TRABAJO ESPECIAL DE GRADO

EVALUACIÓN E IMPLEMENTACIÓN DE ALGORITMOS BASADOS EN REDES NEURONALES A LOS SISTEMAS DE ALARMAS DE UNA "BASE STATION CONTROLLER"

Presentado ante la Ilustre
Universidad Central de Venezuela
Por el Br. Márquez B., José M.
Para optar al título de
Ingeniero Electricista

Caracas, 2013

TRABAJO ESPECIAL DE GRADO

EVALUACIÓN E IMPLEMENTACIÓN DE ALGORITMOS BASADOS EN REDES NEURONALES A LOS SISTEMAS DE ALARMAS DE UNA "BASE STATION CONTROLLER"

Prof. Guía: William La Cruz
Tutor Industrial: Ing. José Salas

Presentado ante la Ilustre
Universidad Central de Venezuela
Por el Br. Márquez B., José M.
Para optar al título de
Ingeniero Electricista

Caracas, 2013

UNIVERSIDAD CENTRAL DE VENEZUELA
FACULTAD DE INGENIERÍA
ESCUELA DE INGENIERÍA ELÉCTRICA
DPTO. DE ELECTRÓNICA, COMPUTACIÓN Y CONTROL

ACTA

Quienes suscriben, Miembros del Jurado designado por el Consejo de Escuela de Ingeniería Eléctrica de la Facultad de Ingeniería de la Universidad Central de Venezuela para examinar el Trabajo Especial de Grado presentado por el Bachiller José M. Márquez B., C.I. No. V-19.387.671, bajo el título "***Evaluación e Implementación de Algoritmos Basados en Redes Neuronales a los Sistemas de Alarmas de una "Base Station Controller"***", a los fines de cumplir con el requisito legal para optar al grado de Ingeniero Electricista, dejan constancia de lo siguiente:

Considerando que el Trabajo Especial de Grado presentando y defendido por el Bachiller en referencia, tiene un nivel académico y un alcance investigativo más allá de lo requerido para el grado en cuestión, acuerda por unanimidad y conforme al Artículo 6 del Reglamento de Menciones Honoríficas, recomendar el otorgamiento de Mención Honorífica a su Trabajo de Grado.

En Caracas, a los cinco días del mes de noviembre de dos mil trece.

Prof. **Ebert Brea**
Jurado

Prof. **Carlos Moreno**
Jurado

Prof. **William La Cruz**
Profesor Guía

Edificio Escuela de Ingeniería Eléctrica, piso 2, oficina 203, Ciudad Universitaria, Los Chaguáramos, Caracas 1051.
Tlf(s).: (58)2 6053164.-.Fax: (58)2 6053105

CONSTANCIA DE APROBACIÓN

Caracas, 05 de noviembre de 2013

Los abajo firmantes, miembros del Jurado designado por el Consejo de Escuela de Ingeniería Eléctrica, para evaluar el Trabajo Especial de Grado presentado por el Bachiller José M. Márquez B., titulado:

"EVALUACIÓN E IMPLEMENTACIÓN DE ALGORITMOS BASADOS EN REDES NEURONALES A LOS SISTEMAS DE ALARMAS DE UNA "BASE STATION CONTROLLER"

Consideran que el mismo cumple con los requisitos exigidos por el plan de estudios conducente al Título de Ingeniero Electricista en la mención de Electrónica, y sin que ello signifique que se hacen solidarios con las ideas expuestas por el autor, lo declaran APROBADO.

Prof. Ebert Brea
Jurado

Prof. Carlos Moreno
Jurado

Prof. William La Cruz
Prof. Guía

DEDICATORIA

Primeramente quiero dedicar este trabajo a Dios quien me ha guiado y enseñado las cosas más importantes durante el transcurso de mi vida. A mi Padre, Jesús Humberto Márquez, quien ha sido un instructor y que con su humildad ha dedicado su vida al bienestar de su familia. También a mi madre, Tibisay Blanco de Márquez, quien estuvo allí cada día para nosotros, con sus consejos y sabiduría que solo una madre puede brindar. Dedico a ellos este y cada uno de los logros que se me permita en la vida.

RECONOCIMIENTOS Y AGRADECIMIENTOS

Quiero agradecer a todas las personas que apoyaron este trabajo y contribuyeron de alguna manera en la elaboración del mismo. Primeramente, al MSc. José Salas quien estuvo siempre al pendiente de esta iniciativo y permitió con sus consejos la elaboración de la misma. También al Profesor William La Cruz, quien supo cómo guiar este trabajo y que mediante sus consejos supo encausar hasta lograr los objetivos.

A mi familia, quien me ha enseñado como salir adelante en cada uno de los desafíos que se presentan.

A mi hermano mayor, quien me ha enseñado siempre y me ha protegido; quien se preocupa por mí en cada momento. Él ha servido de ejemplo durante todos los años de mi carrera.

A la profesora Tamara Pérez, quien se preocupó por darnos una enseñanza de calidad siempre y que con su conocimiento siempre supo cómo darle impulso a la innovación y a la curiosidad.

A mis compañeros, quienes sirvieron de fuente constante de motivación y apoyo en los momentos más difíciles y con los que siempre puedo contar, en especial a Astrid, Víctor, Gabriel, Monic y Argenis, quienes me ayudaron en la parte más difícil de mi carrera.

A los demás profesores de la escuela de Ingeniería Eléctrica quienes nos han acompañado por esta senda llena de gloria.

Márquez B., José M.

EVALUACIÓN E IMPLEMENTACIÓN DE ALGORITMOS BASADOS EN REDES NEURONALES A LOS SISTEMAS DE ALARMAS DE UNA "BASE STATION CONTROLLER"

Profesor Guía: William La Cruz. Tutor Industrial: Ing. José Salas. Tesis. Caracas. U.C.V. Facultad de Ingeniería. Escuela de Ingenieria Eléctrica. Ingeniero Electricista. Opción: Electrónica. Institución: Telefónica Venezuela C.A. Trabajo de grado. 2013. 100 h. + anexos.

Palabras Claves: Redes Neuronales; Sistemas GSM; Sistemas de alarmas, Detección de patrones, Técnicas heurístcias.

Resumen. Con la finalidad de mejorar el sistema de detección de fallas de la Red GSM que tiene la compañía Telefónica Venezolana C.A., en el presente trabajo se plantea la evaluación de la aplicación de algoritmos basados en redes neuronales a su sistema de alarmas. El sistema desarrollado utiliza el conocimiento de expertos que conocen la red y han visto sus fallas y los errores que se producen en la misma. De alguna manera la integración de este conocimiento en un programa de computadora mejora el tiempo de respuesta y el alcance a la hora de diagnóstoicar una falla en alguno de los equipos, lo cual lo realiza en forma automatizada. Para la realización de esto entonces se diseñó un programa que utiliza la información suministrada por los expertos para diagnósticar los posibles errores en la red. Este programa está diseñado en base a redes neuronales y fue entrenado para diagnósticar algunos de los errores más comunes en la Base Station Controller y en las celdas. Para realizar el entrenamiento se utilizó el algoritmo de backpropagation, que busca el error en los patrones y en las salidas esperadas y lo propaga por toda la red de forma que el sistema corrige sus parámetros internos hasta llegar a la salida deseada para un conjunto de patrones presentados. Así mismo se diseñó un programa que se conecta de forma remota a las unidades a analizar con una interfaz gráfica que le muestra al usuario el resultado de los análisis; además, genera un reporte y lo envía por correo electrónico.

ÍNDICE GENERAL

LISTA DE TABLAS

LISTA DE FIGURAS

LISTA DE ALGORITMOS

LISTA DE ACRÓNIMOS

BSC: Base Station Controller (Controlador de estación Base).

RNA: Red Neuronal Artificial

MS: Móvil Suscriptor (Suscritor Móvil)

BSS: Base Station Subsystem (Subsistema de Estaciones Bases)

NSS: Network Switching Subsystem (Subsistema de Conmutación de Red)

NMS: Network Management Subsystem (Subsistema de Administración de Red)

CLS: Clock and Synchronization Unit

BCSU: Base Station Controller Signaling Unit

ET: Exchange Terminal

GSWB: Bit Oriented Group Switch

MCMU: Marker and Cellular Management Unit

OMU: Operation and Maintenance Unit

TCSM: Transcoder Submultiplexer

BTS: Base Transceiver Station

INTRODUCCIÓN

Para brindar los distintos servicios de comunicación a sus clientes, la empresa TELEFÓNICA VENEZOLANA C.A. tiene una red de equipos que integran su plataforma tecnológica, como por ejemplo la BSC (Base Station Controller) que permite realizar las funciones de control de una estación base y manipular todas las señales de control que son necesarias para la correcta comunicación de los usuarios de la misma. Estos equipos cuentan con un sistema que permite asegurar el funcionamiento correcto de los equipos y proporcionar un servicio a cabalidad. Éste es un sistema de alarmas que reporta algunos eventos de los equipos. Tales eventos le indican a los especialistas las fallas que se están produciendo en los equipos y las afectaciones a niveles de servicios y rendimiento. De alguna manera este sistema de alarmas es un sistema que requiere la supervisión de dichos reportes y alguien que constantemente esté realizando una revisión de dichos reportes de forma de hacer un diagnóstico y así determinar una solución. [1]

La idea principal de este trabajo es analizar el uso de redes neuronales para automatizar el proceso de supervisión de reportes del sistema de alarmas y dar un diagnóstico; además, obtener resultados ante situaciones no conocidas, y desarrollar una aplicación que de ser viable pueda mejorar el tiempo de respuesta en obtener una solución y aumentar el alcance de forma de analizar todos los equipos con dicha aplicación.

Las redes neuronales son modelos matemáticos/informáticos simplificados, que emulan la actividad de las redes de neuronas que conforman el cerebro y su funcionamiento consiste en el aprendizaje a partir de los datos que se le suministran. Entre sus aplicaciones más comunes están: el reconocimiento de voz, de caracteres y manuscritos; el análisis de formas y dibujos, diagnóstico, predicción del comportamiento, entre otros. [2]

Para evaluar la aplicabilidad de esta técnica se diseñó un sistema que permite monitorear el sistema de alarmas de una BSC y dar un diagnóstico. Este sistema consta de dos partes, un software de entrenamiento en el que se crean las Redes Neuronales Artificiales (RNA) con funciones de diagnóstico específicas y, la otra parte, consta de un software de diagnóstico que se conecta al sistema de las BSC cada quince minutos y utiliza las RNA ya creadas para diagnosticar el estado de las unidades y servicios de la BSC o identificar posibles afectaciones.

Para el diseño de los programas se seleccionó, entre las RNA disponibles, la topología del perceptrón multicapa por su capacidad de clasificar e identificar patrones y por tener un menor nivel de complejidad a la hora de la implementación. Para el entrenamiento se utilizó el algoritmo de backpropagation, el cual consiste principalmente en comparar patrones con resultados obtenidos e ir propagando el error de las comparaciones por toda la red mientras se ajustan los pesos en función de ese error.

Se observó que el sistema permite reducir el tiempo de respuesta ante una afectación, porque es posible identificarla rápidamente. Además, el sistema tiene la capacidad de detectar problemas no identificados y así avisar de dichas situaciones. En comparación con el sistema tradicional de monitoreo manual de la red se percibió una mejora significativa.

Finalmente la estructura del trabajo es la siguiente. En el Capítulo 1 se trata la propuesta del proyecto así como también los factores que intervienen en la misma. Luego, en el siguiente capítulo se dan los basamentos teóricos sobre los que se fundamenta el diseño de la solución. En el Capítulo 3 se describen los pasos metodológicos necesarios para la obtención efectiva de la solución del problema planteado y, finalmente en el Capítulo 4, se presenta el diseño del sistema y el proceso de evaluación de la aplicabilidad de los algoritmos basados en RNA a la red GSM.

CAPITULO I

DEFINICIÓN DEL PROBLEMA

1.1. PLANTEAMIENTO DEL PROBLEMA

La empresa TELEFONICA VENEZUELA C.A, tiene la necesidad de conocer el estado de los equipos con los que prestan sus servicios para mantenerlos en correcto funcionamiento. Al saber la importancia de tener un sistema de alarmas que permita prevenir una posible falla y actuar en conformidad con la información obtenida, se ha propuesto analizar los sistemas de alarmas de una BSC de la plataforma GSM de Movistar Venezuela y estudiar la aplicabilidad de las redes neuronales que permitan la detección predictiva y notificación de condiciones anormales de funcionamiento de los equipos, para una pronta atención y así evitar afectaciones importantes del servicio.

1.2. JUSTIFICACIÓN

En la actualidad el uso de redes neuronales para predecir el estado de un equipo de una red de telecomunicaciones es muy poco utilizado y no hay compañías que brinden este tipo de servicios porque generalmente se utilizan algoritmos basados en condiciones, por lo que el desarrollo de este proyecto permite abrir nuevas opciones a los sistemas de mantenimiento de estos equipos y minimizar la cantidad de fallas por mantenimiento, debido a que se puede detectar a tiempo la información necesaria para hacerlo de forma precisa y temprana.

Estas mejoras resultan en un mejor servicio y en un beneficio económico, dado que permite que los equipos no se detengan, sino que continuamente estén operando en el punto de trabajo más adecuado.

1.3. ANTECEDENTES

A continuación nombramos algunos trabajos que han utilizado redes neuronales. Nos centramos sólo en los que han sido desarrollados en el ámbito nacional y que tienen cierta relación con el presente trabajo.

En la Escuela de Ingeniería Eléctrica de la Universidad Central de Venezuela, el ingeniero Arcadio Fernández elaboró un trabajo especial de grado en el que desarrolló un modelo computacional para un sistema de reconocimiento de matrículas a través de una imagen proveniente de una cámara de tráfico. El algoritmo de reconocimiento de imágenes estaba basado en redes neuronales. [3]

En el año 2005 el ingeniero Argenis Moreno presentó un trabajo de grado en la Universidad de los Andes donde mostraba la aplicación de redes neuronales artificiales a la resolución de problemas de programación lineal. En el cuál observó las mejoras de implementar dichas técnicas. [4]

En el año 2001 Joanna Collantes presentó un trabajo de maestría en la Universidad de los Andes, donde aplicaba redes neuronales a sistemas predictivos estadísticos comparándolas con las metodologías de Box y Jenkins. [5]

1.4. OBJETIVOS

1.4.1. Objetivo General

Evaluar e implementar algoritmos basados en Redes Neuronales a los sistemas de alarmas de una “Base Station Controller”.

1.4.2. Objetivos Específicos

1. Estudiar e identificar el sistema de reporte de alarmas de la BSC y su sistema de gestión de operación.
2. Determinar la factibilidad de la aplicación del método predictivo en la BSC.
3. Seleccionar el tipo de red neuronal aplicable al elemento seleccionado basado en la velocidad de respuesta y la simplicidad de la implementación de dicho método tanto en la fase de diseño como en la aplicación.
4. Diseñar la red neuronal seleccionada.
5. Diseñar un software de prueba para el elemento seleccionado, haciendo uso de la plataforma y el lenguaje de programación más viable para la implementación y/o exigido por la empresa.
6. Validar la aplicabilidad del software de pruebas en la BSC mediante simulaciones y pruebas.

CAPITULO II

MARCO TEÓRICO

2.1. ARQUITECTURA DEL SISTEMA GSM

2.1.1. Qué es el Sistema GSM

En la actualidad hay muchos estándares de telecomunicaciones que permiten la interconexión de equipos y diferentes servicios, tanto de telefonía como de datos. GSM es el estándar de comunicaciones móviles europeo que ha sido aceptado en el mundo entero dado que conlleva bajo costo en infraestructura, Roaming internacional y un sistema totalmente digital. Este estándar trabaja en las bandas de 850 MHz, 900 MHz, 1800 MHz y 1900 MHz. Algunos de los servicios a los que tenemos acceso en el estándar GSM se dan en la siguiente tabla.

Tabla 1. Servicios del sistema GSM [6]

Teleservicios	Telefonía	Velocidad media de transmisión (6,5 kbit/s)
		Llamada de emergencia
		Telefax
	Mensajes cortos	Mensajes cortos (SMS), punto a punto y punto a multipunto
		Alfanuméricos, usuario-a-usuario y red-a-múltiples usuarios
		Datos Asíncronos (300 a 9600 bit/s y 1200 75 bit/s)
Servicios		Datos síncronos (300 a 9600 bit/s)
de portadora		Acceso asíncrono a un PAD (ensamblador desensamblador de paquetes)
		Acceso síncrono dedicado al servicio de conmutación de paquetes
Servicios		Desvío de llamadas como con usuario ocupado, no alcanzable
suplemntarios		Bloqueo de llamada
		CLIP (calling/connected line identity presentation)
		CLIR (CLI restriction)
		Llamada puesta en espera
		Comunicación multiparticipantes
		AOC (advice of charge)
		Información en vivo sobre tarificación

El propósito detrás de las especificaciones de GSM es definir varias interfaces abiertas. Esto permite que los operadores puedan interconectar elementos como el BSS de un proveedor con el MSC de otro proveedor, poner fin a la incompatibilidad de sistemas en el área de las comunicaciones móviles, crear una estructura de sistemas de comunicaciones a nivel europeo e incluir una amplia variedad de servicios que incluyen transmisiones de voz y servicios de manejo de mensajes entre unidades móviles o cualquier otra unidad portátil. [6]

2.1.2. Elementos que Componen el Sistema GSM

La red está compuesta principalmente por cuatro grupos de elementos como se muestra en la Figura 1, que serán explicados a continuación.

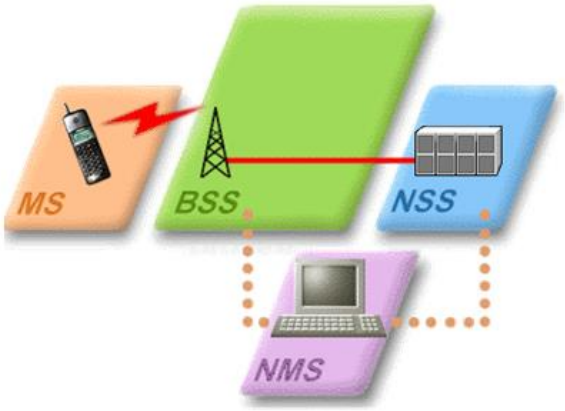

Figura 1. Grupos de elementos de la red GSM [1]

El MS (Mobil Subscriber) o suscriptor Móvil está compuesto por dos unidades, el terminal o equipo que utiliza el suscriptor y un chip donde se encuentra información del suscriptor, la unión de estos da como resultado el MS como un elemento de la arquitectura de la red GSM. [6]

Figura 2. El suscriptor móvil [1]

El BSS (Base Station Subsystem) o Subsistema de Estaciones Bases, está compuesto por tres unidades. Entre sus funciones tenemos:

- Gestionar y controlar la red de acceso
- Establecer la conexión entre el MS y el NSS
- Mantener el contacto con el MS
- Encriptamiento de la Interfaz de aire
- Procesamiento y codificación de la voz
- Señalización de las Interfaces y Aire
- Recolección de datos estadísticos [7]

El NSS (Network Switching Subsystem) o Subsistema de Conmutación de Red, está compuesto por dos elementos: el elemento central que ejecuta las funciones de interconexión y conmutación, y el elemento donde se crean los suscriptores; estos elementos a su vez cuentan con una serie de base de datos. En el caso del HLR, tenemos la base de datos donde están los suscriptores creados y los servicios que éste posee, otra donde se verifica la autentificación del suscriptor (verificación de datos de la SIM), y una última donde se verifica el registro del equipo del suscriptor (verificación del terminal o equipo). En el caso del MSC, la base de datos que este posee es donde se almacena de forma temporal los datos del suscriptor que se encuentre bajo la cobertura de cualquier estación controlado por este MSC. [7]

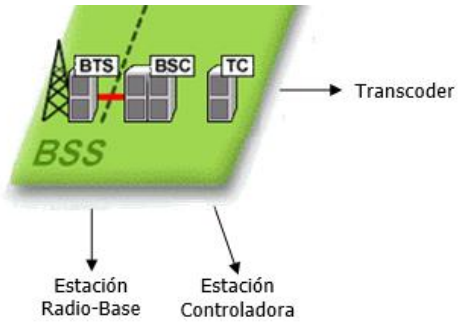

Figura 3. Subsistema de estaciones bases [1]

Figura 4. Subsistema de conmutación de la red [1]

Es necesario destacar que esta arquitectura ha sufrido cierta variación en el elemento MSC, ésta ahora se compone de dos unidades: MSS (Mobiles Service Switching Centre - Server) y MGW (Media Gateway). El MSC separa de su estructura las múltiples interconexiones, adicional a esto permite la interconexión de múltiples redes a este como el caso de 2G-2G en bandas de frecuencias distintas, 2G-3G en tecnologías de accesos al medio distintas (GSM-WCDMA), y muchos otros casos más. [7]

Entre sus funciones tenemos:

- Control de llamadas y del BSS
- Interconexión con otras redes
- Manejo de los datos y servicios del suscriptor

- Manejo de la movilidad y seguridad
- Señalización
- Recolección de datos estadísticos

El NMS (Network Management Subsystem) o Subsistema de Administración de Red, permite controlar y monitorear los recursos de la red, almacenar valores para la generación de reportes de desempeño de la red, realizar cambios masivos a nivel de parámetros de la red de acceso, planificación de red, monitoreo de alarmas y eventos, en conclusión es a través de éste que podemos ser los ojos de la red.

Entre sus funciones tenemos:

- Provee acceso al MSC y a los BSC
- Maneja mensajes de error
- Controla la carga de tráfico de los BSC y BTS
- Configurar las BTS por intermedio de los BSC
- Permite verificar los componentes del sistema

2.2. FUNCIONAMIENTO DE LAS BSC Y SU SISTEMA DE ALARMAS

2.2.1. Qué es una BSC

Como se dijo anteriormente, las BSC son parte del subsistema de estaciones base y es el elemento encargado de llevar toda la información entre las celdas y el subsistema de conmutación de red.

La BSC gestiona los recursos de radio para una o más BTS. Se ocupa de la configuración del canal de radio, el salto de frecuencia (frequency hopping) y los traspasos (handover). En sí, la BSC es la conexión entre el móvil y el MSC. La BSC también traduce el canal de 13 Kbps de voz que se utiliza a través del enlace de radio

al canal estándar de 64 Kbps. Ésta asigna y libera frecuencias e intervalos de tiempo para la MS. La BSC también se ocupa del traspaso intercelular. Controla la transmisión de energía de las MS en su área. La función de la BSC es la asignación de las franjas horarias necesarias entre la BTS y el MSC. Se trata de un dispositivo de conmutación que controla los recursos de radio. [8]

Las funciones adicionales incluyen:

- Control de saltos de frecuencia
- Realización de concentración de tráfico para reducir el número de líneas desde el MSC
- Proporcionar una interfaz para los Centro de Operaciones y Mantenimiento.
- La reasignación de frecuencias entre BTS
- El tiempo y sincronización de frecuencia.
- Administración de energía.
- Las mediciones de retardo de las señales recibidas de los MS.

2.2.2. Elementos que Componen una BSC

Las BSC tienen diferentes unidades que les permiten la normal realización de sus funciones como se muestra en la Figura 5.

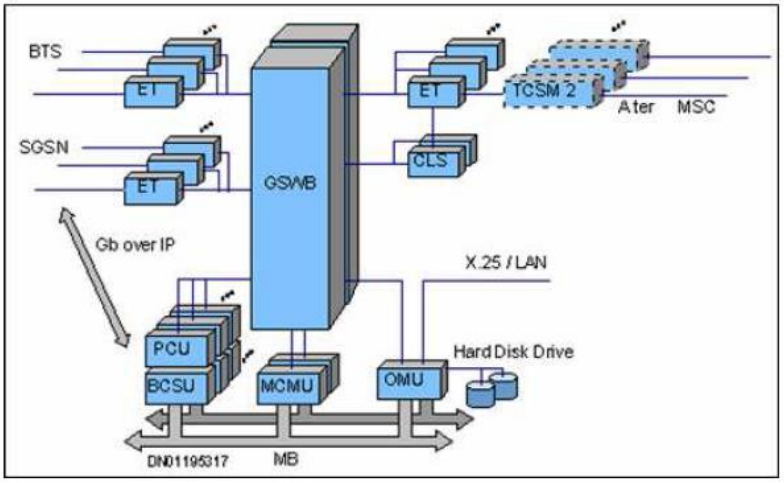

Figura 5. Unidades que componen una BSC [10]

CLS (Clock and Synchronization Unit)

Es la unidad encargada de proporcionar la señal de reloj necesaria para el funcionamiento de la BSC y generalmente se sincroniza con una fuente externa. [9]

BCSU (Base Station Controller Signaling Unit)

Controla la señalización y el tráfico tanto hacia la interfaz Aire, como para la interfaz de la MSC. En orden de prioridades sería la tercera unidad más importante de la BSC ya que se encarga de la administración de recurso de voz. Si esta unidad se encuentra sin funcionar, obviamente no se asignarán recursos de voz y las llamadas completadas en el grupo de congestión se dispararían, adicionalmente se afectará la región a la que está asignada la BSCU. [9]

ET (Exchange Terminal)

Optimiza la sincronización de las líneas, estos son los enlaces que se encargan de llevar las señales a los diferentes puntos de destino. Van directamente unidos a los enlace de radios. [9]

GSWB (Bit Oriented Group Switch)

El Group Switch es el encargado de la conexión física de todas las transacciones de la BSC y el mismo es el corazón de la BSC. En prioridad es la segunda unidad más importante y no falla de forma constante.

Su función es conmutar los canales de voz desde las BTS hacia las líneas troncales y viceversa, así como extraer o insertar la señalización necesaria. [9]

MCMU (Marker and Cellular Management Unit)

Ésta es la unidad más importante de esta central, es el cerebro de todas las transacciones que se llevan a cabo en GSWB. Si esta unidad no controla las conexiones, no se entablarán ningún tipo de conexiones permanentes y las BSCU como no encuentran enrutamiento, se comienzan a conmutar las mismas.

Su función es controlar y supervisar la unidad GSWB. Le indica cuando debe establecer, mantener o finalizar una conexión. [9]

OMU (Operation and Maintenance Unit)

Es la interfaz entre el operador de la red y el equipo. Almacena el software utilizado por las BTS, recopila datos estadísticos y alarmas.

Si esta unidad es afectada, no se tiene afectación de servicio comercial, sin embargo no se puede tener gestión sobre la central para solución ni investigación de la falla. [9]

TCSM (Transcoder Submultiplexer)

Permite comprimir los canales de 64Kbps provenientes del MSC a tazas de hasta 8Kbps. Ciertamente los mismo pueden tomarse como una unidad pasiva ya que no se poseen gestión sobre los mismos a nivel de monitoreo más que bloqueo y desbloqueo.

Si esta unidad es afectada, automáticamente bajarán las llamadas completadas ya que no hay comunicación desde la plataforma BSS a la NSS. [9]

BTS (Base Transceiver Station)

Establece la interfaz a la unidad móvil. Está bajo el control del BSC. Existe una por cada célula y junto a ésta es la interfaz entre la unidad móvil y el MSC. [9]

2.2.3. Conexión con las BSC vía Telnet

Telnet viene de Telecommunication Network. Este protocolo sigue el modelo cliente/servidor y pertenece a la familia de protocolos de Internet. Un servidor permite a los usuarios acceder a un ordenador huésped para realizar tareas como si estuviera trabajando directamente en ese ordenador. Este protocolo es de la capa de aplicación y va sobre TCP/IP. El puerto TCP que utiliza el protocolo Telnet es el 23. Solo sirve para acceder remotamente a un terminal, al acceder el protocolo requiere de una parte de autenticación en la que se le pide al cliente un usuario y una contraseña. Se usa este protocolo mayormente en sistemas UNIX-LINUX y en equipos de comunicaciones como routers. [10]

El sistema de las BSC se comunica a través del protocolo Telnet donde la BSC sirve como servidor y se utilizan programas de gestión como clientes desde donde se monitorea, diagnóstica y actúa sobre estos equipos. El sistema servidor de la BSC responde a ciertos comandos para realizar las acciones antes mencionadas, algunos de los cuáles sirven para reiniciar las unidades o para ver el estado de las mismas y su funcionamiento. Además estos equipos poseen un sistema de alarmas que indica el estado de todas las unidades que el incluye y los eventos que le ocurren a los mismos. Para acceder a este sistema de alarmas se usan algunos comandos y estos tienen una respuesta dependiendo de la unidad de la cual se requiere el reporte o registro de alarmas.

2.2.4. Sistema de Alarmas de las BSC

Como se mencionó anteriormente, para acceder al sistema de alarmas de la unidad es necesario introducir en el software de gestión los comandos para dicha función. Para obtener un reporte de las alarmas que están activas en las unidades internas de las BSC, se procede a usar el comando ZAHO; si se requiere un reporte de las alarmas activas en las unidades externas a la BSC (BTS, TRX, BCF, etc.) se utiliza el comando ZEOL. Si se requiere el historial de todas las alarmas que se han activado en el día en las unidades internas de la BSC, se utiliza el comando ZAHP.

Para cualquiera de los casos anteriores el sistema procederá a enviar un reporte con la información solicitada donde cada alarma aparecerá con el formato que aparece en la Figura 6, y que a continuación se describe de forma detallada:

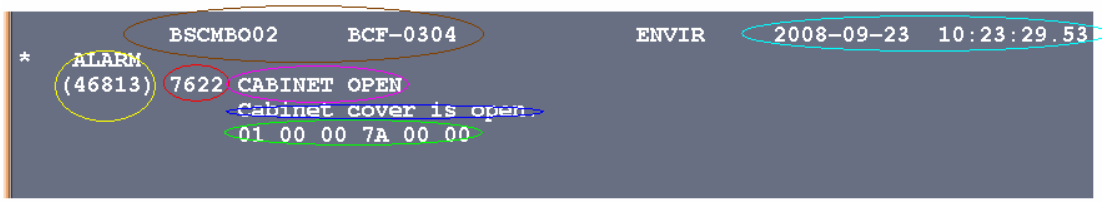

Figura 6. Formato de las alarmas [6]

Encerrado con la elipse de color marrón, se indica el nombre de la central donde se encontró la alarma y el número del subelemento. La elipse de color amarillo encierra el indicador de la criticidad de la alarma que puede estar delimitada con 3, 2 o una estrella, acompañado de un número único que se le da a la alarma propio del sistema.

La elipse de color rojo encierra el número de la alarma fija que se le da al sistema. Este número es el que indica el tipo de la alarma que se está generando. Existen aproximadamente 654 tipos diferentes de alarmas para analizar. Con la elipse de color violeta se encierra el nombre de la alarma.

La elipse azul encierra una breve descripción de la alarma y la elipse de color magenta encierra la fecha y hora del evento. Por último la elipse de color verde encierra la información más importante de la alarma, esta es la información suplementaria y da en detalle la ubicación y causas probables de la falla. [11]

De esta manera las BSC informan de manera continua la situación en la que se encuentra. Los especialistas revisan el registro de alarmas en busca de posibles fallas para realizar los procedimientos adecuados para corregirlas o prevenirlas dependiendo del análisis que ellos hagan en función de la experiencia, la documentación y su propio conocimiento.

El uso eficiente de este registro permite mantener en buen estado de funcionamiento los equipos.

2.3. LAS REDES NEURONALES

Para entender las redes neuronales es importante entender el funcionamiento del cerebro humano.

> "El cerebro humano es el sistema de cálculo más complejo que conoce el hombre. El ordenador y el hombre realizan bien diferentes clases de tareas; así la operación de reconocer el rostro de una persona resulta una tarea relativamente sencilla para el hombre y difícil para el ordenador, mientras que la contabilidad de una empresa es tarea costosa para un experto contable y una sencilla rutina para un ordenador básico.
>
> La capacidad del cerebro humano de pensar, recordar y resolver problemas ha inspirado a muchos científicos intentar o procurar modelar en el ordenador el funcionamiento del cerebro humano. Las redes neuronales artificiales (RNA) proveen un método

general y práctico para aprender funciones de valores reales, vectores y también discretas, a partir de ejemplos de entrenamiento.” [12]

En cierto tipo de problemas, que implican interpretar datos provenientes de entradas del mundo real, las RNA se encuentran entre los métodos que permiten el aprendizaje. Existen diversos ejemplos de la aplicación estas, como el reconocimiento de caracteres escritos a mano, reconocimiento de palabras habladas y reconocimiento de rostros.

Las RNA han sido inspiradas en parte por los sistemas neuronales biológicos. En forma análoga a este tipo de sistemas, las RNA se construyen a partir de un conjunto de unidades simples densamente interconectadas. Intentan además capturar el tipo de procesamiento altamente paralelo que opera sobre representaciones distribuidas que ha sido observado en las redes neuronales biológicas.

Un ejemplo del aprendizaje de RNA, es el sistema ALVINN, que utilizó una RNA entrenada para conducir un vehículo autónomo a velocidades normales en una autopista pública (hasta 70 millas por hora, en tramos de 90 millas con otros vehículos presentes). [13]

Las RNA presentan características propias del cerebro. Por ejemplo, las RNA aprenden de la experiencia, generalizan de ejemplos previos a ejemplos nuevos y abstraen las características principales de una serie de datos. Pueden aprender al adquirir el conocimiento de una cosa por medio del estudio, ejercicio o experiencia. Las RNA pueden cambiar su comportamiento en función del entorno. Se les muestra un conjunto de entradas y ellas mismas se ajustan para producir unas salidas consistentes. Pueden generalizar al extender o ampliar una cosa. Pueden abstraer al aislar mentalmente o considerar por separado las cualidades de un objeto. Las RNA generalizan automáticamente debido a su propia estructura y naturaleza. Estas redes pueden ofrecer, dentro de un margen, respuestas correctas a entradas que presentan pequeñas variaciones debido a los efectos de ruido o distorsión. Algunas RNA son

capaces de abstraer la esencia de un conjunto de entradas que aparentemente no presentan aspectos comunes o relativos. [12]

2.3.1. Estructura Básica de las RNA

El cerebro humano está compuesto por una gran cantidad de elementos básicos denominados neuronas (ver Figura 7). La neurona es la unidad fundamental del sistema nervioso y en particular del cerebro. Cada neurona es una simple unidad procesadora que recibe y combina señales desde y hacia otras neuronas. Si la combinación de entradas es suficientemente fuerte la salida de la neurona se activa. Básicamente las neuronas están formadas por:

- Un cuerpo central o Núcleo
- Un mecanismo de conexión con otras neuronas (sinapsis).
- Axón y dendritas [2]

El cerebro consiste en uno o varios billones de neuronas densamente interconectadas. Los estímulos recibidos en el cerebro son transmitidos entre las neuronas mediante las conexiones sinápticas. Cuando una neurona es estimulada libera una pequeña cantidad de un componente químico (neurotransmisor). Éste viaja a través del axón hasta llegar a las dendritas de otras neuronas en las cuales el proceso se repite. Este proceso sirve para incrementar o disminuir la relación entre las neuronas involucradas en el. Así, ante un determinado estímulo ciertas neuronas se activan y otras se inhiben.

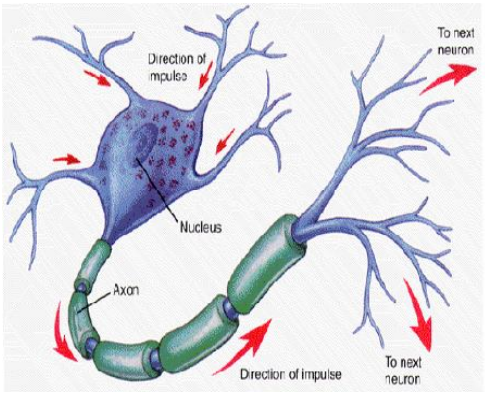

Figura 7. La neurona [2]

Mediante un proceso de aprendizaje se logran establecer los niveles correctos de activación-inhibición de las neuronas. Cuando este proceso se completa entonces ante determinados estímulos sabemos cómo responder y aprendemos. El conocimiento adquirido está entonces en los niveles de relación entre las neuronas logrados durante el proceso de aprendizaje.

En las RNA se tiene un elemento procesador llamado neurona que al igual que en el caso del cerebro, está conectada a un conjunto de entradas y las combina linealmente mediante una suma. La suma de las entradas es modificada por una función de activación y el valor de la salida de esta función se pasa directamente a la salida del elemento procesador.

La salida de la neurona se puede conectar a las entradas de otras neuronas artificiales mediante conexiones ponderadas correspondientes a la eficacia de la sinapsis de las conexiones (ver Figura 8). [14]

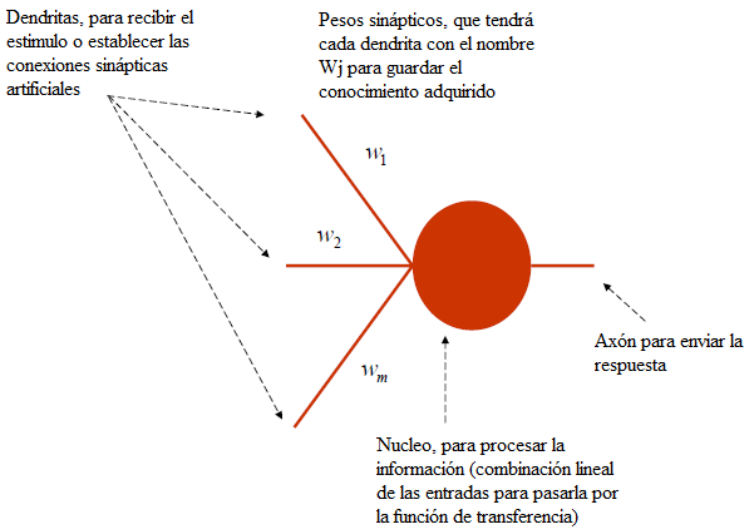

Figura 8. Modelo de una neurona artificial [2]

Los estímulos se consideran vectores de la forma:

$$X = (x_1, x_2, \dots, x_n), \tag{1}$$

donde cada entrada x_i del vector X corresponde a un estímulo variable en particular de la cual se tiene cierta cantidad de observaciones o valores. Los valores $w_1, w_2, \dots, w_n$, denominados *pesos sinápticos*, son factores de peso asociados con cada nodo. Cada entrada se multiplica por el peso asociado de la conexión neurona $X^T W$, donde $W = (w_1, w_2, \dots, w_n)$, es decir, cuando se recibe el estímulo, cada entrada de éste es multiplicada por el correspondiente peso sináptico de la dendrita que recibe dicho valor, y luego estos resultados se suman.

El estímulo es procesado en el núcleo mediante la operación:

$$y = \varphi\left(\sum_{j=1}^{m} w_j x_j + b\right) = \varphi(X^T W + b), \tag{2}$$

donde y es la salida, φ es la función de activación de la neurona y b es el parámetro de sesgo, que representa el umbral de activación de la neurona. Al valor resultante de

la sumatoria de las entradas multiplicadas por sus respectivos pesos, se *denomina activación de la neurona*. Una representación más detallada de la neurona se ve en la Figura 9.

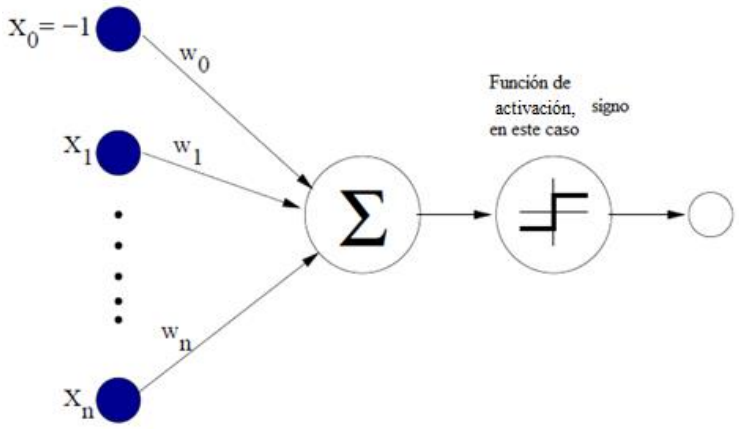

Figura 9. Representación gráfica de la neurona [14]

Entre las funciones de activación más comunes tenemos [15]:

- *Función Escalón*: Coloca la salida en 1 si la activación de la neurona es superior al valor de umbral. Se utiliza comúnmente para redes con resultados netamente discretos. Su expresión matemática es:

$$\varphi(u) = \begin{cases} 0, & si\ u < 0, \\ 1, & si\ u \geq 0. \end{cases}$$

- *Función Lineal*: Se llama función identidad porque la salida es directamente la activación de la neurona y es equivalente a no aplicar una función a su salida. Su expresión matemática es:

$$\varphi(u) = \alpha u.$$

- *Función Mixta*: Con esta función, si la activación es menor o igual que un límite inferior la salida es 0, si la activación es igual o mayor a un límite superior la salida es igual a 1; si la activación está entre los límites

la salida será una función lineal de la activación. Su expresión matemática es:

$$\varphi(u) = \begin{cases} 0, & \text{si } u \leq l, \\ \alpha u, & \text{si } l < u < s, \\ 1, & \text{si } u \geq s. \end{cases}$$

- *Función Sigmoidea*: Produce funciones continuas y proporcionales al nivel de activación de la neurona dentro del rango [0,1]. Cuando el nivel de activación supere el nivel de umbral máximo la salida seguirá siendo 1 y cuando el nivel de activación sea inferior al umbral mínimo la salida seguirá siendo 0. Su expresión matemática es:

$$\varphi(u) = \frac{1}{1 + e^{-\alpha u}}$$

- *Función tangente hiperbólica*: Es similar a la función sigmoidea pero el valor máximo estará en 1 y el mínimo estará en -1. Su expresión matemática es:

$$\varphi(u) = \frac{e^{\alpha u} - e^{-\alpha u}}{e^{\alpha u} + e^{-\alpha u}}$$

También dependiendo de la ubicación de una capa de neuronas en la red podríamos clasificar las capas en tres tipos:

- *De entrada*: Estas capas reciben la información desde el exterior, o se podría decir que esta capa actúa como un órgano sensor.
- *De salida*: Éstas envían la información hacia el exterior, o podríamos decir que actúan como un órgano ejecutor.
- *Ocultas*: Son capas cuya función es procesar la información y comunicar con otras capas. Estas no tienen conexión directa con el entorno.

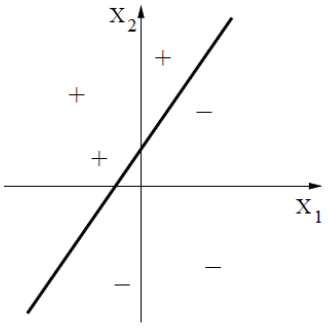

Figura 10. Hiperplano generado por una neurona [14]

Al realizar las diferentes combinaciones lineales la neurona genera una especie de hiperplano. Podemos visualizar a la neurona como representando al hiperplano $X^T W = 0$, donde la neurona retorna 1 para elementos sobre un lado del hiperplano y -1 para las que se ubican del otro como se ilustra en la Figura 10.

Algunos conjuntos de ejemplos positivos y negativos no pueden ser separados por ningún hiperplano. Aquellos que sí pueden ser separados son llamados conjuntos linealmente separables. Mediante el arreglo de las unidades procesadoras (neuronas) se pueden generar múltiples hiperplanos y de esa manera clasificar o trabajar con conjuntos no necesariamente linealmente separables. [2]

La capacidad de cálculo y potencia de la computación neuronal proviene de las múltiples conexiones de las neuronas artificiales que constituyen las RNA. La red más simple es un grupo de neuronas ordenadas en una capa.

Normalmente las redes más complejas y más grandes ofrecen mejores prestaciones en el cálculo computacional que las redes simples. Las configuraciones de las redes construidas presentan aspectos muy diferentes pero tienen un aspecto común, el ordenamiento de las neuronas en capas o niveles imitando la estructura de capas que presenta el cerebro en algunas partes.

Las redes multicapa se forman con un grupo de capas simples en cascada. La salida de una capa es la entrada de la siguiente capa. Se ha demostrado que las redes multicapa presentan cualidades y aspectos por encima de las redes de una capa simple.

Conviene destacar que la mejora de las redes multicapa reside en la función de activación no lineal entre capas, pudiéndose llegar al caso de diseñar una red de una capa simple equivalente a una red multicapa si no se utiliza la función no lineal de activación entre capas.

2.3.2. Aprendizaje de una RNA

Una de las principales características de las RNA es su capacidad de aprendizaje. El entrenamiento de las RNA es similar al desarrollo intelectual de los seres humanos a pesar de que el aprendizaje de las RNA es limitado.

El objetivo del entrenamiento de una RNA es conseguir que una aplicación determinada, para un conjunto de entradas produzca el conjunto de salidas deseadas o mínimamente consistentes. El proceso de entrenamiento consiste en la aplicación secuencial de diferentes conjuntos o vectores de entrada para que se ajusten los pesos de las interconexiones según un procedimiento predeterminado. Durante la sesión de entrenamiento los pesos convergen gradualmente hacia los valores que hacen que cada entrada produzca el vector de salida deseado. [12]

Los algoritmos de entrenamiento o los procedimientos de ajuste de los valores de las conexiones de las RNA se pueden clasificar en supervisados y no supervisados.

Entrenamiento Supervisado

Estos algoritmos requieren el emparejamiento de cada vector de entrada con su correspondiente vector de salida. El entrenamiento consiste en presentar un vector

de entrada a la red, calcular la salida de la red, compararla con la salida deseada, y el error o diferencia resultante se utiliza para realimentar la red y cambiar los pesos de acuerdo con un algoritmo que tiende a minimizar el error. Las parejas de vectores del conjunto de entrenamiento se aplican secuencialmente y de forma cíclica. Se calcula el error y el ajuste de los pesos por cada pareja hasta que el error para el conjunto de entrenamiento entero sea un valor pequeño y aceptable. [12]

Entrenamiento No Supervisado

Los sistemas neuronales con entrenamiento supervisado han tenido éxito en muchas aplicaciones. El sistema en sí mismo contiene un mecanismo que compara las salidas deseadas con las salidas reales. Sin embargo los sistemas no supervisados son modelos de aprendizaje más lógicos en los sistemas biológicos.

Estos sistemas de aprendizaje no supervisado no requieren de un vector de salidas deseadas y por tanto no se realizan comparaciones entre las salidas reales y salidas esperadas.

El conjunto de vectores de entrenamiento consiste únicamente en vectores de entrada. El algoritmo de entrenamiento modifica los pesos de la red de forma que produzca vectores de salida consistentes. El proceso de entrenamiento extrae las propiedades estadísticas del conjunto de vectores de entrenamiento y agrupa en clases los vectores similares. Existe una gran variedad de algoritmos de entrenamiento hoy en día. La gran mayoría de ellos han surgido de la evolución del modelo de aprendizaje no supervisado que propuso Hebb en 1949. El modelo propuesto por Hebb se caracteriza por incrementar el valor del peso de la conexión si las dos neuronas unidas son activadas o disparadas. [12]

2.3.3. Tipos de RNA

Dependiendo de la manera en que estén organizadas las neuronas dentro de una red, pueden variar sus capacidades y su función. Primeramente se pueden clasificar dependiendo de la manera en que se conectan:

- *Feedforward o hacia* adelante: Usualmente, las capas están ordenadas por el orden en que reciben la señal desde la entrada hasta la salida y están unidas en ese orden. Este tipo de redes contienen solo conexiones entre capas hacía adelante. Esto implica que una capa no puede tener conexiones a una que reciba la señal antes que ella.
- *Feedback o hacía atrás (Realimentadas):* Así mismo existen otras redes en las que sus capas, aparte del orden normal, también están unidas desde la salida hasta la entrada en el orden inverso en que viajan las señales de información

La selección de una red se realiza en función de las características del problema a resolver. La mayoría de éstos se pueden clasificar en aplicaciones de Predicción, Clasificación, Asociación, Conceptualización, Filtrado, Optimización, entre otros. [4]

Algunas de las topologías más conocidas de las redes neuronales son las siguientes:

Perceptrón

“La arquitectura del Perceptrón aprende a clasificar modelos mediante un aprendizaje supervisado. Los modelos que clasifica suelen ser generalmente vectores con valores binarios (0,1) y las categorías de la clasificación se expresan mediante vectores binarios.

El Perceptrón presenta dos capas de neuronas y sólo una de ellas presenta la capacidad de adaptar o modificar los pesos de las conexiones. La arquitectura del Perceptrón admite capas adicionales pero éstas no disponen la capacidad de modificar sus propias conexiones" [12] (ver Figura 11).

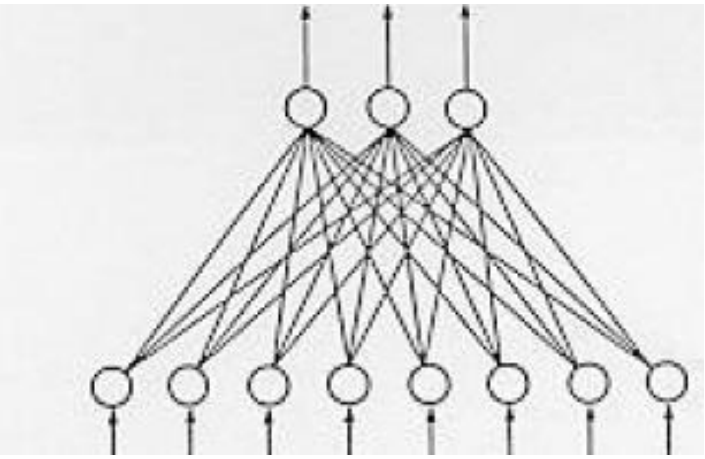

Figura 11. Perceptrón de 2 capas [12]

El mayor inconveniente del Perceptrón, a pesar del éxito que ha tenido en muchas aplicaciones de clasificación de patrones, es la imposibilidad de adaptar los pesos de todas las capas. En los años 50 en los que se realizó el Perceptrón, los investigadores no fueron capaces de diseñar un algoritmo que propagara las correcciones de los pesos a través de redes multicapa. [14]

La principal limitación funcional del Perceptrón es que una unidad de salida sólo puede clasificar patrones linealmente separables.

Adaline – Madaline

Esta estructura está dividida en dos partes. En las neuronas Adaline la unidad procesadora implementa una función umbral. Las conexiones de cada una de las entradas tienen asociadas un valor de ponderación llamado también peso. El mecanismo de ajuste de los pesos consiste en utilizar la diferencia entre el valor de la salida y el valor esperado. La unidad procesadora actúa como un sumador y después

realiza la función umbral. La salida de la unidad Adaline es ±1 a diferencia de la arquitectura del Perceptrón que sólo permite los valores 0 y 1. El entrenamiento se realiza presentando repetidamente una serie de parejas de entradas y salidas.

El objetivo de la Adaline durante el proceso de la adaptación es producir la salida deseada como propia suya.

El sistema Madaline tiene una capa de unidades Adaline que están conectadas a una simple unidad Madaline. Las conexiones entre la capa de entrada y la capa de las unidades Adaline tienen asociadas un peso ajustable por cada una de ellas. Sin embargo, las conexiones entre la capa de Adaline y la unidad Madaline no tienen asociado ningún peso. Cada unidad Adaline transmite su salida (-1 o +1) a la unidad Madaline. La Madaline emplea una regla de mayorías para obtener su salida: si la mitad o más de las unidades Adaline presentan un valor de salida +1, entonces la salida de la Madaline es +1. En caso contrario el valor de salida de la red Madaline es -1. El entrenamiento de los sistemas Madaline es similar al entrenamiento de las Adaline. El conjunto de entrenamiento es un conjunto de patrones de entrada emparejados con las salidas deseadas. Una vez que se presenta el patrón a la entrada, el sistema Madaline calcula su salida y a continuación se compara con la salida deseada. Los pesos son modificados después de que cada patrón sea presentado a la entrada del sistema. [14]

Red Self Organization Map

Esta RNA presenta una topología constituida por dos capas en la cuál la primera capa de la red es la capa de entrada, y la segunda capa, llamada capa competitiva está organizada en una rejilla de dos dimensiones. Las dos capas están totalmente interconectadas. Cada una de las conexiones tiene asociado un peso que será modificado a lo largo de la sesión de entrenamiento. La idea del entrenamiento es localizar a la neurona cuyos pesos se aproximen mejor al patrón de entrada.

La localización de la neurona cuyos pesos se aproximan a la entrada se debe a que el aprendizaje de la capa sigue el modelo de aprendizaje competitivo. El ajuste de los pesos de la neurona ganadora se realiza para que se aproxime más a los datos de la entrada; y por otra parte el ajuste de los pesos de las neuronas vecinas contribuye a mantener el orden del propio espacio de entrada. [12]

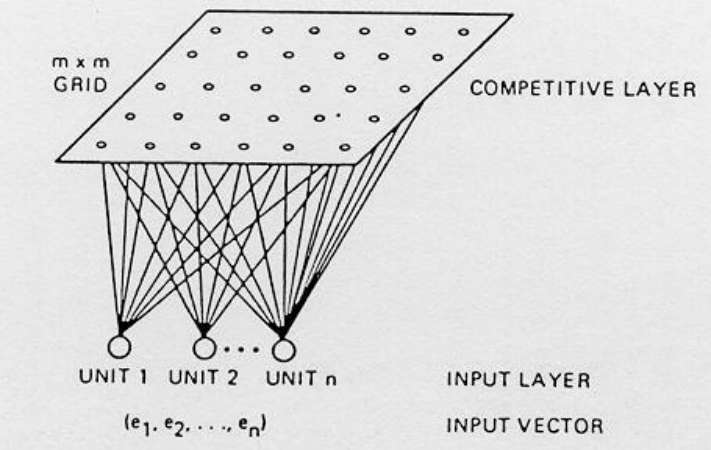

Figura 12. Estructura de una red Self Organization Map [12]

Red Counterpropagation

Se utilizan dos tipos diferentes de capas: la capa oculta es una capa de Kohonen con neuronas competitivas y aprendizaje no supervisado. Y la capa de salida que está totalmente conectada a la capa oculta no es competitiva.

"Una vez entrenada la red Counterpropagation funciona de la siguiente forma: ante un patrón presentado en la entrada, las unidades o neuronas de la capa oculta o de Kohonen compiten por responder a dicha entrada. Una única neurona será la ganadora, presentado el nivel de activación, mientras las demás permanecerán inactivas. La neurona ganadora representa la categoría a la que pertenece la entrada. Esta neurona activa un patrón en la capa de salida convirtiéndose en la salida de la red. Es manifiesto la importancia de los pesos asociados de dicha neurona a la capa de salida, ya que tienen una influencia total en el valor final de las neuronas de la salida." [12]

Red Hopfield

La red Hopfield tiene una única capa de unidades procesadoras. Cada una de las unidades procesadoras tiene un valor o nivel de activación, también llamado estado, que es binario. Se considera que la red Hopfield tiene un estado en cada momento; este estado se define por el vector de unos y ceros constituido por los estados de todas las unidades procesadoras. Las unidades procesadoras de la red Hopfield están completamente interconectadas, cada unidad está conectada con todas las demás unidades.

Esta topología convierte a la red Hopfield en una red recursiva ya que la salida de cada unidad está realimentada con las entradas de las demás unidades. La memoria humana funciona de una manera asociativa. A partir de una porción de información es capaz de obtener la información completa. Por ejemplo, escuchando los primeros acordes de una canción el cerebro es capaz de reproducir toda una experiencia completa, incluyendo escenas, ruidos y olores. Una red recursiva constituye una memoria asociativa.

Al igual que el humano, si se le presenta una porción de datos es capaz de recuperar todos los datos. Para realizar una memoria asociativa mediante una red recursiva (Hopfield propuso originalmente esta aplicación para su red binaria), es necesario elegir los pesos de forma que produzcan un mínimo de energía. Cada vector de estado correspondiente a un mínimo de energía se llama "memoria".

La red parte de un estado inicial y el propio procedimiento de actualización mueve el estado de la red hasta llegar a un estado de energía mínimo. Este mínimo se supone que corresponde a una "memoria" de la red. Entonces, la red converge en una memoria almacenada que es la más similar o la más accesible al estado inicial. En aplicaciones de memoria asociativa, se elige a priori los patrones que van a ser almacenados como memorias.

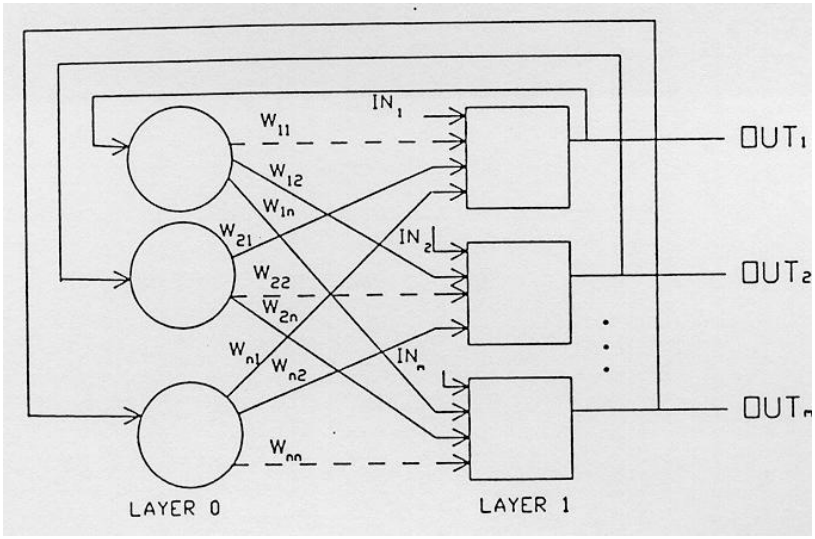

Figura 13. Arquitectura de una Red Hopfield. [12]

El número de unidades procesadoras es igual al número de elementos del vector que representa el patrón que va a ser almacenado. Los pesos se fijan en función de los patrones elegidos. [14]

Red Adaptive Resonance Theory (ART)

Esta red presenta algunas características basadas en las neuronas biológicas y en particular resuelve satisfactoriamente el dilema estabilidad-plasticidad característico del cerebro humano. ¿Cómo el cerebro se muestra flexible para almacenar nuevas memorias que le llegan, y por otra parte es capaz de retener las memorias ya almacenadas sin borrarlas?

El núcleo de la red ART consiste en dos capas interconectadas y en una serie de bloques que realizan funciones de control requeridas en las fases de entrenamiento y clasificación. Esta red es un clasificador de vectores; acepta vectores de entrada y los clasifica en una de las categorías posibles en función del patrón almacenado al que más se aproxime. La capa de Reconocimiento es la responsable de indicar la categoría. En el caso de que el vector de entrada no se aproxime suficientemente a

ningún patrón almacenado se crea una nueva categoría almacenando un patrón idéntico a la entrada.

Una vez encontrado un patrón que se parezca al vector de entrada dentro de una tolerancia especificada (parámetro de vigilancia), se ajusta o se entrena dicho patrón para hacerlo más parecido todavía al vector de entrada. [14]

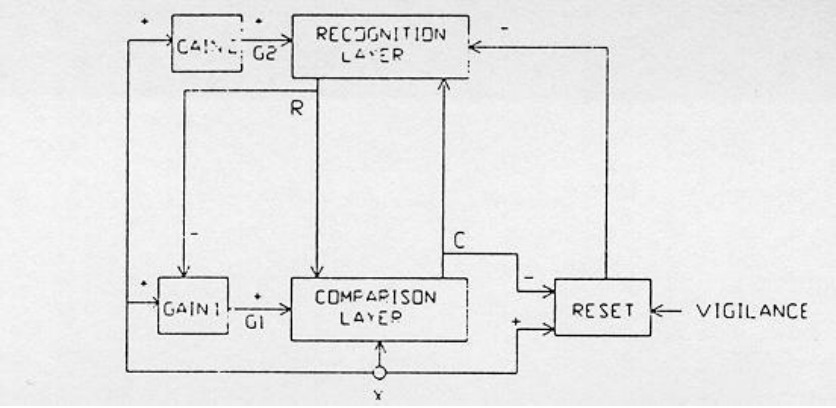

Figura 14. Estructura de la Red ART [12]

2.3.4. Perceptrón Multicapa

El perceptrón multicapa es una RNA que surge por las limitaciones que posee el perceptrón de una sola capa de no poder clasificar conjuntos que no son linealmente independientes y/o separables. Se puede observar en la obra Perceptrons de 1969 por Minsky y Papert las críticas que comentan este punto cuando estas redes no podían aprender la función OR exclusivo.

El perceptrón multicapa es una variación del simple al que se le agregan varias capas a la red y a la salida de dichas capas se colocan funciones de activación no lineales de forma de agregarle no linealidad a la clasificación que realiza el sistema permitiendo resolver el problema antes mencionado. La arquitectura de este perceptrón consta de una capa de entrada y una de salida y una o más capas ocultas. Esta red es de la forma Feedforward, donde la capa de entrada se une con la primera

capa oculta, luego se van uniendo las ocultas hasta la última de ellas y por último se tiene la capa de salida como se observa en la Figura 15.

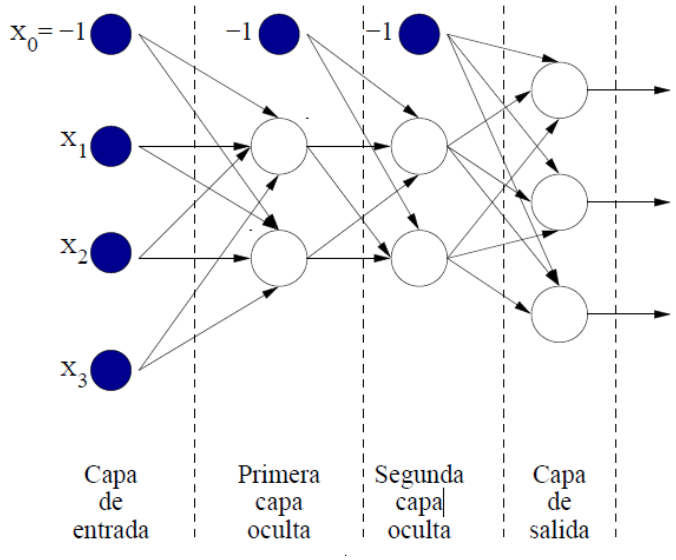

Figura 15. Perceptrón multicapa [14]

De forma que el perceptrón multicapa define una relación entre las variables de entrada y las variables de salida de la red, y esta relación se obtiene propagando a través de toda la red los valores de las variables de entrada de la misma. Para esto cada neurona procesa la información proveniente de cada una de sus entradas y las procesa generando una activación que se propaga a través de las conexiones de la misma.

Supongamos que tenemos una red de C capas donde $c = 1$ representa la capa de entrada y $c = C$ representa la capa de salida. Adicionalmente tenemos n_c neuronas para la capa c. También tenemos e_{ci} entradas para la neurona i en la capa c, con las entradas x_{cij} y adicionalmente tenemos los pesos w_{cij} en la neurona i de la capa c. Por lo que para una neurona i de la capa c tendremos que la activación a_{ci} vendrá dada por la ecuación

$$a_{ci} = \left(\sum_{j=1}^{e_{ci}} x_{cij} w_{cij}\right) + b,$$

donde b es el umbral de activación y se puede entender como una entrada adicional a la neurona y puede ser procesada de la misma manera. [18]

Para procesar la activación se le aplica alguna de las funciones vistas anteriormente (generalmente la sigmoidea para agregar la no linealidad al sistema) que denotaremos como F y obtenemos el resultado de la salida a través de la ecuación:

$$S_{ci} = F(a_{ci}) = F\left(\sum_{j=1}^{e_{ci}} x_{cij} w_{cij}) + b\right). \quad (3)$$

2.3.5. Algoritmo de Backpropagation para el Entrenamiento del Perceptrón Multicapa

Otro de los principales inconvenientes del perceptrón simple era la imposibilidad de modificar los pesos en alguna otra capa que no fuera la capa de salida por lo que el algoritmo de entrenamiento solo actualizaba un grupo de pesos reduciendo toda la red a una sola capa. Este se pudo corregir posteriormente permitiendo evolucionar al perceptrón y así nació el algoritmo de *backpropagation* o de *propagación hacía atrás* para propagar el error cometido a lo largo de toda la red.

El aprendizaje entonces es del tipo supervisado en el que se le presenta un conjunto de patrones a la red para ser analizados y se tiene la salida deseada para cada uno de los patrones y se usa el error cometido para actualizar los pesos. La idea general es minimizar a través del algoritmo alguna función que represente el estado de error de la red, por lo que se usa el error cuadrático como función a minimizar, la cual está dada por:

$$E = \frac{1}{N}\sum_{n=1}^{N} e(n), \tag{4}$$

donde N es el número de patrones presentados a la red y $e(n)$ es el error cometido por la red en el patrón n, el cual viene dado por:

$$e(n) = \frac{1}{2}\sum_{i=1}^{n_C}(S_{dCi} - S_{oCi})^2, \tag{5}$$

donde S_{dCi} representa la salida deseada en la capa C de salida en la neurona i, y S_{oCi} representa la respuesta obtenida en la capa C de salida en la neurona i, todo esto para el patrón n.

Por lo tanto el aprendizaje en la red sería equivalente a encontrar un mínimo en esta función de error, siendo ideal que el error tienda a cero. Está minimización representa una función no lineal debido a la aparición de las funciones de activación no lineales, que convierten a la función de error en no lineal en función de los parámetros variables de la red. La idea general de los algoritmos de minimización para funciones no lineales está basada en la búsqueda de una adaptación de parámetros siguiendo una cierta dirección de búsqueda. Particularmente para este algoritmo la dirección de búsqueda más comúnmente utilizada es la dirección negativa del gradiente, conocido como método del mínimo descenso.

Basados en la técnica anteriormente descrita, la idea es minimizar los errores individuales y de esta manera ir minimizando el error global por lo que la ley de aprendizaje para cada uno de los pesos vendrá dada por la siguiente ecuación:

$$w_{cij}(n) = w_{cij}(n-1) - \alpha\frac{\partial e(n)}{\partial w_{cij}}, \tag{6}$$

donde α es un parámetro conocido como razón de aprendizaje que indica la velocidad a la cual aprende la red e influye en la magnitud del desplazamiento sobre la superficie del error.

Para obtener el gradiente de la función a minimizar, se utiliza la regla delta generalizada y es el corazón del algoritmo de retro propagación o backpropagation, en el cuál el error cometido por la red se propaga hacía atrás a través de la misma red. Es necesario sin embargo hacer una distinción en la manera en la que se implementa el algoritmo en la última capa y en las demás.

Para esto utilizaremos un parámetro adicional δ, propio de cada neurona que representa el descenso que ocurre en el gradiente y permite el ajuste de los pesos por lo que la ecuación quedaría de la siguiente manera:

$$w_{cij}(n) = w_{cij}(n-1) - \alpha\delta_{ci}x_{cij}. \tag{7}$$

Entonces el valor delta en la neurona i lo obtendríamos a partir del error de dicha neurona de la siguiente manera:

$$\delta_{ci} = -e_i(p)f'(a_{ci}), \tag{8}$$

Ahora, extrapolando la ecuación para obtener el valor de los pesos obtendríamos la siguiente ecuación:

$$w_{cij}(n) = w_{cij}(n-1) + \alpha(e_i(p)f'(a_{ci}))x_{cij}. \tag{9}$$

Al incluir al sesgo de la neurona como entrada de valor unitario con un peso ajustable sencillamente, se aplica la fórmula anterior para realizar las modificaciones pertinentes a dicho valor.

Ahora no se posee el valor del error para realizar los ajustes de las siguientes neuronas, por lo que es necesario propagar este delta a través de las dendritas de cada neurona de forma que se utilice el delta propagado para obtener los deltas propios de cada neurona de las otras capas y así modificar los pesos de cada dendrita a través de la fórmula para tal fin, por lo que para el cálculo del nuevo delta existe el parámetro S_{ci} que representa la propagación en retroceso del delta de la última capa:

$$S_{ci} = \sum_{i=1}^{n_{c+1}} \delta_{(c+1)i}\, w_{(c+1)ij} \tag{10}$$

Recordando que esta ecuación se debe aplicar teniendo en cuanto la dendrita j que conecta la neurona en la que se está trabajando con la neurona i a la que se le extrae el delta. Luego de obtener esta sumatoria se aplica la fórmula del delta de la siguiente manera:

$$\delta_{ci} = -S_{ci}(p) f'(a_{ci}) \tag{11}$$

Ahora, la fórmula para el ajuste de peso tomando en cuenta las acotaciones mencionadas anteriormente quedaría definida de la siguiente forma, para todas las demás capas [18]:

$$w_{cij}(n) = w_{cij}(n-1) + \alpha S_{ci}(p) f'(a_{ci})) x_{cij} \tag{12}$$

Resumiendo el algoritmo en una serie de pasos para su posterior aplicación quedaría de la siguiente forma:

1. Inicializar todos los pesos (tomando el umbral como un peso adicional multiplicado por una entrada de valor unitaria).
2. Tomar un patrón n del conjunto de patrones de entrenamiento y se propaga el vector de entrada hacía delante de forma de obtener la salida a la red.
3. Evaluar el error cuadrático cometido por la red para el patrón n.
4. Aplicar la regla delta generalizada para modificar los pesos de la red de la siguiente manera:
    a. Se calculan los valores δ para todas las neuronas de salida de la red.
    b. Se modifican los valores de los pesos de la red de salida
    c. Se van calculando los S para las neuronas de la siguiente capa
    d. Se calcula los valores δ para las neuronas de esa capa.
    e. Se calculan los pesos de las neuronas de esa capa.

f. Se aplican los pasos c, d y e para el resto de las capas hasta llegar a la capa de entrada.

5. Repetir los pasos 2,3 y 4 con todos los patrones de entrenamiento, completando un ciclo de entrenamiento.
6. Repetir varios ciclos de entrenamiento hasta que el error de la red llegue a un mínimo. [18]

Ahora bien, la función que define el error E posee muchos mínimos locales, por lo que el método anterior corre el riesgo de encontrarse con uno de ellos y, en consecuencia, no obtener el entrenamiento adecuado o incluso de no converger en una solución. Para corregir esos problemas se le realizaron mejoras a este algoritmo de las que se describirá una que fue aplicada en el mismo.

Uno de los parámetros más importantes en el entrenamiento de la red es la razón de aprendizaje α, la cual determina la velocidad de aprendizaje y adicionalmente la magnitud del desplazamiento en la superficie del error E. La variación de este parámetro puede acelerar el entrenamiento y adicionalmente sacarlo de los valles o mínimos indeseados. Principalmente consiste en incrementar la velocidad de convergencia al aumentar la velocidad de aprendizaje en superficies planas y disminuye esta razón cuando la pendiente aumenta.

Para la aplicación de esta mejora se aplican 3 reglas que se mencionan a continuación:

1. Si el error cuadrático se incrementa mayor a un porcentaje establecido (1% a 5%) después de haber actualizado los pesos, entonces se descarta la actualización y α se multiplica por $0 < \rho < 1$
2. Si el error cuadrático disminuye después de haber actualizado w, entonces la actualización es aceptada, además α es multiplicado por un factor $\eta > 1$.

3. Si el error cuadrático se incrementa en un valor menor a ζ expresado en porcentaje, entonces la actualización de w se acepta pero α no cambia.

Los valores tradicionales para estos parámetros son η = 1.05, ρ = 0.7 y ζ =4%.

Estas reglas se introducen en el algoritmo al momento del ajuste de pesos de forma de ayudar a la convergencia del entrenamiento. [19]

Es importante recalcar que al aumentar el número de capas ocultas se puede aumentar la probabilidad de convergencia porque aumentamos la capacidad de la red para detectar patrones, pero así mismo el aumento de las capas ocultas produce un retardo en la convergencia del entrenamiento dado que se aumenta el número de variables involucradas. Así mismo el aumento de la cantidad de capas ocultas puede alejar al sistema de parecerse a la salida deseada así como el aumento de la cantidad de patrones puede ayudar a que se corrija el error como se muestra en las Figuras 16 y 17. [20]

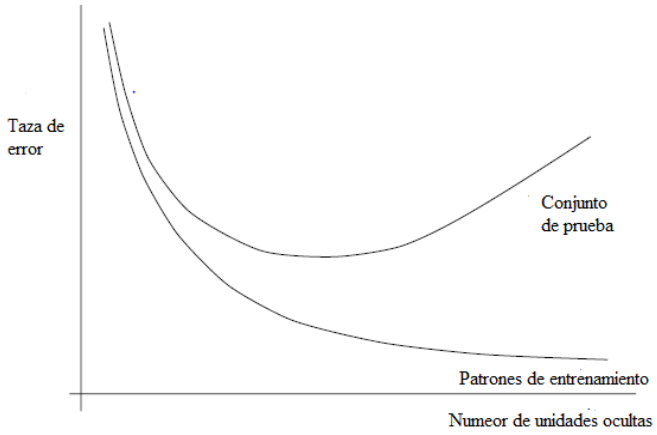

Figura 16. Taza de error en función de la cantidad de unidades ocultas para los parones de entrenamiento y el conjunto de patrones de prueba. [19]

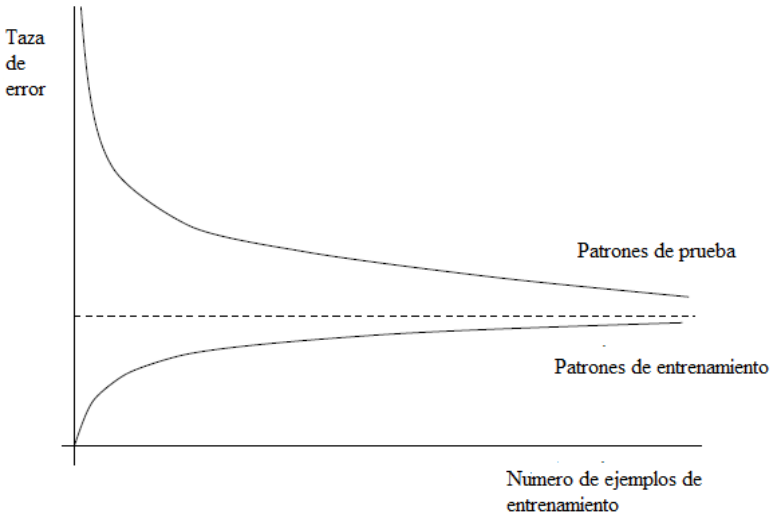

Figura 17. Taza de error en función de la cantidad de ejemplos de entrenamiento para los patrones de entrenamiento y el conjunto de patrones de prueba. [19]

CAPITULO III

MARCO METODOLÓGICO

3.1. TIPO DE INVESTIGACIÓN

La modalidad de la investigación es un proyecto factible según el Manual de la UPEL.

> "El Proyecto Factible consiste en la investigación, elaboración y desarrollo de una propuesta de un modelo operativo viable para solucionar problemas, requerimientos o necesidades de organizaciones o grupos sociales; puede referirse a la formulación de políticas, programas, tecnologías, métodos o procesos. El Proyecto debe tener apoyo en una investigación de tipo documental, de campo o un diseño que incluya ambas modalidades." [20]

3.2. FASES METODOLÓGICAS

3.2.1. Estudio y Selección

3.2.1.1. Recopilación de Información

Se realizó una recopilación de información y documentación técnica acerca de:

- El Sistema de Alarmas de las BSC y su sistema de Gestión.
- Algoritmos basados en redes neuronales.

Esta información se recopiló a través de la documentación suministrada por la empresa sobre su red y sus equipos, tomando información de la documentación apropiada sobre redes neuronales predictivas y la información disponible en la web.

3.2.1.2. Selección

Con la información obtenida en la fase anterior se procedió a seleccionar el tipo de red neuronal, con el método predictivo adecuado para la implementación en la BSC.

Dicha selección se hizo en base a los parámetros del equipo y de la manera en la que se producen las alarmas así como en la factibilidad, viabilidad y sencillez con la que se pueda aplicar la red neuronal.

3.2.2. Evaluación de la Aplicabilidad

Para realizar esta evaluación se procedió a diseñar la red neuronal seleccionada, verificar que fuese aplicable y que se pudiesen producir los resultados deseados. Para este diseñó se realizaron pruebas preliminares que determinaron lo anteriormente expuesto.

3.2.3. Programación y Diseño

En esta fase se procedió al diseño del software de prueba. Para esto se seleccionó un lenguaje de programación adecuado para dicha implementación. Luego se procedió a programar la red neuronal antes diseñada de acuerdo a los parámetros del equipo seleccionado. Además se desarrolló un algoritmo que permite entrenar la red neuronal de forma de obtener los resultados deseados. En esta fase también fue

necesario determinar el tipo de algoritmo de entrenamiento a utilizar, lo cual se hizo de acuerdo a la documentación realizada en las fases anteriores.

3.2.4. Implementación del Software y Pruebas

En esta fase se implementó el software de prueba diseñado en la fase anterior y se probó en la BSC. Se diseñaron pruebas para medir el rendimiento del programa, la velocidad de respuesta del mismo, y verificar los resultados obtenidos mediante el uso del mismo.

CAPITULO IV

DISEÑO DEL SISTEMA DE DIAGNÓSTICO

4.1. CONSIDERACIONES DE DISEÑO PRESENTES

En esta etapa de diseño fue conveniente entender primeramente cómo funciona el equipo al que se le debe analizar el sistema de alarmas mediante las RNA, para luego encontrar la mejor forma de aplicar la herramienta y obtener mejores resultados. También fue necesario entender la aplicación que se le quería dar al sistema para realizar las consideraciones necesarias para el posterior diseño.

4.1.1. Consideraciones Referentes al Sistema de Alarmas de las BSC

Como se estudió en el Capítulo 2, el sistema de alarmas de las BSC es un sistema que permite al usuario saber principalmente dos cosas: el estado de las unidades internas del equipo y su funcionamiento; y conocer el estado o deterioro de los servicios que presta dicho equipo o sus unidades internas. Las alarmas nos pueden indicar varios efectos:

- Afectaciones físicas de la unidad, como por ejemplo la alarma 2693 (WO-EX UNIT FAULTY), la cual le indica al usuario que la unidad ha salido de servicio,
- Afectaciones lógicas de la unidad, como por ejemplo la 1001 (UNIT RESTARTED), la cual nos indica que la unidad se ha reseteado.
- Notificaciones de acciones que ha realizado el equipo.
- Notificaciones del estado de las acciones internas del equipo como por ejemplo la 3030 (BSSGP VIRTUAL CONNECTION BLOCK

PROCEDURE FAILED), la cual nos indica que un bloque de procedimientos de una conexión virtual ha fallado.

- Estado de la conexión del equipo y su interfaz de usuario.

Asimismo, las alarmas pueden mostrar muchas otras cosas que le permiten al operador saber que está ocurriendo con el equipo y adicionalmente que podría ocurrir. Adicionalmente las alarmas se pueden dividir en tres grupos:

- *Notificaciones*: Nos indica que un proceso ha ocurrido o está por ocurrir.
- *Disturbios*: Son alarmas leves que no están generando afectaciones importantes.
- *Alarmas*: Son alarmas importantes que pueden generar afectaciones. Las alarmas también se dividen en tres niveles dependiendo de su criticidad, lo cual dependerá de lo cerca que se está de producirse una afectación a raíz de dicha alarma.

Las alarmas también nos indican cosas que podrían ocurrir. Dependiendo de la situación que pudiera presentarse: en algunas alarmas se pueden presentar síntomas de una afectación futura. Por ejemplo, si el operador quisiera saber si una unidad BCSU pudiera quedar fuera de servicio, él notará que primeramente podrían presentarse, por ejemplo, las siguientes alarmas:

- 2445 UNIT TYPE HAS NO REDUNDANCY – ALTO RIESGO de afectación
- 1178 PREPROCESSOR UNIT DISTURBANCE
- 1001 UNIT RESTARTED
- 3030 BSSGP VIRTUAL CONNECTION BLOCK PROCEDURE FAILED
- 3031 BSSGP VIRTUAL CONNECTION RESET PROCEDUREFAILED
- 3032 BSSGP VIRTUAL CONNECTION PROTOCOL ERROR
- 3012 DIGITAL SIGNAL PROCESSOR FAILURE

De esta manera, al presentarse dichos síntomas ya el operador podrá reconocer situaciones que pueden perjudicar el estado de la unidad. Pero, ya cuando aparezca la alarma 2693 (WO-EX UNIT FAULTY), sabrá que la unidad ha salido de servicio.

Es importante notar que las alarmas que sirven como síntomas no aparecen en todos los casos y no necesariamente predicen el acontecimiento de la afectación, pero si sirven como una buena referencia para ello.

Por lo anteriormente expuesto, los especialistas encargados de analizar el registro de alarmas son capaces de detectar afectaciones y predecir posibles afectaciones en base a las alarmas que aparecen tomando en cuenta los tiempo, las cancelaciones y las unidades donde se presentan, así como también la criticidad de las mismas. La idea general del proyecto entonces es crear una aplicación basada en redes neuronales que pueda realizar estos diagnósticos.

4.1.2. Consideraciones Referentes a las Necesidades de Implementación

Es importante diseñar un sistema que pueda realizar los diagnósticos requeridos. Para ello hay que tomar algunas consideraciones importantes.

Para realizar el diagnóstico es necesario que el sistema obtenga el registro de alarmas, y para ello debe tener la capacidad de comunicarse con el equipo y obtener dicha información de tal manera que realice los diagnósticos de forma automática. Para esto, el sistema debe tener la capacidad de comunicarse a través del protocolo Telnet, que es el protocolo utilizado para la comunicación de las BSC.

Dado que las alarmas se generan de manera espontánea, es preciso que el sistema realice el diagnóstico requerido, no necesariamente en tiempo real, sino cada cierto tiempo de forma que el diagnóstico esté constantemente actualizado.

También es necesario mostrar los resultados que se están obteniendo. Para ello es importante que el sistema sea capaz de realizar un informe de los resultados obtenidos.

4.2. EVALUACIÓN DE LA APLICABILIDAD DE LA RNA AL SISTEMA DE ALARMAS DE LA BSC

4.2.1. Necesidades que debe Cumplir la RNA para el Correcto Funcionamiento

Inicialmente se pensó en generar una red que indicara el estado de sanidad del equipo en base a los patrones que apareciesen en las alarmas, es decir, que el sistema fuera capaz de identificar patrones en un registro de alarmas cualquiera y clasificar dicho patrones para detectar que tan bueno es el funcionamiento del equipo.

Al analizar el sistema de alarmas se observó que sería mucho más eficaz generar una aplicación basada en redes neuronales capaz de reconocer los patrones generados en una afectación para poder predecir o al menos diagnosticar la aparición de la misma. Entonces, la idea es aprovechar la capacidad de procesamiento de un computador para realizar estos diagnósticos utilizando la información de los especialistas. Por lo que si pensamos en que tenemos varios especialistas analizando el registro de alarmas de una BSC, se observará que, dado que todos tienen experiencia, se podrá realizar un mejor diagnóstico y más enriquecido que si se utilizará uno solo, pero hacer eso requeriría de una mayor inversión de personal e incluso tampoco es posible tener uno solo para cada equipo. Por lo que se considera mucho mejor, en cierta forma, colocar el conocimiento de estos especialistas en una red neuronal de forma de aprovechar al máximo tal conocimiento y, de esta forma, minimizar el personal necesario para obtener una respuesta de ese tipo.

Una RNA capaz de identificar y clasificar patrones puede realizar esa función, al ser entrenada con el conocimiento de los patrones a clasificar y las salidas que debe tener para cada uno de esos patrones.

Adicionalmente se consideró que la RNA diseñada fuera capaz de identificar anomalías, es decir, de reconocer afectaciones no identificadas por los especialistas del sistema a manera de enriquecimiento del mismo y hacer a la red más segura y más fácil de diagnosticar. La capacidad de generalización resulta especialmente útil en esta parte de la labor, dado que esa capacidad le permite tomar decisiones ante entradas que no se le han presentado previamente.

Se observó que realizar una sola red que reconociera todas las posibles fallas que se produzcan en el equipo puede resultar muy complicado, debido a la cantidad de salidas que debería tener dicha red y que no es un número constante de salidas, porque dependería de la cantidad de fallas que se quieran identificar quitándole la posibilidad al usuario de ampliar la base de conocimiento de la red para incluir más fallas (que llamaremos en adelante diagnósticos). Esto ocurre debido a que habría que reestructurar la red y reentrenarla cada vez que se quisiera incluir un nuevo diagnóstico, por lo que se pensó en un sistema de dos fases que incluyera un software capaz de entrenar varias RNA, donde cada una de ellas fuera capaz de identificar una falla específica, de forma de que cada vez que se quisiera cargar en el sistema una falla nueva, se entrenara una RNA con los patrones para ello. La segunda fase consiste en un software capaz de integrar todas las RNA entrenadas y utilizarlas como librerías independientes de manera que al realizar un diagnóstico, el sistema solo tendría que llamar a cada una de las RNA e indicar el resultado de cada una de ellas.

4.2.2. Descripción de las Entradas de la RNA

Para la clasificación de patrones es importante tomar en cuenta la información disponible que suministra la BSC. Estas unidades suministran un registro

con una lista de todas las alarmas que se han producido en el día, todas las cancelaciones de dichas alarmas en caso de que hayan ocurrido, las horas en que ocurrieron, las unidades y las criticidades de dichas alarmas.

Analizando la información disponible, notamos que las entradas que se podrían tomar en cuenta son: las alarmas que se han producido en cada unidad correspondiente, la cantidad de cancelaciones que se han producido para dichas alarmas (aunque esto dependerá de la naturaleza del diagnóstico debido a que hay algunas afectaciones en las que no importa si la alarma se canceló, sino el hecho de que haya aparecido dicha alarma) y el tiempo en que aparecieron cada una de las alarmas.

El parámetro de la criticidad no se toma en cuenta debido a que es inherente al tipo de la alarma, es decir, todas las alarmas 2693 (WO-EX UNIT FAULTY) son de criticidad 3 independientemente de la unidad en la que aparezca ni del tiempo; la criticidad solo depende del tipo de alarma. Así que al conocer la alarma que se produjo también se conoce la criticidad.

Para tomar en cuenta las unidades en las que ocurre una falla se pensó en determinar diagnósticos específicos por cada unidad interna de la BSC y, de esta forma, un diagnóstico de una unidad específica solo tomaría en cuenta las alarmas que se produzcan en dicha unidad, a menos que se indique de manera diferente por el especialista para diagnósticos particulares en que la afectación de una unidad afecta otras, como en el caso de las MCMU. Ahora para realizar diagnósticos a nivel de toda la BSC no se tomaría en cuenta la unidad en la que se produjo la alarma.

4.2.3. Resultados que debe Arrojar la RNA

Para la salida de la RNA se consideró que para el diagnóstico era necesario saber de forma predictiva la situación de la unidad, es decir, que el usuario necesitaba saber que probabilidad tiene de ocurrir esa falla en particular. Por lo que se le asignó

un número del 0 al 100, donde 0 representa que no se están presentando síntomas de la falla y 100 representa que la falla está presente. Los valores internos representan que tan posible es que se presente la falla en base de los síntomas que se están presentando en el registro de alarmas. Por ejemplo, si el sistema arroja un 40 para el diagnóstico "Celda fuera de servicio", esto indica que se han estado presentando algunos síntomas de dicha falla por lo que se tendrá un 40% de probabilidad de que dicha falla se presente.

También se tomó en cuenta que el programa debería arrojar alarmas de color dependiendo de qué tan grave se considere la situación, es decir, para un mínimo de presencia de síntomas o ausencia de ellos se muestra el color verde, para un nivel de presencia de síntomas leve a medio le corresponde un verde oscuro. Para alarmas con alto riesgo de producirse o con un nivel de presencia de síntomas alto se le asigna amarillo y el color rojo se utilizó para la presencia de la falla o un muy elevado nivel de aparición de síntomas.

4.2.4. Selección de la RNA

Ahora, tomando en cuenta las necesidades anteriormente expuestas, se decidió seleccionar una topología de RNA que permitiera lo siguiente:

- Reconocimiento de patrones.
- Clasificación de patrones.
- Capacidad de generar funciones (es decir de darle forma a la respuesta, o mejor dicho de obtener respuestas que no sea solamente el valor más alto o más bajo sino que indique que tan alto o que tan bajo).
- Capacidad de realizar predicciones mediante la generalización.

Se observó que el perceptrón multicapa entrenado a través del algoritmo de retro propagación es capaz de realizar esas funciones. En cambio, la implementación

de las otras topologías resulta mucho más complicada y, además, están diseñadas para otras funciones tales como:

- La Redes Adaline-Madaline se utilizan para realizar filtros adaptativos de eliminación de ruido y reconocimiento de patrones de señales.
- La Red Self Organization Map se utiliza comúnmente para realizar mapeos de patrones de ciertas dimensiones a otra de determinada dimensión.
- La red Counterpropagation comúnmente se usa para realizar clasificación de patrones y generalizaciones.
- La red Hopfield se utiliza comúnmente para resolver problemas de optimización.
- La red Adaptative Resonance Teory se utiliza comúnmente para resolver problemas de generalización y optimización.

A pesar de que otras RNA podrían resolver el problema, resulta más complicado programarlas y requieren algoritmos más complejos para el entrenamiento, lo que puede complicar más dicha implementación.

En cuanto a la velocidad de respuesta de la RNA en un software, se identificó que la velocidad con la que actúa la RNA depende de la cantidad de capas de la misma y la cantidad de neuronas por capa, es decir, mientras más neuronas y capas el tiempo de procesamiento aumenta, por lo que no depende de la topología a implementar.

4.3. DISEÑO DE LA RED NEURONAL SELECCIONADA

Parte del diseño de la RNA es determinar las características de los elementos de la misma y como se ejecuta. También es importante determinar de qué manera estará compuesta la base de conocimiento de la red. Para los diagnósticos individuales se tomará el conocimiento de los expertos para que la red aprenda a

interpretar los casos dependiendo de las respuestas que los especialistas a cada uno de los casos, generando una matriz de patrones donde cada caso tendrá una respuesta específica.

Ahora bien, el entrenamiento de las anomalías se realizará usando todo el conocimiento de los especialistas que se ha generado para los respectivos casos, es decir, si se requiere que reconozca las anomalías de las BTS, se entrenará con todos los patrones ya cargados para las BTS de manera que entienda todos esos casos como síntomas y realice generalizaciones en base a este conocimiento.

4.3.1. Descripción General de los Elementos que Componen la RNA

Para la implementación de la RNA seleccionada se creó una estructura en la programación a la que se le dio el nombre de *TNeurona* que consta de los siguientes elementos:

- Un arreglo de una dimensión de nombre *Entradas* del tipo real donde se cargan todas las entradas de la neurona.
- Un arreglo de una dimensión de nombre *W* del tipo real donde se cargan todos los pesos de las conexiones de las entradas.
- Una variable de nombre *sesgo* que representa el umbral de la neurona.
- Una variable de nombre *delta* para utilizarla durante los entrenamientos como representación del descenso del gradiente.
- Una variable de nombre *activación* en la que se almacena el valor de la multiplicación de las entradas y los pesos.
- Una variable de nombre *salida* donde se almacena el valor de la salida procesada de la neurona.
- Una variable que indica la cantidad de entradas y de pesos de la neurona.

La función elegida para la activación de la neurona es la sigmoidea, por lo que la salida estará entre 0 y 1, y la salida de las neuronas estará siempre acotada.

Para generar la red se creó un arreglo bidimensional en donde se alojan todas las neuronas. Cada una de las filas representa una capa de neuronas donde la fila 1 es la capa de entrada. Hay una variable de nombre capas y de tipo entera que representa la cantidad de capas de la red y un arreglo de nombre *Neurpcap* del tipo entero que indica la cantidad de neuronas en cada capa dependiendo de la posición dentro del arreglo en la que se encuentre el número, es decir, si hay un número 8 en la posición 1 del arreglo, indicará que la capa uno tendrá ocho neuronas, de esta manera se logra que la red tenga un número variable de neuronas por capa y así tenga cualquier tipo de geometría no necesariamente cuadrada. Se puede ver el código de la estructura de la neurona en el Anexo 2.

4.3.2. Descripción del Módulo que Ejecuta la RNA

Se creó un módulo encargado de la ejecución de la RNA, que consiste en propagar hacía adelante la información a la entrada de la red hasta llegar a la capa de salida. Esta aplicación en concreto consta de una sola neurona debido a que cada diagnóstico tiene una salida indicando el valor antes explicado.

Cabe destacar para el claro entendimiento de este módulo, que se está tomando el umbral como la entrada en la dendrita cero de la neurona, dejando las dendritas de la uno en adelante para las conexiones con las entradas o con las neuronas de capas anteriores. El nombre que se le da a este módulo es *Red_neuronal*.

Inicialmente se colocan las entradas de la red en las entradas de cada una de las neuronas de la primera capa incluyendo el valor del umbral para la entrada cero mediante un ciclo, de forma de empezar a propagar la información. Luego se empieza el proceso de propagación obteniendo el valor de la activación de cada neurona, para esto se hace un ciclo en el cual se multiplica cada entrada por el peso correspondiente a esa entrada y se van sumando los resultados de forma sucesiva hasta obtener el valor de la activación.

Luego de obtener la activación de esa neurona, se pasa este resultado por la función de activación sigmoidea y el resultado se almacena en el campo salida de la neurona. Finalmente, se pasa el valor obtenido como salida a la entrada de las neuronas de la siguiente capa en la posición de entrada correspondiente para la neurona en cuestión dependiendo de la ubicación de ésta en la capa, es decir, si se trata de la neurona en la posición 3 de la capa, se coloca la salida de esta neurona en la entrada tres de todas las neuronas de la capa siguiente.

Este proceso se repite para todas las neuronas de la capa a través de un ciclo que va desde uno hasta la cantidad de neuronas de esa capa. Luego se va repitiendo el proceso a las capas siguientes hasta propagar la información por toda la red a través de un ciclo que va desde uno hasta el número máximo de capas.

Algoritmo 1. Módulo que ejecuta la RNA

Entradas: El vector de entradas
La cantidad de neuronas por capa (variable *neurpcap*)
La cantidad de capas (variable *capas*)
La cantidad de entradas de la Red (variable *Nentradas)*
La Red (variable *TRed*)

Salidas: El vector de salidas de la red

1. Coloque las entradas en la Red
2. Use ciclo para propagar la información de la red; usar el ciclo más externo para recorrer las capas de la red
3. Use el siguiente nivel del ciclo para recorrer las neuronas de cada capa
4. Active igual a cero
5. Use el siguiente nivel del ciclo para recorrer las entradas de la red
6. Calcule la activación de la neurona a raíz de la multiplicación de las entradas por el peso
7. Calcule la salida de la neurona con la función de activación sigmoidea
8. Coloque la salida en la entrada de la siguiente capa.
9. Repita los pasos 4, 5, 6, 7 y 8 para todas las neuronas de la capa
10. Repita los pasos 4, 5 , 6, 7, 8 y 9 para todas las capas
11. La salida será lo que se encuentre en la salida de la última capa

Al finalizar el proceso antes descrito, en la posición de salida de todas las neuronas de la última capa se tendrá la información correspondiente a la salida de la

red neuronal; esta información se coloca en un arreglo de una dimensión de tipo real generado para tal fin. El algoritmo del proceso se muestra en el Algoritmo 1.

Podemos ver que el módulo solo necesita como entradas el valor de las entradas de la red, el arreglo que representa la red, la variable *capas* y el arreglo *neurpcap* y como Salida solo el vector con todas las salidas de la red, que para el caso en cuestión, solo estará compuesto de un elemento que corresponde al valor de un diagnóstico, que luego para ser presentado se multiplica por cien. El código de este módulo puede ser consultado en el Anexo 2.

4.4. DESARROLLO DEL SOFTWARE DE ENTRENAMIENTO

4.4.1. Descripción general del Software de Entrenamiento

Como se mencionó anteriormente, se diseñaron dos software para asegurar el diseño del sistema, uno que entrenará RNA con funciones específicas y el otro que realizará el diagnóstico en base a las RNA entrenadas.

El software de entrenamiento se diseñó a través de varios módulos donde cada uno de ellos se encarga de una tarea específica. El uso del software se puede dividir en tres fases:

- Etapa de caracterización en la cual el usuario delimita el universo de la RNA.
- Etapa de carga de patrones, en donde el usuario le introduce el conocimiento de los expertos para el posterior entrenamiento.
- Etapa de entrenamiento en donde se utiliza el algoritmo de backpropagation para ajustar los pesos de la red y obtener los resultados esperados.

Sin embargo, es importante en el uso del software que la RNA generada debe ser probada de manera de garantizar su eficacia y que ésta siempre tenga un error asociado a la salida, en función al conocimiento de los especialistas y también al error propio del sistema, debido al hecho de ser un método heurístico basado en la observación.

La filosofía de este programa va orientada en el uso de archivos tipo texto para mantener la menor cantidad de memoria en uso. Esto permite utilizar un elemento y automáticamente colocarlo en un archivo tipo texto, para liberar la memoria utilizada por dicho elemento y usarla con otro fin. Luego, cuando la información sea requerida, se lee del archivo, se usa y se vuelve a colocar en dicho archivo. Muchos de los parámetros del sistema utilizan esta técnica en su funcionamiento, esto permite el uso de diferentes RNA prácticamente en paralelo, con funciones específicas, la carga de la información en sistema y el almacenamiento rápido de la información.

4.4.2. Exportar la RNA, las Configuraciones y Patrones con sus Formatos

Para la aplicación la técnica anteriormente descrita para guardar la información se utilizan 6 módulos llamados exportar_red, exportar_nombres, exportar_patrones, importar_red, importar_nombres e importar_patrones. Los códigos de estos módulos se pueden encontrar en el Anexo 2.

En los casos en los que se exporta la información se escribe directamente en el archivo tipo texto haciendo separaciones a través de espacios en blanco. A diferencia de los casos en los que se importa, en los que la técnica es leer carácter por carácter hasta encontrar espacios en blanco. Lo importante en ambos casos es el orden en el que se coloca la información.

A continuación, se describen los procedimientos donde se implementan dichos módulos; sin hacer una distinción entre los casos en los que se importa o

exporta, recordando que al importar se busca la información y al exportar se escribe la información. El procedimiento se puede ver en los Algoritmos 2, 3 y 4.

Algoritmo 2. Módulo que exporta e importa las RNA

Entradas:	Si se exporta entonces es la RNA; y si se importa entonces es un archivo de texto con la información de la RNA. La cantidad de entradas (variable *Nentradas*). La cantidad de capas (variable *capas*). La cantidad de neuronas por capa (variable *Neurpcap)*.
Salidas:	Si se está importando entonces es la RNA; y si se exporta entonces es un archivo de texto con la información de la RNA

1. Busque o escriba la cantidad de capas de la RNA.
2. Busque o escriba la cantidad de Entradas.
3. Busque o escriba el vector Neurpcap usando un ciclo.
4. Use tres ciclos anidados para buscar o escribir todos los pesos de la RNA en donde el ciclo más interno se usa para escribir los pesos de una neurona. Use el siguiente nivel para ir avanzando entre las neuronas de una capa y el último nivel para ir avanzando entre capas.
5. Exporte tanto la red como las configuraciones de la red a través de los módulos *exportar_red* y *exportar_nombres* hacía la carpeta Config para ser usadas luego por el programa.

Algoritmo 3. Módulo que exporta e importa las características de la RNA

Entradas:	Si se importa entonces es la configuración, los nombres de las entradas y salidas de la Red así como las unidades a las que afecta el estudio implementado en la RNA; y si se exporta entonces es un archivo de texto con la información antes mencionada. Cantidad de entradas (variable *Nentradas*). Cantidad de unidades afectadas (variable *Nunidades*).
Salidas:	Si se exporta la configuración entonces son los nombres de las entradas y salidas de la Red así como las unidades a las que afecta el estudio implementado en la RNA; y si se importa entonces es un archivo de texto con la información antes mencionada

1. Busque o escriba el número de entradas, salidas y de unidades a las cuáles es aplicable el diagnóstico.
2. Haga un ciclo para buscar o escribir los nombres de las entradas de la RNA.
3. Haga un ciclo para buscar o escribir los nombres de las salidas de la RNA.
4. Haga un ciclo para buscar o escribir los nombres de las unidades a las cuales el diagnóstico es aplicable.

Mediante el uso de estos procedimientos se pueden usar múltiples RNA para realizar funciones específicas donde el sistema solo tiene que cargarlas como si estuviera llamando a librerías.

Algoritmo 4. Módulo que exporta e importa los patrones de entrenamiento

Entradas:	En el caso de que se importen los patrones entonces es un archivo de texto con todos los patrones de entrenamiento del sistema. En el caso de exportar entonces son los patrones de entrenamiento del sistema. Cantidad de entradas (variable *Nentradas*). Cantidad de patrones (variable *Patrones*).
Salidas:	En el caso de que se exporten los patrones entonces es un archivo de texto con todos los patrones de entrenamiento del sistema. En el caso de importar entonces son los patrones de entrenamiento del sistema.

1. Busque o escriba el número de patrones.
2. Busque o escriba el número de entradas y salidas de la RNA.
3. Haga dos ciclos anidados. En el más interno, busque o escriba los valores de las entradas para un patrón en particular, use el ciclo más externo para ir avanzando entre los patrones.
4. Haga dos ciclos anidados. En el más interno, busque o escriba los valores de las salidas para un patrón en particular, y use el ciclo más externo para ir avanzando entre los patrones.

4.4.3. Módulo de Caracterización de la RNA

Este módulo permite ingresar a la red los parámetros que definirán a la RNA. Principalmente él módulo dará como resultado un RNA inédita y sin entrenamiento en función de los parámetros introducidos por el usuario. El procedimiento se muestra en Algoritmo 5.

En el caso en que se importe un solo diagnóstico el procedimiento se muestra en el Algoritmo 6.

Algoritmo 5. Caracterización de la RNA

Entradas: El número de capas.
El número de entradas.
El número de salidas.
El número de neuronas por cada capa.
Los nombres de las entradas y el nombre de la salida.

Salidas: Una RNA inédita con pesos aleatorios

1. Pregunte al usuario los valores para la cantidad de capas de la red, la cantidad de entradas y el número de neuronas por cada capa.
2. Pregunte al usuario el nombre de las unidades a las cuales se le realizará el diagnóstico para el que se genera la RNA.
3. Pregunte al usuario el nombre del diagnóstico y el nombre de las entradas (las cuales deben ser el código de las alarmas que se consideren importantes para el diagnóstico. De ser necesario las cancelaciones de las alarmas van seguidas de: _CANCEL, y si es necesario el tiempo en el que se realiza el diagnóstico colocando la variable tiempo_de_estudio).
4. Si se está importando un diagnóstico previamente elaborado, pregunte al usuario si desea que se continúe con la RNA importada.
5. En caso de que se desee usar una RNA importada, traiga la Red usando el módulo *importar_red* y saltar los pasos 6 y 7.
6. Genere las variables *capas*, *Nentradas* y el arreglo *Neurpcap* en función de los datos ingresados por el usuario.
7. Genere una RNA con valores aleatorios para los pesos. En esta RNA se coloca el sesgo igual a -1 y este valor se ingresa en el vector de entrada en la posición cero.
8. Exporte tanto la red como las configuraciones de la red a raves de los módulos *exportar_red* y *exportar_nombres* hacía la carpeta Config para ser usadas luego por el programa.

Algoritmo 6. Módulo que importa un solo juego de características al sistema de caracterización

Entradas: El nombre del diagnóstico del que se desea obtener la configuración, patrones y la RNA

Salidas: La RNA deseada, las configuraciones y los patrones.

1. Pregunte al usuario cual es el diagnóstico que desea importar.
2. Busque toda la información del estudio (incluyendo los patrones con que se entrenó) en la carpeta origen, la cual está identificada con el nombre del diagnóstico dentro de la carpeta Configuraciones; usando los módulos *importar_red*, *importar_nombres* e *importar_patrones*.
3. Exporte esta información hacia la carpeta Config a través de los módulos *exportar_red*, *exportar_nombres* y *exportar_patrones*.

En el caso en que se importe múltiples diagnósticos el procedimiento es el mostrado en el Algoritmo 7.

Algoritmo 7. Módulo que importa las características de varios diagnósticos

Entradas: Los diagnósticos que se desean importar.

Salidas: Los nombres de las entradas de las RNA seleccionadas y sus patrones de entrenamiento.

1. Pregunte al usuario cuál es la unidad para la cual se importan los diagnósticos.
2. Verifique cuantos de los diagnósticos en el sistema aplican para dicha unidad y cárguelos en el sistema.
3. Cargue los nombres de las variables de los diagnósticos de la parte 2, importándolos desde sus carpetas de origen.
4. Cargue todos los patrones de entrenamiento de los estudios de la parte 2 importándolos desde sus carpetas de origen.
5. Exporte los nuevos nombres (configuraciones) y patrones hacía la carpeta Config.

La causa por la cual no se importa la RNA en el último caso, es que este módulo está diseñado para generar redes nuevas, por lo que no es necesario importar la RNA.

4.4.4. Módulo de Carga de los Patrones en la RNA

El entrenamiento de la RNA es supervisado por lo que se necesita de un conjunto de patrones que sirvan de guía para ajustar los pesos. Estos patrones se cargan al sistema a través de una matriz dispuesta para tal fin. Cada fila representa un patrón individual y cada columna representa una entrada o una salida (las salidas generalmente es la última). En cada escenario se introduce el número de veces que se presentó una alarma y, de ser necesario, el número de cancelación y el tiempo en que se produjo el escenario. En la columna de salida, un valor entre 0 y 1 representa la predicción en base a las consideraciones de los especialistas. El procedimiento para introducir los patrones en el sistema se muestra en el Algoritmo 8.

Algoritmo 8. Módulo que carga los patrones al sistema

Entradas:	La cantidad de patrones. Los patrones de entrada. Los patrones de salida.
Salidas:	Los patrones cargados en el sistema.

1. Pregunte al usuario la cantidad de patrones que va a introducir y asigne la cantidad de entradas y de salidas, con sus respectivos nombres en función del módulo de caracterización de la RNA.
2. Pregunte al usuario por la matriz con la información de los patrones.
3. Asigne esa información a una estructura llamado *patrones*.
4. Exporte los patrones a través del módulo *exportar_patrones* hacía la carpeta Config.

También se puede importar los patrones ubicados en la carpeta Config para llenar la matriz de forma automática. Esto permite la carga de patrones utilizados anteriormente o la carga de patrones de varios diagnósticos simultáneamente (para esto ver como cargar configuraciones de varios estudios simultáneamente en el módulo anterior).

4.4.5. Módulo con el Sistema de Backpropagation

Este módulo es el encargado de hacer el ajuste de los pesos mediante la aplicación del algoritmo de backpropagation. Esta técnica se explicó en el capítulo anterior, y en esta sección se muestra cómo se diseñó dicho algoritmo para su implementación.

El algoritmo para un patrón en particular se muestra en el Algoritmo 9. El código de este módulo puede ser consultado en el Anexo 2.

Algoritmo 9. Módulo que ejecuta el algoritmo de Backpropagation

Entradas: Razón de aprendizaje (variable *ap)*.
Cantidad de capas (variable *capas*).
Cantidad de entradas (variable *Nentradas*).
Cantidad de neuronas por capas (arreglo *Neurpcap*).
Entradas de la red (vector de entradas).
Salidas que se esperan de la red (vector *salidas*).
Red (arreglo *TRed* formado de estructuras *TNeurona*).

Salidas: La RNA con los pesos modificados (arreglo *TRed* formado de estructuras *TNeurona*).
El vector de errores (vector *errorc*).

1. Inicialice los pesos de forma aleatoria
2. Presente un patrón de entrada y especificar la salida deseada a la RNA.
3. Ejecute el módulo *Red_neuronal* para ese patrón.
4. Calcule el error restando el valor deseado con el valor obtenido en el paso 3.
5. Calcule el valor del delta de las neuronas de la capa c (última capa) con el error obtenido en el paso 4 y la derivada de la función de activación evaluada en la activación de dicha neurona.
6. Ajuste los pesos de las neuronas de la capa c usando el delta obtenido en el paso 5.
7. Muévase hacía la capa c-1del proceso (la última capa oculta).
8. Obtenga el valor de *S* de las neuronas de la capa al propagar los deltas que están asociados a esas neuronas en la capa siguiente (actual +1).
9. Con el S obtenido en el paso 6, calcule los deltas de las neuronas de ésta capa usando la derivada de la función de activación.
10. Ajuste los pesos de las neuronas usando el delta obtenido en el paso 9.
11. Realice los pasos 8, 9 y 10 para todas las capas restantes de la RNA.
12. Calcule el error cuadrático medio.

4.4.6. Módulo que Realiza el Entrenamiento

Para realizar el entrenamiento se utiliza un temporizador que realiza un ciclo de entrenamiento cada segundo a partir del momento en que se configuren los módulos anteriores. Cabe destacar que la importación de todos los parámetros se hace desde la carpeta Config. El algoritmo de esté módulo se muestra en el Algoritmo 10.

Algoritmo 10. Módulo que entrena la RNA

Entradas:	Valores de *cita*, *neta* y *ro*. Patrones de entrenamiento. RNA a entrenar.
Salidas:	Porcentaje de entrenamiento

1. Aplique el módulo *backpropagation* a todos los patrones.
2. Verifique cuanto fue el error total de todos los patrones de entrenamiento.
3. Si es la primera vez, salte los pasos 4, 5 y 6.
4. Si el error obtenido en este ciclo es menor al obtenido en el ciclo anterior, exporte la RNA para guardar los cambios y multiplique el valor de la *ap* (razón de aprendizaje) por *neta*.
5. Si el error obtenido en este ciclo es mayor la multiplicación del error del ciclo anterior por uno más *cita*, multiplique el valor de la razón de aprendizaje por *ro*.
6. Si el error obtenido en este ciclo es mayor al obtenido en el ciclo anterior pero menor al valor de la multiplicación de dicho error por uno más *cita*, entonces exporte la RNA para guardar los cambios.
7. Calcule el valor de porcentaje de entrenamiento completado a través de la fórmula 13.
8. Repita los pasos del 1 al 7 hasta que el usuario indica el fin del entrenamiento.

$$\%Entrenamiento = \frac{1}{1+error\ cuadrático\ medio\ de\ la\ RNA} * 100 \qquad (13)$$

Se considera que la RNA está entrenada cuando el valor obtenido en el punto 10 supera el 95% de entrenamiento dado que el valor de error en este punto es menor a 0,06. El código de este módulo puede ser consultado en el Anexo 2.

4.4.7. Módulo de Pruebas de la RNA

Cuando la RNA está entrenada el sistema cuenta con un módulo que permite verificar el comportamiento de la RNA. Par esto el usuario puede introducir escenarios y ver como es la salida.

El usuario puede exportar la RNA si considera que el estudio se comporta de manera satisfactoria. El sistema crea de manera automática una carpeta con la

información de la carpeta *Config*, la cual incluye la RNA, las configuraciones de las entradas y salidas y los patrones utilizados para el entrenamiento.

4.5. DESARROLLO DEL SOFTWARE DE DIAGNÓSTICO

4.5.1. Descripción General del Software de Diagnóstico

El programa de diagnóstico permite analizar una BSC en función de todos los diagnósticos que se han cargado en el sistema con el software anterior. Este software almacenará la información de varios ciclos de diagnóstico para poder consultarlos luego. El proceso que realiza es el siguiente: se comunica con las BSC y solicita el registró de las alarmas, lo extrae del archivo que envía la unidad, lo analiza y por último lo almacena.

4.5.2. Conexiones entre el Programa de Entrenamiento y el de Diagnóstico

A manera de conexión, se han creado archivos de tipo texto que, al finalizar de crear un diagnóstico, son modificados y contienen información vital para el correcto funcionamiento del sistema de diagnóstico. El archivo contiene el número de diagnósticos que se han cargado en el sistema, luego contiene el nombre de cada uno de esos diagnósticos con la cantidad de subdiagnósticos asociados a ellos y los nombres de los mismos. Este archivo se llama Estudios_realizables.

Los subdiagnósticos se cargan de la misma manera que los diagnósticos originales con la diferencia de que se indica el diagnóstico al que pertenezca. Los subdiagnósticos dan información adicional sobre un diagnóstico en particular, informando cual es la alarma responsable de tal diagnóstico.

El nombre del diagnóstico que aparece en el archivo Estudios_realizables es el mismo que tiene la carpeta que contiene la información de dicho diagnóstico, es

decir, la configuración de la RNA, los patrones de entrenamiento y los nombres de las variables.

De manera que el software de diagnóstico conoce la ubicación de los archivos que necesita y la existencia de los mismos, por lo que al momento de usar los archivos solo necesita consultar el archivo.

4.5.3. Módulo de Conexión Vía Telnet con los Equipos

El primer paso para realizar el diagnóstico es obtener el registro de alarmas, para esto el sistema debe conectarse a la unidad para obtenerlo vía telnet. Para realizar esta función se utilizan componentes Indy, diseñados para tal fin, de forma que al llamarlos se puede enviar y recibir la información. El componente avisa cuando ha llegado la información para poder trabajar con ella. El procedimiento diseñado para realizar este proceso se muestra en el Algoritmo 11.

Algoritmo 11. Módulo que se conecta con los equipos vía Telnet

Entradas: Nombre de la BSC a analizar.
Dirección Ip de la BSC.

Salidas: Archivo de texto con el reporte de alarmas.

1. Verifique la BSC que será analizada.
2. Coloque la dirección IP de la BSC en el elemento Indy que se conecta con las BSC.
3. Inicie la comunicación Telnet.
4. Envié el Usuario y la Contraseña del sistema.
5. Envíe el comando ZIAG para actualizar la contraseña y enviar la misma contraseña anterior 2 veces (parte del protocolo de actualización de la misma).
6. Envíe de forma consecutiva los comandos ZAHO, ZEOL y ZAHP.
7. Envíe el comando ZZZ, el cuál cierra la comunicación.
8. A medida que llega la información, busque la frase END OF DIALOGUE SESSION dentro del buffer para dar por entendido que el sistema ya ha enviado toda la información.
9. Coloque el buffer en un archivo tipo texto en la carpeta en la que se está ejecutando le software con el nombre de la BSC que se está diagnosticando.

El resultado de este módulo es un archivo tipo texto, ya que es más eficiente almacenar los datos en este archivo y llamarlo cuando sea requerido, que mantener la información en memoria.

4.5.4. Módulo de Lecturas de las Alarmas y Comprensión de Dicha Lectura

Luego que la información para el diagnóstico está en un archivo, el sistema se dispone a la lectura de las alarmas. Para esto se diseñó un algoritmo que pueda entender este formato y extraer efectivamente las alarmas.

Algoritmo 12. Módulo que lee e interpreta la información proveniente de las BSC

Entradas: Archivo tipo texto con el registro de alarmas.
Nombre de la BSC.

Salidas: Estructura con todas las alarmas que se generaron

1. Abra el archivo tipo texto que contiene las alarmas y se asigna para lectura.
2. Busque la cabecera ALARMS CURRENTLY ON.
3. Busque el nombre de la BSC.
4. Lea los espacios en blanco hasta toparse con el nombre de la unidad principal y registre la misma con su respectivo número.
5. Lea la fecha, la cual consta de 22 caracteres.
6. Registre la fecha.
7. Lea el tipo de alarma (DISTUR; NOTICE; ALARM o CANCEL), y regístrela.
8. Lea la otra unidad afectada, y regístrela con su respectivo número de unidad.
9. Haga una lectura previa al código de la alarma que dependerá del tipo de alarma.
10. En caso de ser ALARM o CANCEL busque los caracteres ‘) ‘ y luego lea y almacene el código de la alarma.
11. En caso de ser DISTUR o NOTICE, baje una línea y lea el código el cual es registrado.
12. Genere una estructura con el código de la alarma, la fecha y hora, las unidades y sus números y almacene toda esa información.
13. Repita los pasos del 3 al 12 hasta toparse con el string END OF ALARMS CURRENTLY ON.
14. Repita los pasos del 2 al 13 con la diferencia que para iniciar se usa la cabecera BTS ALARM LISTING y para salir el string END OF ALARMS CURRENTLY ON.
15. Repita los pasos del 2 al 13 con la diferencia que para iniciar se usa la cabecera ALARM HISTORY y para salir el string END OF ALARM HISTORY.

En la Figura 6 se muestra el formato que tienen las alarmas, y en base a este formato se diseñó el algoritmo que se muestra a continuación. Es importante destacar que los comandos ZAHO, ZEOL y ZAHP generan un encabezado y un final para detectar donde es el comienzo y el final del comando. Para ZAHO las cabeceras al principio y al final serían ALARMS CURRENTLY ON y END OF ALARMS CURRENTLY ON respectivamente, para ZEOL serían BTS ALARM LISTING y END OF BTS ALARM LISTING, y para ZAHP serían ALARM HISTORY y END OF ALARM HISTORY, el nombre del módulo es *leer_registros*. Este procedimiento permitirá realizar un filtrado de la información. El procedimiento se muestra en el Algoritmo 12.

Al finalizar la ejecución del módulo se tendrá un arreglo con la cantidad de alarmas producidas y con toda la información relacionada con la alarma.

4.5.5. Módulo para Exportar los Resultados del Diagnóstico para su Utilización

Antes de explicar el módulo que realiza el diagnóstico es necesario hablar del módulo que importa y exporta los resultados obtenidos en los diagnósticos, dado que se utiliza dentro del mismo.

Es muy parecido a los módulos de importar y exportar anteriormente puntualizados y aplican la misma metodología que se aplicó en esos. Estos se almacenan en la carpeta Registros, luego en una carpeta con el nombre de la unidad (BTS, BCSU, etc.) y por último se genera un archivo por cada unidad con el número de la misma.

El nombre del módulo es *importar_resultados* o *exportar_resultandos* dependiendo de cuál sea el caso. El procedimiento para tal fin se muestra en el Algoritmo 13.

Algoritmo 13. Módulo que exporta e importa los resultados

Entradas:	En caso de importar entonces es un archivo de texto con los registros de diagnósticos anteriores; en caso de exportar entonces es una estructura con los registros anteriores
Salidas:	En caso de exportar entonces es un archivo de texto con los registros de diagnósticos anteriores; en caso de importar entonces es una estructura con los registros anteriores.

1. Busque o escriba el número de diagnósticos aplicables a esa unidad.
2. Haga un ciclo para escribir o buscar los diagnósticos aplicables
3. Busque o escriba el nombre del diagnóstico.
4. Busque o escriba el número de subdiagnósticos asociados al diagnóstico.
5. Haga un ciclo para buscar o escribir los valores de los últimos 400 resultados.
6. En caso de tener subdiagnósticos, haga un ciclo para buscar o escribir los resultados de los subdiagnósticos aplicando los pasos 3 y 5.

4.5.6. Módulo que Realiza el Diagnóstico

Este módulo es el corazón del sistema dado que es el que da los resultados. La idea es pasar a través de todas las unidades de la BSC y aplicarles todos los diagnósticos permitidos para cada unidad. El procedimiento para tal fin se encuentra en el Algoritmo 14.

Al finalizar el algoritmo se tendrá una estructura (los resultados) almacenada como archivo tipo texto con los últimos 400 resultados obtenidos en todos los diagnósticos aplicados a las unidades para su posterior utilización.

También se tiene un panel en el que se muestra las afectaciones mayores a 40% para alertar al usuario del software que está ocurriendo. Los resultados de este panel son extraídos directamente durante el proceso de diagnóstico y se muestra la unidad, el número, el diagnóstico y el porcentaje.

Algoritmo 14. Módulo que ejecuta el diagnóstico

Entradas: Archivo con el registro de alarmas.

Salidas: Archivos de texto con los resultados de los diagnósticos.

1. Dependiendo de si el diagnóstico se está haciendo de forma manual o automática, elija la fuente de donde se extraerán las alarmas, si es manual es la seleccionada por el usuario, si es automático es el archivo con el nombre de la BSC que está en el directorio de la carpeta.
2. Aplique el módulo *Leer_registros* para obtener el arreglo con todas las alarmas.
3. Calcule el tiempo en que ocurrieron los eventos.
4. Importe el archivo *Estudios_realizables* con la información de todos los diagnósticos cargados en el sistema.
5. Importe los parámetros básicos de cada diagnóstico mediante el módulo *importar_nombres*.
6. Genere un vector con los nombres y valores de todas las entradas de todos los estudios. Coloque todos los nombres de las entradas ya precargados en el sistema (sin repetir ninguno) y coloque todos los valores en cero. Este vector se llama *Entrada_general*.
7. Mediante dos ciclos anidados, haga un barrido de todas las unidades por nombre y por número (Es decir el primer ciclo barre todos los nombres de la unidades como BTS, BCSU, MCMU, etc. y el segundo barre los números de cada una desde 0 hasta el máximo de cada una).
8. Importe los registros de la unidad con el módulo *importar_resultados.*
9. Coloque los valores de *Entrada_general* en 0 a excepción de la variable *Tiempo_de_estudio* que se coloca según el valor obtenido en el paso 3.
10. Contraste el vector *Entrada_general* con el registro de alarmas y coloque el valor de todas las entradas (N° de veces que ocurrieron alarmas, cancelaciones, etc.) pertenecientes a los diagnósticos aplicables a esta unidad y que además hayan ocurrido en esta unidad.
11. Haga un ciclo con todos los diagnósticos aplicables a la unidad en diagnóstico.
12. Contraste las entradas del diagnóstico con el vector *Entrada_general* y cree un nuevo vector de nombre *entrada* en el que se coloca el valor que tienen las mismas en el vector *Entrada_general* en la misma posición que estas entradas deben tener en la RNA.
13. Importe la RNA del diagnóstico con el módulo *importar_red*.
14. Aplique el módulo *Red_neuronal*.
15. Ingrese el resultado en la estructura de los resultados anteriores colándolo de primero.
16. En caso de que el diagnóstico tenga subdiagnósticos, aplique los pasos del 11 al 15 para los mismos.
17. Exporte los resultados obtenidos.
18. Aplique los pasos del 8 al 17 para todas las unidades

4.5.7. Módulos para visualizar la información

Para visualizar la información se utiliza los registros almacenados. Hay dos formas para hacerlo, la primera es ver el último diagnóstico para todas las unidades del mismo tipo simultáneamente y la segunda es ver el histórico de los registros de una unidad en particular. En ambos casos se utiliza un gráfica. Para el primer tipo el procedimiento es el que se muestra en el Algoritmo 15.

Algoritmo 15. Módulo que muestra los resultados del último diagnóstico

Entradas: Archivo con los resultados de la unidad seleccionada

Salidas: Gráfica que muestra los resultados

1. Importe los registros del tipo de unidad seleccionada.
2. Coloque el nombre y los títulos a la gráfica.
3. Inicie un ciclo con el número total de unidades de ese tipo.
4. Importe los registros del número de unidad correspondiente al valor del ciclo.
5. Obtenga el valor del diagnóstico de la información importada.
6. Asigne un color en función del valor.
7. Agregue la barra a la gráfica.
8. Repita las los pasos del 4 al 7 para todas las unidades.
9. Muestre la gráfica.

El procedimiento para el historial se muestra en el Algoritmo 16.

Algoritmo 16. Módulo que muestra el histórico

Entradas: Archivo con los resultados de la unidad seleccionada

Salidas: Gráfica que muestra los resultados

1. Importe los resultados de la unidad específica.
2. Agregue el nombre y los títulos a la gráfica.
3. Haga un ciclo con todos los valores del diagnóstico y agréguelos a la gráfica desde el más antiguo al más reciente.
4. Muestre la gráfica.

4.6. IMPLEMENTACIÓN DEL SOFTWARE DE ENTRENAMIENTO Y DIAGNÓSTICO

4.6.1. Selección del Lenguaje de Programación

Para la realización del programa se decidió hacerlo en Pascal dado que la empresa no solicitó el uso de un lenguaje específico para el desarrollo del proyecto. El lenguaje seleccionado es el de mayor dominio por el diseñador, con lo que se facilita el desarrollo del proyecto. Además se observó que es posible la implementación de los algoritmos desarrollados.

La plataforma seleccionada fue el propio ambiente de Windows dado que las computadoras de la empresa utilizan dicho ambiente, por lo que usaron librerías propias al sistema.

4.6.2. Diagnósticos Cargados y Especificaciones de Cada Diagnóstico

Todos los diagnósticos fueron cargados al sistema mediante el software de entrenamiento. En la mayoría de los casos se trató de minimizar la cantidad de neuronas para reducir el error. Además, se trató de obtener el mayor número de patrones en cada caso a pesar de que para algunos diagnósticos el número de los mismos es bajo. Esto se debe a que los patrones fueron obtenidos de situaciones reales y no fue posible obtener de dichas situaciones un mayor número de patrones para el entrenamiento.

En la Tabla 2 se muestran los diagnósticos ya cargados al sistema con los respectivos parámetros para cada RNA.

Tabla 2. Diagnósticos incluidos

Nombre del Diagnóstico	Capas	Neuronas por capa	N° Entradas	Error al final del entrenamiento	Cantidad de patrones usados
Anomalías BCSU	4	9,8,7,1	21	0,020156	31
Anomalía BTS	3	10,5,1	28	0,00804	73
BCSU Fuera de Servicio	3	10,10,1	20	0,001	12
Celda Fuera de Servicio	3	10,3,1	12	0,006	16
Degradación de celdas	3	9,10,1	9	0,00637	6
Degradación de CLAB	3	10,5,1	5	0,0027	9
Degradación servicio de voz	3	10,10,1	19	0,0015	25
Degradación de STMU	3	5,2,1	5	0,000000322	4
Degradación servicio de datos BSC	3	10,8,1	10	0,006	5
Degradación servicio de datos BTS	3	10,3,1	19	0,0018	26
Descenso de tráfico	3	7,4,1	7	0,00024	9
Diagnóstico conmutación	3	8,8,1	8	0,00038	5
Fallo de BSC	4	10,10,10,1	57	0,0029	7

Se observa que en la totalidad de los casos la convergencia de la RNA fue posible y el error cuadrático (el mostrado en la tabla) es menor a 0,020156 (diferencia de lo esperado con lo obtenido elevado al cuadrado), por lo que se puede decir que en cuanto a asemejarse a los patrones el entrenamiento es satisfactorio. En 11 de los 13 casos fue necesario realizar más de un entrenamiento para que la RNA se comportara de manera satisfactoria o convergiera en el resultado deseado. Los Patrones identificados pueden ser consultados en el Anexo 1.

4.6.3. Implementación del Sistema en un Equipo

Cuando el software de diagnóstico estuvo terminado se colocó a funcionar en un equipo para obtener data y ver los resultados obtenidos. Dado que la aplicación funciona bien en ambiente Windows, el CPU en el que se montó tenía la versión XP de dicho sistema operativo.

La prueba a realizar para verificar el correcto funcionamiento de la herramienta era colocar la misma a funcionar y observar el comportamiento de la misma. Se conectó el sistema en diversas BSC las cuáles se muestran en la Tabla 3.

Tabla 3. Pruebas realizadas

BSC	FECHA
BSCCAN01	2013-08-29
BSCCAN02	2013-08-29
BSCCAN03	2013-08-29
BSCPLC01A	2013-08-14, 2013-08-07, 2013-07-28
BSCPLM01	2013-10-14
BSCBNS01	2013-08-29

Dicho comportamiento debía ser validado por un especialista. Se hicieron un total de 8 validaciones de las cuáles 7 funcionaron de manera adecuada. En la mayoría de los casos se obtuvieron resultados satisfactorios a excepción de aquellos en los cuáles al sistema se le estaba realizando algún tipo de mantenimiento, situación en la cual indicaba una falla, generalmente en el funcionamiento total de la BSC, o presentaba algún tipo de anomalía.

CAPITULO V

RESULTADOS

A continuación se muestran algunas de las pruebas realizadas. En el primer caso se ve como el sistema detectó el descenso de tráfico en una de las BSC analizadas, el caso es del 8 de julio de 2013 en la BSCPLC01A. Las gráficas referentes al estado de las BCSU indican lo siguiente:

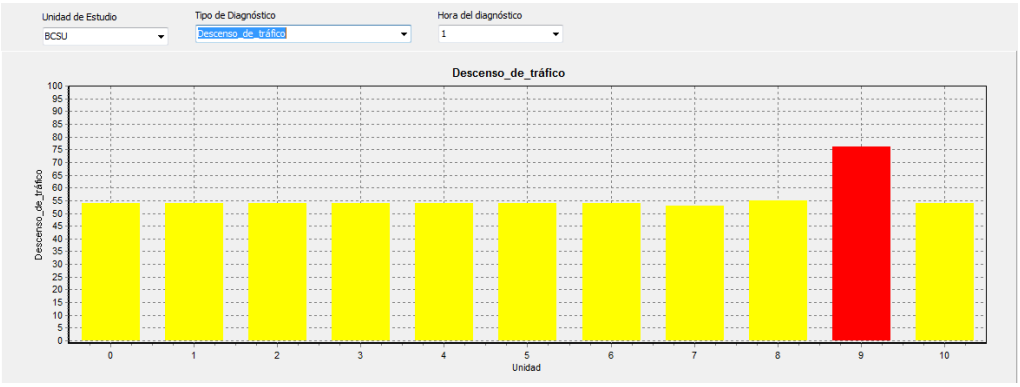

Figura 18. Gráfica Descenso de tráfico

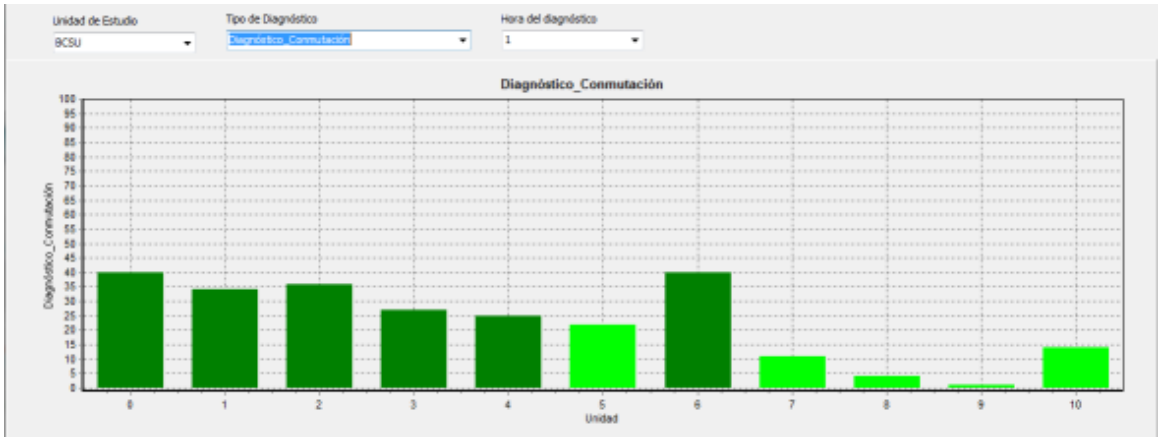

Figura 19. Gráfica Conmutación

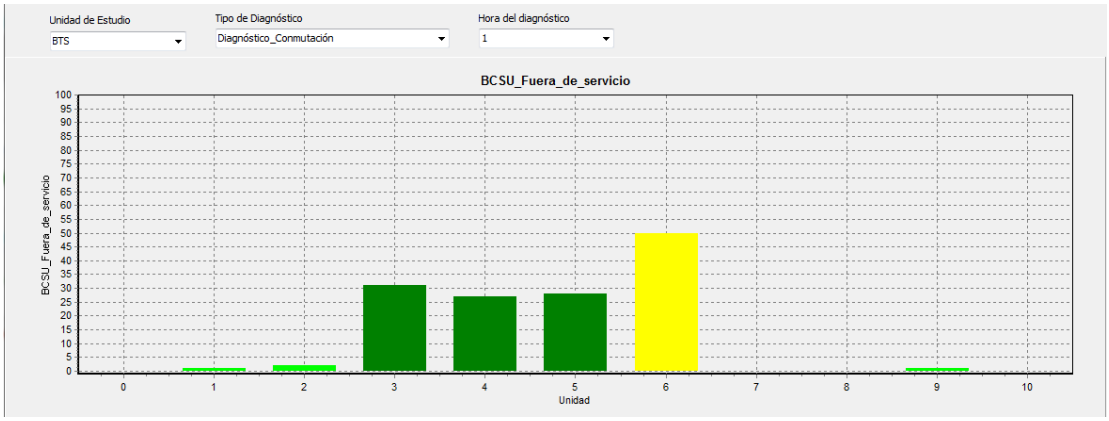

Figura 20. Gráfica BCSU Fuera de servicio.

En este caso podemos ver como el sistema detectó la falla principal de manera efectiva y adicionalmente detectó algunos otros parámetros de interés del estudio.

El siguiente ejemplo se aplicó en la BSCCAN03 el día 29 de agosto de 2013. Se observa como detecta una anomalía producida en una celda:

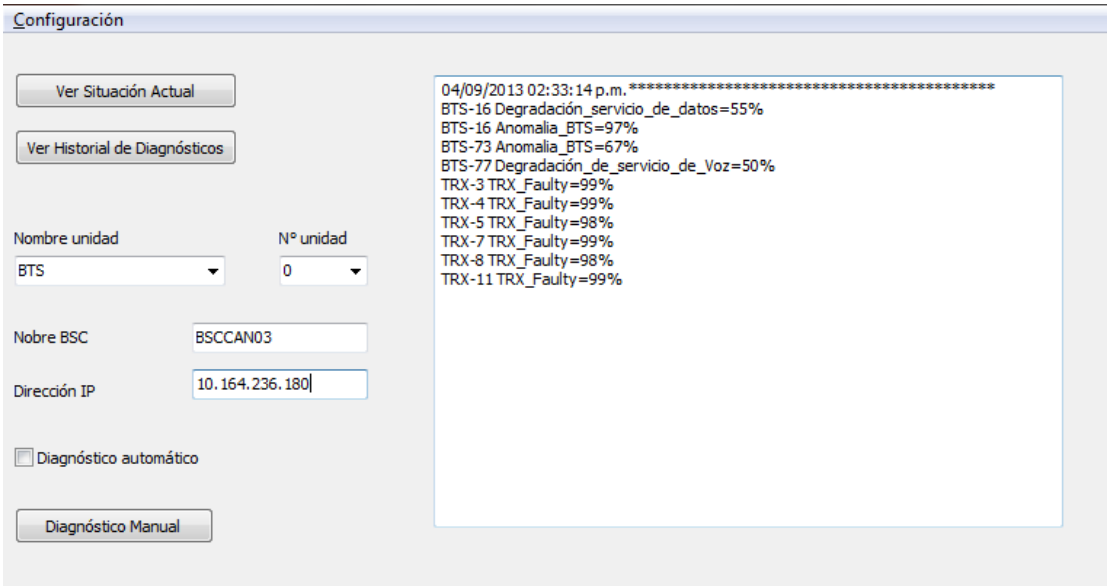

Figura 21. Imagen panel de información

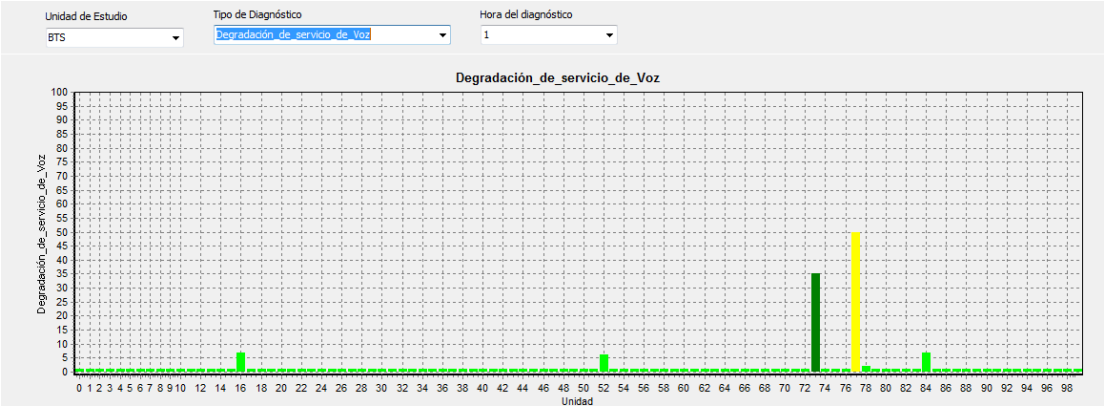

Figura 22. Gráfica de degradación del servicio de voz

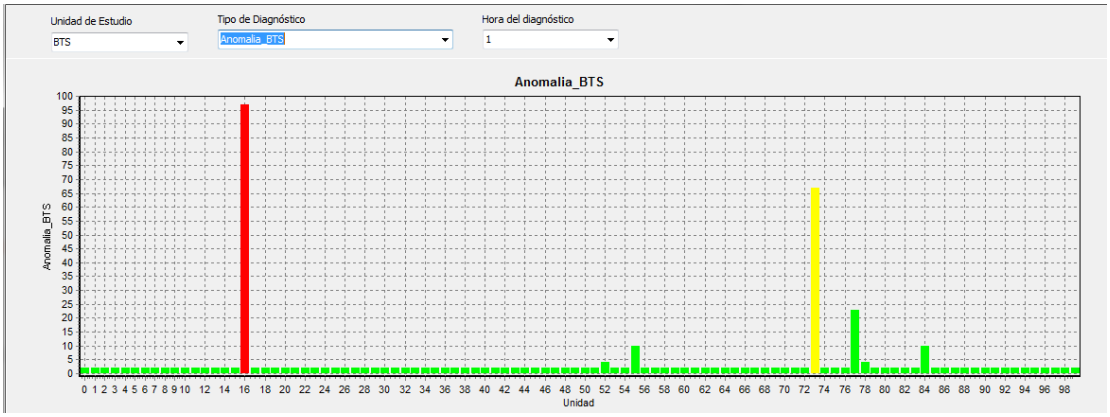

Figura 23. Gráfica de anomalías en la BTS.

Podemos ver en este caso como el sistema detectó una anomalía en la celda 16 que no pudo ser detectado por otros diagnósticos. Una de las cosas que se quería saber era la relación entre los resultados obtenidos por el sistema y los resultados obtenidos por otros métodos. El método tradicional de diagnóstico de la BSC es simplemente monitorear el registro de alarmas por el especialista y de esta manera el especialista informa de lo que ve y de los diagnósticos que él realiza. Ahora bien, dado que el sistema está entrenado con la información que los especialistas suministran, pues el sistema está en capacidad de dar la misma información que ellos, como se vio durante las pruebas. Pero, en este ejemplo se ve que adicionalmente puede mostrarnos situaciones no conocidas brindando ventaja sobre el sistema anterior.

En cuanto a la velocidad de respuesta realmente no se puede medir la velocidad de ejecución del programa dado que varía según la situación y la cantidad de alarmas presentes en el sistema. Se observó que el programa requiere, en promedio, entre 14 y 45 segundos para realizar sus diagnósticos. Ahora, la velocidad en la que los especialistas dan un resultado depende de cuándo fue la última vez en que revisaron el historial de alarmas. En cambio, el sistema se conecta cada 15 minutos y revisa las alarmas, lo cual es una notable mejora, ya que el especialista no tiene necesidad de estar siempre monitoreando las alarmas.

En cuanto al diagnóstico *Fallo_BSC,* en particular se observó que este diagnóstico presentó más problemas y mayor número de confusiones. Una de las razones de esto, es que el número de patrones asociados al entrenamiento es demasiado pequeño.

CONCLUSIONES

En general, la aplicación de RNA como herramienta en predicción y detección eficaz de errores ha tomado mucho interés. La idea principal de este proyecto era identificar la posibilidad de aplicar algoritmos basados en RNA al sistema de alarmas de las BSC de TELEFÓNICA VENEZOLANA C.A., con miras de obtener diagnósticos y respuestas a situaciones no previstas de manera preventiva. Un sistema con estas capacidades puede mejorar la gestión sobre las BSC.

El estudio de los diferentes tipos de RNA permitió identificar una topología adecuada para la implementación del sistema que fuera funcional y al mismo tiempo de fácil implementación. Para poder realizar la implementación del sistema fue necesario el desarrollo de otras RNA más sencillas, pero con la misma topología. También se necesitó conocer el comportamiento de las mismas ante diferentes situaciones, lo cual nos permitió demostrar la versatilidad de la RNA seleccionada siendo esto de gran importancia, ya que se querían realizar diferentes diagnósticos.

Durante el desarrollo del proyecto se pudo observar factibilidad de la aplicación de dichos algoritmos en la situación estipulada, lo cual se comprobó diseñando un software de prueba. Se observa que una de las maneras de aplicarlo es generar RNA con funciones de diagnóstico y entrenamientos específicos para cada una de las situaciones.

Además de cumplir con las exigencias estipuladas, el sistema cuenta con otras bondades como: un mayor nivel de libertad al permitirle al especialista transferir su propio conocimiento al sistema con la información que él considere oportuna para cada particular y corregirlo de manera sencilla, en vez de programar una gran cantidad de casos para cada diagnóstico; independencia del sistema con respecto al especialista debido a que genera sus propias reglas y, permite identificar situaciones que a los especialistas se le escapan gracias a su capacidad de generalizar situaciones.

Durante el desarrollo de las pruebas se pudo observar el comportamiento de las RNA cargadas, cuyo funcionamiento fue satisfactorio con respecto a lo esperado. Lo especialistas validaron en su totalidad 7 de los 8 casos con los que se realizaron las pruebas. En el caso en el que ocurrió la discrepancia, se debió a que se estaba realizando un procedimiento de diagnóstico en la BSC y el sistema reportó anomalías. De alguna manera, el resultado podría ser el esperado, dado que en el procedimiento de diagnóstico, aparecieron alarmas comunes a las que aparecen en algunos casos de fallas; por ello, el sistema al verlas levantó la alerta de anomalía.

También es importante destacar que para un correcto funcionamiento de las RNA, es necesario contar con la fiabilidad de los patrones de entrenamiento y que mientras más patrones de entrenamiento se tengan, los resultados serán más adecuados. Adicionalmente, si bien no se pueden determinar valores exactos para definir la arquitectura de la RNA de forma analítica (en cuanto a número de capas y de neuronas por capa), se pudo observar que a medida que se aumenta el número de capas ocultas en gran proporción, el sistema tiende a dar más errores.

Finalmente, para que el sistema pueda ser más eficaz es necesario cargar un número mayor de diagnósticos de manera que él pueda identificar un mayor número de fallas y/o anomalías; dado que el sistema depende casi en su totalidad del conocimiento de los expertos.

RECOMENDACIONES

Es importante poder continuar con la realización de pruebas del sistema para lograr obtener una validación más completa, es decir, monitorear el comportamiento del sistema y verificar que los resultados sean correctos, de manera de identificar los posibles errores y ubicar una solución o verificar si es posible realizar alguna mejora, tanto a nivel gráfico o de programación.

Dado que la cantidad de información suministrada es crucial para el correcto funcionamiento es importante continuar cargando al sistema con más diagnósticos para ampliar el rango de acción y así lograr hacer al sistema mucho más completo. De esta manera se puede mejorar el sistema de detección de anomalías.

Adicionalmente, viendo el éxito de la aplicación en las BSC sería recomendable poder extender la aplicación a otros equipos de la red GSM de manera de obtener resultados similares, para ello habría que ajustar la comunicación del sistema a la interfaz de los otros equipos y verificar la manera en que se interpretan las alarmas.

También sería interesante verificar la utilización de otra topología de RNA u otro método de entrenamiento para verificar los resultados obtenidos y compararlos con los que se obtienen mediante la herramienta desarrollada. En cuanto al desarrollo de la aplicación, se podría verificar también si hay algún lenguaje de programación y/o compilador que permita un mayor desempeño.

REFERENCIAS

[1] Telefónica, Gerencia CCR, Manual de Procedimientos para atacar fallas a nivel de celdas GSM, 2008.

[2] M. Villasana, Introducción a las redes neuronales, Caracas: Universidad Simón Bolivar, 2000.

[3] A. Fernandez, «Desarrollo de un modelo computacional para un sistema de reconocimiento de matriculas a traves de una imagen proveniente de una cámara de tráfico,» Univercidad Central de Venezuela, Caracas, 2011.

[4] A. Moreno, Redes neuronales artificiales para resolver problemas de programación lineal, Mérida, Venezuela: Universidad de Los Andes, 2005.

[5] J. V. Collantes Duarte, «Predicción con redes neuronales: comparación con las metodologías de Box y Jenkins».

[6] Telefónica, Gerencia CCR, Manual OSS manejo básico de herramientas, 2008.

[7] E. Forner Clavijo, , C. Torrent Cuevas, R. M. Mateu, F. Cordobes Gil y P. Martinez Dimingo, Tecnología GSM.

[8] Telefónica, Gerencia CCR, «Manual básico de MSS,» 2008.

[9] J. Montelius, «GSM Infraestructure».

[10] Telefónica, Gerencia CCR, Manual de procedimientos para Básica BSS, 2008.

[11] E. Lara, «Protocolo Telnet y SSH,» UPC.

[12] X. B. Olabe, «Redes Neuronales Artificiales y sus aplicaciones,» Escruela Superior de Ingeniería de Bilbao, departamento de Ingeniería de sistemas y automática, Bilbao, España.

[13] D. Gu y H. Hu, Wavelet Neural Network based predictive control for mobile robots, Reino Unido: Dept. of Computer Science , University of Essex, 2000.

[14] Departamento de informatica Universidad Nacional de San Luis, Aprendizaje Maquina (Parte 2) Redes neuronales artificiales, San Luis, Argentina.

[15] N. K. Kasabov, Foundations of neural networks, fuzzy systems, and knowledge Engineering, Massachusets: MIT Press, 1998.

[16] A. Zilouchian, Fundamentals of neuronal networks.

[17] Z. Michalewitz y D. B. Fogel, How to solve it: Modern Heuristic, Charlotte: Springer, 2000.

[18] Variantes de Backpropagation, ESCOM IPN, 2002.

[19] B. Krôse y P. Van der Smagt, An introduction to Neural Networks, Amsterdam: University of Amsterdam, 1996.

[20] Universidad Pedagógica Experimental Libertador (UPEL), «Manual de trabajos de grado, Especialización y Maestría y Tesis Doctorales,» Vicerrectorado de Investigación y Postgrado, Caracas. Venezuela, 2006.

BIBLIOGRAFÍA

Telefónica, Gerencia CCR, Manual de Procedimientos para atacar fallas a nivel de celdas GSM, 2008.

M. Villasana, Introducción a las redes neuronales, Caracas: Universidad Simón Bolivar, 2000.

A. Fernandez, «Desarrollo de un modelo computacional para un sistema de reconocimiento de matriculas a traves de una imagen proveniente de una cámara de tráfico,» Univercidad Central de Venezuela, Caracas, 2011.

A. Moreno, Redes neuronales artificiales para resolver problemas de programación lineal, Mérida, Venezuela: Universidad de Los Andes, 2005.

J. V. Collantes Duarte, «Predicción con redes neuronales: comparación con las metodologías de Box y Jenkins».

Telefónica, Gerencia CCR, *Manual OSS manejo básico de herramientas,* 2008.

E. Forner Clavijo, , C. Torrent Cuevas, R. M. Mateu, F. Cordobes Gil y P. Martinez Dimingo, Tecnología GSM.

Telefónica, Gerencia CCR, «Manual básico de MSS,» 2008.

J. Montelius, «GSM Infraestructure».

Telefónica, Gerencia CCR, Manual de procedimientos para Básica BSS, 2008.

E. Lara, «Protocolo Telnet y SSH,» UPC.

X. B. Olabe, «Redes Neuronales Artificiales y sus aplicaciones,» Escruela Superior de Ingeniería de Bilbao, departamento de Ingeniería de sistemas y automática, Bilbao, España.

D. Gu y H. Hu, Wavelet Neural Network based predictive control for mobile robots, Reino Unido: Dept. of Computer Science , University of Essex, 2000.

Departamento de informatica Universidad Nacional de San Luis, Aprendizaje Maquina (Parte 2) Redes neuronales artificiales, San Luis, Argentina.

N. K. Kasabov, Foundations of neural networks, fuzzy systems, and knowledge Engineering, Massachusets: MIT Press, 1998.

A. Zilouchian, Fundamentals of neuronal networks.

Z. Michalewitz y D. B. Fogel, How to solve it: Modern Heuristic, Charlotte: Springer, 2000.

Variantes de Backpropagation, ESCOM IPN, 2002.

B. Krôse y P. Van der Smagt, An introduction to Neural Networks, Amsterdam: University of Amsterdam, 1996.

Universidad Pedagógica Experimental Libertador (UPEL), «Manual de trabajos de grado, Especialización y Maestría y Tesis Doctorales,» Vicerrectorado de Investigación y Postgrado, Caracas. Venezuela, 2006.

L. Veelenturf, Analysis and aplication of artificial neural networks, Herdforshire: Prentice Hall, 1995.

B. Topping, Parallel procesing, neural networks and genetic algorithms, Gran Bretaña: Elsevier Science Ltd., 1998.

S. Haykin, Kalman Filtering and Neuronal Networks, Ontario: Wiley interscience publication, 2001.

P. Garcia Báez, Introducción a las redes neuronales y su aplicación a la investigación astrofísica, Intituto Astrofísica Canarias.

A. Lorente Cablanque, El protocolo Telnet.

A. K. Jain y J. Mao, Antificial Neuronal Networks: A tutorial, 1996.

D. Klerfors y D. T. L. Huston, Artificial Neural Networks, St. Louis, 1998.

R. M. Cunquero, «Algoritmos Heurísticos en optimización combinatoria,» Universidad de Valencia, Valencia, España.

ANEXOS

[ANEXO 1]

[PATRONES UTILIZADOS EN LOS ENTRENAMIENTOS]

A continuación se muestran todos los patrones identificados en el entrenamiento, Se presentarán en forma de matriz en la que cada fila representa un patrón, y cada una de las columnas representa cada una de las entradas. Las entradas están colocadas en el mismo orden en el que se nombran yh todos los patrones tiene una sola salida.

1.1. Anomalías BCSU:

Entradas: 1001, 1007, 1178, 2692, 2693. 3164, 3483, Tiempo_de_estudio, 2445, 3030, 3031, 3032, 3012, 3031_CANCEL, 3030_CANCEL, 2445_CANCEL, 3032_CANCEL, 2693_CANCEL, 1254, 0691, 2770.

Patrones de entrada	*Patrones de salida*
0 0 0 0 0 0 0 2 0 0 0 0 0 0 0 0 0 0 0 0	0
1 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0	0,1
4 1 1 1 0 0 0 2 0 0 0 0 0 0 0 0 0 0 0 0	0,7
6 2 2 1 1 2 1 1 0 0 0 0 0 0 0 0 0 0 0 0	1
1 1 1 1 0 1 1 4 0 0 0 0 0 0 0 0 0 0 0 0	0,3
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	0
0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	0,05
0 0 8 0 1 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0	0,5
0 0 0 0 1 0 0 0 1 1 1 1 0 0 0 0 0 0 0 0	1
0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0	0,7
1 0 0 0 1 0 0 0 0 1 1 1 0 0 0 0 0 0 0 0	1
1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	0,8
0 0 0 0 0 0 0 0 0 0 0 0 2 0 0 0 0 0 0 0	0,5
2 0 1 0 0 0 4 0 0 0 0 0 0 0 0 0 0 0 0 0	0,62
2 0 1 0 0 0 5 0 0 0 0 0 0 0 0 0 0 0 0 0	0,75
0 0 0 0 0 0 0 0 1 1 1 1 0 0 0 1 0 0 0 0	0,8
0 0 0 0 0 0 0 0 1 1 1 1 0 0 1 0 1 0 0 0	0,8
0 0 0 0 0 0 0 0 1 1 1 1 0 1 1 0 0 0 0 0	0,8
0 0 0 0 0 0 0 0 1 1 1 1 0 1 0 0 1 0 0 0	0,8
0 0 1 0 1 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0	0,58
1 0 1 0 1 0 1 0 1 1 1 1 1 1 1 1 1 0 0 0	0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0	0

1 1 1 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 1 0	1
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	0
0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0	0,1
1 1 0 0 0 0 4 0 0 0 0 0 0 0 0 0 0 0 0 0	0,4
0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	0,2
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1	1
1 1 1 0 0 0 1 0 0 0 0 0 0 0 0 0 0 1 1 1	0,8
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1	0,8
2 1 0 0 0 0 10 0 0 0 0 0 0 0 0 0 0 0 0 0	0,6

1.2.Anomalías BTS

Entradas: 7604, Diversidad, 7607, 7725, 7601, Potencia. 7704. Nivel_de_interferencia. Tiempo_de_estudio, 7412, 7401, 7414, falla_motogenerador, 7419, 7403, mayor_rectificación, 7421, 7402, 7405, motogenerador, bajo_nivel_combustible, bloqueo_de_NSEI, Configuración_celda_fallida, Duplexores, 7606, Diversdad, 7705. No_gprs_transation.

Patrones de entrada	*Patrones de salida*
1 2 5 3 1 4 3 2 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	0
0 0 0 0 0 0 0 0 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	0,95
1 3 5 2 4 3 2 1 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	0,85
5 4 2 1 3 2 5 3 4 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	0,65
3 3 2 2 4 5 2 1 3 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	0
0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	1
0 0 0 0 0 0 0 0 2 1 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0	0,7
0 0 0 0 0 0 0 0 2 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	0,5
0 0 0 0 0 0 1 0 2 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	1
0 0 0 0 0 0 5 0 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0	0
0 0 0 0 0 0 0 0 6 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	0
0 0 0 0 0 0 0 0 10 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	0,7
0 0 0 0 0 0 0 0 2 1 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0	0,5
0 0 0 0 0 0 1 0 2 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0	1
0 0 0 0 0 0 5 0 1 1 0 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0	1
0 0 0 0 0 0 0 0 12 1 0 1 0 0 0 0 0 1 1 0 0 0 0 0 0 0	0,3
0 0 0 0 0 0 0 0 5 1 0 1 0 0 0 0 0 1 1 0 0 0 0 0 0 0	1
0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0	0,2
0 0 0 0 0 0 0 0 1 1 0 1 0 0 0 0 0 1 1 0 0 0 0 0 0 0	1
0 0 0 0 0 0 0 0 20 1 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0	1
0 0 0 0 0 0 0 0 1 1 0 1 1 0 0 0 0 1 1 0 0 0 0 0 0 0	1
1 2 5 3 1 4 3 2 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	0

0 0 0 0 0 0 0 0 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	0,95
1 3 5 2 4 3 2 1 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	0,85
5 4 2 1 3 2 5 3 4 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	1
3 3 2 2 4 5 2 1 3 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	0,95
1 2 5 3 1 4 3 2 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 01	0,85
0 0 0 0 0 0 0 0 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	0,65
1 3 5 2 4 3 2 1 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	0
5 4 2 1 3 2 5 3 4 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	0,65
3 3 2 2 4 5 2 1 3 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	0
0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	1
0 0 0 0 0 0 0 0 2 1 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0	0,7
0 0 0 0 0 0 0 0 2 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	0,5
0 0 0 0 0 0 1 0 2 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	1
0 0 0 0 0 0 5 0 1 1 1 0 1 0 1 1 0 0 0 1 1 0 0 0 0 0 0 0 0 0	0
0 0 0 0 0 0 0 0 6 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	0
0 0 0 0 0 0 0 0 10 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	0,7
0 0 0 0 0 0 0 0 2 1 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0	0,5
0 0 0 0 0 0 1 0 2 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0	1
0 0 0 0 0 0 5 0 1 1 0 0 1 0 1 1 1 0 0 1 1 0 0 0 0 0 0 0 0 0	1
0 0 0 0 0 0 0 0 12 1 0 0 0 0 0 0 0 0 1 1 1 0 0 0 0 0 0 0 0 0	0,3
0 0 0 0 0 0 0 0 5 1 0 0 0 0 0 0 0 0 1 1 1 0 0 0 0 0 0 0 0 0	1
0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0	0
0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	1
0 0 0 0 0 0 0 0 2 1 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	0,7
0 0 0 0 0 0 0 0 2 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	0,5
0 0 0 0 0 0 1 0 2 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	1
0 0 0 0 0 0 5 0 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0	0
0 0 0 0 0 0 0 0 6 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	0
0 0 0 0 0 0 0 0 10 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	0,7
0 0 0 0 0 0 0 0 2 1 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0	0,5
0 0 0 0 0 0 1 0 2 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0	1
0 0 0 0 0 0 5 0 1 1 0 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0	1
0 0 0 0 0 0 0 0 12 1 0 1 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0	0,3
0 0 0 0 0 0 0 0 5 1 0 1 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0	1
0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0	0
0 0 0 0 0 0 0 0 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	1
0 0 0 0 0 0 0 0 2 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 0 0 1	1
0 0 0 0 0 0 0 0 3 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0	0,7
0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0	0,25
0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 1 0	0,4
0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0	0,15
0 0 0 0 0 0 0 0 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0	0,8
0 0 0 0 0 0 0 0 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 2 0 0 0	0

0 0 0 0 0 0 0 0 7 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	1
0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1	1
0 0 0 0 0 0 0 0 1,5 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 1 0	1
0 0 0 0 0 0 0 0 1,5 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1	0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	0

1.3. BCSU Fuera de servicio

Entradas: 2693, 2445, 1178, 3030, 3031, 3032, 3483, 1001, 3012, 3031_CANCEL, 3030_CANCEL, 2445_CANCEL, 3032_CANCEL, 3032_CANCEL, 3030_CANCEL, 3031_CANCEL, 2693_CANCEL, 2692_CANCEL, 0690, 0691

Patrones de entrada	*Patrones de entrada*
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	0
0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	0,05
1 0 8 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	0,5
1 1 0 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	1
0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	0,7
1 0 0 1 1 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0	1
0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0	0,8
0 0 0 0 0 0 0 0 2 0 0 0 0 0 0 0 0 0 0 0 0 0	0,5
0 0 1 0 0 0 4 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0	0,62
0 0 1 0 0 0 5 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0	0,75
0 1 0 1 1 1 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0	0,8
0 1 0 1 1 1 0 0 0 0 1 0 1 1 1 0 0 0 0 0	0,8
0 1 0 1 1 1 0 0 0 1 1 0 0 0 1 1 0 0 0 0	0,8
0 1 0 1 1 1 0 0 0 1 0 0 1 1 0 1 0 0 0 0	0,8
1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	0,58
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0	0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0	0
1 1 0 1 1 1 0 0 0 0 0 0 0 0 0 0 1 1 0	0,2
1 1 0 1 1 1 0 0 0 0 0 0 0 0 0 0 1 1 1	0,05
1 0 0 1 1 1 0 1 0 0 0 0 0 0 0 0 1 1 1	0,05

1.4.Degradación de celdas

Entradas: 7604, Diversidad, 7607, 7725, 7601, Potencia, 7704, Nivel_de_interferencia, Tiempo_de_estudio.

Patrones de entrada	*Patrones de salida*
1 2 5 3 1 4 3 2 2	1
0 0 0 0 0 0 0 0 2	0
1 3 5 2 4 3 2 1 2	0,95
5 4 2 1 3 2 5 3 4	0,85
3 3 2 2 4 5 2 1 3	0,65
0 0 0 0 0 0 0 0 0	0

1.5. Degradación de CLAB

Entradas: 0690, 1687, 1001, 0692, 2692_CANCEL.

Patrones de entrada:	*Patrones de salida*
0 0 0 0 0	0
0 0 0 1 1	0
1 1 1 10 10	0
0 1 0 0 0	0,05
4 1 0 0 0	0,2
6 1 1 0 0	0,4
8 2 2 1 0	0,8
10 1 1 2 0	1
10 1 1 2 2	0,05

1.6.Celda fuera de servicio

Entradas: 7412. 7401. 7414. falla_motogenerador, 7419, 7403, mayor_rectificación, 7704, Tiempo_de_estudio, 7421, 7402, 7405

Patrones de entrada	*Patrones de salida*
0 0 0 0 0 0 0 0 1 0 0 0	0
1 0 1 0 1 0 0 0 2 0 0 0	1
1 1 0 0 0 0 0 0 2 0 0 0	0,7
0 1 0 0 0 0 0 1 2 0 0 0	0,5
1 1 1 1 1 1 1 5 1 0 0 0	1
0 0 0 0 0 0 0 0 6 0 0 0	0
0 0 0 0 0 0 0 0 10 0 0 0	0
1 0 0 0 0 0 0 0 2 1 0 0	0,7

0 0 0 0 0 0 0 1 2 1 0 0	0,5
1 0 1 1 1 1 1 5 1 1 0 0	1
1 0 1 0 0 0 0 0 12 0 1 1	1
1 0 1 0 0 0 0 0 5 0 1 1	0,3
1 0 0 0 0 0 0 0 1 0 1 1	1
1 0 1 0 0 0 0 0 1 0 1 1	0,2
1 0 0 0 0 0 0 0 20 0 1 1	1
1 0 1 1 0 0 0 0 1 0 1 1	1

1.7.Degradación de servicio de voz

Entradas: 7604, Diversidad, 7607, 7725, 7601, Potencia, 7704, Nivel_de_interferencia, Tiempo_de_estudio, 7412, 7401, motogenerador, falla_motogenerador, bajo_nivel_combustible, 7403, mayor_rectificación, 7421, 7402, 7405

Patrones de entrenamiento	*Patrones de salida*
1 2 5 3 1 4 3 2 2 0 0 0 0 0 0 0 0 0	1
0 0 0 0 0 0 0 0 2 0 0 0 0 0 0 0 0 0	0
1 3 5 2 4 3 2 1 2 0 0 0 0 0 0 0 0 0	0,95
5 4 2 1 3 2 5 3 4 0 0 0 0 0 0 0 0 0	0,85
3 3 2 2 4 5 2 1 3 0 0 0 0 0 0 0 0 0	0,65
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	0
1 2 5 3 1 4 3 2 2 0 0 0 0 0 0 0 0 0	1
0 0 0 0 0 0 0 0 2 0 0 0 0 0 0 0 0 0	0
1 3 5 2 4 3 2 1 2 0 0 0 0 0 0 0 0 0	0,95
5 4 2 1 3 2 5 3 4 0 0 0 0 0 0 0 0 0	0,85
3 3 2 2 4 5 2 1 3 0 0 0 0 0 0 0 0 0	0,65
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	0
0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0	0
0 0 0 0 0 0 0 0 2 1 0 1 0 1 0 0 0 0	1
0 0 0 0 0 0 0 0 2 1 1 0 0 0 0 0 0 0	0,7
0 0 0 0 0 0 1 0 2 0 1 0 0 0 0 0 0 0	0,5
0 0 0 0 0 0 5 0 1 1 1 1 1 1 1 0 0 0	1
0 0 0 0 0 0 0 0 6 0 0 0 0 0 0 0 0 0	0
0 0 0 0 0 0 0 0 10 0 0 0 0 0 0 0 0 0	0
0 0 0 0 0 0 0 0 2 1 0 0 0 0 0 0 1 0 0	0,7
0 0 0 0 0 0 1 0 2 0 0 0 0 0 0 0 1 0 0	0,5
0 0 0 0 0 0 5 0 1 1 0 1 1 1 1 1 1 0 0	1
0 0 0 0 0 0 0 0 12 1 0 1 0 0 0 0 0 1 1	1
0 0 0 0 0 0 0 0 5 1 0 1 0 0 0 0 0 1 1	0,3
0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 1 1	1

1.8. Degradación de STMU

Entradas: 1598, 1010, 3984, 2693, 2693_CANCEL

Patrones de entrada	*Patrones de salida*
0 0 0 0 0	0
1 1 0 0 0	0,05
1 3 0 0 0	0,2
3 3 0 0 0	0,4

1.9. Degradación de servicio de datos BSC

Entradas: 3019, 3020, 3025, 3030, 3031, 3032, 3053, 3164, 3483, Tiempo_de_estudio

Patrones de estradas	*Patrones de salida*
0 0 0 0 0 0 0 0 0 2	0
1 1 1 1 1 1 1 1 1 2	0,1
2 2 2 2 2 2 2 2 2 2	0,4
2 2 2 2 2 2 2 2 2 1	0,6
3 3 3 3 3 3 3 3 3 1	1

1.10.Degradación de servicio de datos BTS

Entradas: 7412, 7401, 7414, falla_motogenerador, 7419, 7403, mayor_rectificación, 7704, Tiempo_de_estudio, 7421, 7402, 7405, bloqueo_de_NSEI, Configuración_celda_fallida, Duplexores, 7606, Diversdad, 7705, No_gprs_transation

Patrones de entrada	*Patrones de salida*
0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0	0
1 0 1 0 1 0 0 0 2 0 0 0 0 0 0 0 0 0	1
1 1 0 0 0 0 0 0 2 0 0 0 0 0 0 0 0 0	0,7
0 1 0 0 0 0 0 1 2 0 0 0 0 0 0 0 0 0	0,5
1 1 1 1 1 1 1 5 1 0 0 0 0 0 0 0 0 0	1
0 0 0 0 0 0 0 0 6 0 0 0 0 0 0 0 0 0	0
0 0 0 0 0 0 0 0 10 0 0 0 0 0 0 0 0 0	0
1 0 0 0 0 0 0 0 2 1 0 0 0 0 0 0 0 0	0,7
0 0 0 0 0 0 0 1 2 1 0 0 0 0 0 0 0 0	0,5

1 0 1 1 1 1 1 5 1 1 0 0 0 0 0 0 0 0	1
1 0 1 0 0 0 0 0 12 0 1 1 0 0 0 0 0 0	1
1 0 1 0 0 0 0 0 5 0 1 1 0 0 0 0 0 0	0,3
1 0 0 0 0 0 0 0 1 0 1 1 0 0 0 0 0 0	1
0 0 0 0 0 0 0 0 2 0 0 0 0 0 0 0 0 0	0
0 0 0 0 0 0 0 0 2 0 0 0 0 1 1 1 0 0 1	1
0 0 0 0 0 0 0 0 3 0 0 0 1 0 0 0 0 0	1
0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 1 0 0	0,7
0 0 0 0 0 0 0 0 1 0 0 0 0 1 0 0 0 1 0	0,25
0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 1 0 0 0	0,4
0 0 0 0 0 0 0 0 2 0 0 0 0 0 0 0 1 0 0	0,15
0 0 0 0 0 0 0 0 2 0 0 0 0 0 0 2 0 0 0	0,8
0 0 0 0 0 0 0 0 7 0 0 0 0 0 0 0 0 0 0	0
0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 1	1
0 0 0 0 0 0 0 0 1,5 0 0 0 1 0 0 0 0 1 0	1
0 0 0 0 0 0 0 0 1,5 0 0 0 0 0 0 0 0 1 1	1
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	0

1.11.Descenso de tráfico

Entradas: 1178, 1254, 1007, 1001, 3483, 0691, 2770

Patrones de entrada	*Patrones de salida*
1 0 1 1 1 1 0	1
0 0 0 0 0 0 0	0
0 0 0 0 1 0 0	0,1
0 0 1 1 4 0 0	0,4
1 0 0 0 0 0 0	0,2
0 0 0 0 0 0 1	1
1 1 1 1 1 1 1	0,8
0 1 0 0 0 0 1	0,8
0 0 1 2 10 0 0	0,6

1.12. Diagnóstico de conmutación

Entradas: 1001, 1007, 1178, 2692, 2693, 3164, 3483, Tiempo_de_estudio

Patrones de entrada:	*Patrones de salida:*
0 0 0 0 0 0 2	0
1 0 0 0 0 0 0 1	0,1
4 1 1 1 0 0 0 2	0,7

6 2 2 1 1 2 1 1 — 1
1 1 1 1 0 1 1 4 — 0,3

1.13. Fallo de BSC

Entradas: 2915, 2693, 2692, 3019, 2770, 3252, 2018, 2720, 2457, 3068, 2075, 2900, 3273, 3324, 2792, 2909, 2071, 1178, 1020, 1578, 1254, 2915_CANCEL, 0690, 3025, 3020, 3053, 3053_CANCEL, 3254, 2692_CANCEL, 3292, 2863, 3020_CANCEL, 3025_CANCEL, 3019_CANCEL, 3031, 3031_CANCEL, 3292_CANCEL, 2863_CANCEL, 3254_CANCEL, 1569, 1252, Fallo_de_BSC, 2923_CANCEL, 2909_CANCEL, 0691, 1583, 1072, 1582, 2720_CANCEL, 2399, 2399_CANCEL, 2770_CANCEL, 1585, 3273_CANCEL, 2693_CANCEL, 2792_CANCEL, 2900_CANCEL

Patrones de entrada	*Patrones de salida*
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	0
6 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 10 5 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	0,2
12 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 10 18 11 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	0,4
12 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 11 20 11 4 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	0,45
12 0 0 50 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 11 20 11 4 50 50 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	0,6
12 0 0 100 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 11 20 11 4 100 100 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	0,75
12 0 0 100 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 11 20 11 4 100 100 40 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	0,85

[ANEXO 2]

[CÓDIGO DE ALGUNOS MÓDULOS]

En esta sección veremos el código de algunos de los módulos;

2.1. Estructura de la Neurona artificial

```
TNeurona = Record
              Entradas:array[0..100] of Real;
              CantEntradas:integer;
              W:array[0..100]of Real;
              Salida:Real;
              Cesgo:single;
              delta:Real;
             Activación:Real;

  End;

TRed = Array[0..maxcapas,0..maxneur] of TNeurona;
```

2.3. Módulo de ejecución de la RNA

```
Procedure Red_Neuronal (capas,Nentradas:integer;Neurpcap:array of integré ;
          var entradas,Salida:array of Real;var Red:TRed);
var
i,j,k:integer;
begin
  //Colocación de las entradas en la red
  i:=1;
  for i := 1 to Neurpcap[1] do
  begin
   red[1,i].Entradas[0]:=red[1,i].Cesgo;
   for j := 1 to Nentradas do
      red[1,i].Entradas[j]:=Entradas[j];
  end;
  //Revisión de las capas i representa las capas
  for i:= 1 to capas do
     //Revision de las neuronas por capa representada por j
```

```
      for j := 1 to Neurpcap[i] do
      Begin
         red[i,j].Activación:=0;
         //Revisión de las entradas por cada neurona representada por K
         for k := 0 to Red[i,j].CantEntradas do

red[i,j].Activación:=red[i,j].Activación+(Red[i,j].W[k]*Red[i,j].Entradas[k]);
         //Función de activación lineal
         //red[i,j].Salida:=red[i,j].Activación;
         //Función de activación en la salida para sigm
           red[i,j].Salida:=sigm(Red[i,j].Activación);
         //Colocación de entradas en la siguiente capa
         if i<capas then
           for k := 1 to Neurpcap[i+1] do
              Red[i+1,k].Entradas[j]:=Red[i,j].Salida;
      End;
      for I := 1 to Neurpcap[capas] do
   Salida[i]:=Red[capas,i].Salida;
end;
```

2.4. Módulo Backpropagation

```
Procedure backpropagation(ap:Real;capas,Nentradas:integer;Neurpcap:array of
integer;
            entradas,Salida:array of Real;var Red:TRed; var errorc:Real);
var
i,j,k:integer;
salidaob:array[0..100]of Real;
error:array[0..100]of Real;
wn,suma:Real;
begin
  Red_Neuronal(capas,Nentradas,Neurpcap,entradas,salidaob,Red);
  for i := 0 to 100 do
    error[i]:=0;
  //Calculo del error al final de la red neuronal, de los sigmas y de ls pesos de la
última capa
  for i := 1 to Neurpcap[capas] do
  begin
     //Calculo de los errores
     error[i]:=Salida[i]-salidaob[i];
     //calculo de los deltas para función lineal
     //red[capas,i].delta:=error[i];
```

```
    //Para la función sigmuidea quedaría
      red[capas,i].delta:=error[i]*dsigm(red[capas,i].Activación);

    //Calculo de los pesos para cada dendrita en la última capa
     for j := 0 to red[capas,i].CantEntradas do
     begin
       wn:=ap*red[capas,i].delta*red[capas,i].Entradas[j];
       Red[capas,i].W[j]:=Red[capas,i].W[j]+wn;
     end;
  end;
  //Caluclo de los pesos para las capas anteriores a la última
  //Recorrido de las capas
  for i:=(capas-1) downto  1 do
  begin
    //Recorrido por las neuronas
    for j := 1 to neurpcap[i] do
      //Calculo de la sumatoria de la regresión de los pesos para cada neurona
      begin
        suma:=0;
        for k := 1 to neurpcap[i+1] do
          Suma:=Suma+red[i+1,k].delta*red[i+1,k].W[j];
        //Calculo de los delta de las neuronas para función lineal
        //red[i,j].delta:=Suma;
        //Para la función sigmuidea
          red[i,j].delta:=Suma*dsigm(red[i,j].Activación);

        //Calculo de los pesos para cada neurona
        for k := 0 to red[i,j].CantEntradas do
        begin
          wn:=ap*red[i,j].delta*red[i,j].Entradas[k];
          Red[i,j].W[k]:=Red[i,j].W[k]+wn;
        end;
      end;
  end;
  suma:=0;
  for i := 1 to Neurpcap[capas] do
     Suma:=suma+sqr(Error[i]);
  Errorc:=0.5*suma;
end;
```

2.5. Módulo que Realiza el Entrenamiento

```
var
```

```
i,j,k,cant:integer;
entrenamiento,suma,neta,cita,p:Real;
error:array[0..400] of Real;
red1:TRed;
Capasu:Integer;
Neurpcapu:array[0..100]of integer;
Nentradasu:integer;
begin
  cita:=0.04;
  neta:=1.05;
  p:=0.7;
  importar_patrones(config[cont+2],Patrones);
  importar_red(config[cont],capasu,nentradasu,red1,Neurpcapu);
  for i := 1 to Patrones[1].Cantidad_patrones do
  begin
     backpropagation(ap,capas,Nentradas,Neurpcap,Patrones[i].Entradas,
                              patrones[i].Salida,Red1,error[i]);
  end;
  suma:=0;
  for i := 1 to Patrones[1].Cantidad_patrones do
    Suma:=suma+error[i];
  if primeravez then
  begin
     if suma>(errorc2*(1+cita)) then
     begin
        ap:=ap*p;
     end;
     if (suma<=errorc2)and(ap<100) then
     begin
        exportar_red(config[cont],capasu,nentradasu,red1,Neurpcapu);
        ap:=ap*neta;
      end;
      if (suma>errorc2)and(suma<(errorc2*(1+cita))) then
      begin
         exportar_red(config[cont],capasu,nentradasu,red1,Neurpcapu);
      end;
   end;
  errorc2:=suma;
  primeravez:=true;
  entrenamiento:=(1/(1+suma))*100;
  exportar_red(config[cont],capas,nentradas,Red1,Neurpcap);
end;
```

[ANEXO 3]

[FORMATOS DISEÑADOS PARA EXPORTAR]

A continuación se muestran los formatos diseñados para exportar los diagnósticos; lo que incluye la RNA, las configuraciones y los patrones empleados para el entrenamiento de las mismas. También está el formato para el archivo que hace la unión entre los dos programas.

3.1. Formato de las RNA

Capas Nentradas
neurpcapas

W111 W112 W113 ... W11E
W121 W122 W123 ... W12E
.
.
.
W1N1 W1N2 W1N3 ... W1NE

W211 W212 W213 ... W21E
W221 W222 W223 ... W22E
.
.
.

W2N1 W2N2 W2N3 ... W2NE

.
.
.

WC11 WC12 WC13 ... WC1E

WC21 WC22 WC23 ... WC2E
.
.
.
WCN1 WCN2 WCN3 ... WCNE

W = Peso
C = Total de capas
N = Total de neuronas de la capa
E = Total de entradas de la neurona de la capa

3.2. Formato de las características de un estudio

Nentradas(E) Nsalidas(S) Nunidades(U)

Nombre_entrada_1
Nombre_entrada_2
Nombre_entrada_3
.
.
.
Nombre_entrada_E

Nombre_salida_1
Nombre_salida_2
Nombre_salida_3
.
.
.
Nombre_salida_S

Unidad1
Unidad2
Unidad3
.
.
.
UnidadU

3.3. Formato de los patrones

CantPatrones(N) CantEntradas(E) CantSalidas (S)

Entrada_1_1 Entrada_1_2 Entrada_1_3 ... Entrada_1_E
Entrada_2_1 Entrada_2_2 Entrada_2_3 ... Entrada_2_E
Entrada_3_1 Entrada_3_2 Entrada_3_3 ... Entrada_3_E
.
.
.
Entrada_N_1 Entrada_N_2 Entrada_N_3 ... Entrada_N_E

Salida_1_1 Salida_1_2 Salida_1_3 ... Salida_1_S
Salida_2_1 Salida_2_2 Salida_2_3 ... Salida_2_S
Salida_3_1 Salida_3_2 Salida_3_3 ... Salida_3_S
.
.
.
Salida_N_1 Salida_N_2 Salida_N_3 ... Salida_N_S