadores de ventanas (Yeung & Chow, 2002), entre otros. En otros casos, cuando la información es estructurada, la clasificación se puede llevar a cabo mediante árboles de decisión para clasificación (Jin-Seon Lee & Il-Seok Oh, 2003). Un caso muy diferente e interesante se presenta en (Thabtah, Cowling, & Peng, 2005), en donde un clasificador de múltiples clases ha sido diseñado utilizando un sistema basado en reglas, que de hecho no queda en ninguna de las metodologías mencionadas antes. Es particularmente interesante este clasificador, porque ha adaptado un esquema de procesamiento de datos que normalmente es simbólico, para implementar un procesador de datos numéricos. No es extraño, por otro lado, encontrar clasificadores híbridos modulares en cuanto a su funcionamiento, incluyendo entre sus módulos, estilos diferentes de clasificación. Por ejemplo, en (Cheong, Oh, & Lee, 2004) se presenta un clasificador de múltiples clases que utiliza en una primera etapa una red neuronal SOM y luego, en una segunda etapa, se tiene una máquina de vectores soporte, que, a su vez, hace las veces de un clasificador binario cuyas decisiones se correspondan con las de un árbol de decisión binario.

Algunas arquitecturas se basan sólo en parte en las redes MLP, tal como se muestra en (Ciresan, Meier, & Schmidhuber, 2012) donde se presenta un caso de una red neuronal que utiliza una red MLP como parte superior de la jerarquía de su estructura. Esta red utiliza aprendizaje competitivo, que emula el proceso biológico de percepción e interpretación visual en mamíferos, aplicado en la clasificación de imágenes. Esta red neuronal es particularmente interesante porque pertenece a la categoría de redes que emulan muy bien los procesos fisiológicos de mamíferos, para reconocer imágenes, a partir de un clasificador de múltiples clases. En una aplicación en la ingeniería biomédica en (Jayalakshmi & Santhakumaran, 2011), una red MLP ha sido entrenada eficazmente para clasificar patrones que representan pacientes diabéticos. La red debe clasificar dichos patrones en clases que dependen de síntomas y lecturas provenientes de análisis clínicos e historial médico del paciente. La cuestión nueva aquí, es que los patrones deberían ser normalizados, según ciertos criterios, para que la exactitud del clasificador aumente. Los autores entonces, proponen tanto el diseño de la red clasificadora, como de distintos esquemas de normalización para los patrones de los pacientes.

Una red MLP es una red neuronal bastante conocida, cuya forma de entrenamiento básica, algoritmo de propagación hacia atrás, es de los más conocidos y además, existen variantes al algoritmo que son muy utilizadas también. Pero, como se puede concluir a partir de (Singh, Tiwari, & Shukla, 2012), los aspectos referentes a la organización o arquitecturas de la red, pueden ser un predicamento para quien desarrolla y diseña el clasificador. En el presente artículo se hace una revisión sobre los avances existentes en estructura, funcionalidad y entrenamiento de las redes MLP y su desempeño como clasificadores de múltiples clases. En la siguiente sección se presenta lo existente sobre la estructura general y funcionalidad de las redes MLP, así como un análisis de las estructuras específicas utilizadas en la literatura considerando su desempeño. En la tercera sección se revisan los métodos de entrenamiento utilizados. Posteriormente, se muestra un análisis de las variantes, ventajas y dificultades reportadas en la literatura al utilizar una red MLP como red clasificadora. Por último, se presentan las conclusiones emanadas de la revisión realizada.

Arquitectura de una red MLP

Una red MLP es una red neuronal constituida por capas de neuronas. Cada *k*-ésima capa de neuronas representa un conjunto de neuronas que reciben todas la misma

información de entrada, X_k, por medio de conexiones o pesos, W_k, y cada neurona produce su propia respuesta, $Z_{k,j}$, donde $j=1,2,...,m_k$, siendo el total de neuronas en la *k*-ésima capa de neuronas. Cada neurona en la red utiliza la misma función de activación para procesar el producto del patrón de entrada con las conexiones asociadas a dicho patrón de entrada. En ciertos casos se consideran entradas extras fijas, asociado con un patrón, b_k, de valores llamados *bias*, para cada una de las *m* neuronas en la capa. Si se agregan todas las respuestas individuales $z_{k,j}$ en un patrón, $Z_k \in R^{m_k}$, de respuesta, entonces se puede representar el proceso llevado a cabo por la capa de neuronas como:

$$\boldsymbol{Z_k} = f_k(\boldsymbol{W_k X_k}+\boldsymbol{b_k}) \quad (1)$$

Donde $f_k(.)$ es la función de activación que todas las neuronas de la *k*-ésima capa utilizan. Toda red neuronal tipo MLP presenta una sola capa de entrada, que es la capa con el patrón de entrada X, una o más capas ocultas, una seguida de la otra, conectada cada una con la respuesta de la capa anterior, y una capa de salida, conectada con la última capa oculta, que presenta la repuesta final de la red.

En la expresión (2), se muestra el procesamiento de una red MLP con dos capas. Existe el patrón de entrada X, luego la primera capa, después la segunda capa, y finalmente la capa de salida. Se denota con Z_1 la respuesta de la primera capa, Z_2 la respuesta de la capa 2, y como Y la respuesta de la red neuronal.

$$\begin{aligned} \boldsymbol{Z_1} &= f_1(\boldsymbol{W_1 X}+\boldsymbol{b_1}) \\ \boldsymbol{Z_2} &= f_2(\boldsymbol{W_2 Z_1}+\boldsymbol{b_2}) \\ \boldsymbol{Y} &= f_3(\boldsymbol{W_3 Z_2}+\boldsymbol{b_3}) \end{aligned} \quad (2)$$

Así pues, por su forma de procesamiento, una red MLP es una red de propagación hacia adelante, esto es, la información de entrada se propaga a través de la red, desde la capa de entrada, hasta la capa de salida, pasando por una o más capas de neuronas ocultas. Determinar la cantidad de capas ocultas y el total de neuronas en cada capa, son dos parámetros cuyo criterio de selección no está completamente definido. Estos dos parámetros son un verdadero reto en lo que a diseñar redes neuronales se refiere, y deben ser elegidos muy cuidadosamente. Más capas ocultas hacen que la red funcione más lentamente. Aunque en (Cybenko, 1989) se demuestra que una sola capa oculta es suficiente para lograr la aproximación de cualquier función continua, existen trabajos que muestran un mejor desempeño al utilizar más capas ocultas, específicamente para funciones no convexas (Huang *et al.*, 2000), dependiendo de la función de activación utilizada. Así mismo, si una capa contiene mayor cantidad de neuronas que un número óptimo, el mapeo será llevado a cabo de forma ineficiente, pues las neuronas redundantes llevan a cabo cálculos innecesarios y sin significado (Lee, Oh, & Kim, 1993). También, en este mismo artículo, los autores demuestran que menos capas de neuronas ocultas o menos neuronas ocultas en cada capa, de las absolutamente necesarias provocan un desempeño pobre, pues el mapeo es incompleto o incongruente. En resumen, debe diseñarse la red MLP para que contenga la cantidad de capas ocultas, y en cada capa, la cantidad de neuronas que proporcionen el mejor desempeño. Además, lo que sucede cuando una capa produce su respuesta a partir de la información de entrada es un *mapeo* (Hagan, Demuth, & Beale, 1995) que convierte los datos de entrada en datos de salida con un significado y sentido diferente. Por ejemplo, ese mapeo puede resultar en una mayor o menor dimensión a la salida (Müller, Mika, Rätsch, Tsuda, & Schölkopf, 2001).

Es común definir la cantidad de capas y neuronas en cada capa por prueba y error, aunque distintos autores han utilizado métodos y procedimientos muy variados. Algunos de estos métodos se mencionan a continuación, por ser notables. Se puede ver en (Stathakis, 2009) el uso de un proceso de optimización a partir de un algoritmo genético. La organización y arquitectura de la red es determinada a partir de un proceso de optimización estocástica

evolutiva. En (Panchal, Ganatra, Kosta, & Panchal, 2011) se presenta un análisis comparativo sobre algunos criterios que, a manera de valores heurísticos, proponen una arquitectura u organización para la red MLP en cuanto a la cantidad de neuronas ocultas, si la red sólo contiene una capa oculta. Dependiendo del tipo de aplicación que se le asignará a la red, algunos criterios son fáciles de aplicar, otros más bien implican cálculos o decisiones complejas. En este estudio sobresalen los siguientes criterios: una simple fórmula que relaciona las entradas con las salidas, otro utiliza una red de Hopfield para encontrar una relación entre entradas y salidas, determinando con esto la cantidad de neuronas ocultas. Otro enfoque busca minimizar entropías entre entrada y salida, conforme la cantidad de neuronas ocultas aumenta, con ello, optimizan de manera constructiva la mejor red con la cantidad de neuronas ocultas necesarias. Un dato interesante en este artículo es la conclusión de que una sola capa de neuronas ocultas es suficiente casi siempre y en el resto de los casos, dos cuando mucho, es necesario. Los autores concluyen, además, que el uso de estos valores, de estas decisiones heurísticas, están muy lejos de ser determinantes, ni siquiera son generales, y su aplicación en tal caso, es tan solo para otorgar un valor inicial a un proceso como el de prueba y error; por otro lado, en (Huang, 2003) se presenta un estudio sobre cómo una red neuronal con dos capas ocultas puede aprender información, y esta misma información puede ayudar a determinar cuántas neuronas deben existir en cada capa oculta. Similarmente, en (Singh *et al.*, 2012) la red clasificadora ha sido diseñada siguiendo el enfoque de prueba y error, para definir la cantidad de capas y total de neuronas en la red, utilizando hasta dos capas ocultas y la cantidad inicial de neuronas es determinada a partir de información tácita del problema de aplicación.

En (Chattopadhyay & Chattopadhyay, 2008) se tiene un ejemplo de una aplicación que consiste en predecir el promedio de precipitación pluvial debido a los monzones anuales en la India, datos que ya se tienen disponibles a través de servicios meteorológicos. En ella, los autores han utilizado una sola capa de neuronas ocultas, y la cantidad de neuronas en ella es determinada nuevamente con el método de prueba y error. Probaron múltiples versiones de la red MLP con diferentes cantidades de neuronas en la capa oculta, y llegan a la conclusión de que once neuronas es lo ideal para que la predicción de precipitaciones monzónicas sea lo más correcto posible. En (Sheela & Deepa, 2013), se tiene un estudio muy sustancioso sobre técnicas estadísticas para fijar la cantidad de neuronas ocultas en una red MLP con una sola capa de neuronas ocultas. En este caso se repasan 101 criterios estadísticos que sirven a modo de heurísticas para fijar la cantidad de neuronas ocultas. Estos criterios están basados en el cálculo de valores estadísticos procedentes de los errores que la red comete, específicamente, cuando ésta aproxima a una función continua. Los autores luego proceden a proponer su propio criterio estadístico basado nuevamente en estos errores. La versión de la red MLP que ellos utilizan es una red llamada red de Elman, para predecir velocidades del viento, de manera que comparándolos con los registrados en bases de datos públicas, estas predicciones sean lo más parecidas posibles.

Mención aparte merece la estructura de las llamadas *máquinas de aprendizaje extremo,* o *ELM's*, por sus siglas en inglés de Extreme Learning Machine. Estas son una modificación de lo que en principio es una red neuronal estilo MLP con una sola capa de neuronas ocultas. Estas ELM's han sido aplicadas muy exitosamente en ambas áreas. Lo novedoso y muy sobresaliente acerca de las ELM's es que su capa de neuronas ocultas junto con sus respectivas conexiones con la capa de entrada, no requieren ser entrenadas, ni requieren ajustes durante una etapa de entrenamiento. El valor de las conexiones y el tipo de función de activación en cada neurona ni siquiera se relacionan directamente con la información de entrada y, de hecho, ambos aspectos se determinan

aleatoriamente. Por lo mismo, el aprendizaje en una ELM sólo aplica a las conexiones entre la capa oculta y la capa de salida, lo que hace que el entrenamiento sea en general mucho más rápido que el de una red MLP regular. Para un análisis detallado sobre el tema de las redes ELM véase (G.-B. Huang, Wang, & Lan, 2011). Si bien las máquinas ELM's muestran una velocidad de aprendizaje más rápida que las redes MLP regulares, también es verdad que, en general, la cantidad de neuronas en la capa oculta tiende a ser demasiado grande (Huynh, Won, & Kim, 2008), lo que es una clara desventaja con respecto a las redes neuronales tradicionales en lo referente a su implementación. Algo que resulta muy obvio en todos estos artículos, es la clara preferencia por utilizar una sola capa de neuronas ocultas. En realidad, se ha demostrado que una capa oculta es suficiente para la aproximación de funciones, véase (Barron, 1993; Cybenko, 1989; Anastassiou, 2011). Un estudio detallado respecto a la cantidad de capas ocultas en una red MLP, se muestra en (G. Bin Huang *et al.*, 2000), cuando se utilizan como máquinas clasificadoras. El estudio muestra que las redes MLP con una sola capa de neuronas ocultas pueden aprender a clasificar categorías cuya disposición en el espacio del problema es convexa, mientras que para aprender a distinguir categorías disjuntas o con distribuciones no convexas es necesario una segunda capa de neuronas ocultas. Aunque también los autores han encontrado que, si la función de activación de las neuronas es continua, monotónicamente creciente y acotada, una capa oculta es suficiente para que la red aprenda incluso categorías con distribuciones no convexas. En cuanto a su implementación, la gran mayoría de las redes neuronales se realizan en dispositivos programables de propósito más o menos general, debido a su dificultad de implementarlos en estructuras de hardware específico para redes neuronales (Gardner & Dorling, 1998). Aún con el avance en disponibilidad y facilidad de implementación en dispositivos programables de propósito general, lo que mantiene el interés en la realización de hardware de uso específico para redes neuronales es su potencial en rapidez de procesamiento, costo en producción masiva y confiabilidad de los sistemas con hardware paralelo o distribuido. En (Misra & Saha, 2010) se muestra una revisión de los avances de implementación de las redes neuronales en dispositivos dedicados, conocidos como *Hardware Neural Networks*.

Técnicas de entrenamiento en redes MLP

La obtención del valor óptimo de conexiones y *bias* en una red MLP se hace por medio de un proceso iterativo llamado entrenamiento. La idea consiste en optimizar algún criterio, que casi siempre es una medida del error de clasificación. En caso de redes MLP, lo que originalmente se propuso para implementar aprendizaje fue la técnica de *propagación hacia atrás* o *backpropagation*, en inglés (M T Hagan *et al.*, 1995). Según el algoritmo de propagación hacia atrás, el entrenamiento implica aplicar iterativamente la *regla delta generalizada* (3), para actualizar cada conexión, W_i, y *bias*, b_i, de cada capa, i, en la red, hasta que el error cuadrático medio (4), de clasificación entre el patrón muestra, Y, y el patrón de la respuesta deseada, Y_d, sea el mínimo posible.

$$W_i(k+1) = W_i(k) - \alpha \nabla_{w,i}(k) \qquad (3)$$
$$b_i(k+1) = b_i(k) - \alpha \nabla_{b,i}(k)$$
$$donde$$
$$0 < \alpha < 1$$
$$\nabla_{w,i}(k) = s_i X_i^T$$
$$\nabla_{b,i}(k) = s_i$$
$$s_i = \begin{cases} -2\dfrac{\partial f_i(net_i)}{\partial net_i} e_j & \text{si } i \text{ es la capa de salida} \\ \dfrac{\partial f_i(net_i)}{\partial net_i} W_{i+1}^T \dfrac{\partial f_{i+1}(net_{i+1})}{\partial net_{i+1}} & \text{si } i \text{ es la capa oculta} \end{cases}$$
$$net_i = W_i X_i + b_i$$
$$ecm = \frac{\sum_{j=1}^{P} e_j^T e_j}{P} \qquad (4)$$
$$con \qquad e_j = Y_{d,j} - Y_j$$

En (3), α es el factor de aprendizaje, mientras que $\nabla_{W,i}(k)$ y $\nabla_{b,i}(k)$ son las direcciones hacia donde ocurre la adaptación de los pesos y *bias* en la i-ésima capa de neuronas, respectivamente; s_i se refiere a la sensibilidad en las neuronas en la i-ésima capa. En (4), T denota la transpuesta y P es el total de pares patrón muestra y de la respuesta deseada. El valor del factor de aprendizaje controla el cambio del valor del parámetro a ajustar, de una iteración a otra (M T Hagan *et al.*, 1995). Si es demasiado grande, el ajuste es desmedido, lo que puede provocar que el algoritmo divague mucho. Si es muy pequeño, el ajuste es muy poco, lo que puede provocar que el algoritmo sea muy lento. No existe manera de establecer un buen valor predeterminado para tal parámetro, casi siempre esto se ajusta con un proceso de prueba y error. Desde que esta técnica fue originalmente propuesta, han surgido otros métodos de entrenamiento, algunos basados directamente en propagación hacia atrás y otros utilizando conceptos diferentes. Por ejemplo, en (Jing, Ji-hang, Jing-yuan, & Fei, 2012) se utiliza factor de aprendizaje adaptivo, esto es, no permanece constante en toda la aplicación de la regla delta, sino que varía dependiendo de la magnitud y dirección de los gradientes. En este tipo de algoritmos, el factor de aprendizaje α se obtiene en cada iteración, llamada época, dependiendo tanto de la magnitud como de la dirección del gradiente del parámetro en turno. En los ejemplos reportados donde se utiliza el factor de aprendizaje adaptivo, el algoritmo de propagación hacia atrás converge más rápido.

Levenberg-Marquardt

Otros definitivamente prefieren un enfoque distinto. Por ejemplo, en (M T Hagan *et al.*, 1995; Kumar, 2012) no se utiliza el gradiente descendiente, sino la técnica *Levenberg-Marquardt*. Esta es una técnica que, en lugar de utilizar sólo la información de la primera derivada, como lo hace propagación hacia atrás, utiliza la segunda derivada como en la técnica de optimización de Newton, si así le conviene, de lo contrario, usa la primera derivada como en propagación hacia atrás básico. Así que su comportamiento es ajustable, en determinados casos se comporta como propagación hacia atrás, y en otros como la técnica de Newton, lo que le hace un método rápido y también seguro (Martin T. Hagan & Menhaj, 1994; Kumar, 2012). La desventaja es que, en general, es más complejo tanto en su implementación como en sus requerimientos computacionales.

Algoritmos genéticos

En la técnica de propagación hacia atrás, además de los problemas del factor de aprendizaje, existen otros dos aspectos a considerar: el hecho de que se base en la primera derivada le hace proclive a establecerse en un mínimo local muy pronto en el entrenamiento. Además, si los valores iniciales de las conexiones y los *bias* están muy lejos de un óptimo, la convergencia puede ser muy lenta (Jadav & Panchal, 2012). Esto último ha encaminado a no pocos investigadores a buscar formas alternas de llevar a cabo la optimización de los valores de las conexiones y los *bias* de la red MLP evitando el uso de las derivadas. Así, algunos de estos métodos de entrenamiento de redes MLP se basan en *algoritmos genéticos*, tal como en (Che, Chiang, & Che, 2011; Irani & Nasimi, 2011; Jadav & Panchal, 2012). En (Jadav & Panchal, 2012), los resultados demuestran que el valor óptimo encontrado para las conexiones y los *bias*, como quedan determinados por el algoritmo genético, representa un óptimo mejor al que normalmente encuentra la propagación hacia atrás. Por su parte, una comparación más detallada entre los resultados obtenibles por la propagación hacia atrás y el algoritmo genético se presenta en (Che *et al.*, 2011). Aquí los autores han aplicado ambas formas de entrenamiento en una red MLP con una organización prestablecida. Se han limitado a entrenar tal red con ambos enfoques. Según sus resultados, por un lado, la propagación hacia atrás tiene una velocidad de convergencia mayor a la de los algoritmos genéticos, sin embargo, este último presenta una solución menos proclive al problema del sobreajuste.

Por otro lado, en (Irani & Nasimi, 2011) se ha propuesto un enfoque híbrido de entrenamiento para una red MLP. En este nuevo método, se evita que los valores iniciales de las conexiones y de los *bias* se localicen muy cercanos a un óptimo local, utilizando un algoritmo genético para encontrar los valores iniciales de las conexiones y *bias*. Así, se espera que tales parámetros inicien el proceso clásico de entrenamiento según la propagación hacia atrás, pero con unos valores iniciales más cercanos a un óptimo global. También en (Harp & Tariq, 1992) se tiene otro ejemplo de la aplicación de un algoritmo genético como método para encontrar los valores óptimos tanto en la arquitectura de la red, como en los valores de las conexiones y los *bias* de la misma. Se ha determinado que tal enfoque, que incluye tanto el diseño de la red como el entrenamiento de la misma, arroja excelentes resultados en redes más bien pequeñas, con conjuntos de entrenamiento pequeños, según sus resultados experimentales.

Colonias de hormigas

También se han usado otras técnicas estocásticas de optimización, además de algoritmos genéticos. Algunos han preferido utilizar la técnica de *optimización mediante colonias de hormigas*, o *ACO*, por las siglas en inglés de *Ant Colony Optimization*. En (Blum & Socha, 2005) se muestra un método de entrenar redes MLP utilizando la técnica de optimización ACO. En este caso la aplicación de la red es la de un clasificador de patrones para un sistema médico. Las conexiones y los *bias* de la red son determinados por el optimizador ACO, en lugar del método de propagación hacia atrás. Al igual que en (Harp & Tariq, 1992), los autores demuestran que en la aplicación específica que tienen sus respectivas redes, la solución encontrada por sus técnicas de optimización representan un óptimo mejor al que normalmente localiza propagación hacia atrás. Luego se tiene en (Mavrovouniotis & Yang, 2015) otra aplicación muy interesante. Aquí los autores han decidido entrenar una red MLP utilizando una red con una sola capa de neuronas ocultas y una cantidad específica de neuronas en tal capa. Los autores han utilizado dos enfoques: entrenar la red utilizando sólo una optimización ACO, y luego, utilizan un híbrido entre propagación hacia atrás y optimización ACO. Para este último, de manera similar a lo hecho en (Irani & Nasimi, 2011) con algoritmos genéticos, la técnica ACO sólo se usa para establecer valores iniciales en las conexiones y *bias*, asegurando un punto de inicio mejor. Los autores han aplicado su red a ciertos problemas de clasificación de patrones, y han comparado sus resultados con los obtenidos con otros estilos de clasificadores, por ejemplo, una red MLP clásica, un clasificador basado sólo en un algoritmo genético y otro basado en *swarm intelligence* clásico. Los resultados muestran que el clasificador entrenado con ACO y propagación hacia atrás es más exacto en sus clasificaciones, pero más lento en su entrenamiento. Un estudio muy interesante acerca del uso de las llamadas *metaheurísticas*, como métodos de optimización en el diseño de una red MLP se presenta en (Khan & Sahai, 2012). Aquí se ha entrenado una red MLP, con una arquitectura muy bien definida, utilizando los clásicos métodos de propagación hacia atrás y el algoritmo Levenberg-Marquardt y otros tres métodos basados en metaheurísticas: un algoritmo genético, un algoritmo llamado *Particle Swarm Optimization*, y uno conocido como *Bat Algorithm*. Todos son métodos de optimización estocástica. Según sus resultados, para una aplicación muy particular que implica determinar preferencias de ventas, en clientes en ambientes de compras en línea, para empresas específicas, nadie supera los resultados obtenidos por el método *Bat Algorithm*, tanto en velocidad de aprendizaje como en calidad del resultado final.

Aplicación y comportamiento de las redes MLP como multiclasificadores

En lo referente a multiclasificadores con redes MLP, es oportuno revisar aquello que justifique su uso, muestre las ventajas y también

que exponga sus desventajas. Un estudio clásico sobre esta comparación, enfocada sobre todo a las aplicaciones de las redes neuronales MLP para predicciones de metas financieras, se expone en (Vellido, 1999). En este artículo, los autores revisan y analizan artículos y publicaciones existentes en el uso de redes MLP en el contexto de clasificación en aplicaciones financieras. Ellos concluyen que la principal ventaja es la notable capacidad de las redes MLP para clasificar patrones aún con datos incompletos o contaminados. También hacen notar que las redes MLP no exigen ninguna información *a priori* para clasificar un patrón. La principal desventaja para ellos es el hecho de que las redes MLP, una vez que han encontrado una relación entre datos de entrada y datos de salida, luego del entrenamiento, son como una caja negra. En cualquier caso, el éxito o fracaso asociados a la utilidad o la aplicabilidad de una red neuronal MLP, se ha asociado en gran medida de su aplicación específica y muy rara vez a una aplicación en general. En una aplicación que involucra el diseño de medicamentos (Gertrudes *et al.*, 2012), se presenta una comparación entre distintos métodos de aprendizaje de máquina utilizados con este fin. Uno de ellos es una red MLP, y los resultados en tal comparación para con ella son alentadores. En estos resultados se observa que no es sencillo preferir una SVM a una red MLP, pues en cuanto a sus resultados son ambas muy similares.

Además del problema de determinación de cuántas capas ocultas, y en cada capa, cuántas neuronas deben existir, existen otros aspectos relacionados con el uso de redes MLP que algunos autores consideran una eventual desventaja. Por ejemplo, en (Huynh *et al.*, 2008) se menciona que las principales desventajas de las redes MLP se refieren a la velocidad de aprendizaje, por un lado, y a la capacidad de generalizar, por otro. Los autores afirman que una máquina ELM tiene mejor desempeño en estos dos aspectos. El problema del lento aprendizaje luego es retomado por (Moraes, Valiati, & Gavião Neto, 2013) en un uso comparativo de redes neuronales junto con SVM's, para clasificación de emociones incluidas en textos, además los autores concluyen que las redes neuronales MLP tienden a ser muy susceptibles a datos ruidosos, en contradicción con lo encontrado en (Vellido, 1999) para predicciones de metas financieras.

Por otro lado, en (Khan & Sahai, 2012) se muestra un estudio del uso de una red neuronal MLP en comparación con métodos estadísticos de regresión para una aplicación que implica predecir valores y clasificar pacientes. Es interesante cómo los autores concluyen que, a pesar de las dificultades en la interpretación del proceso llevado a cabo por la red neuronal, su capacidad inédita para encontrar relaciones entre conjuntos de datos es simplemente sorprendente, lo que les hace una mejor opción, en general, a los métodos tradicionales.

Los mismos autores en (Moraes *et al.*, 2013) coinciden con las desventajas que ya se comentaron en párrafos anteriores, sin embargo, agregan que, en general, las redes MLP tienen mejores resultados de clasificación que las SVM, al menos en ese campo de aplicación. En (Karlaftis & Vlahogianni, 2011) concluyen que, en lo referente a clasificación, en aplicaciones estadísticas como detección de incidentes de transporte las redes MLP tienen un desempeño superior al que presentan las técnicas estadísticas, por ejemplo, los modelos *logit*, los de análisis de discriminantes, regresión binomial negativa y regresión logística por escalón. Un enfoque muy interesante se presenta en (Valtierra-Rodriguez, De Jesus Romero-Troncoso, Osornio-Rios, & Garcia-Perez, 2014), sobre cómo acoplar dos redes neuronales, una para procesamiento de información, y otra conectada en serie, para clasificar los patrones que contienen información sobre el desempeño de una planta eléctrica, buscando clasificar las fallas en la planta. Aquí se demuestra cómo una red MLP clasifica de manera muy confiable patrones que representan ruido y fallas en la planta, y evitar así que la misma planta se congestione o falle por sobrecargas.

En una aplicación para las ciencias agro tecnológicas, en (Zhang, Wang, Ji, & Phillips, 2014), se ha diseñado un clasificador de frutas a partir de una red MLP. Esta red recibe como patrones de entrada los componentes principales extraídos luego de un procesamiento de imágenes, que incluye un cambio a escala de gris, una segmentación y una descripción de momentos sobre las formas simplificadas, todo lo anterior llevado a cabo sobre las fotografías de un conjunto de frutas. Los autores demuestran que, dado este pre procesamiento, el clasificador de frutas tiene un desempeño muy elevado, es decir, identifica correctamente las frutas de forma bastante confiable. De esta aplicación, se puede deducir que no existe claramente una diferencia entre el uso de una red MLP, y un clasificador basado en SVM´s, en cuanto a la exactitud del clasificador. Sin embargo, en cuanto al tiempo de entrenamiento, las SVM tienden a entrenarse más rápido que una red MLP. En (Cunha Palácios, da Silva, Goedtel, & Godoy, 2015) se muestran resultados para detección de diferentes fallas en motores de inducción utilizando seis diferentes clasificadores: k-Nearest Neighbors classifier, RIPPER (Rule Incremental Reduced Error Pruning), C4.5 Decision Tree, Naive Bayes, SVM y MLP. Se presentan ocho casos y en cinco de ellos las redes MLP tienen el mejor o segundo mejor desempeño en cuanto a exactitud. En los otros tres casos, todos relacionados con generalización, las redes MLP tienen el tercer o cuarto lugar en cuanto a exactitud. En dos de ellos, el desempeño se puede considerar pobre. En cuanto al tiempo de entrenamiento, las redes MLP mostraron el mayor tiempo requerido en todos los casos, tal como se esperaba. Todos los clasificadores tuvieron desempeño pobre, en al menos uno de los casos.

Conclusiones

Las redes neuronales MLP han sido utilizadas satisfactoriamente para la implementación de clasificadores de múltiples clases. Los principales aspectos que afectan el uso de una red MLP son: la arquitectura exacta de la red, el método de entrenamiento y la aplicación específica. En todos los casos revisados, pareciera que es la aplicación específica la que decide finalmente cómo se desempeña la red y con qué parámetros debe funcionar, por lo que parece que bastaría con decidir en qué aplicación la red ha de funcionar y aprovechando la información local del problema, todas las particularidades de la red quedarían claramente definidas; aunque es necesario un análisis más profundo. Esto es aún un problema abierto, así como la mejor elección de cantidad de capas ocultas y la cantidad de neuronas en cada una de ellas. Igualmente, se requiere mayor investigación sobre los métodos de entrenamiento, para determinar claramente la elección específica. Además, en cuanto al desempeño de las redes neuronales MLP como máquinas clasificadoras, es difícil afirmar categóricamente cuando una red MLP es mejor o peor que alguna otra técnica. La comparación realizada en diferentes trabajos sobre el desempeño de diferentes máquinas clasificadoras ha sido asociada al campo en donde se aplica. Es por ello que pareciera ser que esto es un factor importante, aunque no se han encontrado resultados determinantes al respecto.

Referencias

Allwein, E. L., Schapire, R. E., & Singer, Y. 2001. Reducing Multiclass to Binary: A Unifying Approach for Margin Classifiers. *Journal of Machine Learning Research*, *1*, 113–141. http://doi.org/10.1162/15324430152733133

Anastassiou, G. A. 2011. Multivariate sigmoidal neural network approximation. *Neural Networks*, *24*(4), 378–386. http://doi.org/10.1016/j.neunet.2011.01.003

Angulo, C., Parra, X., & Català, A. 2003. K-SVCR. A support vector machine for multi-class classification. *Neurocomputing*, *55*(1-2), 57–77. http://doi.org/10.1016/S0925-2312(03)00435-1

Barron, A. R. 1993. Universal Approximation Bounds for Superposition of a Sigmoid Function, *39*(3), 930–945.

Blum, C., & Socha, K. 2005. Training feed-forward neural networks with ant colony optimization: An application to pattern classification. *His'05*, 6.

Chattopadhyay, S., & Chattopadhyay, G. 2008. Identification of the best hidden layer size for three-layered neural net in predicting monsoon rainfall in India. *Journal of Hydroinformatics*, *10*(2), 181. http://doi.org/10.2166/hydro.2008.017

Che, Z.-G., Chiang, T.-A., & Che, Z.-H. 2011. Feed-forward neural networks training: A comparison between genetic algorithm and back-propagation learning algorithm. *International Journal of Innovative Computing, Information and Control*, *7*(10), 5839–5850.

Cheong, S., Oh, S., & Lee, S. 2004. Support vector machines with binary tree architecture for multi-class classification. *Neural Information Processing - Letters and Reviews*, *2*(3), 47–51. Retrieved from http://logos.mokwon.ac.kr/pub/NIPLR2004.pdf.

Ciresan, D., Meier, U., & Schmidhuber, J. 2012. Multi-column Deep Neural Networks for Image Classification. *Cvpr*, 3642–3649. http://doi.org/10.1109/CVPR.2012.6248110

Cunha Palácios, R. H., da Silva, I. N., Goedtel, A., & Godoy, W. F. 2015. A comprehensive evaluation of intelligent classifiers for fault identification in three-phase induction motors. *Electric Power Systems Research*, *127*, 249–258. http://doi.org/10.1016/j.epsr.2015.06.008.

Cybenko, G. 1989. Degree of approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals and Systems*, *9*(3), 303–314. http://doi.org/10.1007/BF02836480.

Galar, M., Fernández, A., Barrenechea, E., Bustince, H., & Herrera, F. 2011. An overview of ensemble methods for binary classifiers in multi-class problems: Experimental study on one-vs-one and one-vs-all schemes. *Pattern Recognition*, *44*(8), 1761–1776. http://doi.org/10.1016/j.patcog.2011.01.017.

Gardner, M. W., & Dorling, S. R. 1998. Artificial Neural Networks (the Multilayer Perceptron)— a Review of Applications in the Atmospheric Sciences, *32*(14), 2627–2636.

Gertrudes, J. C., Maltarollo, V. G., Silva, R. a, Oliveira, P. R., Honório, K. M., & da Silva, a B. F. 2012. Machine learning techniques and drug design. *Current Medicinal Chemistry*, *19*(25), 4289–97. http://doi.org/10.2174/09298671280288 4259.

Hagan, M. T., Demuth, H. B., & Beale, M. H. 1995. Neural Network Design, Boston, PWS Publishing Company. Retrieved from http://books.google.ru/books?id=bUNJAAAACAAJ.

Hagan, M. T., & Menhaj, M. B. 1994. Training feedforward networks with the Marquardt algorithm. *IEEE Transactions on Neural Networks*, *5*(6), 989–993. http://doi.org/10.1109/72.329697.

Harp, S. A., & Tariq, S. 1992. Optimizing neural networks with genetic algorithms. In *Proceedings of the 1992 INNS summer workshop* (pp. 41–43).

Huang, G. Bin, Chen, Y. Q., & Babri, H. a. 2000. Classification ability of single hidden layer feedforward neural networks. *IEEE Transactions on Neural Networks*, *11*(3), 799–801. http://doi.org/10.1109/72.846750

Huang, G. B. 2003. Learning capability and storage capacity of two-hidden-layer feedforward networks. *IEEE Transactions on Neural Networks*, *14*(2), 274–281. http://doi.org/10.1109/TNN.2003.809401.

Huang, G.-B., Wang, D. H., & Lan, Y. 2011. Extreme learning machines: a survey. *International Journal of Machine Learning and Cybernetics*, *2*(2), 107–122. http://doi.org/10.1007/s13042-011-0019-y.

Huynh, H. T., Won, Y., & Kim, J.-J. 2008. An improvement of extreme learning machine for compact single-hidden-layer feedforward neural networks. *International Journal of Neural Systems*, *18*(5), 433–441. http://doi.org/S0129065708001695 [pii].

Irani, R., & Nasimi, R. 2011. Evolving neural network using real coded genetic algorithm for permeability estimation of the reservoir. *Expert Systems with Applications*, *38*(8), 9862–9866. http://doi.org/10.1016/j.eswa.2011.02.046.

Jadav, K., & Panchal, M. 2012. Optimizing Weights of Artificial Neural Networks using Genetic Algorithms, *1*(10), 47–51.

Jayalakshmi, T., & Santhakumaran, a. 2011. Statistical normalization and back propagation for classification. *International Journal of Computer …*, *3*(1), 1–5. Retrieved from http://www.ijcte.org/papers/288-L052.pdf.

Jing, L., Ji-hang, C., Jing-yuan, S., & Fei, H. 2012. Brief introduction of backpropagation (BP) neural network algorithm and its improvement. *Advances in Computer Science and Information Engineering*, *169*, 553–558.

Jin-Seon Lee, & Il-Seok Oh. 2003. Binary classification trees for multi-class classification problems. In *Seventh International Conference on Document Analysis and Recognition, 2003. Proceedings.* (Vol. 1, pp. 770–774). IEEE Comput. Soc. http://doi.org/10.1109/ICDAR.2003.1227766.

Karlaftis, M. G., & Vlahogianni, E. I. 2011. Statistical methods versus neural networks in transportation research: Differences, similarities and some insights. *Transportation Research Part C: Emerging Technologies*, *19*(3), 387–399. http://doi.org/10.1016/j.trc.2010.10.004.

Khan, K., & Sahai, A. 2012. A Comparison of BA, GA, PSO, BP and LM for Training Feed forward Neural Networks in e-Learning Context. *International Journal of Intelligent Systems and Applications*, *4*(7), 23–29. http://doi.org/10.5815/ijisa.2012.07.03.

Kumar, M. P. 2012. B Ackpropagation L Earning a Lgorithm B Ased O N L Evenberg M Arquardt, 393–398. http://doi.org/10.5121/csit.2012.2438.

Lange, T., Mosler, K., & Mozharovskyi, P. 2014. Fast nonparametric classification based on data depth. *Statistical Papers*, *55*(1), 49–69. http://doi.org/DOI 10.1007/s00362-012-0488-4.

Lee, Y., Oh, S.-H., & Kim, M. W. 1993. An analysis of premature saturation in back propagation learning. *Neural Networks*, *6*(5), 719–728. http://doi.org/10.1016/S0893-6080(05)80116-9.

Lorena, A. C., De Carvalho, A. C. P. L. F., & Gama, J. M. P. 2008. A review on the combination of binary classifiers in multiclass problems. *Artificial Intelligence Review*, *30*(2008), 19–37. http://doi.org/10.1007/s10462-009-9114-9.

Martínez, J., Iglesias, C., Matías, J. M., Taboada, J., & Araújo, M. 2014. Solving the slate tile classification problem using a DAGSVM multiclassification algorithm based on SVM binary classifiers with a one-versus-all approach. *Applied Mathematics and Computation*, *230*, 464–472. http://doi.org/10.1016/j.amc.2013.12.087.

Mavrovouniotis, M., & Yang, S. 2015. Training neural networks with ant colony optimization algorithms for pattern classification. *Soft Computing*, *19*(6), 1511–1522. http://doi.org/10.1007/s00500-014-1334-5.

Mayoraz, E., & Alpaydin, E. 1999. Support vector machines for multi-class classification. *Engineering Applications of Bio-Inspired Artificial …*. Retrieved from http://link.springer.com/chapter/10.1007/BFb0100551.

Mazurowski, M. A., Habas, P. A., Zurada, J. M., Lo, J. Y., Baker, J. A., & Tourassi, G. D. 2008. Training neural network classifiers for medical decision making/: The effects of imbalanced datasets on classification performance $, *21*, 427–436. http://doi.org/10.1016/j.neunet.2007.12.031.

Misra, J., & Saha, I. 2010. Artificial neural networks in hardware: A survey of two decades of progress. *Neurocomputing*, *74*(1-3), 239–255. http://doi.org/10.1016/j.neucom.2010.03.021.

Moraes, R., Valiati, J. F., & Gavião Neto, W. P. 2013. Document-level sentiment classification: An empirical comparison between SVM and ANN. *Expert Systems with Applications*, *40*(2), 621–633. http://doi.org/10.1016/j.eswa.2012.07.059.

Müller, K. R., Mika, S., Rätsch, G., Tsuda, K., & Schölkopf, B. 2001. An introduction to kernel-based learning algorithms. *IEEE Transactions on Neural Networks*, *12*(2), 181–201. http://doi.org/10.1109/72.914517.

Ou, G., & Murphey, Y. L. 2007. Multi-class pattern classification using neural networks. *Pattern Recognition*, *40*(1), 4–18. http://doi.org/10.1016/j.patcog.2006.04.041

Panchal, G., Ganatra, A., Kosta, Y., & Panchal, D. 2011. Behaviour analysis of multilayer perceptrons with multiple hidden neurons and hidden layers. *International Journal of Computer Theory and Engineering*, *3*(2), 332–337. Retrieved from http://www.ijcte.org/papers/328-L318.pdf

Sheela, K. G., & Deepa, S. N. 2013. Review on methods to fix number of hidden neurons in neural networks. *Mathematical Problems in Engineering*, *2013*. http://doi.org/10.1155/2013/425740

Singh, A. K. A. K., Tiwari, S., & Shukla, V. P. 2012. Wavelet based Multi Class image classification using Neural Network. *International Journal of Computer Applications*, *37*(4), 21–25. Retrieved from http://www.academia.edu/download/30872301/pxc3876555.pdf

Stathakis, D. 2009. How many hidden layers and nodes? *International Journal of Remote Sensing*, *30*(8), 2133–2147. http://doi.org/10.1080/01431160802549278

Tax, D. M. J., & Duin, R. P. W. 2002. Using two-class classifiers for multiclass classification. *Object Recognition Supported by User Interaction for Service Robots*, *2*, 124–127. http://doi.org/10.1109/ICPR.2002.1048253

Thabtah, F., Cowling, P., & Peng, Y. 2005. MCAR: multi-class classification based on association rule. *International Conference on Computer Systems and Applications*, 1–7. http://doi.org/10.1109/AICCSA.2005.1387030.

Valtierra-Rodriguez, M., De Jesus Romero-Troncoso, R., Osornio-Rios, R. A., & Garcia-Perez, A. 2014. Detection and classification of single and combined power quality disturbances using neural networks. *IEEE Transactions on Industrial Electronics*, *61*(5), 2473–2482. http://doi.org/10.1109/TIE.2013.2272276.

Vellido, a. 1999. Neural networks in business: a survey of applications (1992–1998). *Expert Systems with Applications*, *17*(1), 51–70. http://doi.org/10.1016/S0957-4174(99)00016-0.

Windeatt, T., & Ghaderi, R. 2003. Coding and decoding strategies for multi-class learning problems. *Information Fusion*, *4*(1), 11–21. http://doi.org/10.1016/S1566-2535(02)00101-X.

Wu, T.-F., Lin, C.-J., & Weng, R. C. 2004. Probability {Estimates} for {Multi}-class {Classification} by {Pairwise} {Coupling}. *J. Mach. Learn. Res.*, *5*, 975–1005. Retrieved from http://dl.acm.org/citation.cfm?id=1005332.1016791.

Yeung, D.-Y. Y. D.-Y., & Chow, C. 2002. Parzen-window network intrusion detectors. *Object Recognition Supported by User Interaction for Service Robots*, *4*, 385–388. http://doi.org/10.1109/ICPR.2002.1047476.

Zhang, Y., Wang, S., Ji, G., & Phillips, P. 2014. Fruit classification using computer vision and feedforward neural network. *Journal of Food Engineering*, *143*, 167–177. http://doi.org/10.1016/j.jfoodeng.2014.07.001.

Este artículo es citado así:

Majalca-Martínez, R. y P. R. Acosta-Cano de los Ríos. 2015. Una revisión de redes MLP como clasificadores de múltiples clases. *Tecnociencia Chihuahua 9(3)*: 148-159.

Resumen curricular del autor y coautores

Ricardo Majalca Martínez. Terminó su licenciatura en 1997, año en que le fue otorgado el título de Ingeniero en Sistemas Computacionales opción Hardware por la Facultad de Ingeniería de la Universidad Autónoma de Chihuahua (UACH). Realizó su posgrado en Chihuahua, México, donde obtuvo el grado de Maestro en Ciencias de la Ingeniería Electrónica en el área de Computación y Procesamiento Digital de Señales en el año 2000, por el Instituto Tecnológico de Chihuahua. Actualmente cursa su tercer semestre en el Doctorado en Ciencias en Ingeniería Electrónica en el Instituto Tecnológico de Chihuahua. Desde el año 2001 labora en la Facultad de Ingeniería de la UACH y posee la categoría de Académico titular C. Su área de especialización son las ciencias computacionales, en general, y, en particular, el aprendizaje de máquina.

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.