Universidad de las Ciencias Informáticas
Facultad 2

Diseño de la arquitectura e inicialización de los pesos de Redes Neuronales Artificiales de tipo Perceptrón Multicapa aplicando el algoritmo conceptual RGC

TRABAJO DE DIPLOMA PARA OPTAR POR EL TÍTULO DE INGENIERO EN CIENCIAS INFORMÁTICAS

Autores:

Bienvenido Hanley Roque Orfe

Wilber Luna Jiménez

Tutores:

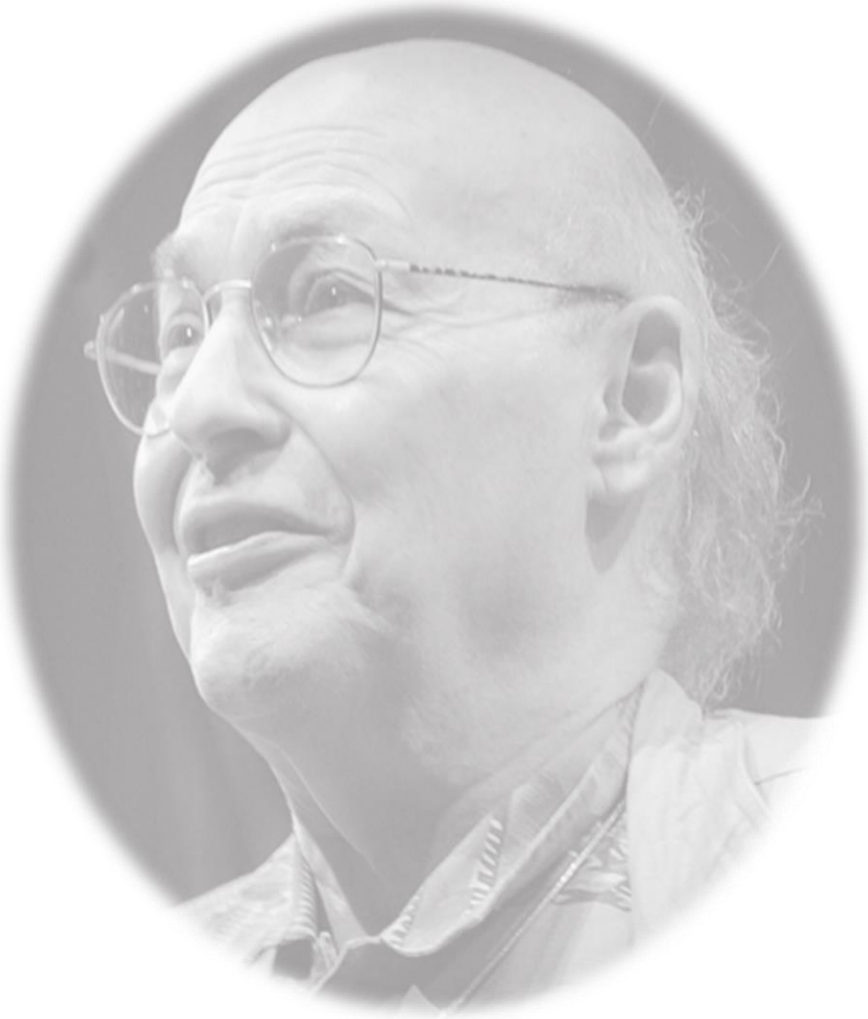
Dra.C. Yunia Reyes González

Dra.C. Natalia Martínez Sánchez

MSc. Maidelis Milanés Luque

La Habana, junio 2018

"Año 60 de la Revolución"



"Las leyes del pensamiento no solo dependen de las propiedades de las células cerebrales, sino del modo en que están conectadas."

Marvin Minsky (1927-2016)

Uno de los padres de la inteligencia artificial

DECLARACIÓN DE AUTORÍA

Declaramos ser autores de la presente tesis y reconocemos a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo. Para que así conste firmo la presente a los ____ días del mes de _____ del año _____.

_____	_____
Firma del Autor	Firma del Autor
Bienvenido Hanley Roque Orfe	Wilber Luna Jiménez
_____	_____
Firma del Tutor	Firma del Tutor
Dra.C. Yunia Reyes González	Dra.C. Natalia Martínez Sánchez

_____

Firma del Tutor

MSc. Maidelis Milanés Luque

DATOS DE CONTACTO

Dra.C. Yunia Reyes González: Graduada de Ingeniera en Ciencias Informáticas en el 2008, actualmente realizando investigaciones en el área de la Inteligencia Artificial, específicamente en Algoritmos de Agrupamiento Conceptual, utilizando el enfoque lógico combinatorio y los Sistemas Basados en el Conocimiento. Profesora de la disciplina de Inteligencia Artificial. Máster en Ciencias y Doctora en Ciencias Técnicas. Vicedecana de Investigaciones y Postgrado de la Facultad 2. Correo electrónico: yrglez@uci.cu

Dra.C. Natalia Martínez Sánchez: Graduada de Licenciatura en Cibernética Matemática en la Universidad Central de Las Villas. Máster en Computación Aplicada y Doctora en Ciencias Técnicas. Profesora Titular. Investiga en el área de la Inteligencia Artificial e Informática Educativa. Ha impartido docencia tanto en pregrado como en postgrado en las ramas de la Inteligencia Artificial, las Matemáticas y Programación. Ha impartido conferencias de pregrado y postgrado en Universidades de Colombia, Perú y Mozambique. Ha participado en eventos nacionales e internacionales, publicando trabajos científicos en revistas y bases de datos de prestigio internacional. Vicerrectora de Formación en la Universidad de las Ciencias Informáticas. Correo electrónico: natalia@uci.cu

MSc. Maidelis Milanés Luque: Graduada de Ingeniera en Ciencias Informáticas en el 2007, actualmente realizando investigaciones en el área de la Inteligencia Artificial, específicamente en el enfoque lógico combinatorio del reconocimiento de patrones y las Redes Neuronales Artificiales. Máster en Ciencias en el año 2017. Profesora de la disciplina de Inteligencia Artificial. Jefa del Departamento de Programación de la Facultad 2. Correo electrónico: mmilanes@uci.cu

Dedicatoria

A nuestro invencible Comandante Fidel Castro Ruz, a la Revolución Cubana, a la Universidad de las Ciencias Informática y a la Federación Estudiantil Universitaria por darme la oportunidad de cumplir mi sueño. A mis padres por ayudarme a formarme como un profesional. A toda mi familia por contribuir a mi formación. A todas las personas que ven, con esta tesis, sus sueños realizados.

Bienvenido Hanley Roque Orfe

A mi madre y mi hermanita Brianna . A mis dos bisabuelas Lume y Zoila . A todos mis profesores en especial a Zenaida y Odilia mi maestra de preescolar. A mi primo Eidel Ramos. A Fidel y Raúl . A la Revolución Cubana, a la Universidad de las Ciencias Informáticas y todos los amigos que de una forma u otra me apoyaron en mis estudios.

Wilber Luna Jiménez

Agradecimientos

La agradezco el presente resultado:

A nuestro eterno comandante Fidel Castro Ruz por haber sido el líder de la Revolución Cubana, por haber creado la batalla de idea con ese fervor haber creado la universidad de mis sueños, la Universidad de las Ciencias Informáticas. A la FEU por ser la organización que me ha enseñado y contribuir en mi formación.

A mis tutoras: Yunia por tener confianza en mí, por ayudarme cuando más lo necesitaba a pesar de aguantar mi impertinencia, por lograr que yo aprendiera y dedicar parte de su tiempo para lograr este resultado. Maydelis por ayudarme y brindarme todo su tiempo para lograr este resultado, por darme el apoyo en las buenas y las malas. Natalia por ser comprensiva y brindar de su tiempo para atenderme, por ayudarme siempre que la he molestado. Muchas gracias a las tres por ser quienes son.

A mi mamá por apoyarme en todas las decisiones que tome, por su infinito cariño, por sufrir mis reveses y gozar mis logros como si fuera yo, por existir y apoyar en un 100% mi formación. A mi papá por apoyar todas mis decisiones y contribuir en toda mi formación. A mis abuelos por estar ahí conmigo en la etapa más en los momentos que me hace falta. A toda la familia en general por estar ahí y de una forma u otra han aportado algo a mi formación como profesional.

A mi hermano Manchón a pesar de pelear demasiado, gracias por el apoyo y la ayuda que me has brinda al igual que Lourdes y Victor R. A Gema por ser como mi segunda madre y darme consejos bueno para aplicar para la vida. A el profe Lester y Yuya por ser mis guías en todos los pasos que he dado en la uci y tener que aguantar mi resabio. A la profe Madelin Haro por haberme apoyado, enseñado y por haber formado parte de mi vida y de la cual he aprendido sabias lecciones para la vida. A profe Darvis por ser como un padre para mí, mucho que me enseño y guio en todas las cosas aquí en la universidad.

A mis hermanos de por vida José E. Lenzano y Victor A. Mesa Cisneros por estar desde primer año ahí conmigo en todos los momentos malos y buenos, por contribuir en mi formación y enseñarme a tomar las cosas con calma. A Roman a por apoyarme en las decisiones y darme consejo cuando lo he necesitados.

A todos los profesores que he tenido por encaminarme en el área del saber. A todas mis amistades, en especial Jennifer O, Glenda, Yunior, Irialys, Raydel, Odeynis. A mis compañeros de aula, en especial: Anamelis, Analaura, Oswald, Glenda por el apoyo y la confianza que me brindaron los todos estos años de la universidad. A mi grupo de trabajo de la FEU gracias por apoyarme en todo.

Por último y más importante son a mis compañeros de tesis: Wilber por su tolerancia a mi resabio, por estar ahí en todo momento, por apoyarme en todo sin duda ninguna las actividades, sin él no hubiese podido lograr este resultado y Osniel por enseñarme a coger las cosas con calma, también me apoyo en las buenas y las malas. Gracias a los dos por aportar en un 100% a mi formación.

Gracias a todos por existir!!!

Bienvenido Hanley Roque Orfe.

Agradecimientos

La agradezco el presente resultado:

Quiero agradecer primeramente a mi mamá por todo el amor y la dedicación que me ha brindado desde que estuve en sus brazos por primera vez. Por estar conmigo en cada momento difícil de mi vida dándome ánimos y aliento para seguir adelante superando cualquier obstáculo.

A mis dos bisabuelas Lume y Zoila que de seguro estuvieran orgullosas de verme en este momento y que el tiempo no les permitió acompañarme físicamente.

A mi hermanita Brianna que es mi motivo de superación para darle el mejor de los ejemplos. A mi padrastro Osleydi por ser como un padre para mí y del que nunca me ha faltado el apoyo. A mi abuela Ramona por todo lo que me ha ayudado, a mis tíos Eugenio, Marlenis, Virgilio, Frank, por estar siempre presentes ayudando en todo lo que necesite. A mi prima María quien me dio la bienvenida y me guio los primeros meses en la Universidad.A mi prima Mirian que muchas veces ocupé de su tiempo para ayudar en mis cosas de la escuela.

Agradezco a mi papá por preocuparse por mis estudios y darme sus consejos siempre que los he necesitado. A mi abuela Xiomara por su inmenso cariño, a mi tía Zaida y Dianelis que han estado siempre al tanto de mi formación así como mi prima Zeleida.A mi tía Zenaida quien fue además mi maestra desde primer grado y que me enseñó los primeros pasos, las primeras letras, los primeros cálculos. A mi primo Eidel Ramos y mi amigo Jose Rodríguez por ser ambos mi ejemplo a seguir como persona y profesional, por toda su ayuda espiritual y material. A toda mi familia en general que de una forma u otra ha aportado su granito de arena para mi formación.

A nuestras tutoras Yunia, Maidelis y Natalia, por su paciencia y sabiduría y por su esfuerzo para lograr el mejor resultado.

A los profesores: Víctor Roque, Lester Gonzáles, Madelín Haro,Yanio Hernández, Darvis Dorvigni, María Teresa, Sayda Coello, Abel Belázquez, Abelino, y a los que de una forma u otra apoyaron nuestra educación.

A Jose Ernesto que no escatimó de su tiempo para brindarnos su ayuda. A Osciel, Yoanys, Jessica y todos los compañeros del grupo en general.

A mi compañero de tesis Bienvenido por acompañarme en este proyecto siempre dando lo mejor de sí para que todo saliera lo mejor posible así como a Osniel por también poner su parte en este trabajo y brindar su mano amiga en todo momento.

Gracias a todos por existir!!!

Wilber Luna Jiménez.

Resumen

Las redes neuronales artificiales son sistemas conexionistas que se basan en el funcionamiento de las redes neuronales biológicas para solucionar problemas complejos. Dentro de ellas, uno de los modelos más utilizados para la solución de problemas supervisados es el Perceptrón Multicapa teniendo en cuenta su gran capacidad como aproximador universal. En este modelo se presentan dos problemas que son significativos tanto en la convergencia como en la ejecución de la red neuronal artificial, uno relacionado con el diseño de la arquitectura y otro referente a la determinación de los pesos sinápticos iniciales. El presente trabajo se propone como objetivo general aplicar el algoritmo RGC en el diseño de la arquitectura e inicialización de los pesos del Perceptrón Multicapa para mejorar la eficacia en la clasificación. Se utilizaron como métodos científicos el analítico-sintético, hipotético-deductivo, observación, histórico-lógico y preexperimento, los que permitieron dirigir la investigación. Se describen las dos etapas del algoritmo RGC: determinación extensional y determinación intencional. Para validar los resultados se utiliza el método de validación cruzada y la prueba no paramétrica de Friedman. Se establecen comparaciones en cuanto a la cantidad de objetos clasificados correctamente entre tres configuraciones del Perceprón Multicapa, uno sin aplicar los algoritmos conceptuales, otro aplicando el algoritmo LC-Conceptual y el último aplicando el algoritmo RGC. Los resultados demuestran que la solución propuesta garantiza una mayor eficacia en la solución de problemas.

Palabras clave: algoritmo RGC, clasificación, Perceptrón Multicapa.

Abstract

Artificial neural networks are connectionist systems that rely on the functioning of biological neural networks to solve complex problems. Among them, one of the most commonly used models for supervised problem solving is the Multilayer Perceptron, given its high capacity as a universal approximator. This model presents two problems that are significant both in the convergence and in the execution of the artificial neuronal network, one related to the design of the architecture and the other referring to the determination of the initial synaptic weights. The general objective of this work is to apply the RGC algorithm in the design of the architecture and initialization of the Multilayer Perceptron weights to improve the efficiency of the classification. Analytical-synthetic, hypothetical-deductive, observation, historical-logical and pre-experimental methods were used as scientific methods, which allowed the research to be conducted. The two stages of the GCR algorithm are described: extensional determination and intentional determination. The cross validation method and Friedman's non-parametric test are used to validate the results. Comparisons are made in terms of the number of objects correctly classified between three configurations of the Multilayer Percepron, one without applying the conceptual algorithms, another applying the LC-Conceptual algorithm and the last applying the RGC algorithm. The results show that the proposed solution guarantees greater efficiency in problem solving.

Keyword: RGC algorithm, classification, Multilayer Perceptron.

Contenido

Índice de figuras

Índice de tablas

Introducción

Las Redes Neuronales Artificiales (RNA) son modelos computacionales aplicables en determinados problemas complejos. Requieren de poco espacio de almacenamiento y presentan una rápida respuesta computacional. Son capaces de aprender de la experiencia, de generalizar el conocimiento, de abstraer características esenciales de un conjunto de entradas que pueden contener información irrelevante. Estas ventajas posibilitan su utilización en problemas prácticos concretos, que normalmente no pueden ser resueltos con sistemas tradicionales, como los relacionados con la clasificación, la estimación funcional y optimización en general.

Las RNA son sistemas conexionistas que se basan en el funcionamiento de las redes neuronales biológicas para la solución de problemas complejos (del Brío y Molina 2001;Domínguez et al. 2015). Los tres conceptos claves de los sistemas nerviosos, que se pretenden emular en los artificiales, son: paralelismo de cálculo, memoria distribuida y adaptabilidad al entorno (del Brío y Molina 2001). Entre las ventajas que se presentan al usarlas están la tolerancia a fallos, el aprendizaje adaptativo, la autoorganización y el trabajo con información incompleta.

Están compuestas por un conjunto de unidades de procesamiento (U), llamadas neuronas, que se encuentran conectadas entre sí. Por lo general las conexiones están establecidas mediante arcos dirigidos con un peso asociado (W_{ij}), que indica la importancia de la información que llega desde una neurona U_i hasta la neurona U_j. Entre los elementos que definen un modelo neuronal se encuentran la arquitectura, los pesos sinápticos iniciales y la regla de aprendizaje. Se denomina arquitectura a la topología, estructura o patrón de conexionado de una red neuronal (del Brío y Molina 2001). Por su parte los pesos sinápticos iniciales son valores numéricos que se le asignan a las conexiones antes del entrenamiento y la regla de aprendizaje es el mecanismo por el cual un modelo neuronal artificial es capaz de reajustar sus pesos en función de las entradas presentadas, hasta que permanezcan estables.

El Perceptrón Multicapa (MLP) es uno de los modelos neuronales más utilizados en la solución de problemas. Esto se debe fundamentalmente a su capacidad como aproximador universal, ya que cualquier función continua puede aproximarse con un Perceptrón Multicapa con al menos una capa oculta de neuronas. Se utiliza en problemas con aprendizaje supervisado tales como: clasificación y predicción de enfermedades, temperaturas, deslizamiento de terrenos, clasificación de patrones (Pérez-Valls 2013), entre otros. El MLP está determinado mediante una arquitectura estructurada por un conjunto de neuronas organizadas en tres tipos de capas (entrada, oculta y salida), con conexiones pesadas entre sí y flujo de datos unidireccional, desde la capa de entrada hasta la capa de salida (del Brío y Molina 2001). Los pesos

sinápticos iniciales son generados en valores pequeños y de forma aleatoria y el algoritmo de aprendizaje que emplea este modelo es el de Retropropagación del error (Backpropagation) (del Brío y Molina 2001).

El funcionamiento del Backpropagation se basa en la minimización del error, mediante una función que relaciona las entradas a la red y los pesos asociados a las conexiones, a partir de una arquitectura fija que se debe establecer con anterioridad. Por lo tanto, lograr una combinación eficiente o no entre el diseño de la arquitectura y la inicialización de los pesos sinápticos, puede conllevar al éxito o al fallo del modelo neuronal, que además es una problemática que se analiza en la actualidad en varias investigaciones (Milanés-Luque 2017).

La elección del número de neuronas correspondientes a las capas de entrada y salida, está en correspondencia con las variables que definen el problema. Sin embargo, para determinar la cantidad de neuronas en la capa de entrada, en disímiles situaciones se dispone de variables que no aportan información relevante y su utilización podría atentar negativamente contra el tiempo de aprendizaje de la red, siendo conveniente el análisis previo de las mismas para determinar aquellas que son necesarias y suficientes para la solución del problema (Milanés-Luque 2017).

Por otra parte, la selección del número de capas ocultas y el número de neuronas pertenecientes a las mismas deben ser elegidos por el diseñador, puesto que no existe una regla que determine el número óptimo de neuronas ocultas para resolver un problema dado. En la mayor parte de las aplicaciones prácticas, estos parámetros se determinan por prueba y error. Partiendo de una arquitectura ya entrenada, se realizan cambios aumentando o disminuyendo el número de neuronas ocultas y el número de capas hasta conseguir una arquitectura adecuada para el problema a resolver, que pudiera no ser la óptima, pero que proporciona una solución (Pérez-Valls 2013).

Para la inicialización de los pesos sinápticos, generalmente se destinan valores aleatorios pequeños, debido a que los grandes valores absolutos de pesos pueden hacer que las neuronas sean altamente activas o inactivas para todas las muestras de entrenamiento y por ende insensibles a este proceso. Además, la elección del vector de peso inicial (W_0) puede acelerar la convergencia del proceso de aprendizaje hacia un mínimo global o local si se ubica dentro de la atracción basada en ese mínimo. Por el contrario, si W_0 comienza la búsqueda en una región relativamente plana de la superficie de error, la adaptación de los pesos de conexión puede ser lenta (Cabrera et al. 2011;Coello et al. 2016).

El problema del modelo de la arquitectura e inicialización de los pesos en el MLP con Backpropagation como algoritmo de aprendizaje, ha sido abordado con diferentes técnicas de la Inteligencia Artificial, aprovechando las ventajas de una y otra para mitigar las deficiencias y alcanzar un modelo eficaz en la clasificación. Entre

ellas se encuentran: combinación de algoritmos genéticos (Correa y Gonzalez 2011), colonia de hormigas (Ghanou y Bencheikh 2016). Sin embargo, en dichas técnicas no se aprecia el uso del enfoque lógico combinatorio para el diseño de la arquitectura e inicialización de los pesos del MLP.

En Milanés-Luque,(2017) luego de haberse realizado un estudio sobre las problemáticas presentadas en las técnicas anteriores, se propone la aplicación de los algoritmos conceptuales en el diseño de un modelo neuronal, específicamente el algoritmo LC-Conceptual del Reconocimiento Lógico Combinatorio de Patrones (RLCP), como resultado en esta investigación se aprecia una mejora de la eficacia en la clasificación. La cantidad de patrones clasificados correctamente por el modelo propuesto es mayor con respecto al modelo sin aplicar el agrupamiento conceptual, comportándose mejor en bases de datos numéricas que en bases de datos con atributos nominales y mezclados.

El algoritmo LC-Conceptual (Martínez-Trinidad y Sánchez-Díaz 2001) tiene la peculiaridad de estar basado en los conceptos de la clasificación no supervisada del enfoque lógico combinatorio y retoma algunas ideas propuestas por Michalski para generar conceptos, interpretables por los especialistas, en términos del conjunto de rasgos original (Ruiz-Shulcloper 2009). En el LC-Conceptual la creación de los conceptos es realizada a través de los testores y/o testores típicos, siendo los rasgos que los componen excluyentes y clasificadores. También son utilizados como alternativa para el cálculo de su propia importancia informacional y la de los conceptos.

Este algoritmo tiene como problemática que al aplicar el operador Refunión Condicionada (RUC) a cada agrupamiento pueden quedar objetos no cubiertos por el concepto construido (Pons-Porrata 2004). Esto significa que el concepto que se construye no caracteriza a todos los objetos del grupo, esta limitación afecta la eficacia en la clasificación de nuevos objetos. Por ello la solución propuesta por (Milanés-Luque 2017) puede verse afectada por dicha condición.

En Pons-Porrata,(2004) se propone el algoritmo de agrupamiento conceptual RGC, que pretende responder a un conjunto de descripciones de objetos en términos de variables simbólicas para encontrar una estructura conceptual en el espacio de representación inicial. El mismo consta de dos etapas: determinación extensional y determinación intencional. En la primera etapa los agrupamientos son creados usando alguna medida de similitud entre objetos y un criterio de agrupamiento para generar la estructuración. En la segunda, los conceptos asociados a cada agrupamiento son construidos y generalizados. Según Pons-Porrata,(2004) los conceptos que se obtienen por el RGC logran caracterizar a todos los objetos del grupo, alcanzando un mayor grado de eficacia al clasificar nuevos objetos.

Debido a lo expuesto anteriormente se plantea el siguiente **problema a resolver:** ¿Cómo contribuir a mejorar la eficacia en la clasificación en una Red Neuronal Artificial de tipo Perceptrón Multicapa?

El problema se enmarca en el **objeto de estudio:** Redes neuronales artificiales de tipo Perceptrón Multicapa. Se determina como **objetivo general:** Aplicar el algoritmo RGC en el diseño de la arquitectura e inicialización de los pesos del Perceptrón Multicapa para mejorar la eficacia en la clasificación de los objetos. La investigación está enmarcada en el siguiente **campo de acción**: Arquitectura e inicialización de los pesos del Perceptrón Multicapa utilizando el algoritmo conceptual RGC.

Para cumplir con los objetivos propuestos se plantean los siguientes **objetivos específicos:**

1. Sistematizar los principales referentes teóricos y metodológicos relacionados con las redes neuronales artificiales de tipo Perceptrón Multicapa y el algoritmo conceptual RGC.
2. Diseñar la arquitectura e inicializar los pesos de Redes Neuronales Artificiales de tipo Perceptrón Multicapa aplicando el algoritmo conceptual RGC
3. Valorar el resultado de la aplicación práctica de la arquitectura e inicialización de los pesos de Redes Neuronales Artificiales de tipo Perceptrón Multicapa, aplicando el algoritmo conceptual RGC a partir de la ejecución de preexperimentos en conjuntos de datos internacionales.

Como hipótesis en la presente investigación se plantea que:

Con la determinación de la arquitectura e inicialización de los pesos de Redes Neuronales Artificiales de tipo Perceptrón Multicapa aplicando el algoritmo conceptual RGC se mejorará la eficacia en la clasificación de objetos.

Variable independiente: Arquitectura e inicialización de los pesos de Redes Neuronales Artificiales de tipo Perceptrón Multicapa.

Variable dependiente: Eficacia en la clasificación de objetos.

Para el desarrollo de la investigación se combinan diferentes **métodos teóricos y empíricos.** Los fundamentales son**:**

Métodos Teóricos:

- **Analítico–Sintético:** para el análisis de documentos y teorías, permitiendo la extracción de los elementos más importantes que se relacionan con el algoritmo RGC para la clasificación no supervisada y las redes neuronales artificiales de tipo Perceptrón Multicapa. Además, es empleado

para analizar las tecnologías, herramientas y metodologías para la aplicación del algoritmo.

- **Sistémico-Estructural-Funcional:** para relacionar hechos aparentemente aislados y se formula una teoría que unifica los diversos elementos.
- **Hipotético-Deductivo**: para hacer predicciones siguiendo reglas lógicas de deducción, las que posteriormente son sometidas a verificaciones empíricas con bases de datos internacionales.
- **Histórico-Lógico:** para el estudio del surgimiento del Perceptrón Multicapa y analizar su funcionamiento con el fin de dar cumplimiento al objetivo de la investigación.
- **Modelación:** para representar gráficamente los elementos que componen la propuesta de solución.

Método Empírico:

- **Observación:** se observan las deficiencias que presenta la aplicación del algoritmo LC-Conceptual en la determinación de la arquitectura de una red neuronal de tipo Perceptrón Multicapa.
- **Preexperimento:** se utiliza para validar la propuesta de solución haciendo una comparación de la eficacia en la clasificación entre tres configuraciones de arquitectura e inicialización de los pesos del Perceptrón Multicapa, sin aplicar los algoritmos conceptuales, aplicando el algoritmo LC-Conceptual y aplicando el algoritmo RGC. Los preexperimentos son utilizados en bases de datos internacionales con diferentes características.

El trabajo realizado está estructurado en tres capítulos:

En el primero se describen los referentes teóricos relacionados con el diseño de la arquitectura e inicialización de los pesos de una red neuronal de tipo Perceptrón Multicapa y sobre los algoritmos conceptuales del Reconocimiento Lógico Combinatorio de Patrones. Además, se describen tecnologías, metodología y lenguajes de programación que se adecuan a las características de la herramienta computacional que se implementa como resultado de esta investigación.

En el segundo capítulo se explica la propuesta de diseño de la arquitectura e inicialización de los pesos del Perceptrón Multicapa, así como los elementos que se tuvieron en cuenta del agrupamiento conceptual según el RGC para desarrollar la propuesta de solución.

En el tercer capítulo se describen los preexperimentos realizados para validar la variable eficacia en la clasificación con la propuesta de solución. Se muestran los resultados obtenidos al aplicar tres configuraciones de un MLP en distintas bases de datos internacionales, la primera con el MLP sin aplicar los algoritmos conceptuales, la segunda con el MLP y el LC-Conceptual y la tercera con el MLP y el RGC.

Al finalizar se muestran los principales resultados obtenidos, se realizan las recomendaciones pertinentes para futuros trabajos, se muestran las referencias bibliográficas y los anexos que complementan la investigación.

Capítulo 1. Marco teórico referencial sobre las Redes Neuronales Artificiales de tipo Perceptrón Multicapa y los algoritmos conceptuales

En el presente capítulo se abordan los elementos de la base teórico-conceptual de la investigación para el desarrollo de la solución propuesta, basándose en el análisis de las técnicas y antecedentes asociados con la arquitectura e inicialización de los pesos del Perceptrón Multicapa. Se realiza una caracterización de distintas soluciones existentes que están en correspondencia con la problemática de las RNA de tipo Perceptrón Multicapa. Se caracterizan los algoritmos de agrupamiento conceptual LC-Conceptual y RGC. Finalmente se describen las características fundamentales de las herramientas y tecnologías de software utilizadas para el desarrollo del sistema.

1.1 Redes Neuronales Artificiales

Una RNA es definida como un modelo matemático-computacional que trata de imitar la estructura y la funcionalidad de las redes neuronales biológicas (Rosenblatt 1962).

Los elementos básicos que componen una RNA son:

1. Un conjunto de unidades de procesamiento, denominadas neuronas.
2. La arquitectura o topología; forma en que se organizan las neuronas.
3. Una regla de aprendizaje.

Las redes neuronales artificiales son sistemas de procesamiento, que copian esquemáticamente la estructura neuronal del cerebro para tratar de reproducir sus capacidades. Estos sistemas son capaces así de aprender de la experiencia a partir de las señales o datos provenientes del exterior, dentro de un marco de computación paralela y distribuida, fácilmente implementable en dispositivos hardware específicos (del Brío y Molina 2001).

Por otra parte, según el autor Montaño-Moreno, (2017) las RNA son sistemas de procesamiento de la información cuya estructura y funcionamiento están inspirados en las redes neuronales biológicas. Consisten en un conjunto de elementos simples de procesamiento llamados nodos o neuronas conectadas entre sí por conexiones que tienen un valor numérico modificable nombrado peso.

Atendiendo a las definiciones anteriores en la presente investigación se considera que una RNA es un modelo matemático computacional, que procesa información tratando de similar el cerebro humano, y está compuesta según las elementos básicos que expresa (Rosenblatt 1962).

Las RNA son capaces de detectar relaciones complejas y no lineales entre variables. Las variables se dividen en variables de entrada y de salida, relacionadas por algún tipo de correlación o dependencia (no necesariamente causa-efecto). Las neuronas se pueden disponer en diferentes capas, definiendo el tipo de capa según el modelo neuronal al que se esté enfrentando.

La arquitectura en una red neuronal es la forma en que son organizadas las neuronas en su interior, estas se pueden clasificar atendiendo a dos criterios de acuerdo con (del Brío y Molina 2001) :

Criterio 1: En relación a su estructura en capas, existen las redes monocapa y multicapa. Las redes monocapa *(single-layer networks)* son aquellas compuestas por una capa de neuronas. Las redes multicapa *(layered networks)* son aquellas cuyas neuronas se organizan en varias capas.

Criterio 2: Atendiendo al flujo de datos en la red neuronal, existen las redes unidireccionales y las redes recurrentes. En las redes unidireccionales *(feedforward)*, la información circula en un único sentido, desde las neuronas de entrada hacia las de salida. En las redes recurrentes o realimentadas *(feedback)* la información puede circular entre las capas en cualquier sentido, incluido el de salida-entrada.

Las RNA presentan dos modos de operación, entiéndase por modo de operación la forma de cómo puede ser utilizada la red:

Aprendizaje o convergencia: está estrechamente relacionado con la regla de aprendizaje definida y es el encargado a partir de un conjunto de patrones de entrada, enseñar a la red un dominio específico de aplicación, mediante el reajuste de sus pesos. El aprendizaje se basa en definir un conjunto de pesos sinápticos que permitan a la red realizar correctamente la tarea para la que fue entrenada, sin embargo, en algunos modelos neuronales se incluye la creación o destrucción de neuronas como parte del aprendizaje.

Recuerdo o ejecución: comienza cuando el sistema ya ha sido entrenado y el aprendizaje termina (los pesos y la arquitectura quedan fijos), quedando listo para solucionar el problema correspondiente al dominio de aplicación sobre el que fue entrenado.

De acuerdo a lo anterior se puede deducir que un modelo neuronal es un modelo matemático que trata de reproducir algunas de las capacidades de las redes neuronales biológicas para la resolución de problemas.

Está determinado según el modelo de neurona, la arquitectura y la regla de aprendizaje que utilice. Un ejemplo de clasificación de modelos neuronales se muestra en la figura 1.

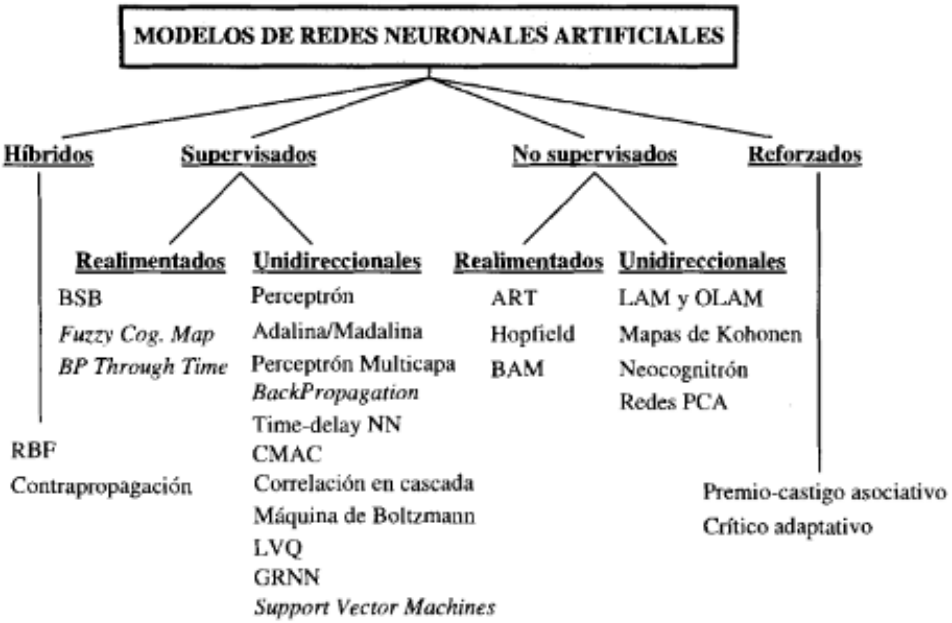


Figura 1Clasificación de los modelos neuronales atendiendo al tipo de aprendizaje y flujo de datos (tomada de (del Brío y Molina 2001))

Dentro de los distintos tipos de modelos de redes neuronales artificiales se encuentra el Perceptrón Multicapa que es uno de los más utilizados por su capacidad como aproximador universal.

1.2 Perceptrón Multicapa

Según Montero-Montes,(2012) las RNA de tipo Perceptrón Multicapa son las redes con múltiples capas que funcionan con encadenamiento hacia delante. Esta red está compuesta por un conjunto de nodos de entrada que componen la capa de entrada, un conjunto de una o más capas ocultas de neuronas y una capa de neuronas de salida. La señal de entrada se propaga desde la capa entrada hasta la capa salida.

El MLP surge como una alternativa para resolver el problema de la separabilidad no lineal que presentaba el perceptrón simple capa. En (del Brío y Molina 2001) se referencia que Minsky y Papert demostraron que una solución para el tratamiento de algunos problemas de este tipo podía ser la combinación de varios perceptrones simples lo que conllevó a la aparición de las capas ocultas (ver figura 2).

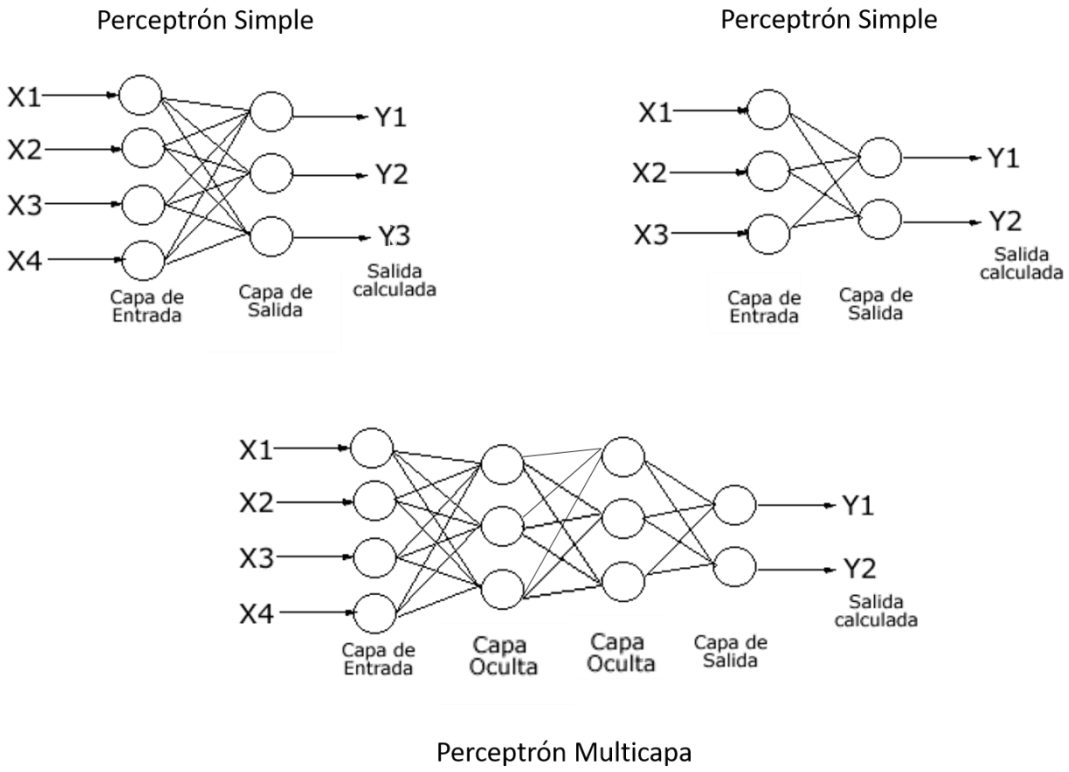


Figura 2 Surgimiento de un MLP (Fuente: elaboración propia)

Una de las principales ventajas del Perceptrón Multicapa es que se pueden separar regiones no lineales de decisión tan complicadas como se desee, dependiendo del número de neuronas y capas. Por lo tanto, las redes neuronales artificiales sirven para resolver problemas de clasificación de alta complejidad.

Sin embargo, en ese momento aún no se tenía un recurso que permitiera la adaptación de los pesos, porque no era posible utilizar en esta nueva propuesta la regla de aprendizaje del Perceptrón Simple. En 1986, Rummelhart, Hinton y Wilians, presentaron una nueva regla que permitió que el Perceptrón Multicapa aprendiera a través de retropropagación de los errores medidos en la salida de la red hacia las neuronas de entrada.

A partir de este momento varios autores demostraron matemáticamente que el Perceptrón Multicapa es un aproximador universal, ya que cualquier función continua en un espacio R^n puede aproximarse teniendo en cuenta este modelo, lo cual lo convierte en un modelo matemático útil a la hora de aproximar o interpolar relaciones no lineales entre datos de entrada y salida (del Brío y Molina 2001; Pérez-Valls 2013). Su aplicación está centrada en problemas con aprendizaje supervisado: clasificación y predicción de enfermedades (Jurado y Asbusac 2015; Rabuñal 2002), temperaturas (Comesaña-Campos 2013; Villamil-

Bahamón 2017), deslizamiento de terrenos (Vargas-Bejarano 2017; Cajamarca-Palma y García-Lema 2017), clasificación de patrones (Tipan y Fernando 2018; Mendoza, Fernanda y Zapata-Sarzosa 2018), entre otros.

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. En la presente investigación se utiliza el MLP con una sola capa oculta considerando el criterio expuesto por (Majalca Martínez y Acosta-Cano de los Ríos 2015) en el que se demuestra que más capas ocultas hacen que la red funcione más lentamente y que una sola capa oculta es suficiente para lograr la aproximación de cualquier función continua.

El MLP está determinado mediante una arquitectura estructurada por un conjunto de neuronas organizadas en tres tipos de capas (entrada, oculta y salida), con conexiones pesadas entre sí y flujo de datos unidireccional, desde la capa de entrada hasta la capa de salida. El algoritmo de aprendizaje que emplea este modelo es el Backpropagation.

Backpropagation emplea un ciclo de aprendizaje en dos fases, en la primera la señal de entrada se propaga desde la capa de entrada hasta generar una salida, se compara con la deseada y se calcula una señal de error para cada salida. La segunda fase es la encargada de distribuir la señal de error entre las distintas capas, recibiendo cada neurona una fracción total del error en función de la contribución relativa que haya aportado a la salida original. Para ello, se utiliza una función de error (ver Ecuación 1), que será derivada no solo en función de los pesos de la capa de salida, sino también en función de los pesos de las neuronas ocultas(del Brío y Molina 2001).

Ecuación 1: Función error

$$E[w_{ij}] = \frac{1}{2}\sum_{\mu=1}^{P}\sum_{i=1}^{n}\left(t_i^{\mu} - y_i^{\mu}\right)^2$$

El funcionamiento del Backpropagation se basa en la minimización del error, mediante una función que relaciona las entradas a la red y los pesos asociados a las conexiones, a partir de una arquitectura fija que se debe establecer con anterioridad.

1.2.1 Arquitectura e inicialización de los pesos del MLP

La arquitectura del MLP se caracteriza por tener las neuronas organizadas en tres tipos de capas: la capa de entrada, la capa oculta y la capa de salida (ver figura 3), las neuronas entre las distintas capas están interconectadas mediante arcos dirigidos con un escalar denominado peso.

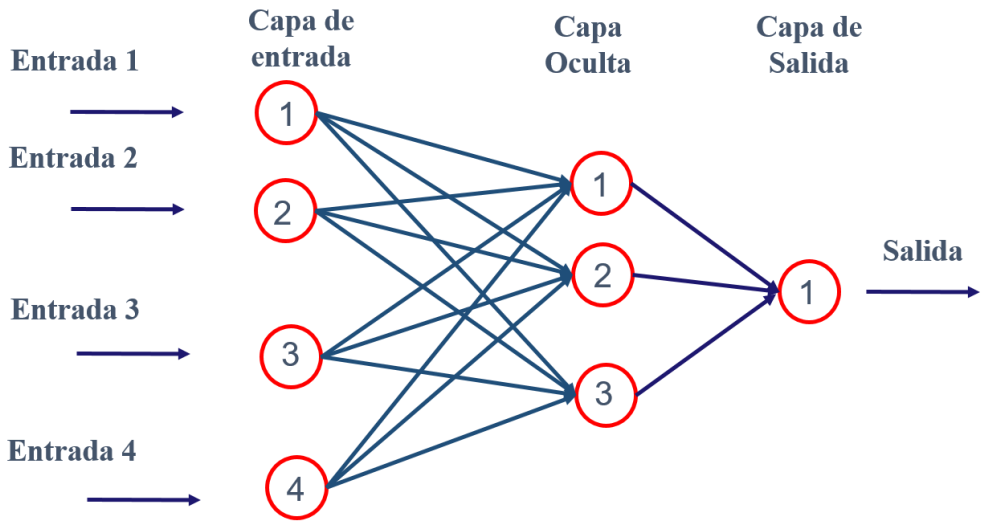


Figura 3 Ejemplo de un Perceptrón Multicapa con una capa oculta (Fuente: elaboración propia)

Las conexiones del Perceptrón Multicapa siempre están dirigidas hacia adelante, es decir, las neuronas de una capa se conectan con las neuronas de la siguiente capa, de ahí que reciban también el nombre de redes alimentadas hacia adelante o redes *feedforward*. Las conexiones entre las neuronas de la red llevan también asociado un umbral, que en el caso del Perceptrón Multicapa suele tratarse como una conexión más a la neurona, cuya entrada es constante e igual a 1 (Pérez-Valls 2013).

El número de neuronas correspondientes a las capas de entrada y salida se define a partir de la cantidad de variables (entradas y salidas) presentes en el problema que se desea resolver.

Según los autores (Jurado y Asbusac 2015; Tallón-Ballesteros 2013; Rabuñal 2002; Comesaña-Campos 2013) la selección del número de capas ocultas, así como la cantidad de neuronas pertenecientes a las mismas es un conflicto para el diseñador de la red, puesto que no existe una regla en correspondencia con la naturaleza del problema a resolver. Tanto es así que esta elección se realiza por lo general de manera empírica y se cambia aleatoriamente en dependencia de los resultados del entrenamiento.

En lo referente a la inicialización de los pesos sinápticos, generalmente se destinan valores aleatorios pequeños, debido a que los grandes valores absolutos de pesos pueden hacer que las neuronas sean altamente activas o inactivas para todas las muestras de entrenamiento y por ende insensibles a este proceso. Además, la elección del vector de peso inicial (W_0) puede acelerar la convergencia del proceso de aprendizaje hacia un mínimo global o local si se ubica dentro de la atracción basada en ese mínimo. Por el

contrario, si W_0 comienza la búsqueda en una región relativamente plana de la superficie de error, ralentizará la adaptación de los pesos de conexión (Coello et al. 2016). Los estudios han demostrado que una buena inicialización podría conducir a una convergencia y / o capacidad de generalización rápida, incluso con una simple técnica de minimización de errores basada en gradientes (Cabrera et al. 2011).

En Cabrera et al.,(2011) además, se hace referencia a que en problemas de clasificación multiclases, cuando todos los pesos se establecen en valores aleatorios, las curvas de aprendizaje tienen una característica típica; después de disminuir rápidamente al principio, la suma de errores al cuadrado mantiene un valor casi constante durante mucho tiempo, haciendo que las redes de MLP muestren un peor desempeño en el caso del problema de clasificación multiclases que en los de dos clases. Se afirma que las inicializaciones determinísticas pueden mejorar la convergencia del entrenamiento en comparación con la inicialización aleatoria.

Otra de las problemáticas que puede traer la inicialización aleatoria de W_0 independiente de la topología de la red, es que, para una misma arquitectura, diversas inicializaciones de W_0, pueden conllevar a resultados diferentes.

1.3 Técnicas utilizadas en la arquitectura e inicialización de los pesos del MLP

Para abordar la problemática de la arquitectura de la red y la inicialización de los pesos en el MLP se emplean diversas técnicas de la inteligencia artificial, las cuales se explican a continuación.

1.3.1 Algoritmos Genéticos

El Algoritmo Genético (AG) es una técnica de optimización que intenta replicar procesos de evolución natural en los que los individuos con las mejores características consideradas para adaptarse al entorno tienen más probabilidades de reproducirse y sobrevivir. Estos individuos ventajosos se aparean entre sí, produciendo descendientes de características similares, por lo que se conservan las características favorables y se destruyen las desfavorables, dando lugar a una evolución progresiva de la especie. El objetivo del algoritmo genético artificial es mejorar la solución de un problema manteniendo la mejor combinación de variables de entrada. Comienza con la definición del problema a optimizar, generando una función objetiva para evaluar las posibles soluciones candidatas (cromosomas), es decir, la función objetiva es la forma de determinar qué individuo produce el mejor resultado (Correa, Gonzalez y Ladino 2011).

El autor (Correa, Gonzalez y Ladino 2011).lo integró con el MLP para la determinación de la arquitectura, pero en esta propuesta destacan las siguientes diferencias:

1. No realiza un análisis de las variables relevantes.

2. Cada individuo representa un diseño arquitectónico diferente, generado a través de inicializaciones aleatorias que engloban tanto la definición de la cantidad de capas y unidades ocultas como el uso de sus respectivas funciones de activación.

3. Es sensible a la determinación de los parámetros iniciales: tamaño de la población, probabilidades de mutación y cruzamiento; lo que puede atentar con la convergencia del algoritmo.

1.3.2 Algoritmo basado en la Optimización de Colonia de Hormiga

La optimización de la colonia de hormigas (OCA) es una metaheurística introducida por Marco Dorigo en 1992 para resolver problemas de optimización combinatoria. La OCA consiste en una colonia de hormigas artificiales, que construyen iterativamente soluciones a una instancia dada de una optimización combinatoria y usan rastros de feromonas para comunicarse. Cada hormiga de la colonia representa una solución del problema como un camino y pone una cantidad de feromonas en su recorrido de acuerdo a la calidad de la solución. Sin embargo, para evitar el refuerzo de trayectorias de mala calidad, se aplica un paso de evaporación al final de cada iteración del algoritmo (Ghanou y Bencheikh 2016).

La OCA se integra con la RNA para resolver el problema de la arquitectura donde se determina la cantidad de neuronas por capa. En el 2017 en la investigación (Ghanou y Bencheikh 2016) se presenta una propuesta para dar solución a la problemática presentada y se detectó las siguientes deficiencias:

1. No realiza un análisis de las variables relevantes.

2. Cada hormiga representa un diseño arquitectónico diferente, generado a través de inicializaciones aleatorias que engloban tanto la definición de las unidades ocultas, los vectores de pesos, las entradas a la red.

3. Es sensible a la determinación de los parámetros iniciales: función de costo, regla para la liberación de feromonas, funciones de probabilidad para la modificación de los parámetros pesos y unidades ocultas.

1.3.3 Algoritmo LC-Conceptual

En Milanés-Luque,(2017) se propone un nuevo enfoque atendiendo a las problemáticas vistas anteriores para determinar el diseño arquitectónico y la inicialización de los pesos en un Perceptrón Multicapa, mediante el uso de algoritmos del agrupamiento conceptual del reconocimiento lógico combinatorio de patrones, específicamente el algoritmo LC-Conceptual que contribuye a la eficiencia temporal en el proceso de aprendizaje y la eficacia en la clasificación.

Esta propuesta selecciona los rasgos más significativos en el problema, mediante el cálculo de los testores típicos los cuales le permiten definir el número de neuronas en la capa de entrada. Luego aplica el operador Refunión Condicionada (Martínez-Trinidad y Ruiz-Shulcloper 1999) para calcular los conceptos que están asociados al número de neuronas en la capa oculta. El número de neuronas correspondientes a la capa de salida está en correspondencia con las clases definidas inicialmente en el problema.

Para la inicialización de los pesos se utilizan dos algoritmos: el primero para determinar los pesos entre las neuronas de la capa de entrada y la capa oculta y el segundo para definir los pesos entra la capa oculta y la capa de salida.

En la investigación de referencia se integra el LC-Conceptual con el MLP para la definición de la arquitectura e inicialización de los pesos sinápticos correspondiente a la relación entre las diferentes capas, demostrando que se mejora tanto en la eficiencia en el tiempo de entrenamiento como eficacia de la clasificación. Sin embargo, el algoritmo tiene la peculiaridad de que puede dejar objetos sin cubrir por los conceptos construidos, la cual puede ser una deficiencia a la hora de clasificar.

1.4 Reconocimiento Lógico Combinatorio de Patrones

En la bibliografía consultada se aprecia que el RLCP puede ser aplicado para resolver problemas de la mayoría de las áreas del saber: reconocimiento de caracteres, diagnóstico médico, teledetección de la tierra, identificación de rostros humanos y huellas digitales, pronóstico de roturas en equipos y maquinarias, análisis de señales e imágenes biomédicas, inspección automática, conteo de células sanguíneas, análisis de los registros de pozos, arqueología, pronóstico de depósito de minerales, análisis de la actividad sismológica, clasificación de documentos, entre otras (Reyes-González 2014).

Según Ruiz-Shulcloper,(2009) los problemas del reconocimiento de patrones se centran en la selección de rasgos, la clasificación supervisada, no supervisada y semisupervisada. En el RLCP se destaca la representación de los objetos en términos de un conjunto de rasgos (propiedades, características, variables), lo cual permite la resolución de muchos problemas presentados en las denominadas ciencias poco formalizadas, donde la descripción de estos objetos puede ser mezclada e incompleta.

1.4.1 Selección de los rasgos

La selección de rasgos es uno de los pasos fundamentales en cualquier problema de clasificación debido a que la mayoría de los problemas de reconocimiento de patrones están basados en la descripción de los objetos en términos de un conjunto de rasgos. En la literatura se identifican dos problemas diferentes pero muy vinculados: la selección de rasgos para la clasificación y la selección de rasgos para la descripción (Ruiz-Shulcloper 2009).

La selección de rasgos para la clasificación, es la determinación del mejor subconjunto de rasgos para la clasificación de nuevos objetos (no clasificados). Esto conlleva la reducción del conjunto de todos los posibles rasgos (reducción de la dimensionalidad) sobre la base de las diferencias que estos rasgos presentan en cuanto a mejor reconocer, clasificar a los nuevos objetos y otros problemas de optimización adicionales del subconjunto de rasgos a emplear (Ruiz-Shulcloper 2009).

La selección de rasgos para la clasificación es un elemento importante para definir las neuronas de la capa de entrada, en problemas donde estén presentes variables que no sean relevantes. Inicialmente se considera la cantidad de neuronas correspondientes a la clase de entrada igual a la cantidad de rasgos definidos en el problema, pudiendo reducirse la dimensión del vector de entrada al prescindir de aquellos que no tienen importancia para el dominio de aplicación. Otro aspecto a tener en cuenta sería la posible influencia que pueda existir entre la importancia informacional del rasgo y el peso asociado de las conexiones entre la neurona de la capa de entrada y las pertenecientes a la capa oculta.

En el RLCP una alternativa de solución al problema de la selección de variables es a partir de la utilización del conjunto de testores típicos (Reyes-González 2014). En esencia, un testor es un conjunto de características (rasgos) que diferencia a elementos (objetos) de clases distintas. Un testor típico es un testor al que si se le elimina cualquiera de sus rasgos pierde la propiedad de ser testor (Pons-Porrata 2004).

1.4.2 Clasificación Supervisada

Un problema de clasificación supervisada consiste en, dado un universo de objetos estructurados en clases, de cada una de las cuales se tiene una muestra no vacía, que permite construir un algoritmo a partir de esta muestra, decidirá a cuál de las clases pertenece un objeto nuevo que se quiera clasificar.

Según Ruiz-Shulcloper en el 2009, un problema de Reconocimiento de Patrones en la clasificación supervisada es un procedimiento efectivo donde la clasificación de un objeto $O \in U$ por un clasificador supervisado A, se entiende la acción de asignar un r-uplo de pertenencias a las clases de U, tomando dicho objeto como entrada de A. En algunos textos también se entiende por clasificación el resultado de esta acción. La clase que A, con matriz de entrenamiento M, le asigna a un objeto O se denotará por $\alpha A(M,O)$. A esto último se le llamará r-uplo de pertenencia del objeto asignado por A. En la descripción de los algoritmos se utiliza la notación funcional de un clasificador supervisado (Ruiz-Shulcloper 2009).

1.4.3 Clasificación no Supervisada

En la clasificación no supervisada o agrupamiento el propósito es juntar (agrupar) los objetos según su analogía (parecido, semejanza, cercanía si se está hablando de un espacio de representación con distancia definida). Los tres elementos esenciales que lo constituyen son: el espacio de representación de los objetos,

la medida de similaridad (β, función de semejanza) y el criterio de agrupamiento Π, es decir, la manera en que es utilizada la similaridad para la solución del problema planteado (Reyes-González 2014).

El objetivo de los algoritmos de agrupamiento es dado un conjunto de objetos definidos en términos de un conjunto de rasgos, intentar construir particiones o cubrimientos de este conjunto, donde la semejanza intragrupo sea máxima y la semejanza inter-grupos sea mínima (Reyes-González, Y. et al. 2016).

En la literatura consultada se aprecia que existen dos tipos de algoritmos de agrupamientos: los que solo agrupan a los objetos, denominados convencionales, y los que además de agrupar a los objetos, brindan información relevante sobre las agrupaciones obtenidas, llamados algoritmos de agrupamiento conceptual.

Según Reyes-González, (2014) los algoritmos de agrupamiento convencionales tienen las siguientes limitaciones:

1 Dejan el problema de la interpretación de los grupos al analista de datos.
2 No tienen en cuenta los métodos que los humanos emplean para agrupar objetos. Las personas tienden a agrupar objetos en categorías caracterizadas por conceptos, en grupos similares teniendo en cuenta algún atributo relevante o más importante que el resto.
3 No toman en consideración ningún concepto o construcciones lingüísticas que las personas usan para describir colecciones de objetos (Michalski y Stepp 1981).
4 Para enfrentar estas limitaciones a finales de los años 70 e inicios de los 80, Rysazard S. Michalski introdujo un conjunto de ideas que dieron origen al agrupamiento conceptual (Michalski 1980).

1.4.4 Clasificación semisupervisada

Los problemas de clasificación semisupervisada combinan tanto la clasificación supervisada como la no supervisada, su definición formal plantea: la clasificación semisupervisada se considera una extensión de la clasificación supervisada donde el conjunto de entrenamiento está formado por un conjunto L de objetos clasificados y un conjunto U de objetos sin clasificar, donde se asume que el número de objetos no clasificados es mayor que los clasificados.

El objetivo de la clasificación semisupervisada es entrenar un clasificador f a partir de los conjuntos L y U, de manera que se podrá obtener una clasificación más exacta que la cantidad de objetos ya clasificados por los problemas de clasificación supervisados.

1.5 Algoritmos Conceptuales

Los algoritmos de agrupamiento conceptual se componen de dos tareas fundamentales, las cuales no tienen necesariamente que ser independientes ni realizarse en un orden determinado (Reyes-González 2014):

1. La estructuración o determinación extensional: se lleva a cabo el proceso de agrupar entidades, en el que se determinan grupos a partir de una colección de objetos, esto no es más que la enumeración de los objetos que lo componen los grupos.
2. La caracterización o determinación intencional: se determina el concepto de cada grupo de la estructuración, las propiedades que caracterizan el agrupamiento.

Los algoritmos de agrupamiento conceptual se pueden dividir en dos grandes grupos, a saber, los algoritmos incrementales y los no incrementales (Pérez-Valls 2013). Los algoritmos incrementales basan su funcionamiento en la adaptación de los agrupamientos (o conceptos) con los nuevos objetos que se le van presentando, es decir, cada vez que llega un nuevo objeto mediante una cierta estrategia éste es clasificado en los agrupamientos ya existentes o se crean nuevos agrupamientos. Por otro lado, los algoritmos no incrementales estructuran una muestra de objetos sin presuponer que éstos llegan de uno en uno (Ruiz-Shulcloper 2009).

En los trabajos de Ruiz-Shulcloper,(2009) y Reyes-González,(2014) se realiza un análisis crítico de diferentes algoritmos de agrupamiento conceptual, atendiendo a sus características y funcionamiento, destacándose entre sus principales resultados los siguientes:

1. La principal desventaja de los algoritmos conceptuales de tipo incremental es la dependencia del resultado (la estructuración) en función del orden de presentación de los objetos al algoritmo mientras que en los de tipo no incremental se determina el número de las agrupaciones de manera aleatoria, lo que constituye una dificultad en la vida real, debido al desconocimiento de los posibles agrupamientos en ciertos tipos de problemas.
2. Los algoritmos que no pertenecen al enfoque lógico combinatorio del reconocimiento de patrones construyen sus conceptos en función de criterios probabilísticos o estadísticos, por lo que su interpretación puede ser engorrosas para personas no especializadas en esas áreas del conocimiento.
3. Los algoritmos LC-Conceptual y RGC (ambos pertenecientes al RLCP) construyen sus conceptos en función de propiedades lógicas, basadas en los rasgos de los objetos en estudio. Además de que no requieren que sean especificados a priori el número de agrupamientos. La dificultad que poseen estos algoritmos, es la complejidad computacional en el cálculo de los testores típicos.

La descripción de los conceptos en función de propiedades lógicas basadas en los rasgos de los objetos, puede constituir una variante para la explicación del significado que puede tener una neurona de la capa oculta en función de las entradas de la red. En la bibliografía consultada se aprecia que la selección de la cantidad de capas ocultas y sus correspondientes neuronas han sido abordadas con diferentes técnicas de la IA, sin embargo, no dejan claro el significado que puede tener una neurona de la capa oculta en función de sus entradas.

Un aspecto a tener en cuenta para afrontar esta situación, sería asociar las neuronas de la capa oculta a los conceptos que se generan en la parte intencional del algoritmo de agrupamiento. Otro elemento a revisar sería asociar la importancia de cada concepto (o neurona) a la inicialización del peso entre las conexiones de la capa oculta y la de salida.

1.5.1 Algoritmo LC-Conceptual

El algoritmo LC-Conceptual fue propuesto en el año 2001 por Martínez-Trinidad y Ruiz-Shulcloper,(1999), está basado en los conceptos de la clasificación no supervisada en el enfoque lógico-combinatorio y retoma algunas ideas propuestas por Michalski para generar conceptos, interpretables por los especialistas, en términos del conjunto de rasgos original.

El algoritmo LC-Conceptual incluye dos etapas:

Etapa de estructuración extensional: se construyen los agrupamientos de objetos basándose en la semejanza entre los mismos y utilizando un criterio de agrupamiento.

Etapa de estructuración intencional o conceptual: se construyen las propiedades (conceptos) que caracterizan a cada agrupamiento de objetos utilizando el operador de refunión condicionada y los testores típicos.

Paso 1: Se determinan, los criterios de comparación por rasgos, la función de semejanza y el criterio de agrupamiento. Se estructura la muestra inicial en función de estos parámetros, siendo considerado cada grupo formado como una clase. Posteriormente se procede a calcular todos los testores típicos para cada clase.

Paso 2: Se forma una matriz de entrenamiento ME o matriz de aprendizaje MA a partir de la matriz inicial MI y de los agrupamientos o clases K_1, …, K_c, luego se calcula el conjunto de los testores típicos de MEт = {t_1,…, t_p}. Para cada clase K_i, i = 1,.. , c se emplea el operador de Refunión Condicionada para construir los l-complejo (entiéndase por l-complejo como el producto lógico de todos los rasgos que cubren un concepto o el conjunto de todos los valores diferentes que pueden tomar los objetos de dicho concepto (Pons-Porrata 2004)) y a su vez construir los conceptos (Reyes-González 2014) (ver figura 4).

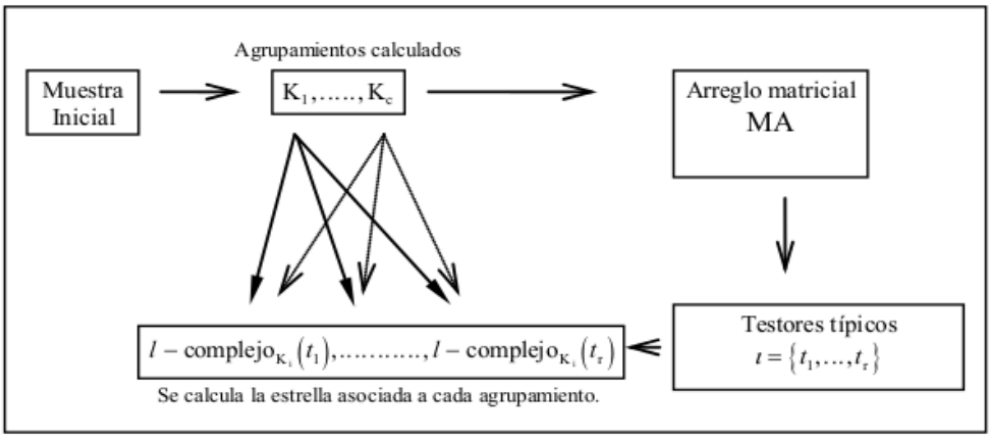


Figura 4 Proceso del agrupamiento conceptual utilizando el algoritmo LC-Conceptual (Martínez-Trinidad y Sánchez-Díaz 2001)

El operador de Refunión Condicionada (RUC): forma los l-complejos garantizando que los l-complejos de cada agrupamiento no satisfacen a ningún objeto de otros grupos, pero no de manera independiente, sino en combinación con el resto de las variables (Pons-Porrata 2004).

Este algoritmo presenta las siguientes limitaciones:

1. Se aplica el operador de refunión condicionada, el cual garantiza que el concepto de un agrupamiento no sea satisfecho por ningún objeto de otro agrupamiento, pero entonces, no logra que este concepto cubra a todos los objetos de dicho agrupamiento.
2. No permite el empleo de otro algoritmo de reducción de rasgo que no sea el testor típico.
3. No incluye la regla de generalización como parte del proceso del algoritmo.

1.5.2 Algoritmo RGC

El algoritmo conceptual RGC fue propuesto en el año 2004 por Aurora Pons Porrata (Pons-Porrata 2004) está basado en los conceptos de la clasificación no supervisada del Reconocimiento Lógico Combinatorio de Patrones y retoma algunas de las ideas planteadas por Michalski para generar los conceptos, en función del conjunto de rasgos original.

El algoritmo RGC cuanta con dos etapas como se muestra en la figura 5:

I Etapa de determinación extensional: Primeramente, se realizan los agrupamientos, usando los criterios de comparación entre los rasgos, la función de semejanza entre objetos y un criterio de agrupamiento para generar la estructuración por clases. Para ello, podrá emplearse cualquier algoritmo de Clasificación no supervisada como grupo de apoyo.

II Etapa de determinación intencional: Después de haber aplicado la etapa extensional se emplea el operador de refunión extendida para el cálculo de los l-complejos, se aplican las reglas de generalización y se construyen los conceptos que corresponde a cada clase K_i.

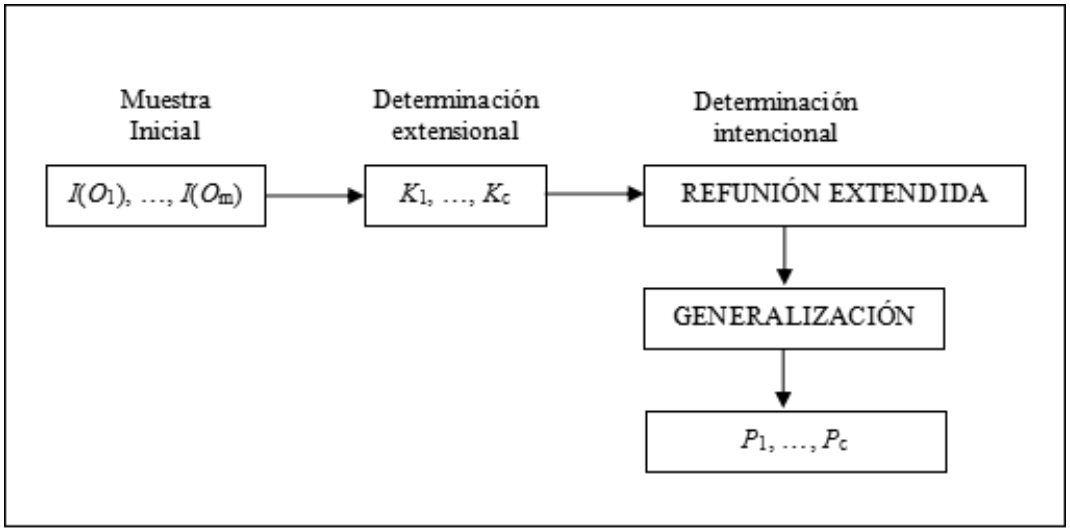


Figura 5 Proceso del agrupamiento conceptual del RGC (tomado de (Pons-Porrata 2004))

Los pasos para aplicar el algoritmo RGC son:

1- Se construye las propiedades (conceptos) que caracterizan a cada grupo de objetos.

2- Se forma una matriz de entrenamiento a partir de MI y de los grupos $K_1,\ldots, K_c$, luego se calcula el conjunto de apoyo en este caso es los testores típicos de $ME_T = \{t_1,\ldots, t_p\}$.

3- Para cada clase K_i, i = 1,…, c se calcula la estrella G_T ($K_i \setminus K_1,..., K_{i-1}, K_{i+1},..., K_c$) que estará formada por l- complejos.

4- Para el cálculo de los l-complejos se emplea el operador de Refunión Extendida (RUE) (Pons-Porrata 2004). Luego se aplican las reglas de generalización.

5- Los conceptos que caracterizan a cada agrupamiento K_i serán los l-complejos obtenidos en el paso anterior.

El operador de refunión extendida (RUE): forma los conceptos garantizando que los l-complejos de cada agrupamiento no satisface a ningún objeto de otros grupos; pues si un nuevo objeto no puede ser añadido a ninguno de los l-complejos existentes se crea, al menos, un nuevo l-complejo que lo cubre (Pons-Porrata 2004).

1.5.3 Comparación entre los algoritmos conceptuales

En Pérez-Suárez y Medina-Pagola,(2014) y Reyes-González,(2017) se presenta un estado del arte en la temática de los algoritmos conceptuales, se realiza una comparación entre ellos. En esta investigación se presentan solo la comparación del LC- Conceptual y RGC teniendo en cuenta que estos son los algoritmos de referencia para el trabajo.

Los criterios que los autores tienen en cuenta para establecer la comparación entre los algoritmos son:

La complejidad computacional: la métrica de medición se va a establecer en fácil, medio, alta y muy alta. En este caso los dos algoritmos tienen una complejidad computacional muy alta establecido expresado en la investigación (Reyes-González 2017).

La estructuración: la métrica se establece en conjunto o no conjunto. El algoritmo conjunto está definido por las instrucciones ordenadas y finitas que permite realizar una tarea (Montero-Montes 2012). El no conjunto es el que no cumple con las instrucciones ordenadas y tiene distintas salidas. En el caso de los dos algoritmos de objeto de comparación según sus autores (Martínez-Trinidad y Sánchez-Díaz 2001;Pons-Porrata 2004) son conjunto.

Tipo de atributo: se definen los tipos de atributos que puede tener los rasgos de la base de conocimiento que se puede trabajar. En Pérez-Suárez y Medina-Pagola,(2014) se considera que una de las características fundamentales, a los efectos de su investigación, radica en el tipo de atributo que admiten los algoritmos y en este sentido LC- Conceptual y RGC (basados en el Reconocimiento Lógico Combinatorio de Patrones) son los de mejores resultados pues permiten trabajar con datos mezclados e incompletos, lo cual es muy frecuente en los problemas prácticos.

Regla de generalización: se basa en tener en cuenta las reglas de generalización. En este caso según sus creadores (Martínez-Trinidad y Sánchez-Díaz 2001)(Pons-Porrata 2004), en el caso del LC-Conceptual no lo tiene incluido y el RGC sí lo tiene presente.

Excluyente: se analiza si en la forma de construir los conceptos el cual indica que no son satisfechos por ningún objeto de otra clase (Pons-Porrata 2004). Por lo que los dos algoritmos lo tienen presente cuando se construyen los conceptos.

Caracterizante: se refiere a que cubren a todos los objetos de la clase que describen (Pons-Porrata 2004). Esto ocurre en la forma de construir los l-complejos.

Sin embargo, la diferencia radica en que el RGC cumple con las propiedades de ser excluyente y caracterizante a diferencia del LC-Conceptual que solo puede ser excluyente, no garantiza que sea caracterizante (Pons-Porrata 2004), al igual el algoritmo RGC tiene incluida las reglas de generalización y el LC-Conceptual no, como se muestra en la Tabla 1. Por lo expuesto en la presente investigación se propone aplicar el algoritmo RGC para mejorar la eficacia en la clasificación de los objetos de un MLP.

Tabla 1: Comparación de los algoritmos conceptuales

Criterios de Comparación	**Algoritmos Conceptuales**	
	LC- Conceptual	**RGC**
Complejidad	Muy alta	Muy alta
Estructuración	Conjunto	Conjunto
Atributos	Mezclados incompletos	Mezclados incompletos
Regla de generalización	no	si
Excluyentes	si	si
Caracterizante	no	si

(Fuente: elaboración propia)

1.6 Herramientas y tecnologías utilizadas

En la presente investigación se utiliza para aplicar el Reconocimiento Lógico Combinatorio de Patrones (RLCP) la herramienta Entorno Cubano para el Reconocimiento Lógico Combinatorio de Patrones (CEPAR), En el diseño del modelo neuronal, así como para la inicialización de los pesos se emplea el toolboox de redes neuronales artificiales de MatLab y en la validación probabilística el RStudio.

1.6.1 Herramienta CEPAR para la implementación del Algoritmo RGC

La herramienta CEPAR (Entorno Cubano para el Reconocimiento Lógico Combinatorio de Patrones) es un Sistema Herramienta Universal (SHU) desarrollado según las teorías y presupuestos del RLCP. Tiene como objetivo fundamental apoyar en las labores cotidianas de los investigadores, docentes y estudiantes de estas teorías; además de proveer de una herramienta que pueda ser embebida en desarrollos propios de una investigación (Yero-Oses, Reyes-González y Martínez-Sanchez 2016).

Implementado en JAVA como lenguaje de programación, dado sus comodidades como multiplataforma, CEPAR solo requiere de la Máquina Virtual de Java (JVM) para poder ser utilizado, plantea un diseño modular, de manera que permite la incorporación de nuevos rasgos, funciones de semejanza, o algoritmos. Cada módulo es independiente y se relaciona con los demás a través de las clases e interfaces definidas en su núcleo (cepar.core), que contiene las características más generales para modelar un problema de RLCP.

La herramienta permite el trabajo simultáneo con variables cualitativas y cuantitativas (todos los tipos de rasgos soportan la ausencia de información o valor “?”), y maneja de forma nativa los tipos:

- BooleanFeature: valores booleanos.

- DateFeature: valores de tipo fecha (día-mes-año).

- DoubleFeature: valores reales.

- FloatFeature: valores de punto flotante.

- IntegerFeature: valores enteros.

- StringFeature: valores alfanuméricos.

Esta cantidad de tipos de rasgos resulta minúscula, en comparación con los que pueden aparecer en problemas reales del RLCP, por lo que resalta en CEPAR el permitir la extensión según las interfaces definidas de nuevos rasgos por los usuarios, no limitando la herramienta solo al trabajo con los seis rasgos nativos.

Para cada uno de los tipos de rasgos nativos, o implementados, es posible definir dominios de pertenencia con posibilidades de extensión atendiendo a las necesidades propias del especialista. Además, cuenta con diversos criterios de comparación y funciones de semejanzas, que de no cumplir con las necesidades del problema modelado pueden ser extendidos (Yero-Oses, Reyes-González y Martínez-Sanchez 2016).

La herramienta cuenta con las funcionalidades de cargar dataset, seleccionar criterio de comparación entre rasgos, seleccionar funciones de semejanza, calcular el umbral por diferentes criterios, establecer criterio de agrupamientos y calcular testores típicos. Por lo antes expuesto se decide utilizar el CEPAR como librería en java para la implementación de la primera etapa del algoritmo RGC.

1.6.2 Herramienta MatLab para el diseño de la arquitectura e inicialización de los pesos sinápticos del Perceptrón Multicapa

La herramienta MATLAB (abreviatura de ***MAT****rix* ***LAB****oratory*, "laboratorio de matrices") es una herramienta de software matemático que ofrece un entorno de desarrollo integrado (IDE) con un lenguaje de programación propio (lenguaje M). La herramienta es multiplataforma ya que se puede usar en varios sistemas operativos como Unix, Windows, Mac OS X y GNU/Linux.

La herramienta de MATLAB se desarrolla en un lenguaje de programación propio. Este lenguaje es interpretado, y puede ejecutarse tanto en el entorno interactivo, como a través de un archivo de script (archivos *.m). Este lenguaje permite operaciones de vectores y matrices, funciones, cálculo lambda, y programación orientada a objetos.

MATLAB puede llamar funciones y subrutinas escritas en C o Fortran. Se crea una función envoltorio que permite que sean pasados y devueltos tipos de datos de MATLAB. Los archivos objeto dinámicamente cargables creados compilando esas funciones se denominan "MEX-files", aunque la extensión de nombre de archivo depende del sistema operativo y del procesador.

Esta herramienta es utilizada como asistente matemático en la mayoría de los casos, pero otras de las aplicaciones que tiene es el gran aporte que brinda a la inteligencia artificial en el área de las redes neuronales. El mismo cuenta con un Toolbox de Redes Neuronales Artificiales que facilita el trabajo en la presente investigación.

1.6.3 Entorno de desarrollo integrado (IDE)

Se utiliza NetBeans (en su versión 8.0.2): teniendo en cuenta que es una solución muy completa para programar en Java, aunque soporta otros lenguajes como C/C++, JavaScript, Ruby, Groovy, Phython y PHP. El proyecto de NetBeans está apoyado por una comunidad de desarrolladores y ofrece una amplia documentación y recursos de capacitación y consta de una gran cantidad de módulos. Además, puede ser usado tanto por programadores con poca experiencia como por expertos y permite trabajar sobre el mismo código a más de un programador. Además, puede ejecutarse en Windows, OS X, Linux, Solaris y otras plataformas que soportan una Java Virtual Machine compatible. Existe además un número importante de módulos para extender el NetBeans IDE. NetBeans IDE es un producto libre y gratuito sin restricciones de uso.

También está disponible NetBeans_Platform; una base modular y extensible usada como estructura de integración para crear grandes aplicaciones de escritorio. Empresas independientes asociadas, especializadas en desarrollo de software, proporcionan extensiones adicionales que se integran fácilmente

en la plataforma y que pueden también utilizarse para desarrollar sus propias herramientas y soluciones(Marshall y Thomson 2015).

RStudio es un entorno de desarrollo integrado (IDE) para el lenguaje de programación R, dedicado a la computación estadística y gráficos. Incluye una consola, editor de sintaxis que apoya la ejecución de código, así como herramientas para el trazado, la depuración y la gestión del espacio de trabajo. Es multiplataforma para los sistemas operativos Windows, Mac y Linux o para navegadores conectados a RStudio Server o RStudio Server Pro. Permite un análisis y desarrollo para que cualquiera pueda analizar los datos con R (Santana y Nieves-Hernándes 2018) Este IDE es utilizado para realizar la validación de los datos.

1.6.4 Lenguaje de programación

Se utiliza **Java 8** como lenguaje de programación teniendo en cuenta que la herramienta que se emplea como biblioteca (CEPAR) está implementada en dicho lenguaje, además es la versión más reciente de Java, que incluye nuevas características, mejoras y correcciones de bugs para mejorar la eficacia en el desarrollo y la ejecución de programas Java. La nueva versión de Java primero se pone a disposición de los desarrolladores, para dar tiempo suficiente para realizar las pruebas y certificaciones, antes de que esté disponible para su descarga en el sitio web java.com.

El lenguaje Java está diseñado para permitir el desarrollo de aplicaciones portátiles, de elevado rendimiento, para el más amplio rango de plataformas informáticas posible. Al poner a disposición de todo el mundo, aplicaciones en entornos heterogéneos, las empresas pueden proporcionar más servicios y mejorar la productividad, las comunicaciones y colaboración del usuario final y reducir drásticamente el costo de propiedad, tanto para aplicaciones de usuario, como de empresa. Java se ha convertido en un valor impagable para los desarrolladores, ya que les permite:

- Escribir software en una plataforma y ejecutarla virtualmente en otra.
- Crear programas que se puedan ejecutar en un explorador y acceder a servicios Web disponibles.
- Desarrollar aplicaciones de servidor para foros en línea, almacenes, encuestas, procesamiento de formularios HTML y mucho más.
- Combinar aplicaciones o servicios que utilizan el lenguaje Java para crear aplicaciones o servicios con un gran nivel de personalización.

Se utiliza **R** como leguaje de programación para la validación de los datos aplicando la prueba de Friedman. R es un lenguaje y un entorno para computación y gráficos estadísticos que proporciona una amplia variedad de técnicas estadísticas y gráficos, y es altamente extensible. El lenguaje R suele ser el vehículo de elección para la investigación en metodología estadística y proporciona una ruta de código abierto para la

participación en esa actividad. R está diseñado en torno a un verdadero lenguaje informático y permite a los usuarios agregar funciones adicionales definiendo nuevas funciones. Gran parte del sistema está escrito en el dialecto R, lo que facilita a los usuarios seguir las elecciones algorítmicas realizadas. Para tareas intensivas en cómputo, el código C, C++ y Fortran se puede vincular y ejecutar en tiempo de ejecución. Los usuarios avanzados pueden escribir código C para manipular objetos R directamente (Santana y Nieves-Hernándes 2018).

1.7 Conclusiones del Capítulo

A partir del análisis de la información presentada en el presente capítulo se puede arribar a las siguientes conclusiones:

- En el MLP tanto la definición de la arquitectura, como la inicialización de los pesos son determinante para mejorar la eficacia en la clasificación.
- La importancia informacional de rasgos y conceptos, puede ser un elemento influyente para calcular los pesos sinápticos iniciales entre las distintas capas del MLP.
- El RGC-Conceptual es un algoritmo que puede resultar útil para abordar la problemática planteada por su propiedad de ser caracterizante.
- Se determinó a CEPAR como herramienta que trabaja con el RLCP la cual se va a utilizar como librería y a MatLab para el diseño de la arquitectura e inicialización de los pesos del Perceptrón Multicapa. Para el desarrollo de la solución se determinó emplear Java y R como lenguajes de programación y NetBeans en su versión 8.0.2 como entorno de desarrollo integrado para la implementación de la solución.

Capítulo 2: Propuesta para el diseño de la arquitectura e inicialización de los pesos de un Perceptrón Multicapa aplicando el algoritmo conceptual RGC

En el presente capítulo se describe la propuesta de solución al aplicar el algoritmo conceptual RGC para el diseño de la arquitectura del Perceptrón Multicapa, haciendo énfasis en la definición de las neuronas correspondientes a las capas entrada y oculta, así como a la inicialización de los pesos sinápticos del modelo neuronal, mediante la importancia informacional de rasgos y conceptos.

2.1 Descripción de la propuesta de solución

Se considera un Perceptrón Multicapa compuesto por una capa de entrada, una capa oculta y una capa de salida en la que sus neuronas se encuentran conectadas con conexiones pesadas entre sí. Para el diseño de un MLP con estas características primeramente es necesario determinar el número de neuronas correspondientes a cada una de las capas y posteriormente los dos conjuntos de pesos sinápticos iniciales; los que se encuentran entre la capa de entrada y la capa oculta y aquellos entre la capa oculta y la capa de salida.

El cálculo de los elementos para diseñar una red con las características anteriormente descritas se realiza utilizando el algoritmo conceptual RGC, teniendo en cuenta sus dos etapas. La primera etapa (determinación extensional) se encarga de agrupar los objetos por clases, mientras que su segunda etapa (determinación intencional) calcula los conceptos que caracterizan a los grupos de clases. La propuesta de solución la cual cuenta con tres fases se muestra en la figura 6.

La primera fase "Cálculo de los conceptos" comienza con la determinación extensional del algoritmo RGC en caso de trabajar con un problema no supervisado, agrupando los objetos en clases (algoritmo 1), luego basándose en la etapa intencional se calculan los conceptos (algoritmo 2,3 y 4), los cuales representan las características distintivas de los objetos que conforman los grupos. Si el problema es supervisado se parte directamente de la determinación intencional (algoritmo 2,3 y 4) puesto que ya las clases se encuentran agrupadas.

En la segunda fase "Selección de las neuronas por capa" se determina el número de neuronas pertenecientes a cada una de las capas del MLP (entrada, oculta y salida), en correspondencia con los resultados arrojados luego de aplicar el RGC.

En la tercera fase “Determinación de los pesos sinápticos iniciales” como su nombre lo indica se definen los pesos de las conexiones entre la capa de entrada y la oculta (algoritmo 1) así como entre la capa oculta y la de salida (algoritmo 2).

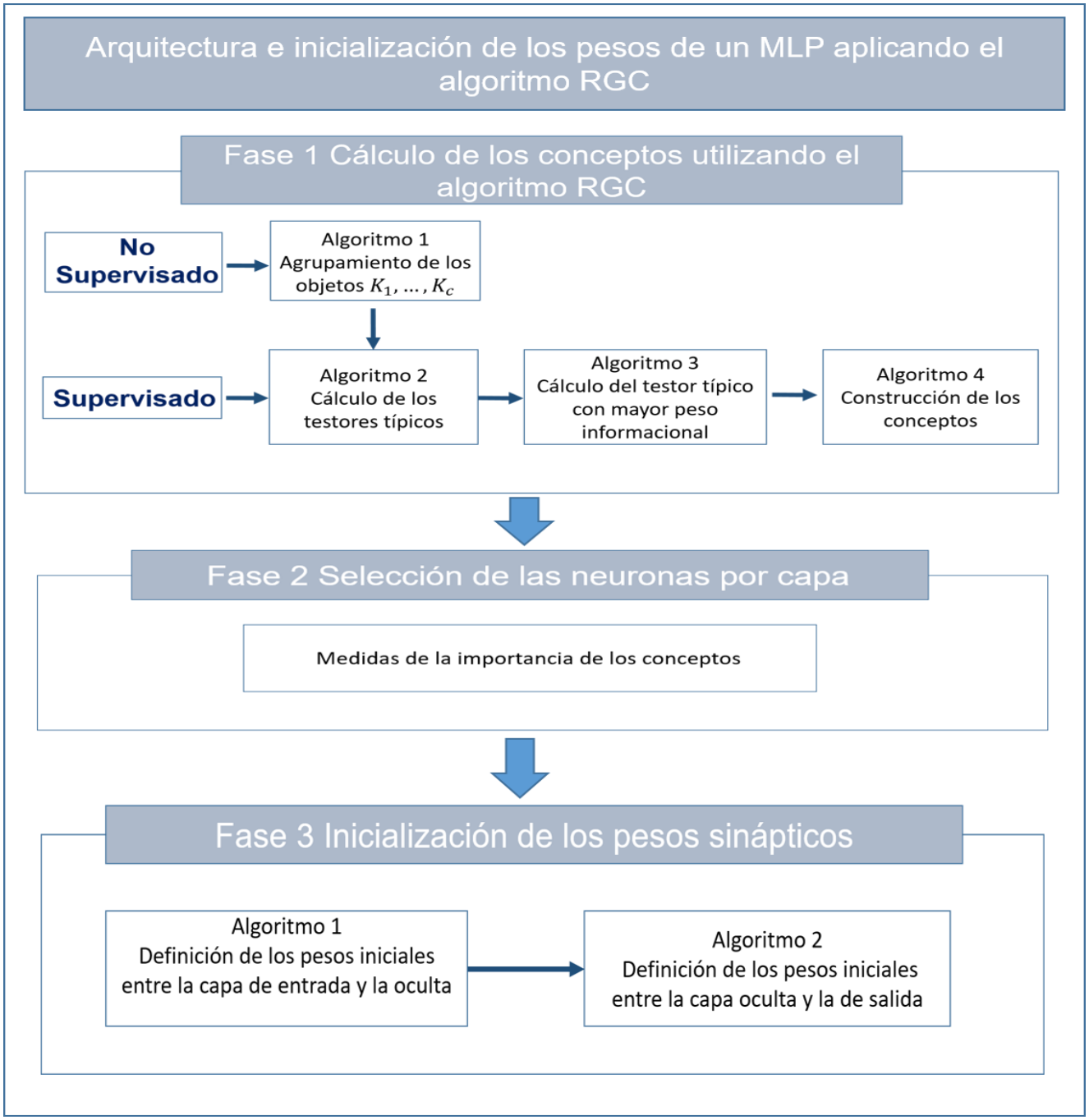


Figura 6 Esquema general para la solución (Fuente: elaboración propia)

Fase 1: Cálculo de los conceptos utilizando el RGC

En esta fase se aplica el algoritmo RGC para tomar el testor típico de mayor peso informacional y el número de conceptos formados.

Objetivo: Calcular los testores típicos y seleccionar el de mayor peso informacional y los conceptos asociados a cada clase.

Independientemente del tipo de problema a resolver lo constituye el planteamiento formal del mismo, con lo que se persigue precisar el objetivo, así como determinar las fuentes de información con las cuales se trabaja. La presencia del especialista del área de aplicación es de vital importancia, pues sus criterios deciden en gran medida varios de los aspectos a tener en cuenta, tales como: cuáles son los rasgos y su importancia, cómo se comparan las variables y los objetos, determinar la forma en que se representan los objetos de estudio, si se tienen clases, si son disjuntas o no, duras o difusas. Para este proceso se propone adoptar la metodología descrita en (Ruiz-Shulcloper 2009).

Del enfoque lógico-combinatorio para problemas de clasificación sin aprendizaje, se conoce que los objetos pueden ser representados en una matriz inicial (MI), $MI = \{I(O_1), I(O_2), \dots, IO_m\}$ conjunto de descripciones de los objetos $O_1, O_2, \dots, O_m$ de un universo U, dadas como $I(O_n) = (x_1, (O_i), \dots, x_n(O_n))$, como se puede observar en la tabla 2. Para cada x_i se tiene asociado un conjunto de valores admisibles M_i $i = 1, \dots, n$, consecuentemente el espacio de representación inicial de los objetos, no es otra cosa que $M_1 \times M_2 \times , \dots, \times M_n$ el producto cartesiano de los conjuntos admisibles de valores de los rasgos $x_1, .. x_n$. Sobre M_i se define un criterio de comparación de valores:

C_i: donde G es un conjunto dado($G = [0,1], G = \{0,1\}, G = \{1, \dots, k\}$ y $otros$).

Tabla 2 Matriz inicial (Fuente: elaboración propia)

	x_1	x_2	...	x_n
O_1	$x_1\ (O_1)$	$x_2\ (O_1)$	...	$x_n\ (O_1)$
O_2	$x_1\ (O_2)$	$x_2\ (O_2)$	...	$x_n\ (O_2)$
⋮		⋮		
O_m	$x_1\ (O_m)$	$x_2\ (O_m)$	...	$x_n\ (O_m)$

Cada rasgo constituye una variable aleatoria con valores discretos (nominales u ordinales) o continuos, y se admite la ausencia de información, representada por el símbolo *. En dependencia de la naturaleza del

rasgo pueden utilizarse diferentes criterios de comparación tales como los mostrados en las ecuaciones 2 a 5 ó pueden construirse nuevas funciones que no necesariamente son booleanas.

Ecuación 2: Criterio de comparación igualdad estricta

$$C_s\left(X_s(O_i), X_s(O_j)\right) = \begin{cases} 1 \quad si\ X_s(O_i) = X_s(O_j)\ \vee\ X_s(O_i) = *\vee\ X_s(O_j) = * \\ 0\ en\ otro\ caso \end{cases}$$

Ecuación 3: Intervalos de valores semejantes

$$C_s\left(X_s(O_i), X_s(O_j)\right) = \begin{cases} 1 \quad si\ X_s(O_i), X_s(O_j)\ \in\ \left[a_{p,} a_{p+1}\right] \vee\ X_s(O_i) = *\vee\ X_s(O_j) = * \\ 0\ en\ otro\ caso \end{cases}$$

Ecuación 4: Umbral de error admisible de semejanza

$$C_s\left(X_s(O_i), X_s(O_j)\right) = \begin{cases} 1 \quad si\ \left|X_s(O_i) - X_s(O_j)\right| \leq \varepsilon_s\ \vee\ X_s(O_i) = *\vee\ X_s(O_j) = * \\ 0\ en\ otro\ caso \end{cases}$$

Ecuación 5: Conjunto de valores semejantes

$$C_s\left(X_s(O_i), X_s(O_j)\right) = \begin{cases} 1 \quad si\ X_s(O_i), X_s(O_j)\ \in\ A_p\ \vee\ \ X_s(O_i) = *\vee\ X_s(O_j) = * \\ 0 \quad en\ otro\ caso \end{cases}$$

Entre las descripciones de objetos se define una función de semejanza:

$$\beta: (M_1 \times, \dots, \times M_n)^2 \longrightarrow \beta$$

A partir de MI y β se puede construir una matriz que refleje las relaciones de semejanza entre todos los objetos sujetos a estudio. A esta matriz se le llama matriz de semejanza (MS) y es:

$$MS = \left|\left|\beta(I(O_i), I(O_n))\right|\right|_{m \times n}$$

MS es simétrica y $\beta(I(O_i), I(O_n) = 1, i = 1, \dots, m.$

La función de Semejanza β determina una medida numérica del grado de similaridad de un objeto con respecto al otro teniendo en cuanta las similitudes entre los rasgos. Esta función unida a la MI son usadas para hallar la matriz de semejanza.

Para determinar la similitud entre rasgos, es importante considerar la naturaleza de los mismos, en dependencia de esta, se utilizan diferentes criterios de comparación (Ruiz-Shulcloper 2009).

Umbral de Semejanza: El umbral es la cota inferior de los posibles valores de semejanza existentes para cada uno de los casos recuperados con respecto al nuevo caso. Su función consiste en restringir el intervalo en caso de que se desee lograr una mayor precisión en la obtención de los casos más semejantes.

La magnitud $\beta_0 \in \Delta$ (Δ=[0,1], Δ={0,1}, Δ={1,..,k}, y otros) se denomina umbral de semejanza y puede ser calculada, por ejemplo, según las ecuaciones 6, 7 y 8 (Ruiz Shulcloper 2013).

Ecuación es 6 ,7 y 8 Cálculo del umbral de semejanza:

$$a)\ \beta_0 = \frac{2}{\mathrm{m}(\mathrm{m}-1)} \sum_{\mathrm{i}=1}^{\mathrm{m}-1} \sum_{\mathrm{j}=\mathrm{i}+1}^{\mathrm{m}} \beta(\mathrm{I}(\mathrm{O_i}), \mathrm{I}(\mathrm{O_j}))$$

$$b)\ \beta_0 = \max_{\substack{i=1\ldots m-1 \\ i \neq j}} \left\{ \min_{j=i+1\ldots m} \{\beta(I(O_i), I(O_j))\} \right\}$$

$$c)\ \beta_0 = \min_{\substack{i=1\ldots m-1 \\ i \neq j}} \left\{ \max_{j=i+1\ldots m} \{\beta(I(O_i), I(O_j))\} \right\}$$

El criterio de agrupamiento unido a la función de semejanza y a la existencia de otros objetos, es la razón por la cual un objeto va a pertenecer a un agrupamiento o por qué dos objetos pertenecerán a una misma agrupación. De esta manera se puede apreciar que la selección del criterio a usar, es determinante en la calidad de la solución del problema de clasificación no supervisado.

La definición del criterio de agrupamiento debe estar basada en el conocimiento que se tenga sobre el problema en concreto que se está tratando, para poder definir así el tipo de comportamiento entre los objetos a partir de sus semejanzas, que resulte significativo, según el problema en particular. Por tanto, al seleccionar algún criterio de agrupamiento, dado un conjunto de objetos y la función de semejanza, se ha definido indirectamente, la familia de agrupaciones, es decir, la estructura del universo ha sido conformada.

El planteamiento formal de la estructuración de universos, de la clasificación no supervisada, consiste en encontrar un criterio de agrupamiento que responda a los intereses del problema en cuestión.

β_0-semejantes

Dos descripciones (objetos) $I(O_n), I(O_j)$ se denominan β_0 -semejantes si $\beta(I(O_i), I(O_j)) \geq \beta_0$ .Sea un espacio de representación $\Phi = \{\Xi, \beta\}$, sea dado un conjunto de descripciones de objetos $MI = \{I(O_1), I(O_2), \ldots, IO_m\}$, una función de semejanza $\beta: MI \times MI \longrightarrow \Delta$, y un umbral $\beta_0 \in \Delta$ que define la β_0-semejanza entre los elementos de MI.

Criterio agrupacional β_0-conexo

Se dice que $C \subseteq MI, C \neq \emptyset$ es una componente β_0-conexa si y sólo si:

a) $\forall\, O_i\, O_j \in C \in\ O_{i_1}, \ldots, O_{i_q} \in C\ [O_i = O_{i_1} \wedge O_j = O_{i_q} \wedge\ \forall\, p \in \{1, \ldots, q-1\}\, \beta\ (O_{i_p}, \ldots, O_{i_{p+1}}) \geq \beta_0]$

b) $\forall O_i \in MI\big[\big(O_j \in C, \beta\big(O_i, O_j\big) \geq \beta_0\big) \Rightarrow O_i \in C\big]$

c) Todo elemento β_0-aislado es una componente β_0-conexa (degenerada).

La condición a) expresa que para cualquier par de elementos de C existe una sucesión de elementos en C, que empieza en O_i y termina en O_j tales que uno es β_0-semejante al siguiente, b) significa que no existe fuera de C un elemento β_0-semejante a un elemento de C.

Criterio Agrupacional β_0-compacto

Se dice que $\mathcal{B}$ (F 9) cMI, $\mathcal{B} \neq$o es un conjunto β_0-compacto si y sólo si:

a) $\forall \mathrm{O_j} \in \mathrm{MI}[\mathrm{O_i} \in \mathcal{B} \wedge \max\limits_{\substack{O_t \in MI \\ O_t \neq O_i}} \{\beta(O_i, O_t)\} = \beta\big(O_i, O_j\big) \geq \beta_0\ ] \Rightarrow O_i \in \mathcal{B}$

b) $[\max\limits_{\substack{O_t \in MI \\ O_t \neq O_i}} \{\beta(O_P, O_i)\} = \beta(O_P, O_t) \geq \beta_0 \wedge O_t \in \mathcal{B}] \Rightarrow O_i \in \mathcal{B}$

c) $|\mathcal{B}|$ es la mínima

d) Todo elemento β_0-aislado constituye un conjunto β_0-compacto (degenerado).

La condición a) expresa que todo elemento de $\mathcal{B}$ tiene en $\mathcal{B}$ al elemento que más se le parece que es β_0-semejante con él. La condición b) significa que no existe fuera de $\mathcal{B}$ un elemento cuyo elemento más parecido que sea β_0-semejante esté en $\mathcal{B}$, la tercera condición c) especifica que $\mathcal{B}$ debe ser el conjunto más pequeño de cardinalidad mayor que 1.

El algoritmo 1 realiza el agrupamiento de los objetos semejantes, a partir de la MI se obtiene la MS y posteriormente se utiliza un umbral de semejanza $\beta_0 \in \Delta$ y un criterio de agrupamiento para conformar la ME.

Algoritmo 1: "Agrupamiento de los objetos"

Entrada: MI // Matriz inicial

β // Función de semejanza

Π // Criterio de agrupamiento

$C_s\left(X_s(O_i), X_s(O_j)\right)$ // Funciones de comparación por rasgos

Salida: ME // Matriz estructurada en c agrupamientos (K_c).

Paso 1: Construir la matriz de semejanza utilizando la función de semejanza β.

Paso 2: Calcular el umbral de semejanza utilizando un criterio β_0.

Paso 3: Agrupar siguiendo un criterio agrupacional Π (Ruiz Shulcloper 2013).

Como dato de salida del algoritmo 1 se obtiene la ME, la cual está estructurada en c agrupamientos (K_c) de objetos similares, culminando así la primera etapa con la determinación extensional.

Para la construcción de los conceptos, es necesario seleccionar en primer lugar los rasgos determinantes en el problema, esto es realizado mediante el cálculo de los testores, lo que permite la reducción eficiente de la cantidad de rasgos que describen los objetos. A partir de la Matriz de Entrenamiento (ME) se obtiene la Matriz de Diferencias (MD) y posteriormente la Matriz Básica (MB), mediante la cual se hallan los testores. Se aplica un algoritmo para obtener los testores típicos, luego se obtienen los l-complejos de cada agrupamiento con el operador Refunión Extendida (Martínez-Trinidad y Ruiz-Shulcloper 1999), después el operador Generalización (GEN) y finalmente se forman los conceptos para cada clase perteneciente al problema.

La ME es una matriz estructurada en m clases de objetos similares. La MD una matriz booleana que se obtiene de la ME comparando los respectivos valores de los rasgos en objetos de clases diferentes por medio de los criterios de comparación de valores de las variables (Shulcloper et al. 1995).

La MB es una matriz formada únicamente por filas básicas de la MD, las cuales constituyen el conjunto de testores. Se entiende por fila básica según (Ruiz-Shulcloper 2009).

Se entiende por fila básica según (Ruiz-Shulcloper 2009).

La fila i_t es básica sí y sólo sí en MD no existe fila i_p alguna que sea subfila de i_t.

Sea i_p, i_t filas de MD Se dice que i_p es subfila de $_{it}$ sí y sólo sí:

a) $\forall\forall j\ (ai_tj = 0 \longrightarrow ai_pj = 0)$

b) $\forall\exists j_0\ (ai_tj_0 = 1 \wedge ai_pj_0 = 0)$

Algoritmo 2: "Cálculo de los testores típicos"

Entrada: ME// Matriz de Entrenamiento/

Salida: TT//Conjunto de Testores Típicos

Paso 1: Calcular la matriz de diferencias.

Paso 2: Calcular la matriz básica.

Paso 3: Aplicar algoritmo para el cálculo de los TT.

El número de testores típicos para ciertos problemas puede ser muy grande. Esto da como resultado que cada clase podría tener asociado una gran cantidad de conceptos o propiedades, por lo cual, el algoritmo 3 calcula la utilidad de cada testor típico utilizando las propuestas descritas en (Ruiz-Shulcloper 2009). (Reyes-González 2014) para seleccionar el mejor.

Algoritmo 3. Cálculo del testor típico de mayor peso informacional.

Entrada: TT//Conjunto de Testores Típicos

Salida: TT' // Testor típico de mayor peso informacional

Paso 1: Calcular el peso ε_i de los rasgos X_i que aparecen en la familia de testores típicos según:

$$\varepsilon(x_i) = \alpha F(x_i) + \gamma L(x_i)$$

α>O, β>O y α+β=1. α y β parámetros que ponderan la participación o influencia de frecuencia de aparición y la longitud de los testores típicos respectivamente.

$$\alpha = \beta = 0.5 \qquad p(x_i) = \frac{T_i}{|T|} \qquad L(X_i) = \frac{\sum_{t \in T} \frac{1}{|T_i|}}{|T|}$$

T_i : número de testores donde aparece rasgo i.

$|T|$: número de testores.

$|T_i|$: número de rasgos que forman el testor T_i

Paso 2: Seleccionar los testores típicos de menor longitud.

Paso 3: $\Psi(t_i) = \sum_{x \in I} \frac{\varepsilon(x)}{|TT|}$ Calcular el peso de los testores típicos de menor longitud según:

Paso 4: Ordenar descendentemente los testores típicos según su peso.

Paso 5: Seleccionar el testor típico de mayor peso informacional (Ψ), a partir de un umbral previamente definido.

La importancia de utilizar el testor típico de mayor peso informacional radica en que el mismo no contiene todos los rasgos, sino los más relevantes al problema en cuestión, por tanto cuando se haga referencia al conjunto de rasgos, debe entenderse que son sólo aquellos rasgos presentes en el testor típico con que se trabaje (Reyes-González, Yunia et al. 2016).

En el algoritmo 4, se obtienen los l-complejos de cada agrupamiento con el operador Refunión Extendida (Pons-Porrata 2004) luego se aplica el operador Generalización (GEN) y finalmente se forman los conceptos para cada clase perteneciente al problema.

Algoritmo 4. Construcción de los conceptos

Entrada: TT' // Testor típico de mayor peso informacional

ME //Matriz de entrenamiento

Salida: l-complejos //conceptos para cada clase

Paso 1: Calcular l-complejos

Paso 2: Calcular la estrella G_τ ($K_i \setminus K_1$, ..., K_{i-1}, K_{i+1}, ..., K_c) para cada clase K_i, i = 1,…, *c.*

Paso 3: Aplicar reglas de generalización en dependencia del tipo de variable.

La complejidad computacional de este algoritmo depende de la naturaleza de las variables que describen a los objetos, de los criterios de comparación por rasgos y la función de semejanza entre objetos seleccionados, del criterio de agrupamiento que se utilice y de las características que posean los objetos de un mismo agrupamiento. El operador de mayor complejidad computacional es el de Refunión Extendida. En el caso peor la complejidad es exponencial. No obstante, los pasos del operador de Refunión Extendida pudieran optimizarse explotando las características propias de la función de semejanza y del criterio de agrupamiento empleados(Pons-Porrata 2004).

Fase 2: Selección de las neuronas por capa

Esta fase corresponde a la estructura física del modelo en función de la cantidad de unidades de procesamiento (neuronas) correspondientes a las capas de entrada, oculta y salida.

Objetivo: Seleccionar el número de neuronas de la capa de entrada y la capa oculta.

Aspectos a examinar:

- Capa de entrada: las neuronas coinciden con los rasgos que componen los conceptos construidos en función de los testores típicos utilizados.
- Capa oculta: las neuronas están asociadas a los conceptos construidos.
- Capa de salida: las neuronas coinciden con el número de clases definidas inicialmente en el problema.

Fase 3: Determinación de los pesos sinápticos iniciales

En esta etapa se establecen los pesos sinápticos iniciales del modelo.

Objetivo: Seleccionar el conjunto de pesos iniciales utilizando métricas definidas como una heurística en función de la importancia informacional de rasgos y conceptos.

Para la inicialización de los pesos se proponen dos algoritmos, el primero referente a los pesos entre las neuronas de la capa de entrada y las correspondientes a la capa oculta, mientras que el segundo define los pesos iniciales entre la capa oculta y la de salida. Esta división está dada debido a que entre las dos primeras capas el que rige el proceso es el rasgo, y por ende se utiliza el peso informacional del mismo en relación con la asociación interpretativa que tiene con los conceptos que definen las neuronas de la capa oculta. Mientras que entre las neuronas de la capa oculta y las de salida la relación está entre los conceptos y las clases, siendo dirigidos por la importancia informacional dada por los conceptos.

Para la determinación de los pesos sinápticos iniciales entre las dos primeras capas el algoritmo 5 utiliza la métrica definida por (Ruiz-Shulcloper 2009), donde la importancia informacional del rasgo $\varepsilon(x_i)$ se calcula según la ecuación $\varepsilon(x_i) = \alpha F(x_i) + \gamma L(x_i)$ α>O, β>O y α+β=1. α y β parámetros que ponderan la participación o influencia de frecuencia de aparición y la longitud de los testores típicos respectivamente.

$$\alpha = \beta = 0.5 \qquad p(x_i) = \frac{T_i}{|T|} \qquad L(X_i) = \frac{\sum_{t \in T} \frac{1}{|T_i|}}{|T|}$$

T_i : número de testores donde aparece rasgo i.

$|T|$: número de testores.

$|T_i|$: número de rasgos que forman el testor T_i

Algoritmo 1. Definición de $W_{0e/o}$ entre las neuronas de entrada y las ocultas

---

Entrada: N_e // Neuronas de entrada

N_o // Neuronas ocultas

Salida: W_{0eo} // Vector inicial de pesos entre las neuronas de la capa de entrada y la capa oculta.

Paso 1: Calcular el peso de cada rasgo presente en N_e aplicando la ecuación:

$$\varepsilon(x_i) = \alpha F(x_i) + \gamma L(x_i)$$

Paso 2: Inicializar los pesos sinápticos entre la capa de entrada y la capa oculta.

$W_{0eo} = [W_{011}, \ldots, W_{0eo}]$

La inicialización de los pesos sinápticos entre la capa oculta y la de salida es determinada por el algoritmo 6, a partir de las métricas propuestas en (Reyes-González, Y. et al. 2016) que fija la importancia informacional de los conceptos identificados en las neuronas de la capa oculta. Para determinar el peso informacional de los subconceptos y los conceptos se incorpora la semántica del significado de los valores de los rasgos a las ecuaciones con similares propósitos descritas en (Ruiz-Shulcloper 2009).

El peso informacional de un subconcepto $PI(b_i)$: Se define utilizando la ecuación del peso informacional de un objeto incorporándole la medida denominada importancia informacional del valor del rasgo y se calcula:

$$PI(b_i) = \frac{1}{|k_j|} \cdot \sum_{j=1}^{n} a_j^i \cdot \varepsilon(x_i) \cdot t_{xi}(v_i)$$

Cuando se comparan dos rasgos con distintos valores, la medida $\iota_r(v_i)$ se calcula como la media de ambos. En la ecuación anterior el subíndice i recorre la cantidad de valores de los rasgos, j la cantidad de clases y $a_j^i = |\{b_t \in k_j | b \; y \; b_t \; son \; semejantes \; en \; el \; rasgo \; x\}|$

Con los valores calculados a nivel de cada subconcepto se aplica la regla del máximo peso para calcular el peso informacional de un concepto.

El peso informacional del concepto $PI\ (P_i)$: Se determina a partir del subconcepto de mayor peso informacional y se calcula:

$$\mathrm{PI}(\mathrm{P_i}) = \max_{i=1\ldots\gamma}\{\mathrm{PI_j}(b)\}$$

Algoritmo 2. Definición de $W_{0o/s}$ entra las neuronas ocultas y las de salidas

Entrada: N_o // Neuronas ocultas

N_s // Neuronas de salida

Salida: W_{0os} // Vector inicial de pesos entre las neuronas de la capa oculta y la de salida.

Paso 1: Determinar los subconceptos asociados a cada concepto

Paso 2: Calcular el peso informacional de cada subconcepto:

$$\mathrm{PI}(\mathrm{b_i}) = \frac{1}{|\mathrm{k_j}|} \cdot \sum_{\mathrm{j=1}}^{\mathrm{n}} \mathrm{a_j^i} \cdot \varepsilon(\mathrm{x_i}) \cdot \mathrm{t_{xi}}(\mathrm{v_i})$$

Paso 3: Calcular el peso informacional de cada concepto:

$$\mathrm{PI}(\mathrm{P_i}) = \max_{i=1\ldots\gamma}\{\mathrm{PI_j}(b)\}$$

Paso 4: Inicializar los pesos sinápticos entre la capa oculta y la capa de salida.

$W_{0os} = [W_{011}, \ldots, W_{0os}]$

Culminada la segunda fase, la red neuronal se encuentra lista para ser entrenada. La figura (figura 2.3) muestra el diseño arquitectónico del MLP después de aplicada la propuesta de solución.

La definición del MLP propuesto estará en concordancia con la definición de redes neuronales artificiales en término de grafos dado en (del Brío y Molina 2001). La figura 7 define el diseño arquitectónico del MLP como un grafo dirigido con las siguientes propiedades:

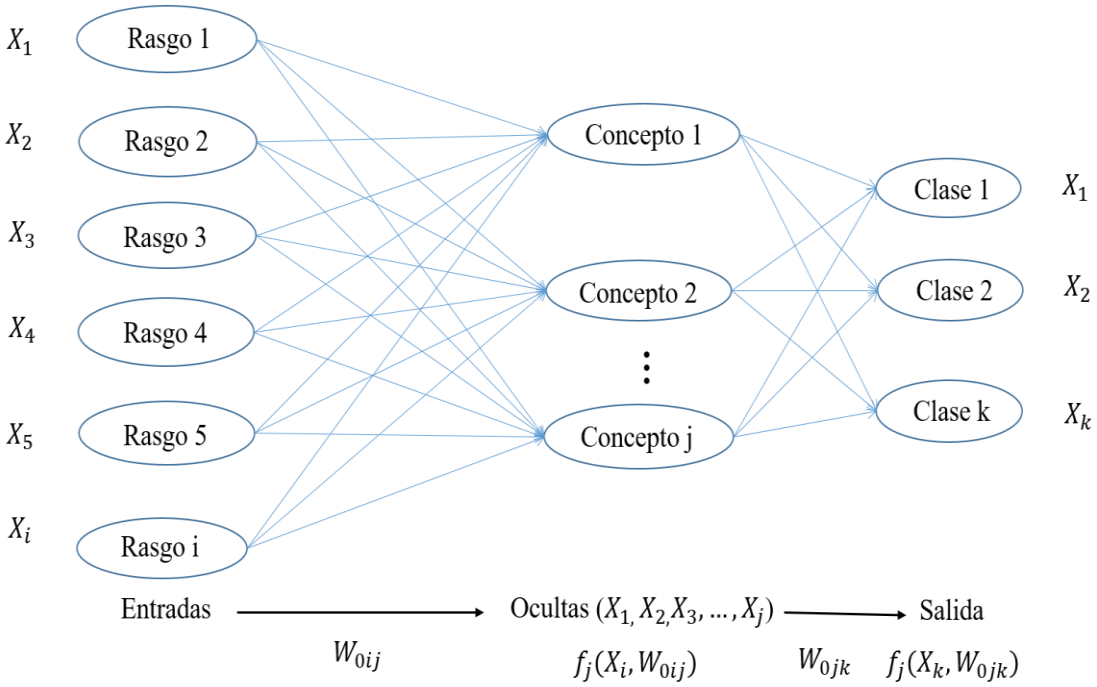


Figura 7 Diseño de la arquitectura e inicialización de los pesos del MLP aplicando la propuesta de solución (Fuente: elaboración propia)

a) A cada nodo i de entrada se la asocia una variable de estado x_i tal que $x_i \in$ {conjunto de rasgos presentes en los conceptos}
b) A cada nodo j oculto se le asocia una variable de estado x_j tal que $x_j \in$ {conjunto conceptos}
c) A cada nodo k de salida se le asocia una variable de estado x_k tal que $x_k \in$ {conjunto de clases definidas en el problema}
d) A cada conexión ij de los nodos i y j se le asocia su correspondiente W_{0ij}.
e) A cada conexión jk de los nodos j y k se le asocia su correspondiente W_{0jk}.
f) Para cada nodo j y k se define una función $f_j(x_i, W_{0ij})$ y $f_j(x_k, W_{0jk})$ respectivamente, que dependerá de los pesos de sus conexiones y de los estados de los nodos i y j a ellos conectados (el Backpropagation exige que la función debe ser diferenciable, y el tipo lo define el rango de valores en los que se presentan las entradas y salidas según la naturaleza del problema).

2.2 Ejemplo al aplicar el algoritmo RGC y determinar los pesos con la base de datos wine.

A continuación se muestra un ejemplo al aplicar la base de datos wine tomado de (Merz y Murphy 1998). La figura 8 representa la matriz inicial al cargar la base de datos wine que cuenta con 177 objetos descritos por 13 rasgos y clasificados en tres tipos de clases.

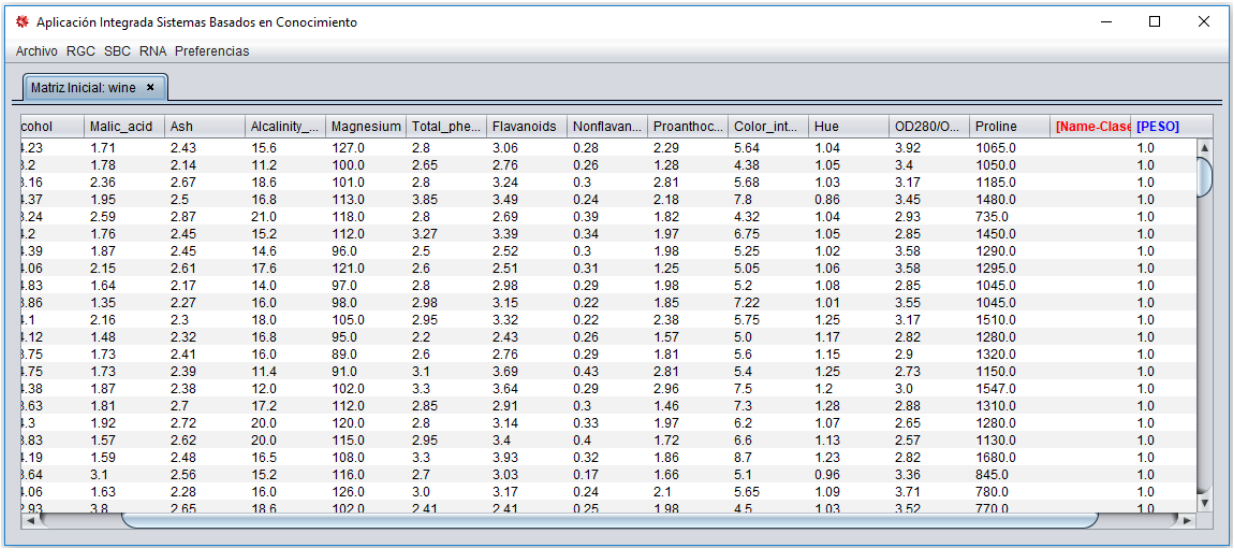


Figura 8 Matriz inicial con los objetos clasificados de la base de datos wine cargados en la aplicación (Fuente: elaboración propia)

Si la base de datos es no supervisada se comienza con la determinación extensional para agrupar los objetos por clases como se muestra en la figura 9.

En este caso al estar en presencia de datos supervisados se parte de la determinación intencional del algoritmo RGC puesto que ya los objetos están agrupados.

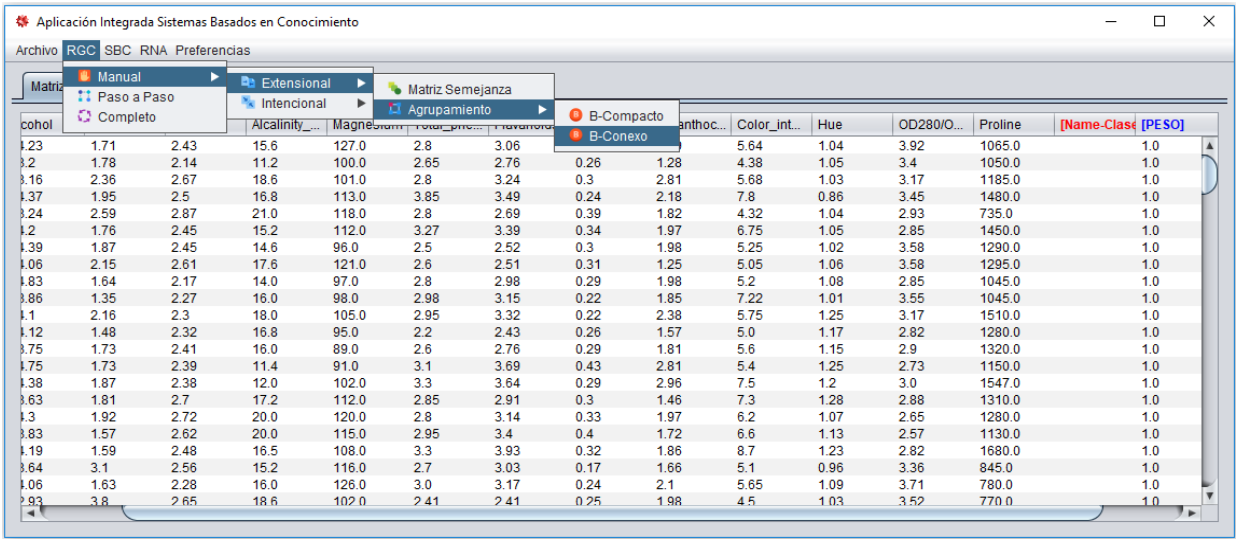


Figura 9 Agrupación de los objetos por clases para bases de datos no supervisadas (Fuente: elaboración propia)

En las siguientes figuras (10-13) se presentan los pasos a seguir para obtener finalmente los datos necesarios para determinar la arquitectura e inicializar los pesos del MLP.

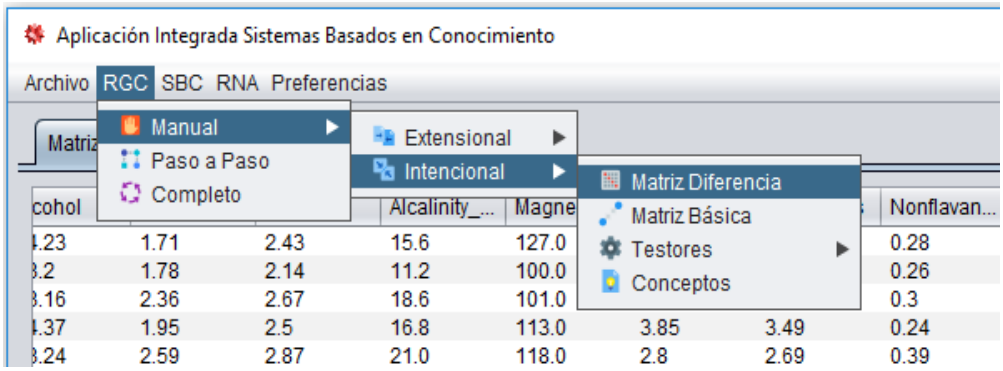


Figura 10 Pasos para obtener la Matriz de diferencia (Fuente: elaboración propia)

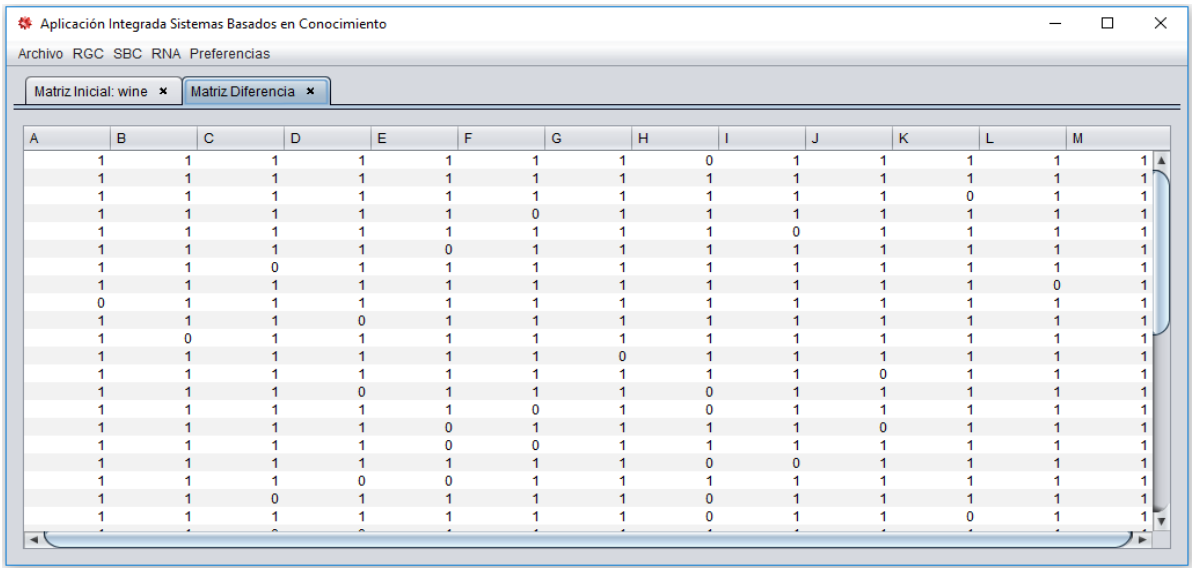


Figura 11 Matriz de diferencia (Fuente: elaboración propia)

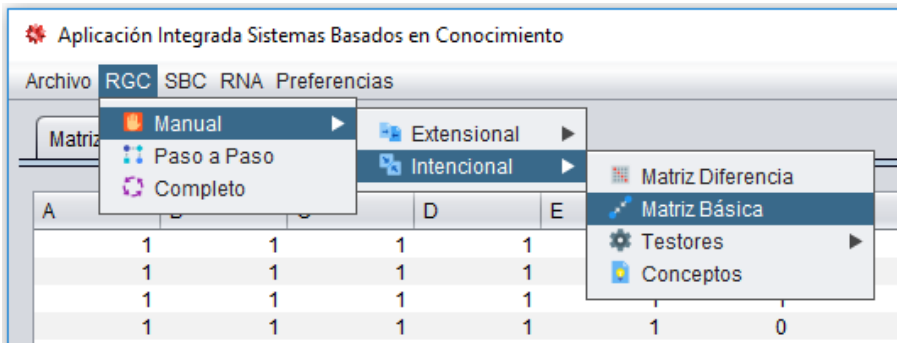


Figura 12 Pasos para obtener la Matriz básica (Fuente: elaboración propia)

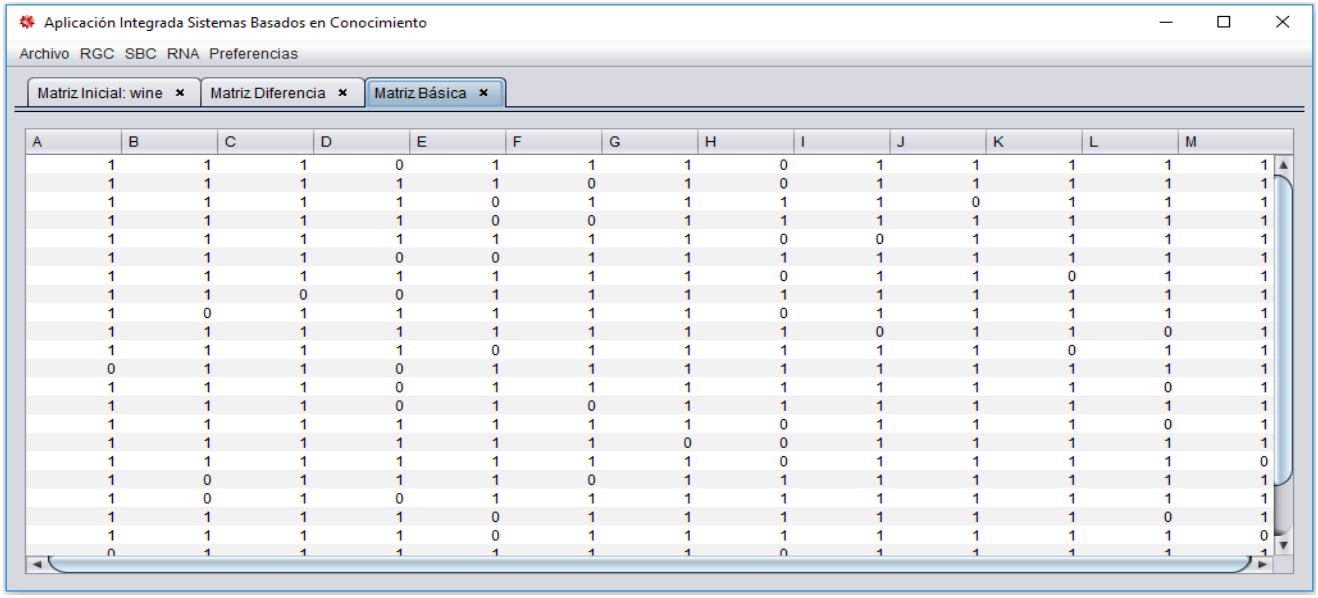


Figura 13 Matriz básica (Fuente: elaboración propia)

Para el cálculo de los testores típicos se debe seleccionar uno de los algoritmos propuestos (Ruíz-Shulcloper, Alba y Lazo 1994) como se muestra en las figuras 14 y 15.

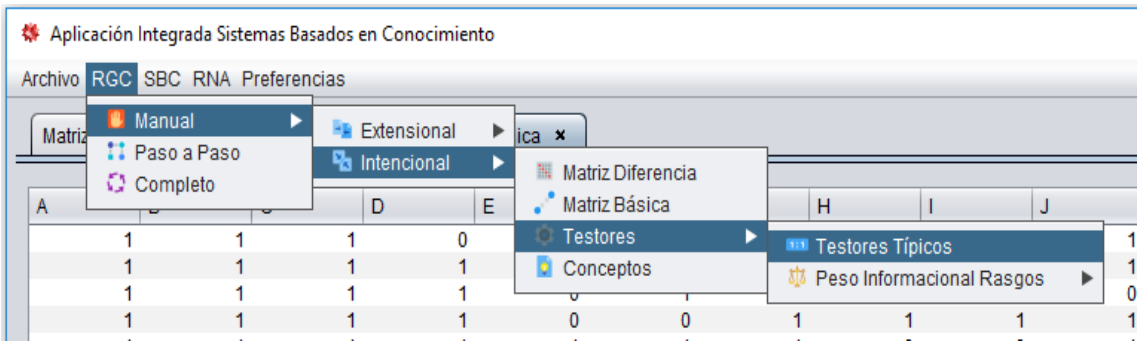


Figura 14 Pasos para calcular los testores típicos

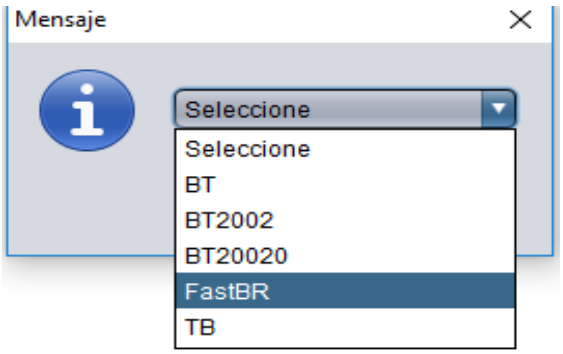


Figura 15 Selección del algoritmo para el cálculo de los testores típicos

En la figura 16 se presenta cómo calcular el peso informacional de los rasgos. Para ello se cuenta con tres opciones: Frecuencia, Longitud y Frecuencia +Longitud, de las cuales el experto debe escoger una de ellas considerando su criterio para la solución del problema.

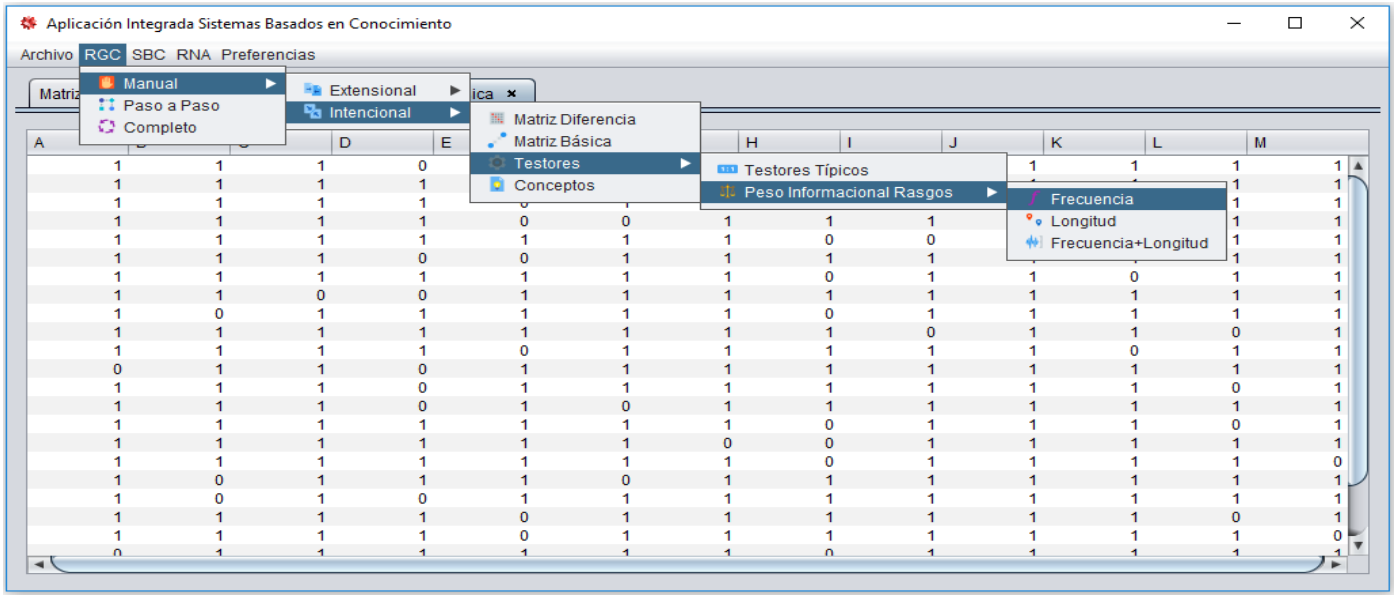


Figura 16 Peso Informacional

En la figura 17 y 18 se muestra cómo calcular los conceptos, los cuales permitirán determinar el número de neuronas correspondiente a la capa oculta y se generan los pesos iniciales.

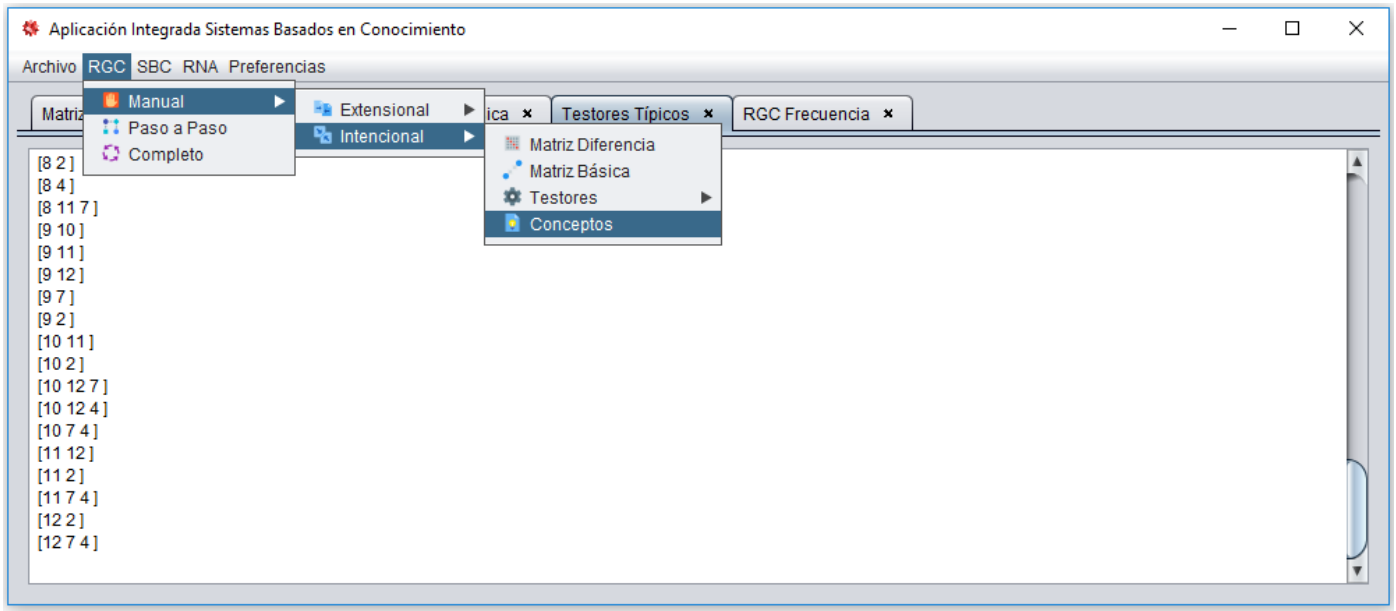


Figura 17 Cálculo de los conceptos

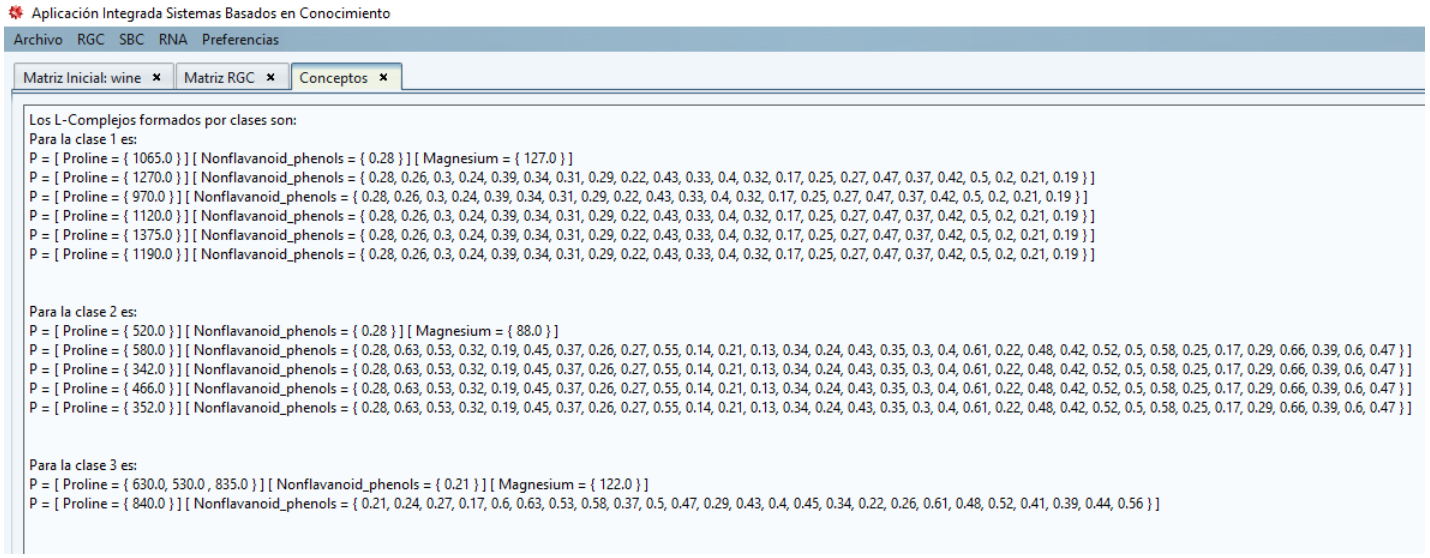


Figura 18 Conceptos generados (Fuente: elaboración propia)

Luego de ser calculados los conceptos y generados los pesos los resultados son exportados en Excel para cargar en la herramienta MatLab y con el ToolBox de Redes Neuronales comprobar la eficacia del mismo (ver figuras de la 19 a la 21).

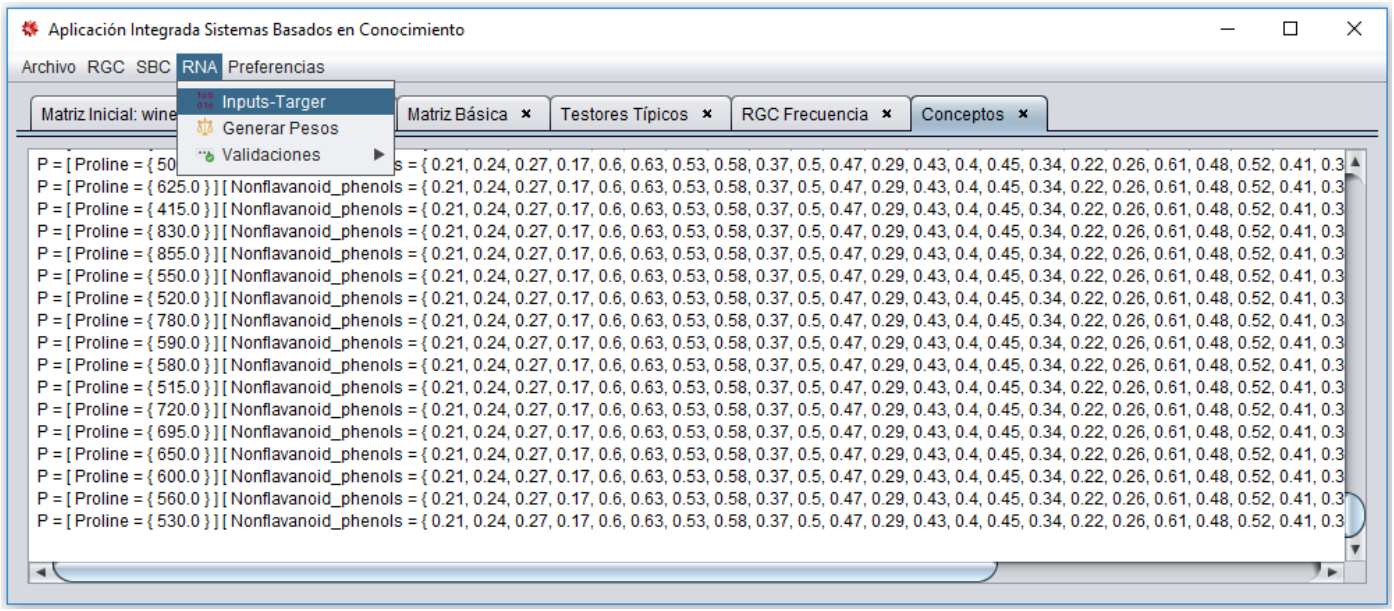


Figura 19 Exportar datos (Fuente: elaboración propia)

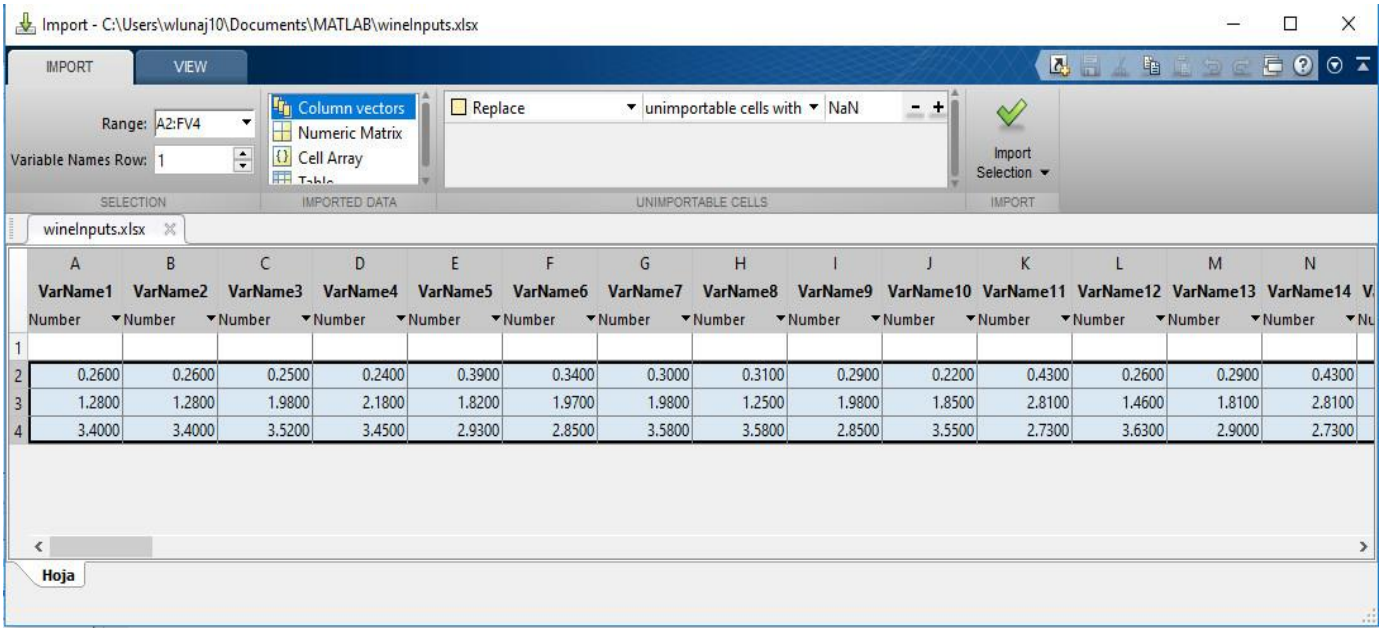


Figura 20 Datos cargados en Matlab

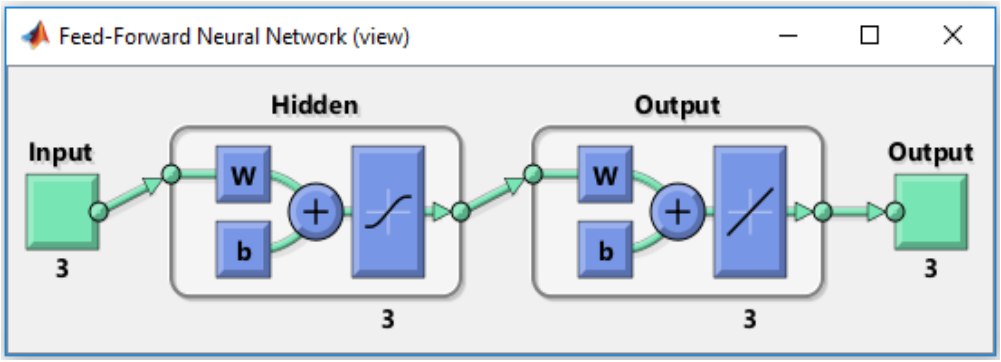


Figura 21 Red neuronal conformada lista para entrenar

2.3 Conclusiones parciales del capítulo

Como resultado de aplicar el algoritmo RGC en el diseño de la arquitectura de un Perceptrón Multicapa con una capa oculta se obtienen las siguientes conclusiones:

- Con la aplicación de los testores típicos se lograron seleccionar las variables de entrada, reduciendo la dimensión de los rasgos iniciales presentes en el problema.
- El cálculo de los conceptos permite identificar la cantidad de neuronas correspondientes a la capa oculta.
- La importancia informacional de rasgos y conceptos proporciona una heurística para calcular los pesos sinápticos iniciales de forma determinística.

Capítulo 3. Preexperimento y resultados

En el presente capítulo se exponen los resultados experimentales al aplicar la propuesta de solución. Se explican las técnicas y criterios a considerar para realizar los preexperimentos, posteriormente se detallan las características de las bases de datos utilizadas, y finalmente los resultados alcanzados al aplicar la propuesta de solución sobre las mismas.

3.1 Descripción de los preexperimentos

Para el análisis de la propuesta de diseño de un MLP se utilizaron 10 bases de datos de reconocimiento internacional disponibles en el repositorio para aprendizaje automatizado de la Universidad de Irvine, California (Merz y Murphy 1998). En la selección se consideran conjuntos de datos con variadas características como: la presencia de rasgos numéricos y no numéricos, la ausencia de información, y diversos en cuanto a la cantidad de rasgos y de objetos. El Anexo II muestra un resumen de las bases de datos seleccionadas y su descripción. Para medir la validez de la propuesta de solución se establece como criterio la eficacia en la clasificación (aumento de la cantidad de patrones clasificados correctamente).

Se compararon tres diseños de redes MLP: el primero donde se seleccionan las neuronas de entrada directamente proporcional a las variables definidas en el problema, las neuronas de las capas ocultas se escogieron inicialmente teniendo en cuenta cantidad de variables de entrada + cantidad de salidas / 2 y los pesos iniciales en valores aleatorios, denominado MLP1. El segundo MLP se crea aplicando el algoritmo LC-Conceptual mientras que el tercero se forma aplicando el algoritmo conceptual RGC reflejado en la propuesta de solución.

Para el entrenamiento y prueba de los diseños, se aplicó el método validación cruzada (k-fold cross validation). Dicho método consiste en dividir los datos maestrales en dos partes; una parte se utiliza como conjunto de entrenamiento para determinar los parámetros del clasificador neuronal y la otra parte, llamada *conjunto de validación*, se utiliza para estimar el error de generalización, es decir, la tasa de clasificación incorrecta del clasificador con datos diferentes a los utilizados en el proceso de entrenamiento. Se selecciona el 85% de los datos para entrenar la red y el 15% restante para la simulación. Se utiliza además la prueba de Friedman según lo sugerido en García et al. (2010), para comprobar si existen diferencias significativas entre la eficacia de cada modelo.

3.2 Descripción de las bases de datos utilizadas

Para los preexperimentos se escogieron 10 bases de datos, 5 supervisadas y 5 no supervisadas con diferentes características en cuanto a cantidad de rasgos. La selección se realiza en función de analizar los

resultados del diseño propuesto en problemas con diferentes dominios en las variables de entrada, como se muestra en las tablas 3 y 4.

Tabla 3 Bases de datos supervisadas.

Base de Datos	Cantidad de Objetos	Cantidad de rasgos	Cantidad de clases
Wine	178	13	3
Annealing	798	16	38
Zoo	101	18	7
Iris	150	5	3
Glass	214	10	7

(Fuente: Elaboración propia)

Tabla 4 Bases de datos no supervisadas.

Base de Datos	Cantidad de Objetos	Cantidad de rasgos
Tae	151	6
Sonar	207	60
Solarflare	324	13
Lymph	148	19
Liver-disorders	345	8

(Fuente: Elaboración propia)

3.3 Eficacia del diseño de la arquitectura e inicialización de los pesos según la cantidad de objetos clasificados correctamente.

El análisis de la eficacia se realiza en función de la cantidad de patrones clasificados correctamente por el modelo con la arquitectura propuesta con respecto al modelo sin aplicar el agrupamiento conceptual. A continuación se muestra un ejemplo al aplicar la base de dato wine tomado de (Merz y Murphy 1998) que cuenta con 177 objetos descritos por 13 rasgos y clasificados en tres tipos de clases.

Parte del resultado del entrenamiento se muestra en la figura 22, en los anexos se presentan las gráficas de los comandos: nntraintool, plotperforms y plotregression, los cuales exponen otros datos de interés.

Puede observarse que en este caso se necesitaron solo 6 iteraciones y un tiempo menor de 1 segundo para entrenar correctamente la red.

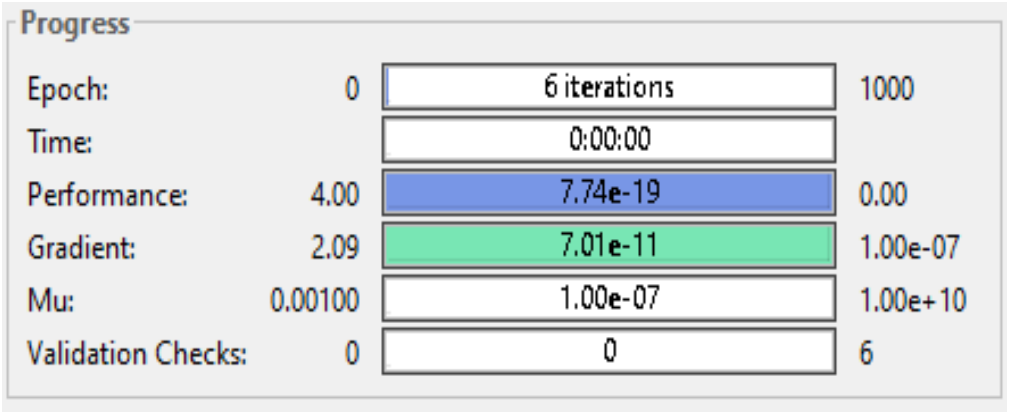


Figura 22 Resultado del entrenamiento con la base de datos Wine aplicando el algoritmo RGC

Las figuras 23 y 24 reflejan la eficacia en la clasificación tomando el 85 % de los datos (151 objetos) para el entrenamiento de la red y el 15 % (27 objetos) para la simulación, de los cuales 10 se agruparon en la clase 1, 10 en la clase 2 y 7 en la clase 3 coincidiendo todas las agrupaciones con la base de datos original. Este resultado representa el 100% de eficacia en la clasificación.

Figura 23 Imagen del comando plotconfusion (base de datos wine, Matlab)

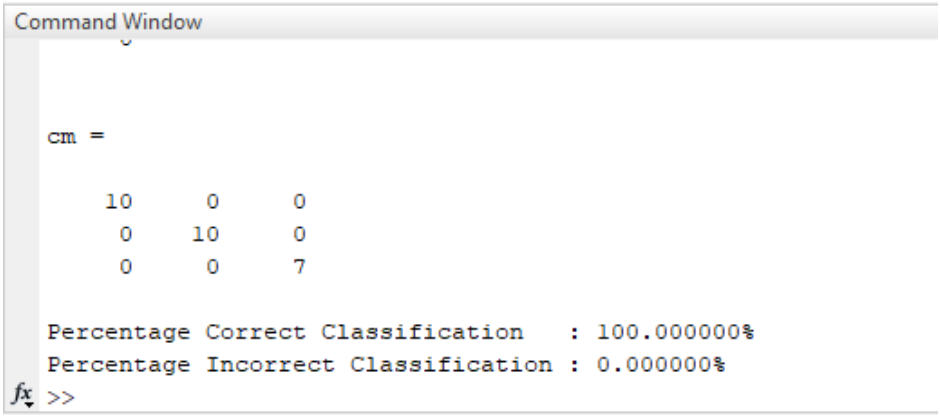


Figura 24 Eficacia en la clasificación de la base de datos wine aplicando el algoritmo RGC

3.3.1 Preexperimento 1

Se realizó el preexperimento 1 que se basa en la comparación de la eficacia del Perceptrón Multicapa sin aplicar el agrupamiento conceptual (MLP1) y el Perceptrón Multicapa aplicando el algoritmo RGC (MLP-RGC) para bases de datos supervisadas. Se evidencia en la tabla 5 y figura 25 que el comportamiento de la eficacia en la clasificación de los objetos, se comporta mejor en el MLP-RGC en el 100% de los casos.

Tabla 5 Datos del preexperimento 1 para bases de datos supervisadas

Base de Datos	Cantidad de variables en MLP 1	Cantidad de variables en MLP-RGC	Eficacia MLP 1 (%)	Eficacia MLP-RGC (%)
Wine	13	3	95.02	100
Annealing	38	11	98.51	98.99
Zoo	18	7	90.23	93.33
Iris	5	3	96.34	100
Glass	10	4	98.32	100

(Fuente: Elaboración propia

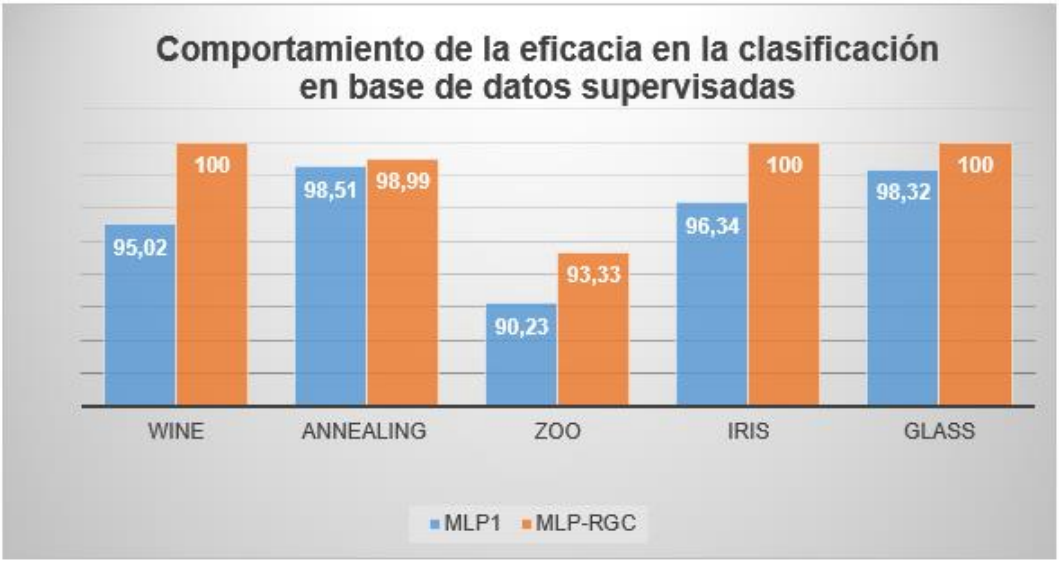


Figura 25 Resultados de la clasificación en bases de datos supervisadas. (Fuente: Elaboración propia)

3.3.2 Preexperimento 2

El segundo preexperimento se basa en la comparación de la eficacia del Perceptrón Multicapa sin aplicar el agrupamiento conceptual (MLP1) y el Perceptrón Multicapa aplicando el algoritmo RGC (MLP-RGC) para bases de datos no supervisadas. Se observa en la tabla 6 y figura 26 que el comportamiento de la eficacia en la clasificación de los objetos, se comporta mejor en el MLP-RGC en el 80% de las bases de datos de prueba.

Tabla 6 Datos del preexperimento 2 para bases de datos no supervisadas

Base de Datos	Cantidad de variables en MLP 1	Cantidad de variables en MLP-RGC	Eficacia MLP 1 %	Eficacia MLP-RGC %
Tae	6	4	92.50	97.33
Sonar	60	3	78.33	92.14
Solarflare	13	12	89.50	88.60
Lymph	19	11	85.20	99.11

Liver-disorders	8	3	98.24	99.45

(Fuente: Elaboración propia)

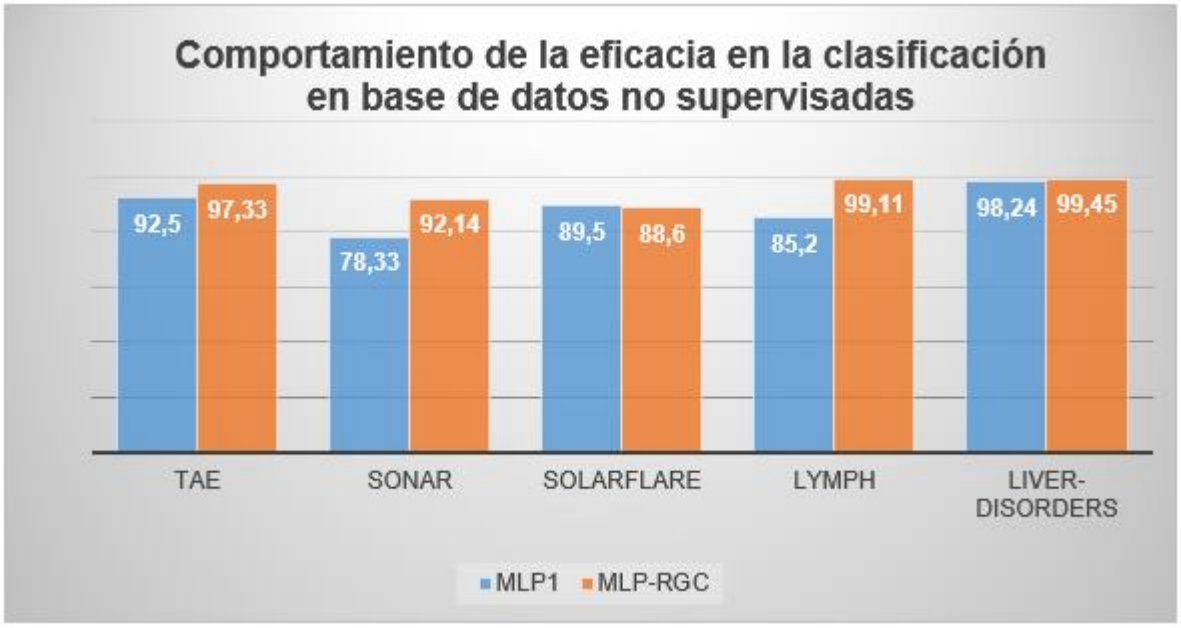


Figura 26 Resultados de la clasificación en bases de datos no supervisadas. (Fuente: Elaboración propia)

3.3.3 Preexperimento 3

En el tercer preexperimento se compara la eficacia del Perceptrón Multicapa aplicando el algoritmo LC-Conceptual (MLP-LC) y el Perceptrón Multicapa aplicando el algoritmo RGC (MLP-RGC) para bases de datos supervisadas y no supervisadas. En la tabla 7 y la figura 27 se visualiza que el comportamiento de la eficacia en la clasificación de los objetos, se comporta mejor en el MLP-RGC en el 90% de las bases de datos de prueba.

Tabla 7 Datos del preexperimento 3 para bases de datos supervisadas y no supervisadas

Base de Datos	Cantidad de variables en MLP-LC	Cantidad de variables en MLP-RGC	Eficacia MLP-LC %	Eficacia MLP-RGC %
Wine	3	3	95.60	100
Annealing	11	11	84.43	98.99

Zoo	7	7	92.60	93.33
Iris	3	3	96.34	100
Glass	4	4	99.32	100
Tae	4	4	96.24	97.33
Sonar	3	3	90.23	92.14
Solarflare	12	12	90.33	88.60
Lymph	11	11	88.21	99.11
Liver-disorders	3	3	90.34	99.45

(Fuente: Elaboración propia)

Figura 27 Resultados de la clasificación aplicando los algoritmos LC-Conceptual y RGC (Fuente: Elaboración propia)

Luego de realizar los preexperimentos descritos correspondientes al MLP1, MLP-LC y MLP-RGC se obtienen los resultados mostrados en la figura 28 correspondientes al porcentaje de clasificaciones correctas.

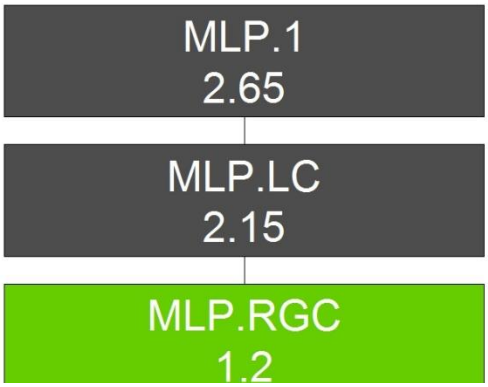


Figura 28 Ranking

Se aplica la prueba de Friedman para la variable por ciento de clasificaciones correctas, con un intervalo de confianza de 95%. La hipótesis nula consiste en asumir que todos los algoritmos son estadísticamente equivalentes. La prueba se realiza con un p-value = 0.0008781 como este valor está por debajo de 0.05 se debe rechazar la hipótesis nula y por tanto se puede afirmar que existen diferencias significativas entre los diseños de redes neuronales. Para detectarlas se aplica la prueba post hoc con la corrección de Finner

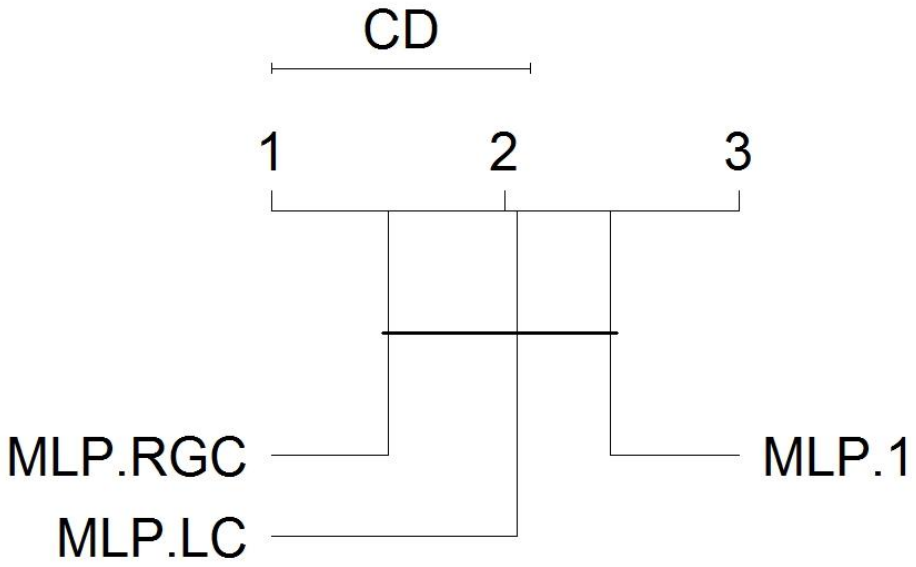


Figura 29 Gráfica que muestra las diferencias significativas entre los modelos neuronales

(García et al. 2010), que arroja como resultado que el modelo propuesto en esta investigación es estadísticamente significativo en comparación con el modelo neuronal MLP1. En la figura 29 se puede observar que la distancia crítica que marca el límite de significación no incluye al modelo del MLP1, por otra parte las diferencias no son significativas con respecto al modelo creado aplicando el algoritmo LC-Conceptual aunque los resultados son superiores. Por tanto a partir del análisis resultante de los

preexperimentos se validan los componentes del modelo y se comprueba la hipótesis planteada en la investigación.

3.4 Conclusiones parciales del capítulo

1 En el 90% de las bases de datos de prueba se evidencia que al aplicar el algoritmo RGC en la determinación de la arquitectura e inicialización de los pesos mejora la eficacia en la clasificación con respecto al LC-Conceptual.

2 El desempeño de la variable eficacia, se comporta mejor en bases de datos supervisadas que, en bases de datos no supervisadas.

3 La heurística para la inicialización de los pesos sinápticos, en función de la importancia informacional de rasgos y conceptos, garantiza una mejor eficacia en la clasificación de objetos.

4 La arquitectura propuesta y la inicialización de los pesos del MLP utilizando el RGC es estadísticamente significativo en comparación con el modelo neuronal sin aplicar algoritmos conceptuales, sin embargo las diferencias no son significativas con respecto al modelo creado aplicando el algoritmo LC-Conceptual aunque los resultados son superiores.

Conclusiones

A partir de la sistematización de los principales referentes teóricos que sustenta la investigación se confirma que la determinación de la arquitectura del Perceptrón Multicapa aplicando el algoritmo LC-Conceptual presenta limitaciones. En dicho algoritmo al aplicar el operador Refunión Condicionada pueden quedar objetos fuera de la clasificación y por ende afecta la eficacia. Todo ello fundamenta la necesidad del desarrollo de una nueva propuesta de solución.

El diseño de la arquitectura e inicialización de los pesos sinápticos de Redes Neuronales Artificiales de tipo Perceptrón Multicapa aplicando el algoritmo conceptual RGC permitió disminuir la cantidad de rasgos para el análisis del número de neuronas pertenecientes a la capa de entrada, identificar el número de neuronas que debía tener la capa oculta y calcular los pesos sinápticos de las conexiones entre las neuronas.

Los métodos científicos empleados para la valoración de la propuesta de solución permitieron comprobar que la solución propuesta contribuye a mejorar la eficacia en la clasificación con respecto al modelo formado aplicando el algoritmo LC-Conceptual y sin aplicar el agrupamiento conceptual.

Recomendaciones

1. Abordar el entrenamiento del MLP utilizando el agrupamiento conceptual.
2. Estudiar la aplicación del agrupamiento conceptual en la definición de la arquitectura y entrenamiento de otros modelos neuronales.

Referencias Bibliográficas

CABRERA, Y.F., PÉREZ, R.B., MOTA, Y.C. y JIMENEZ, G.R., 2011. Improving the MLP learning by using a method to calculate the initial weights of the network based on the quality of similarity measure. *Mexican International Conference on Artificial Intelligence*. S.l.: Springer, pp. 351-362.

CAJAMARCA-PALMA, L.A. y GARCÍA-LEMA, C.E., 2017. *Evaluación de la Estructura De Daños de Un Edificio Postsísmico Aplicando Técnicas de Mapas Cognitivos Difusos y Redes Neuronales de la Universidad de Guayaquil.* S.l.: Universidad De Guayaquil. Facultad de Ciencias Matemáticas y Físicas. Carrera de Ingeniería en Sistemas Computacionales.

COELLO, L., FERNÁNDEZ, Y., FILIBERTO, Y. y BELLO, R., 2016. Impact of Weight Initialization on Multilayer Perceptron Using Fuzzy Similarity Quality Measure. *Workshop on Engineering Applications*. S.l.: Springer, pp. 115-122.

COMESAÑA-CAMPOS, A., 2013. *Metodología para la generación y selección de alternativas de diseño considerando múltiples factores de un modo holístico*. S.l.: Deseño na enxeñaría.

CORREA, A., GONZALEZ, A. y LADINO, C., 2011. Genetic Algorithm Optimization for Selecting the Best Architecture of a Multi-Layer Perceptron Neural Network: A Credit Scoring Case. *SAS Global Forum 2011 Data Mining and Text Analytics*. S.l.: Citeseer,

CORREA, B.A. y GONZALEZ, A.M., 2011. Evolutionary algorithms for selecting the architecture of a MLP neural network: A credit scoring case. *Data Mining Workshops (ICDMW), 2011 IEEE 11th International Conference on*. S.l.: IEEE, pp. 725-732. ISBN 1-4673-0005-5.

DEL BRÍO, B.M. y MOLINA, A.S., 2001. *Redes neuronales y sistemas difusos Bonifacio Martín del BRio, Alfredo Sanz Molina*. S.l.: Alfaomega Ra-Ma. ISBN 970-15-0733-9.

DOMÍNGUEZ, L., ERNESTO, J., SOBERANES, A., JUÁREZ LANDÍN, C. y RUEDA PAZ, J., 2015. Sistema Detector de Intrusiones ocupando una red Neuronal Artificial. [en línea], Disponible en: http://ri.uaemex.mx/bitstream/handle/20.500.11799/49966/Sistema%20Detector%20de%20Intrusiones%20Ocupando%20una%20Red%20Neuronal%20Artificial.pdf?sequence=1#page=1&zoom=auto,-99,741.

GARCÍA, S., FERNÁNDEZ, A., LUENGO, J. y HERRERA, F., 2010. Advanced nonparametric tests for multiple comparisons in the design of experiments in computational intelligence and data mining: Experimental analysis of power. *Information Sciences*, vol. 180, no. 10, pp. 2044–2064. ISSN 0020-0255.

GHANOU, Y. y BENCHEIKH, G., 2016. Architecture Optimization and Training for the Multilayer Perceptron using Ant System. *International journal of computer science*, vol. 43, no. 1, pp. 10.

JURADO, F. y ASBUSAC, J., 2015. *Desarrollo de Videojuegos: Desarrollo de Componentes*. S.l.: Cursos en Español.

LUDERMIR, T.B., YAMAZAKI, A. y ZANCHETTIN, C., 2006. An Optimization Methodology for Neural Network Weights and Architectures. *IEEE TRANSACTIONS ON NEURAL NETWORKS*, vol. 17, no. 6.

MARSHALL, J.S. y THOMSON, M.A.E.P.J.C., 2015. *The Pandora software development kit for pattern recognition*. S.l.: s.n.

MARTÍNEZ-TRINIDAD, J.. y RUIZ-SHULCLOPER, J., 1999. Algoritmo LC Conceptual Duro. *IV Simposio Iberoamericano de Reconocimiento dePatrones*. La Habana, Cuba: s.n.,

MARTÍNEZ-TRINIDAD, J.F. y SÁNCHEZ-DÍAZ, G., 2001. LC: a conceptual clustering algorithm. *International Workshop on Machine Learning and Data Mining in Pattern Recognition*. S.l.: Springer, pp. 117–127.

MENDOZA, G., FERNANDA, J. y ZAPATA-SARZOSA, M.B., 2018. *Desarrollo de una red neuronal para la clasificación y reconocimiento del lenguaje de signos ecuatoriano.* S.l.: Universidad de las Fuerzas Armadas ESPE. Extensión Latacunga. Carrera de Ingeniería en Electrónica e Instrumentación.

MERZ, C.J. y MURPHY, P.M., 1998. UCI repository of machine learning datasets. , pp. 92697–3425.

MICHALSKI, R., 1980. Knowledge acquisition through conceptual clustering: A theoretical framework and algorithm for partitioning data into conjunctive concepts. *International Journal of Policy Analysis and Information Systems*, vol. 4, pp. 219-243.

MICHALSKI, R.S. y STEPP, R.E., 1981. *Concept-based clustering versus numerical taxonomy*. S.l.: Department of Computer Science, University of Illinois at Urbana-Champaign.

MILANÉS-LUQUE, M., 2017. *Diseño de la arquitectura en un Perceptrón Multicapa utilizando el agrupamiento conceptual*. Tesis presentada en opción al Título de Máster en Informática Aplicada. La Habana: Universidad de las Ciencias Informáticas.

MONTAÑO-MORENO, J.J., 2017. Redes neuronales artificiales aplicadas al análisis de datos. ,

MONTERO-MONTES, G., 2012. *Desarrollo de un simulador de tráfico ferroviario sobre cartografía descargada dinámicamente*. S.l.: s.n.

PÉREZ-SUÁREZ, A. y MEDINA-PAGOLA, J.E., 2014. Algoritmos para el agrupamiento conceptual de objetos. ,

PÉREZ-VALLS, J., 2013. *HERRAMIENTA MATLAB PARA LA SELECCIÓN DE ENTRADAS Y PREDICCIÓN NEURONAL DE VALORES DE BOLSA* [en línea]. Tesis. S.l.: Universidad de Sevilla. Disponible en: http://bibing.us.es/proyectos/abreproy/12166/fichero/Volumen+1+-+Memoria+descriptiva+del+proyecto%252F3+-+Perceptron+multicapa.pdf.

PONS-PORRATA, A., 2004. *DESARROLLO DE ALGORITMOS PARA LA ESTRUCTURACIÓN DINÁMICA DE INFORMACIÓN Y SU APLICACIÓN A LA DETECCIÓN DE SUCESOS.* Trabajo de Tesis en opción al grado científico de Doctor en Ciencias Técnicas. S.l.: s.n.

RABUÑAL, J.R., 2002. Metodología para el desarrollo de sistemas de extracción de concimiento en RNA. ,

REYES-GONZÁLEZ, Y., 2014. *Modelo para la adaptación de las soluciones en un Sistema Basado en Casos utilizando el agrupamiento conceptual*. S.l.: Tesis de Maestría.

REYES-GONZÁLEZ, Y., 2017. *Modelo basado en casos utilizando algoritmos conceptuales del Reconocimiento Lógico Combinatorio de Patrones*. Tesis en opción al grado científico de Doctor en Ciencias Técnicas. La Habana: Universidad de las Ciencias Informáticas.

REYES-GONZÁLEZ, Y., ARCEO, A.C., MARTÍNEZ-SÁNCHEZ, N. y HERNÁNDEZ-DOMINGUEZ, A., 2016. Agrupamiento conceptual lógico combinatorio: Una alternativa para la toma de decisiones. *INTELIGENCIA ARTIFICIAL*, vol. 19, pp. 82-96.

REYES-GONZÁLEZ, Y., RODRÍGUEZ-VALLEJO, L., MARTÍNEZ-SÁNCHEZ, N. y YERO-OSES, E.A., 2016. Métricas para la validación de los conceptos en el Reconocimiento Lógico Combinatorio de Patrones. ,

ROSENBLATT, F., 1962. Principles of neurodynamics: perceptrons and the theory of brain mechanisms (Spartan, Washington DC). ,

RUIZ SHULCLOPER, J., 2013. Acerca del surgimiento del Reconocimiento de Patrones en Cuba. , vol. 7, no. 2, pp. 169–192.

RUIZ-SHULCLOPER, J., 2009. Reconocimiento lógico combinatorio de patrones: teoría y aplicaciones (Tesis en opción al grado científico de Doctor en Ciencias). ,

RUÍZ-SHULCLOPER, J., ALBA, E. y LAZO, M., 1994. Introducción a la teoría de testores. *Serie Verde Cinvestav-IPN*, no. 50.

SANTANA, A. y NIEVES-HERNÁNDES, C., 2018. *Presentación del Curso: El entorno estadístico R (R4ULPGC)* [en línea]. Gran Canaria: Universidad de las Palmas. [Consulta: 2 junio 2018]. Disponible en: http://www.dma.ulpgc.es/profesores/personal/stat/cursoR4ULPGC/1-presentacion.html.

SARDIÑAS, A.D. y PÉREZ, R.B., 2001. Generalización del uso de prototipos para el diseño de redes neuronales tipo MLP con una capa oculta. *Revista Facultad de Ingeniería*, no. 22, pp. 126-133. ISSN 2422-2844.

SHULCLOPER, J.R., ALBA-CABRERA, E., LAZO-CORTÉS, M.S. y GRUPO DE RECONOCIMIENTO DE PTRONES CUBA-MÉXICO, 1995. Introducción al Reconocimiento de Patrones. Grupo de Reconocimiento de Patrones Cuba-México. , ISSN CINVESTAV-IPN.

TALLÓN-BALLESTEROS, A.J., 2013. Nuevos modelos de Redes Neuronales Evolutivas para Clasificación Aplicación a Unidades Producto y Unidades Sigmoide. ,

TALUKDAR, J. y MEHTA, B., 2017. Human Action Recognition System using Good Features and Multilayer Perceptron Network. *arXiv preprint arXiv:1708.06794*,

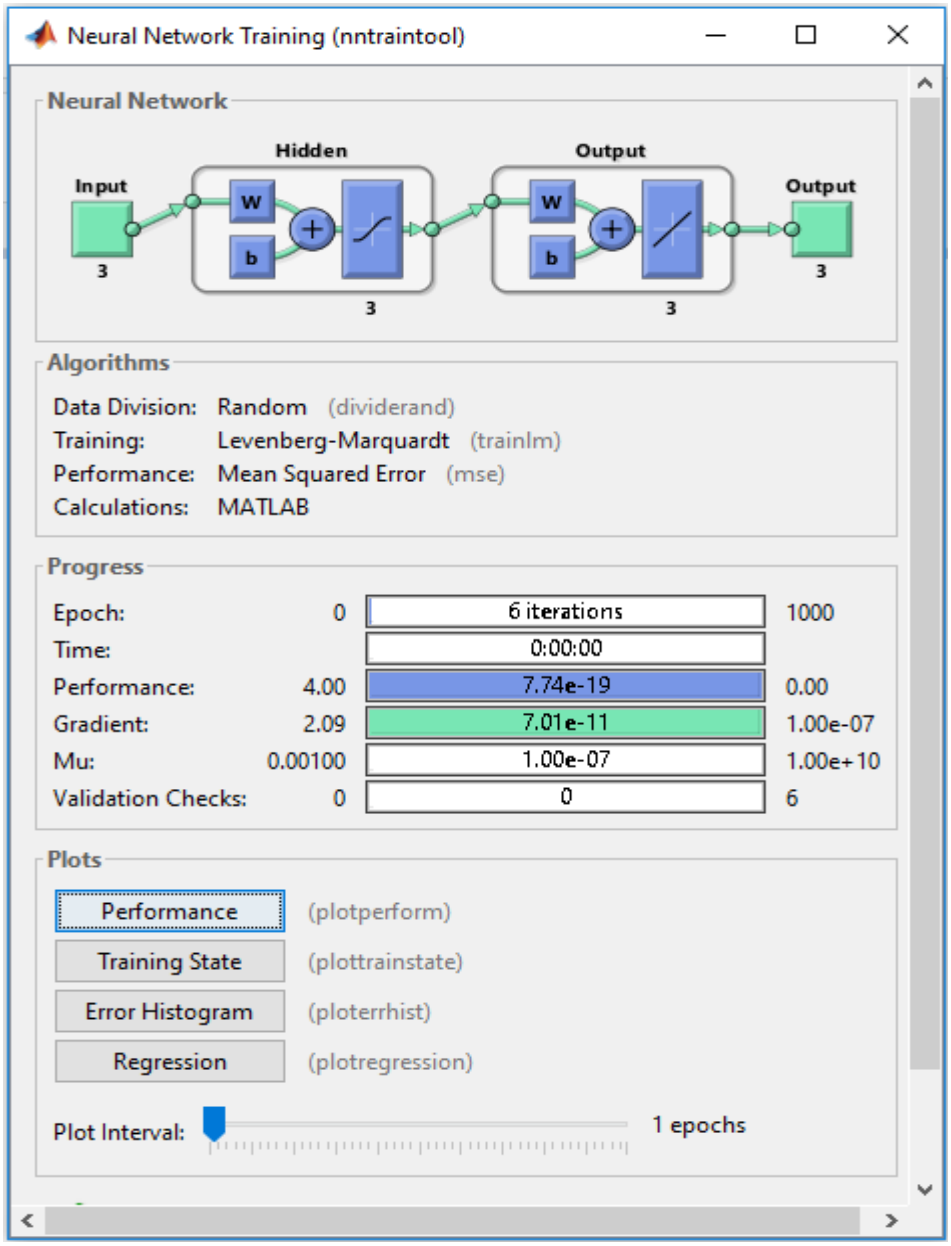
TIPAN, C. y FERNANDO, M., 2018. *Ubicación de estabilizadores de potencia y el control realimentación de estados para amortiguar oscilaciones electromecánicas de baja frecuencia utilizando redes neuronales*. S.l.: Quito, 2018.

VARGAS-BEJARANO, C.I., 2017. Evaluación del desempeño sísmico de puentes continuos. ,

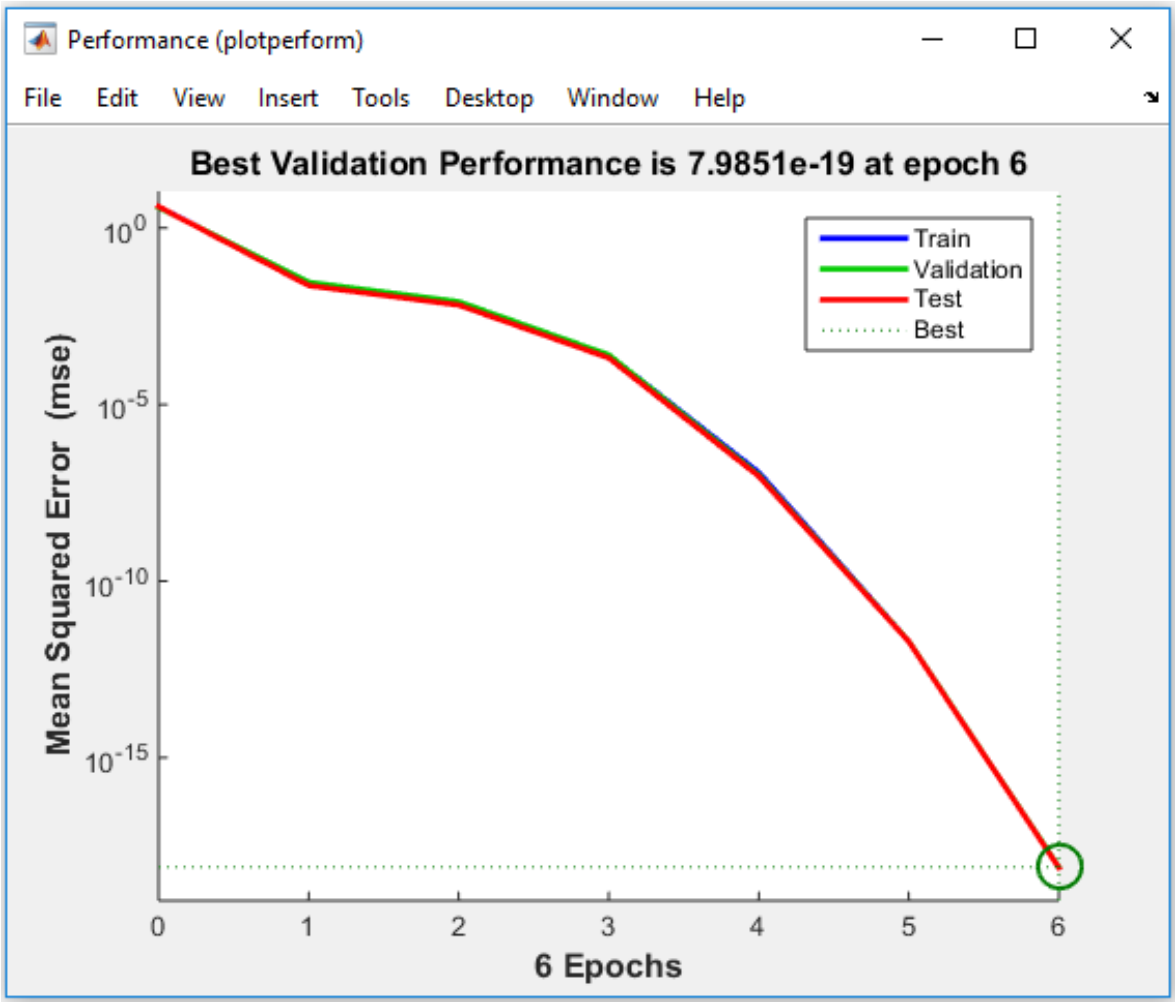
VILLAMIL-BAHAMÓN, R., 2017. *Modelo predictivo neuronal para la evaluación del riesgo crediticio*. S.l.: Universidad Nacional de Colombia.

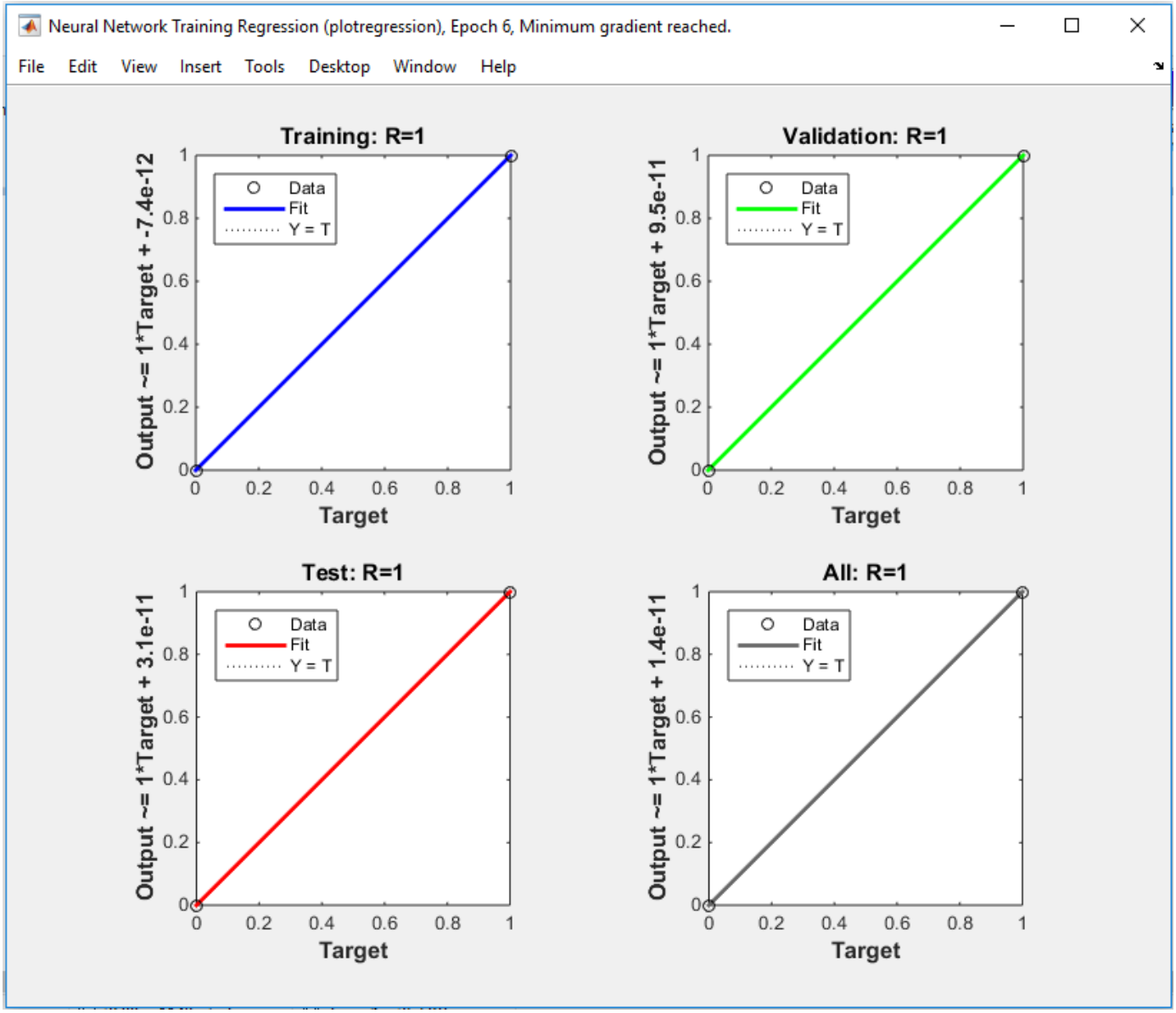
YERO-OSES, E.A., REYES-GONZÁLEZ, Y. y MARTÍNEZ-SANCHEZ, N., 2016. *CEPAR:Un sistema herramienta de apoyo al docente del reconocimiento lógico combinatorio de patrones.* S.l.: s.n.

Anexos



Anexo I: Imágenes de los comandos: nntraintool, plotperforms y plotregression en Matlab del entrenamiento de la base de datos wine (Merz y Murphy 1998).


Performance (plotperform)
File Edit View Insert Tools Desktop Window Help
Best Validation Performance is 7.9851e-19 at epoch 6
Train
Validation
Test
Best
Mean Squared Error (mse)
10^0
10^-5
10^-10
10^-15
0 1 2 3 4 5 6
6 Epochs


Neural Network Training Regression (plotregression), Epoch 6, Minimum gradient reached.
File Edit View Insert Tools Desktop Window Help
Training: R=1
Output ~= 1*Target + -7.4e-12
Target
Data
Fit
Y = T
Validation: R=1
Output ~= 1*Target + 9.5e-11
Target
Data
Fit
Y = T
Test: R=1
Output ~= 1*Target + 3.1e-11
Target
Data
Fit
Y = T
All: R=1
Output ~= 1*Target + 1.4e-11
Target
Data
Fit
Y = T

Anexo II Características de los conjuntos de datos internacionales utilizados en la experimentación.

Bases de Datos	Cantidad de Objetos	Cantidad de Rasgos	Cantidad de clases	Rasgos numéricos	Rasgos no numéricos	Ausencia de información
Wine	178	13	3	si	no	no
Annealing	798	16	38	si	si	si
Zoo	101	18	7	si	si	no
Iris	150	5	3	si	no	no
Glass	214	10	7	si	no	no
Tae	151	6	-	si	no	no
Sonar	207	60	-	si	no	no
Solarflare	324	13	-	si	no	no
Lymph	148	19	-	si	si	no
Liver-disorders	345	8	-	si	no	no