UNIVERSIDAD POLITÉCNICA SALESIANA

SEDE CUENCA

CARRERA DE INGENIERÍA DE SISTEMAS

Tesis previa a la obtención del Título de:

INGENIERO DE SISTEMAS.

TÍTULO:

"ESTUDIO DE LOS PRINCIPALES TIPOS DE REDES NEURONALES Y LAS HERRAMIENTAS PARA SU APLICACIÓN"

AUTOR:

EVA CRISTINA ANDRADE TEPÁN

DIRECTOR:

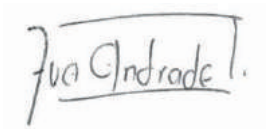
ING. VLADIMIR ROBLES BYKBAEV.

CUENCA, FEBRERO DE 2013.

DECLARATORIA DE RESPONSABILIDAD.

Los conceptos desarrollados, el estudio, análisis, aplicaciones y las conclusiones del presente trabajo, son de exclusiva responsabilidad de la autora. Los textos de otros autores llevan su correspondiente cita bibliográfica.

Cuenca, febrero de 2013.



Eva Cristina Andrade Tepán.

C.I.: 0105024566

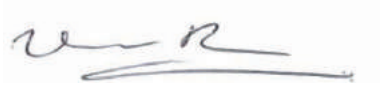
DERECHOS RESERVADOS © 2013, Universidad Politécnica Salesiana CUENCA-ECUADOR.

Ing. Vladimir Robles

CERTIFICA:

Que la tesis denominada: **"ESTUDIO DE LOS PRINCIPALES TIPOS DE REDES NEURONALES Y LAS HERRAMIENTAS PARA SU APLICACIÓN."** ha sido desarrollada en su totalidad por la Srta. Eva Cristina Andrade Tepán, he dirigido y revisado prolijamente cada uno de los capítulos del informe de tesis, y por cumplir los requisitos autorizo su presentación.

Atentamente:



Ing. Vladimir Robles.
DIRECTOR DE TESIS

DEDICATORIA

Dios en su bondad nos regala una familia y amigos con quienes aprendemos a lo largo de nuestra vida.

Mi trabajo de tesis lo dedico a toda mi familia; a mi mamá Josefina Tepán, mi amiga, mi apoyo, quien ha estado a mi lado en todo momento, en cada paso que nos ayuda avanzar y en cada caída de la cual se aprende.

Mi papi Vicente que a pesar de las circunstancias, lo quiero mucho.

Un ser que sin siquiera darse cuenta ha hecho tanto por mi, M.A.

Y toda mi vida, mis alegrías, experiencias y esfuerzos son por el amor de Dios, a Él va dedicado todo lo que me permite lograr.

AGRADECIMIENTO

Cada instante de vida lo compartimos con alguien, cada esfuerzo, cada derrota, las alegrías y tristezas; muchas gracias a todos ellos que han formado parte de ese caminar.

Gracias mami Josefina por permitirme contar siempre con usted; a mi valiente abuelita Dorinda, con sus grandes experiencias nos llena la vida. Gracias a mis tías queridas Cuca y Nora por su apoyo y amistad.

Solo pensar en él me hace sonreir. Mil gracias Ber Cueva por ser parte de mi vida, por tu apoyo y confianza siempre; gracias, porque has llenado mi alma de amor y felicidad, te amo.

Mis amigos queridos, compañeros de carrera y de la vida, gracias por todo lo compartido, aprendizajes que contribuyeron a nuestro crecimiento profesional y personal. Siempre los llevaré en mi corazón:

Berna León con quien compartí toda mi carrera universitaria, los logros y disgustos, gracias por tus brillantes ideas y consejos, por tu ayuda, aprecio y comprensión. Eres una gran persona.

Albilla, compañera de clase, confidente, consejera, cómplice de aventuras y locuras; mil gracias por todo lo vivido y aprendido.

A todos mis amigos Rosalba, Zuly, Edy, Diego, Patín, Carlines, Fabián, Dianita, Sole, Gaby y Carlitos; muchas gracias por su amistad.

Gracias Ingeniero Vladimir Robles, por la ayuda y paciencia brindada en este trabajo final.

La U, en ella conocí personas geniales con quienes viví grandes momentos; en cada espacio existe un recuerdo, una experiencia que forman parte de mi historia.

TABLA DE CONTENIDOS

ÍNDICE DE ILUSTRACIONES

ÍNDICE DE TABLAS

CAPÍTULO I

INTRODUCCIÓN

CAPÍTULO I

INTRODUCCIÓN

1.1 ANTECEDENTES

Existen problemas del mundo real: reconocimiento de formas, toma de decisiones, que no pueden ser descritos fácilmente mediante un "enfoque algorítmico tradicional"; para lograrlo se ha establecido "un nuevo campo de la computación que tiene su origen en la emulación de sistemas biológicos". La metodología que nos ayuda en este desarrollo son las redes neuronales, que buscan la resolución de problemas complejos a través de sistemas de computación inspirados en el cerebro humano [1].

Siempre se ha buscado diseñar y construir máquinas que tengan la capacidad de realizar procesos con "cierta inteligencia"; y a pesar de que existen las herramientas y lenguajes de programación necesarios; no se ha logrado dicho objetivo, pues hay un problema de fondo: "estas máquinas se apoyan en una descripción secuencial del proceso de tratamiento de la información" [1].

Las redes neuronales artificiales están inspiradas en la estructura y funcionamiento de los sistemas nerviosos, donde la neurona es el elemento principal.

Se dará una explicación del modelo biológico, pues es la base de todo el estudio de las redes neuronales artificiales.

Modelo Biológico.

Como se menciona en [2], una neurona tiene tres partes principales:

- **Ramas de Extensión o Dendritas:** Reciben estímulos de entrada.
- **Cuerpo de la Neurona**: Procesa estímulos de entrada.
- **Axón:** Emite estímulos de salida a las dendritas de otras neuronas.

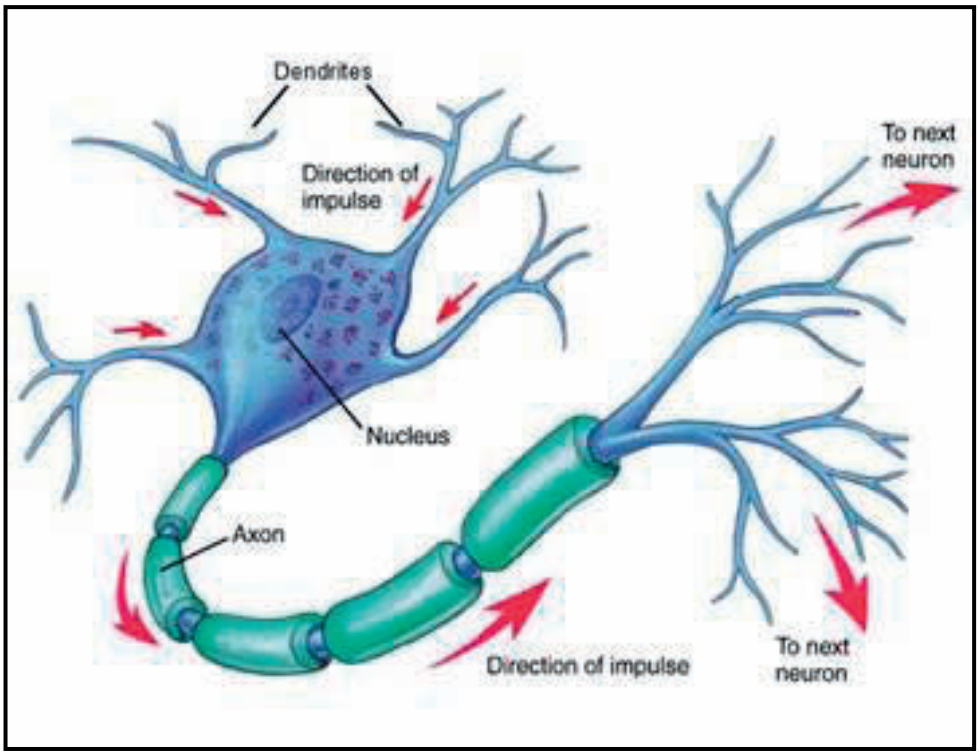


Ilustración 1. 1: Representación de una neurona biológica [11].

Una neurona recibe estímulos de entrada mediante las dendritas, estos estímulos son procesados en el cuerpo, para posteriormente emitir un estímulo de salida a través del axón [2].

La neurona utiliza dos tipos de señales: las que se generan y transportan a través del axón son impulsos eléctricos. La señal transmitida entre los terminales axónicos y las dendritas de otra neurona es de origen químico. Esta conexión entre el axón de una neurona y las dendritas de otra se llama Sinapsis; y se da gracias a que existen dos tipos de neuronas: aquella que suministra el impulso se llama presináptica; y las que reciben el impulso son conocidas como postsinápticas.

Todas las neuronas siguen un proceso similar para conducir la información, "ésta viaja a lo largo de los axones en breves impulsos eléctricos, denominados potenciales de acción, que alcanzan una amplitud máxima de unos 100mv y duran 1ms". La neurona que se encuentra en reposo "mantiene un potencial eléctrico de -70mv".

Los potenciales de acción no pueden saltar de una neurona a otra; para que sea posible la comunicación entre neuronas se necesita transmisores químicos que son liberados en la sinapsis. Cuando un potencial de acción llega al terminal de un axón

se libera transmisores que son alojados en una hendidura muy pequeña que separa la célula presináptica de la postsináptica; durante este proceso son liberados también neurotransmisores "que se enlazan con receptores postsinápticos", lo que da origen a la comunicación entre dos neuronas.

Las redes neuronales artificiales tratan de imitar la funcionabilidad de un cerebro biológico, aunque el sistema artificial no alcanza la complejidad del mismo.

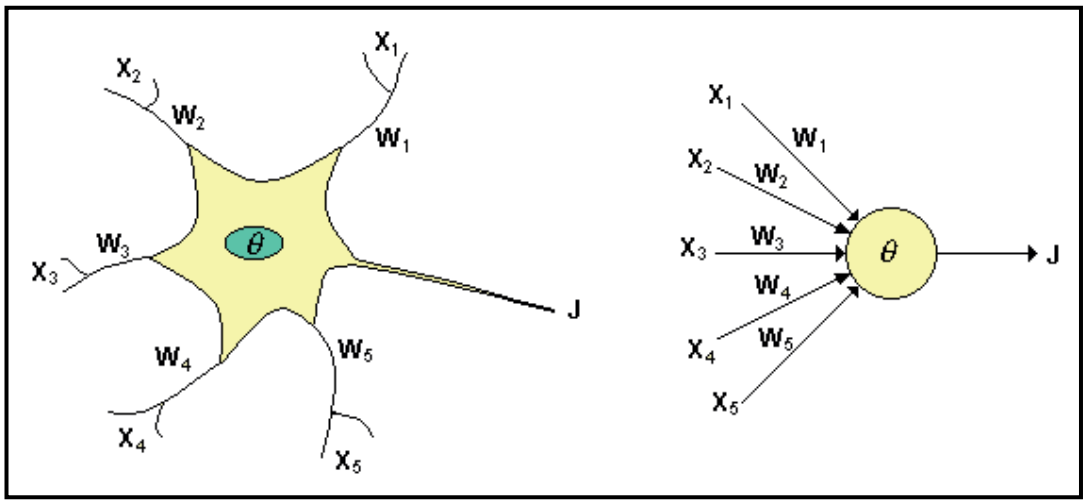


Ilustración 1. 2: Representación de la neurona biológica a la neurona artificial [3].

Como observamos en la ilustración existen algunas analogías entre las redes neuronales artificiales y las redes neuronales biológicas, como explica [3], estas son:

- Las entradas Xi representan las señales que provienen de otras neuronas y que son capturadas por las dendritas.
- Los pesos Wi son la intensidad de la sinapsis que conecta dos neuronas; tanto Xi como Wi son valores reales.
- ϑ es la función umbral que la neurona debe sobrepasar para activarse; este proceso ocurre biológicamente en el cuerpo de la célula.
- Las señales de entrada a una neurona artificial X1, X2,.., Xn son variables continuas.
- Cada señal de entrada pasa a través de una ganancia o peso.
- Los pesos pueden ser positivos (excitatorios), o negativos (inhibitorios).

- El nodo sumatorio acumula todas las señales de entradas multiplicadas por los pesos o ponderadas y las transfiere a la salida a través de una función umbral o función de transferencia.

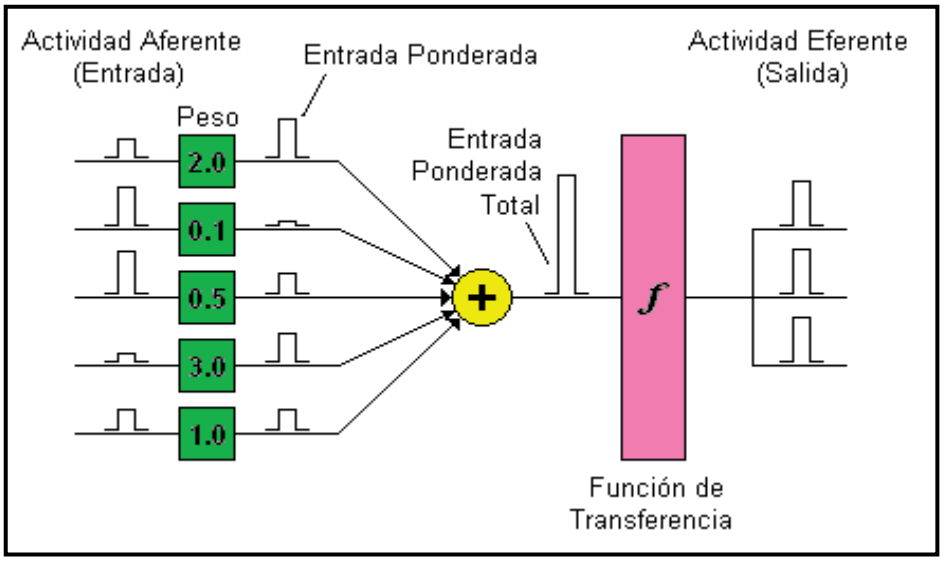


Ilustración 1. 3: Proceso de una red neuronal artificial [3].

1.2 OBJETIVOS

a. Objetivo General

Realizar un estudio de los principales tipos de Redes Neuronales que existen, y las herramientas necesarias para su respectiva aplicación.

b. Objetivos Específicos

- ✓ Conocer las características, aplicaciones, ventajas y desventajas de una red neuronal artificial.
- ✓ Analizar el proceso de una red neuronal por tipo de aprendizaje.
- ✓ Analizar las herramientas para la implementación de una red neuronal artificial.

- ✓ Revisar y preparar un corpus que servirá para realizar la aplicación de cada tipo de red.

- ✓ Seleccionar las redes neuronales y las herramientas adecuadas para realizar la experimentación.

- ✓ Probar cada tipo de red neuronal, en la herramienta seleccionada.

1.3 INTRODUCCIÓN A LA INTELIGENCIA ARTIFICIAL

La Inteligencia Artificial permite simular los procesos del pensamiento y las acciones del hombre, mediante procedimientos que logran un adecuado comportamiento y una toma de decisiones de manera precisa e inmediata.

Los principales procesos que abarca la Inteligencia Artificial, como se menciona en [4], son:

- Ejecución de una respuesta predeterminada por cada entrada (análogas a actos reflejos en seres vivos).
- Búsqueda del estado requerido en el conjunto de los estados producidos por las acciones posibles.
- Algoritmos genéticos (análogo al proceso de evolución de las cadenas de ADN).
- Redes neuronales artificiales (análogo al funcionamiento físico del cerebro de animales y humanos).
- Razonamiento mediante una lógica formal (análogo al pensamiento abstracto humano).

El principal esfuerzo de la Inteligencia Artificial radica en la posibilidad de imitar las diferentes capacidades del ser humano, y así exhibir un comportamiento inteligente, propio del hombre; para demostrarlo se han desarrollado sistemas que logran en cierta parte realizar estas actividades. La I.A., abarca distintos estudios: como hacer que las computadoras hagan cosas que la gente hace mejor hoy en día, tratar de que las computadoras piensen, imitar las acciones de las personas, el estudio de las

facultades mentales; es decir, la inteligencia artificial trata de construir "sistemas que actúan como humanos, que piensan como humanos, que actúan racionalmente, y que piensan racionalmente" [6].

El comportamiento inteligente es muy amplio, para lograr un desarrollo óptimo, se han clasificado como sub áreas de la Inteligencia Artificial, que se menciona en [5]:

- Procesamiento de lenguaje natural
- Visión artificial
- Resolución de problemas
- Representación del conocimiento y razonamiento
- Aprendizaje
- Robótica

Los primeros intentos de la I.A., se fundamentaron en la creación de programas que tuvieran la capacidad de realizar únicamente actividades propias de la mente humana; es decir se buscaba crear un cerebro artificial; sin embargo, como se puede observar hoy en día las máquinas inteligentes ya existen, y se busca avanzar cada vez más en este proceso; "según Moravec 'las máquinas estarán en condiciones de alcanzar el nivel de la inteligencia humana', teniendo en cuenta que el cerebro humano trabaja a una frecuencia de 100 hertzios, y que no se vislumbra una cota que limite la velocidad de procesamiento de la computadora". Las máquinas van a superar al hombre por su rapidez al momento de pensar [7].

Existen personas que se mantienen a favor de las máquinas inteligentes, creen que serán estas máquinas las que construyan una nueva generación de computadoras con mayores capacidades, y así vaya aumentando la inteligencia en sistemas no biológicos; sin que el hombre tenga participación en este proceso; entonces la pregunta es ¿no sería el fin de la humanidad?

Ellos se basan en la evolución de los animales y la comparan con la evolución de las máquinas, "ya que los humanos fueron el producto de una larga evolución desde los organismos unicelulares pasando por los mamíferos hasta llegar al homo sapiens. ¿A partir de qué momento surgió la inteligencia? ¿Por qué no aceptar que las máquinas también son organismos en evolución que puedan llegar a 'pensar'?"[7].

1.4 BREVE REVISIÓN DEL ESTADO DEL ARTE ACTUAL DE LA I.A.

Muchos científicos creen que en un futuro no muy lejano la inteligencia artificial podría alcanzar los niveles del ser humano, así el científico José Cordeiro investigador de Singularity University en Sillocon Valley, indicó: "es posible que suceda entre el 2029 y el 2045, explica que su teoría se basa en la tendencia histórica de que cada dos años se duplica el número de transistores por computadora"; una máquina superará al cerebro humano por su velocidad [8].

Actualmente tenemos máquinas inteligentes capaces de realizar actividades propias del hombre; ASIMO, desarrollado por la empresa de origen japonés Honda; es un robot de baja estatura (1.30 mts), que puede realizar varias acciones como se indica en [9]:

> *"Saludar o patear un balón de fútbol, puede caminar sobre una superficie irregular sin perder estabilidad, correr hacia atrás, saltar en una o en ambas piernas en repetidas ocasiones. El robot viene equipado con una tecnología de control autónoma única en el mundo, que le permite caminar a una velocidad máxima de 2,7 Km/h, y correr a 9 Km/h".*

Ilustración 1. 4: ASIMO, el robot más avanzado creado por Honda [12].

Asimo tiene instalado sensores de espacio, que le permite determinar con rapidez una ruta alternativa para evitar choques, puede identificar múltiples rostros y voces debido a la coordinación entre los sensores visuales y auditivos que lo componen [9].

Utiliza reconocimiento de imágenes y realiza un análisis de sonido; obedece órdenes, y puede distinguir una persona que desea café caliente, otra jugo de naranja y otra, té; es un robot que puede ayudar a la gente, a personas minusválidas. Asimo es un avance muy significativo dentro del área de inteligencia artificial, y ha sido tan bien recibido que ya tiene trabajo, es recepcionista en el Museo Nacional de las Ciencias Emergentes, guía a las personas en su recorrido.

El caminar es una tarea sencilla para los humanos, sin embargo lograrlo en un robot es algo más complicado; los ingenieros de Honda dedicaron gran parte de su tiempo en estudiar la forma de andar de las personas, "el caminar no es más que una serie de caídas controladas, interrumpidas antes que el cuerpo se incline lo suficiente para hacernos caer". Cada paso es una caída incompleta. De esta manera se logró que Asimo camine sobre dos piernas y además realice giros mientras camina, exactamente como lo hace una persona [10].

> *"Al momento de efectuar un giro, el robot de Honda mueve la posición de su centro de gravedad, mediante un algoritmo llamado "predictive movement control", conocida también como "Honda's Intelligent Walking Technology". Este algoritmo permite a ASIMO predecir donde deberá situar su tronco en cada paso, en base a su inercia, longitud de sus pasos, ángulo del giro, etc"* [10].

Asimo tiene en su cabeza cámaras y "un algoritmo propiedad de Honda que le permite interpretar, reconocer objetos y moverse entre ellos, aun si su orientación o iluminación no es exactamente la misma que tiene almacenada en su base de datos". Este algoritmo también le permite reconocer caras de las personas que le son familiares [10].

No es un robot totalmente autónomo, no puede tomar decisiones en un entorno que desconoce, no sabe como desenvolverse en un medio, si antes no ha sido programado cada detalle del ambiente en el cual se encuentra; sin embargo lo que se ha logrado es la base para lo que en un futuro nos podría esperar, robots autómatas en un 100%.

Reconocimiento automático del habla

Cuando hablamos de reconocimiento de voz, se debe mencionar a Watson; la supercomputadora creada por IBM que cuenta con un potencial técnico grandioso, es capaz de entender el lenguaje hablado, incluso juegos de palabras, sarcasmos, y responder preguntas complejas al instante, relacionándolas y sacando sus propias conclusiones; es capaz de aprender de la experiencia, y una vez que ha sido entrenada sigue aprendiendo mientras trabaja, y cada vez sabe más. En su desarrollo se han empleado algoritmos que le permitan a Watson interpretar y comprender las preguntas, haciendo posible que procese los datos y responda de manera precisa. El potencial de esta máquina ha sido puesto a prueba hasta ahora, en un concurso de preguntas y respuestas de un programa de televisión, donde ganó a los dos campeones históricos del concurso [13] [15].

La computadora cuenta con un potencial enorme, por lo que sus creadores la han puesto a trabajar en campos donde puede ayudar a mejorar la vida de las personas; y es así como Watson empezó con su formación, IBM está construyendo una base de conocimientos dirigida a la lucha contra el cáncer de mama y de colon. Los ingenieros ingresan datos de este tipo en la computadora, la cual "aprende de la misma manera que un niño" a través de la lectura y las preguntas respondidas por humanos expertos. Es un proceso cognitivo[1] que busca que el sistema pueda comprender además de aprender. El equipo recibe los registros, información y pruebas de los pacientes de los últimos cuarenta años mediante un diálogo en lenguaje natural, Watson les brinda un diagnóstico con un porcentaje exacto de acierto, sugiere pruebas y explica su razonamiento [13][15].

Watson es capaz de discernir entre los datos que posee, debido a su diseño que imita los patrones de pensamiento humano; entiende y razona la pregunta para dar una respuesta más exacta. El objetivo es que en los próximos años, Watson pueda ser una herramienta y apoyo para los médicos en el tratamiento de enfermedades. La aplicación de este equipo en un futuro supone un gran avance en la lucha contra enfermedades como el cáncer de mama y de colon [13].

[1] Relacionado al conocimiento, que es un conjunto de información que se dispone gracias a un proceso de aprendizaje o a la experiencia [14].

La tecnología detrás de Watson es conocida como DeepQA, cuenta con 2 racks de servidores IBM Power 7 con un total de 2500 núcleos, que procesan los contenidos, extraen la información y brinda una respuesta en menos de 3 segundos [16].

Agentes Inteligentes

Un agente inteligente[2], actúa en distintos medios sin la intervención humana o ayuda de otros sistemas externos, dando respuestas al instante. Se menciona por ejemplo, un agente que intervenga en problemas de seguridad en sistemas bajo posible ataque, éste puede determinar qué tipo de incidente está ocurriendo basándose en los síntomas que presenta, y así proporcionará alertas, soluciones inmediatas de forma automática, para solucionar inconvenientes [19].

Hoy en día son muchas las organizaciones que utilizan un agente inteligente, pues facilitan la gestión de incidentes, de diversos tipos, garantizando de forma razonable, eficacia y eficiencia [19].

Visión por computador

La visión por computador tiene la finalidad de interpretar escenarios o ambientes mediante imágenes obtenidas con cámaras; esta interpretación se logra usando las prestaciones y la potencia de procesamiento de un computador [18].

En la actualidad existen diversas aplicaciones que utilizan visión por computador para llevar a cabo tareas: dentro de la industria farmacéutica, química, automovilística; se requiere que ciertos procesos sean automatizados, y se ha logrado mediante el uso de visión por computador, con un alto porcentaje de éxito.

La interpretación visual de las acciones humanas es utilizada en sistemas de video vigilancia, control gestual de robots. El sistema que se implementa localiza imágenes principales de la persona como cabeza, codos, manos dentro de un ambiente natural, luego utiliza técnicas de segmentación por movimiento y creación de un modelo cilíndrico del cuerpo superpuesto en la imagen, aunque sigue abierto a investigaciones, se ha usado con éxito en el guiado gestual de robots.

[2] Sistemas de computación capaces de actuar de manera autónoma y racional, es de cir de manera correcta en un ambiente dinámico [17].

CAPÍTULO II

INTELIGENCIA ARTIFICIAL. REDES NEURONALES

CAPÍTULO II

INTELIGENCIA ARTIFICIAL. REDES NEURONALES

2.1 HISTORIA DE LAS REDES NEURONALES

A lo largo de la historia se ha tratado de construir máquinas que puedan realizar tareas con cierta inteligencia; su funcionamiento se ha basado en distintos procesos que realiza el ser humano, por ejemplo las redes neuronales artificiales tratan de emular a la neurona biológica.

El primero en estudiar el cerebro como una forma de ver el mundo de la computación fue Alan Turing en el año 1936 [20].

En 1943, Warren McCulloch, un neurofisiólogo, y Walter Pitts, un matemático, dieron los primeros fundamentos de la computación neuronal, explicaron la posible forma de trabajar de las neuronas y modelaron una red neuronal simple mediante circuitos eléctricos [20].

Donal Hebb en 1949 fue el primero en explicar los procesos del aprendizaje, desde un punto de vista psicológico, desarrollando una regla de como el aprendizaje ocurría. Estos trabajos formaron las bases de la teoría de redes neuronales artificiales [20].

Entre los años 1950 y 1956, karl Lashley la información no era almacenada en forma centralizada en el cerebro, sino que era distribuida encima de él. El Congreso de Dartmouth se conoce como el inicio de la inteligencia artificial [20].

Frank Rosenblatt en 1957, comenzó el desarrollo del Perceptrón, la red neuronal más antigua que se conoce, es usado actualmente en el reconocimiento de patrones [20].

En 1960, Bernard Widrow y Marcial Hoff, desarrollaron la red neuronal ADALINE, (Adaptative Linear Elements), el primer modelo que fue utilizado para resolver un problema real: filtros adaptativos para eliminar ecos en las líneas telefónicas [20].

Stephen Grossberg, en 1967, desarrolló la red neuronal Avalancha, que se utilizó para actividades como reconocimiento continuo del habla [20].

Marvin Minsky y Seymour Papert en 1967, demostraron que el Perceptrón era una red muy débil pues no podía resolver problemas sencillos como el aprendizaje de una función no-lineal. Sin embargo los estudios sobre redes neuronales artificiales continuaron, como es el caso de James Anderson que desarrolló un modelo lineal llamado Asociador Lineal [20].

En 1974 Paul Werbos, desarrolló la idea básica del algoritmo Backpropagation; sin embargo su estudio quedó totalmente claro en 1985 [20].

Stephen Grossberg en 1977, desarrolló la teoría de resonancia adaptada, es una arquitectura diferente que simula otras habilidades del cerebro como memoria a largo y corto plazo. En este mismo año Teuvo Kohonen desarrolló un modelo similar al de Anderson, pero de manera independiente [20].

En 1980 Kunihiko Fukushima, desarrolló un modelo neuronal para el reconocimiento de patrones visuales.

Desde 1985, el panorama mejoró en cuanto a la investigación y desarrollo de una red; John Hopfield con su libro "Computación neuronal de decisiones en problemas de optimización", dio paso al renacimiento de las redes neuronales. Un año más tarde, 1986; David Rumelhart, G. Hinton redescubrieron el algoritmo de aprendizaje backpropagation [20].

Las investigaciones y el desarrollo de redes neuronales artificiales son muy amplios, cada vez se obtienen nuevos trabajos y se publican numerosos estudios acerca de las redes.

Algunos de los científicos que iniciaron con el estudio de una RNA, continúan aún, como es el caso de Grossberg que trabaja en compañía de Carpenter en la Universidad de Boston; de igual forma Teuvo Kohonen en la Universidad de Helsinki [21].

Se han creado grupos de investigación de las redes neuronales artificiales, entre los más grandes están: PDP (Parallel Distibuted Processing), formado por Rumelhart, McClelland y Hinton. Rumelhart de la Universidad de Stanford es uno de los principales impulsores de la famosa red neuronal Backpropagation; McClelland con su grupo de investigación en la Universidad de Carnegie-Mellon destacan por el estudio de las posibles aplicaciones de la red Backpropagation; Hinton y Sejnowski en la Universidad de Toronto han desarrollado la máquina Boltzmann que se basa en la red de Hopfield. Otros de los grupos de mayor investigación son: California Institute of Technology, Massachussets Institute of Technology, University of California Berkeley y University of California San Diego [21].

2.2 CARACTERÍSTICAS DE UNA RED NEURONAL.

La Neurona Artificial

Una RNA está compuesta por un conjunto de neuronas artificiales, dispositivos simples de cálculo que a partir de un vector de entrada, ya sea del mundo exterior o bien a partir de estímulos recibidos de otras neuronas generan una respuesta única. Se puede distinguir tres tipos de neuronas, como se menciona en [54]:

Neuronas de entrada: reciben señales del entorno, ya sea de otras partes del sistema o de sensores.

Neuronas de salida: emiten una salida fuera del sistema una vez que ha finalizado el tratamiento de la información.

Neuronas ocultas: reciben estímulos y emiten salidas dentro del sistema, es decir no tienen ningún contacto con el exterior; son las encargadas de realizar el procesamiento de la información.

Estado de activación

Es necesario conocer los estados del sistema en un tiempo t, esto se especifica mediante un vector de N números reales A(t), que indica el estado de activación del conjunto de neuronas. Cada elemento del vector representa el estado de activación de una unidad en un tiempo t, es decir $a_i(t)$; entonces, como describe [56] tenemos:

$$A(t) = (a_1(t), a_2(t), \ldots., a_i(t), \ldots., a_N(t))$$

Ecuación 2. 1

Todas las neuronas que conforman la red se encuentran en un estado: excitado o de reposo, llamados estados de activación, que han sido asignados un valor discreto o continuo [56].

Entradas a la neurona

Las variables del exterior que se presenten a la neurona de entrada pueden ser de distinto tipo, dependiendo del tipo de red y la tarea que se vaya a realizar; como se menciona en [54]:

Binarias: Cuando tienen dos valores, por ejemplo la variable sexo que toma los valores hombre o mujer.

Continuas: Cuando la variable toma valores en un intervalo numérico; por ejemplo la edad.

Las neuronas que se encuentran después de la capa de entrada reciben como inputs las salidas que generan las capas previas con un valor de peso que indica su importancia; estas salidas pueden ser también binarias o continuas. Cada conexión entre la neurona i y la neurona j es llamada sinapsis y está ponderada por un peso w_{ij} [54].

Función de propagación

La función de propagación nos indica el procedimiento que se debe seguir para combinar los valores de entrada y los pesos de las conexiones que llegan a una neurona. Todos los pesos w_{ij} se suelen agrupar en una matriz W, indicando la influencia que tiene la neurona i sobre la neurona j; este conjunto de pesos puede ser positivo, negativo o nulo, como nos indica [54]:

W_{ij} **Positivo:** La interacción entre las neuronas i y j es excitadora, es decir cuando la neurona i esté activa emitirá una señal a la neurona j que tienda a excitarla.

W_{ij} **Negativo:** Entonces la sinapsis o conexión entre la neurona i y j es inhibitoria, es decir si la neurona i está activa emitirá una señal a la neurona j que la desactivará.

W_{ij} **Nulo:** Cuando $W_{ij} = 0$, entonces se considera que no existe conexión entre ambas neuronas.

La función de propagación permite obtener el valor de potencial postsináptico Net_j de una neurona en un momento t, de acuerdo con una función σ; el valor Net_j se calcula en base a los valores de entrada y pesos recibidos. La función más utilizada es de tipo lineal y consiste como se indica en [54]: la suma ponderada de las entradas con los pesos sinápticos a ellas asociados,

$$Net_j(t) = \sum_i w_{ij} * x_i(t)$$

Ecuación 2. 2

Existe otra regla de propagación que también se utiliza, la distancia euclídea:

$$Net_j(t) = \sqrt{\sum_i \left(w_{ij} - x_i\right)^2}$$

Ecuación 2. 3

La entrada neta que recibe una neurona o el potencial postsináptico Net_j es la suma ponderada de las entradas y un valor de umbral θ_j, en caso de existir; de tal forma que:

$$Net_j = \sum_{i=1}^{N} w_{ij}(t) * x_i(t) + \theta_j(t)$$

Ecuación 2. 4

Ahora, el umbral puede ser representado por un valor $x_0 = 1$, con un peso asociado w que determina el signo positivo o negativo y su fuerza; entonces tendríamos:

$$Net_j = \sum_{i=0}^{N} w_{ij}(t) * x_i(t)$$

Ecuación 2. 5

Función de Salida o de Transferencia

Existe un conjunto de conexiones que unen las neuronas que componen la red, cada neurona emite señales a aquellas que están conectadas con su salida. Una neurona tiene asociada una función de salida $f_i(a_i(t))$; por lo que el vector que contiene todas las salidas de las neuronas en un instante t sería, como nos indica [56]:

$$Y(t) = (f_1(a_1(t)), f_2(a_2(t)), \ldots, f_i(a_i(t)), \ldots\ldots, f_N(a_N(t)))$$

Ecuación 2. 6

En algunos casos, la función de salida o de transferencia es igual al nivel de activación de la neurona, por lo que

$$f_i(a_i(t)) = a(t)$$

Ecuación 2. 7

Existen cuatro funciones de transferencia, como se describe en [56] son:

Función Identidad: o función lineal es equivalente a no aplicar función de salida y es muy poco utilizada. La salida es igual a su entrada

Función Escalón: (hardlim, hardlims) o umbral, es utilizada únicamente cuando las salidas de la red son binarias. La salida de la neurona se activa solo cuando el estado de activación de dicha neurona sobrepasa o es igual a un valor umbral. La función puede ser desplazada sobre los ejes. Esta función crea neuronas que clasifican entradas en dos categorías distintas.

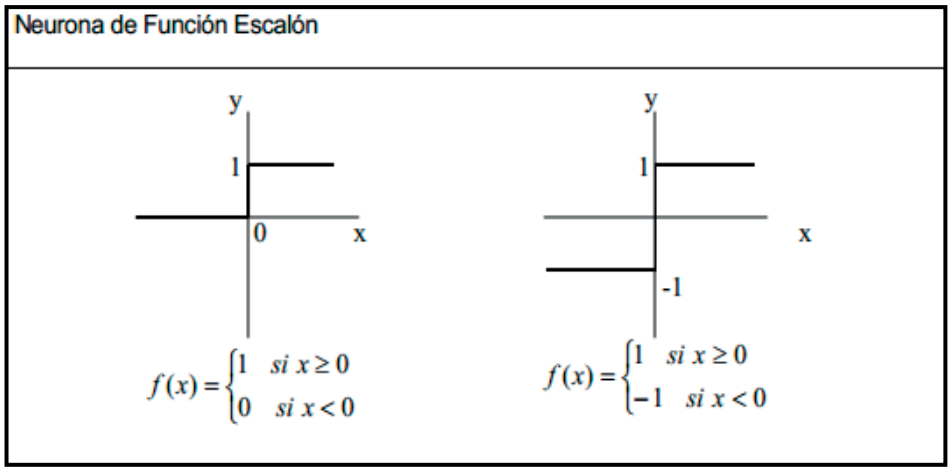


Ilustración 2. 1: En ambos casos se ha tomado que el umbral es cero; en caso de no serlo, el escalón quedaría desplazado [56].

Función lineal y mixta: esta función al igual que la sigmoidal es la más apropiada cuando se requiere como salida información analógica.

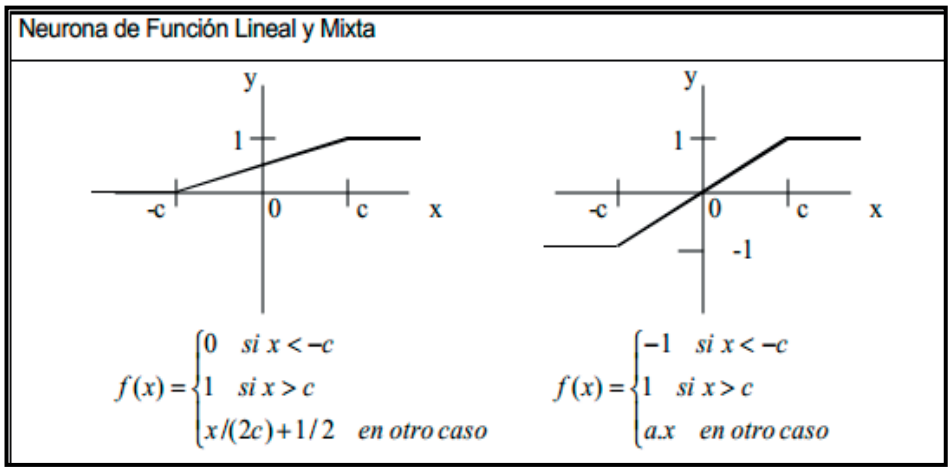


Ilustración 2. 2: c= límite superior de la suma de todas las entradas de activación

-c= límite inferior [56].

Función Sigmoidal: Cuando se requiere una salida de información analógica, esta función es la más apropiada. Los valores de entrada pueden variar entre más y menos infinito, y devuelve como salida valores entre 0 y 1; se frecuentemente en redes multicapa como la Backpropagation

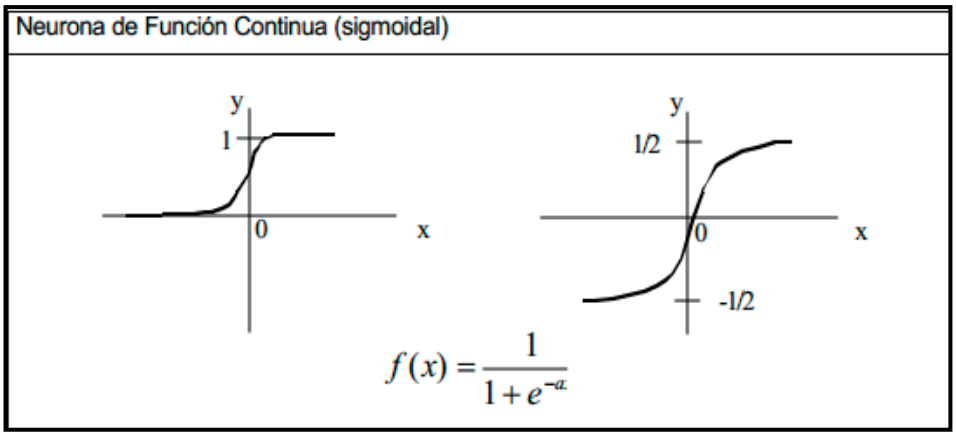


Ilustración 2. 3: Cuando x=0 toma su valor máximo, siendo muy útil para definir métodos de aprendizaje en los cuales se usan derivadas [56].

Función Gaussiana: Existen mapeos que requieren dos niveles ocultos con neuronas de transferencia sigmoidales; en algunos casos este tipo de función Gaussiana nos permite realizar mapeos con un solo nivel de neuronas.

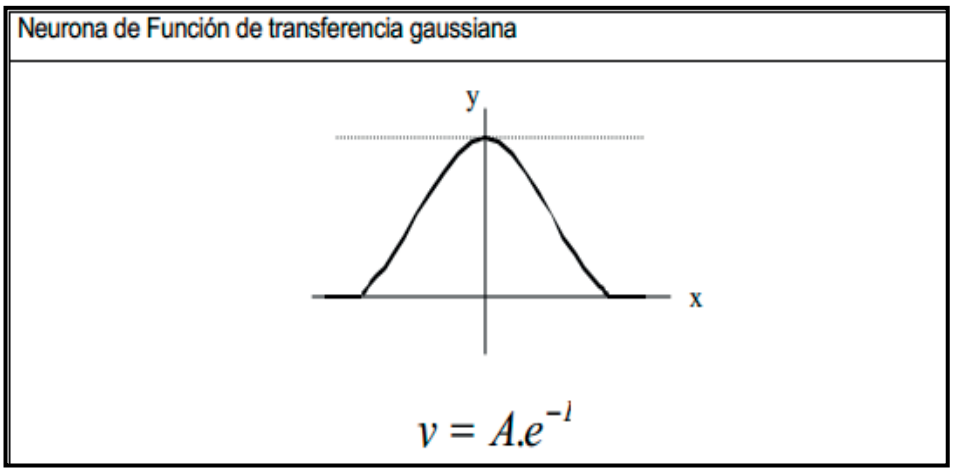


Ilustración 2. 4: Los centros y anchura de estas funciones pueden ser adaptados, lo cual las hace mas adaptativas que las funciones sigmoidales [56].

Función o regla de Activación

Es necesario que exista una regla que combine las entradas con el estado actual de la neurona, para producir en nuevo estado de activación. La función de activación F, produce un nuevo estado a partir del estado que ya existía a_j y la combinación de las entradas con los pesos de las conexiones Net_j, para determinar el nuevo estado de activación $a_j(t+1)$ de la neurona teniendo en cuenta el estado actual $a_j(t)$ y la entrada total que se calculó, se aplica la función de activación F [54] [55] [56].

$$a_j(t+1) = Fa_j(t), Net_j$$

Ecuación 2. 8

En algunos casos, F es la función identidad, por lo que el estado de activación de una neurona en t+1 coincide con el Net de la misma neurona en un tiempo t. A continuación se describe la fórmula para la salida de una neurona i, como se menciona en [56]:

$$y_i(t+1) = f(Net_i) = f(\sum_{j=1}^{N} w_{ij} y_j(t))$$

Ecuación 2. 9

Se debe tener en cuenta el valor umbral de activación de la neurona i θ_i, que no es igual en todas ellas.

$$y_i(t+1) = f(Net_i - \theta_i) = f(\sum_{j=1}^{N} w_{ij} y_j(t) - \theta_i)$$

Ecuación 2. 10

Características de una Red Neuronal Artificial

Las redes neuronales artificiales se caracterizan de acuerdo a 4 aspectos principales: topología, el mecanismo de aprendizaje, tipo de asociación realizada entre la información de entrada y salida, y la forma de representación de esta información [22].

Topología.- Hace referencia a la organización y disposición de las neuronas en red, formando agrupaciones llamadas capas. Los parámetros fundamentales son: "el número de capas, el número de neuronas por capa, el grado de conectividad y el tipo de conexiones entre neuronas". Este último se utiliza para conocer si las redes son de propagación hacia adelante, Feedforward; o hacia atrás, Backpropagation; el número de capas permite saber si son mono capa o multicapa [22].

Mecanismo de aprendizaje.- "El aprendizaje es el proceso por el cual una red neuronal modifica sus pesos en respuesta a una información de entrada." Durante

este proceso los pesos de las conexiones de la red se modifican, cuando estos permanecen estables quiere decir que la red aprendió [22].

Existen diferentes mecanismos de aprendizaje que le permiten a la red ir modificando sus pesos de acuerdo a una salida deseada; o interpretar de diferente manera las salidas que la red genere.

Tipo de asociación realizada entre la información de entrada y salida.- La asociación entre la información de entrada y salida se refiere a los datos que la red aprende, y asocia las entradas con una salida correspondiente.

Existen dos formas de realizar esta asociación:

Heteroasociación.- "La red aprende parejas de datos [(A1, B1), (A2, B2)… (An, Bn)], cuando se presenta una entrada Ai, la red deberá responder generando la correspondiente salida Bi" [22].

Auto asociación.- "La red aprende ciertas informaciones A1, A2…An, de tal manera que cuando se le presenta una información de entrada realizará una auto correlación, respondiendo con uno de los datos almacenados, el más parecido al dato de la entrada" [22].

Forma de representación de la información.- Los datos de entrada y salida de una red neuronal pueden ser representados de maneras distintas: "pueden ser analógicos, cuando esto ocurre, las funciones de activación de las neuronas son continuas, de tipo lineal o sigmoidal". Otras redes tienen como datos de entrada valores discretos, entonces la función de activación son de tipo escalón. También existen redes híbridas, donde las entradas son continuas y las salidas discretas [22].

2.3 TIPOS DE REDES NEURONALES

Existen diferentes tipos de redes neuronales, a continuación se describen las principales:

Perceptrón.- Es la red más antigua que se conoce, fue desarrollada en 1943. Este tipo de red consiste en sumar las señales de entrada y multiplicar por los valores de pesos escogidos aleatoriamente; este valor es comparado con un patrón para

determinar si la neurona es activada o no, si el valor comparado es mayor, la salida es 1, caso contrario es 0.

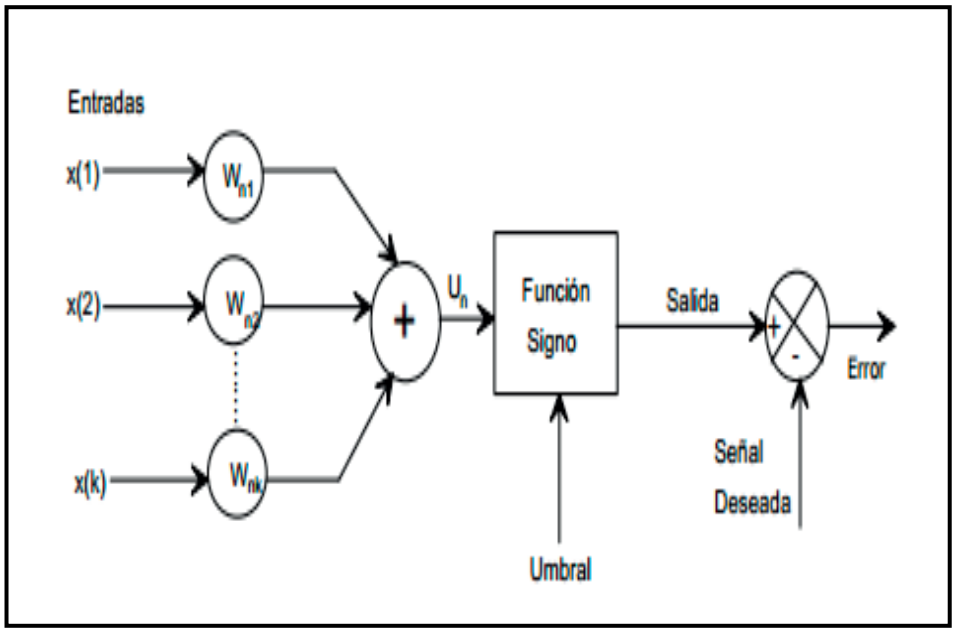


Ilustración 2. 5: Estructura de la red Perceptrón simple [58].

La única neurona de salida del Perceptrón: "realiza la suma ponderada de las entradas, resta el umbral y pasa el resultado a una función de transferencia de tipo signo". Esta red utiliza aprendizaje supervisado, es decir necesita conocer los valores esperados para cada entrada que se presenta a la red [23].

Esta estructura es usada en problemas de clasificación y se obtendrá resultados perfectos si los patrones son linealmente separables, como podemos observar en la ilustración.

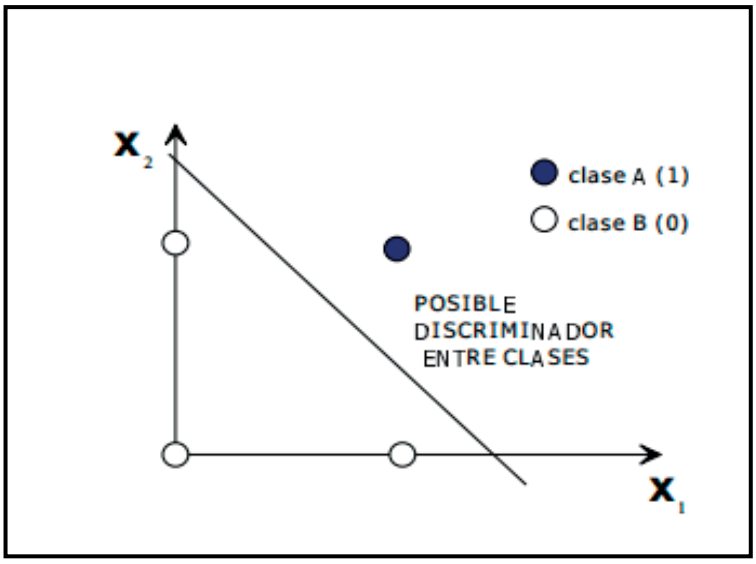


Ilustración 2. 6: Patrones linealmente separables: la recta deja a cada lado los elementos de diferentes clases [58].

Limitaciones del Perceptrón

Si los patrones no son linealmente separables, el algoritmo del Perceptrón no podrá converger hacia un error nulo. No es posible clasificar elementos que no se encuentren claramente separados de otros elementos, es decir no puede categorizar elementos no lineales [59].

Regla del Perceptrón.

Llamada también como Procedimiento de Convergencia del Perceptrón, es conocida como la primera regla para actualizar los pesos en una red neuronal; esta regla modifica los elementos W_k de acuerdo al algoritmo básico de la regla del Perceptrón [57].

$$W_{k+1} = W_k + \alpha\left(\frac{\varepsilon_k}{2}\right)X_k$$

Ecuación 2. 11

"Esta regla desarrollada por Rosenblatt, actualiza W_k, solo si el error ε_k [3] es diferente de 0. El vector de entrada es X_k, el nuevo vector de

[3] El error ε_k se calcula restando la salida deseada menos la salida real. Es decir este valor indica cuanto se equivocó la red.

pesos W_{k+1} y α es la tasa de aprendizaje del sistema, que es un valor constante muy pequeño que no cambia en el tiempo" [57].

Como sabemos el Perceptrón es un tipo de red supervisado, necesita conocer los valores esperados para cada una de las entradas presentadas a la red, por lo que se tiene pares de entrada – salida:

$$\{p_1, t_{1,}\}, \{p_{2,}, t_2\}, \dots \dots \{p_q, t_q\}$$

Ecuación 2. 12

Cuando las entradas p son presentadas a la red, las salidas son comparadas con el valor esperado t, y la salida de la red está dada por [73]:

$$a = f\left(\sum_i w_i p_i\right)$$

Ecuación 2. 13

Adaline

Este tipo de red neuronal es muy parecida al Perceptrón, pero utiliza otra función de transferencia, una de tipo lineal. El gran aporte de esta red es que sirvió de base para el desarrollo de nuevos algoritmos.

"El elemento de procesamiento realiza la suma de los productos de los vectores de entrada y de pesos, y aplica una función de salida para obtener un único valor de salida, el cual debido a su función de transferencia lineal será +1 si la sumatoria es positiva o –1 si la salida de la sumatoria es negativa" [23].

Esta red es uno de los principales elementos en el procesado digital de señales. Su estructura es muy parecida a la del Perceptrón, como se observa en la ilustración.

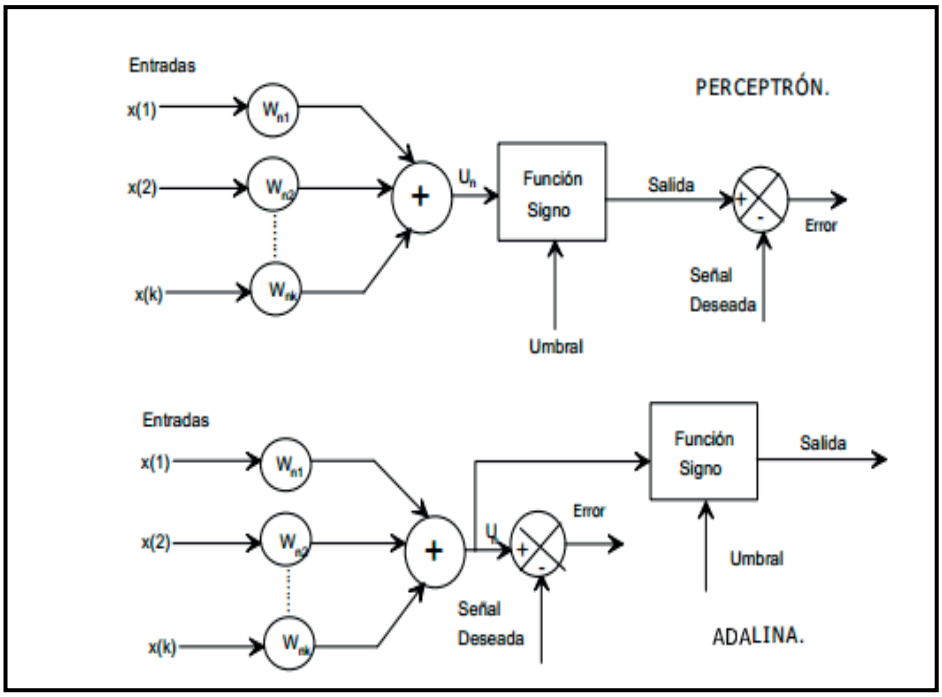


Ilustración 2. 7: Comparación de la estructura de la red Perceptrón y Adaline, ésta última tiene el detalle que la señal de error se calcula antes de aplicar la función signo [21].

Perceptrón Multicapa

Es una red muy antigua que utiliza aprendizaje supervisado y la que en mayor aplicaciones se ha utilizado; el MultiLayer Perceptrón MLP, está formado por una capa de entrada, al menos una capa oculta y una capa de salida. Este tipo de redes usan para su entrenamiento propagación hacia atrás, conocido también como retropropagación del error o regla delta generalizada.

Como se menciona en [21], las principales características de esta red son:

- Se trata de una estructura altamente no lineal.
- Presenta tolerancia a fallos.
- Es capaz de establecer una relación entre dos conjuntos de datos.

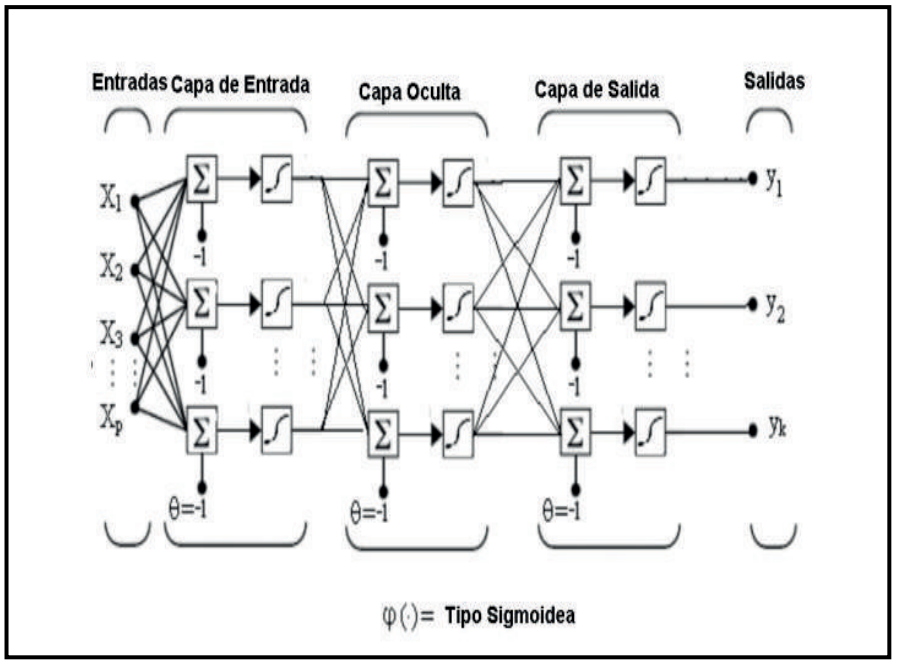


Ilustración 2. 8: Estructura del Perceptrón multicapa, utilizando como funciones de salida Sigmoide [57].

El conjunto de nodos o neuronas propagan la señal hacia la salida, las conexiones que conectan las neuronas son optimizadas por el algoritmo de aprendizaje. Se debe considerar el número de neuronas que forman las capas de la red; para la capa de entrada y salida se considera el problema a resolver; el número de capas ocultas y de neuronas en cada una de ellas decide el diseñador de acuerdo a la aplicación que va hacer de la red.

Cada neurona realiza la propagación, haciendo una combinación de las señales que vienen de las neuronas de la capa anterior utilizando como coeficientes los pesos sinápticos.

Las funciones no lineales más utilizadas, como se describe en [21] son:

Sigmoide**:** Toma valores entre 0 y 1.

Tangente hiperbólica: Toma valores entre -1 y 1.

Se suele pensar en implementar una red con muchas capas ocultas y gran cantidad de neuronas en cada una, sin embargo existe algunos inconvenientes, como se menciona en [58]:

Aumento de la carga computacional: Dificultad de implementación en tiempo real y crecimiento en el tiempo de aprendizaje por parte de la red.

Pérdida en la capacidad de generalización: Si se aumenta el número de neuronas en la capa oculta, aumenta el número de pesos sinápticos; entonces la red contiene mayor número de parámetros, lo que ayuda a una mejor modelización de los patrones utilizados pero se pierde capacidad de generalización debido a que un patrón no usado en la modelización tendrá más dificultad de ajustarse a un modelo con muchos parámetros.

Algoritmo de entrenamiento Backpropagation (Regla delta generalizada)

La red Perceptrón multicapa utiliza un algoritmo de aprendizaje supervisado por corrección del error con aprendizaje offline conocido como propagación hacia atrás de errores o retropropagación (Backpropagation). Consiste en retropropagar la señal desde la capa de salida hasta la capa de entrada, optimizando los valores de los pesos a través de un proceso que se basa en la minimización de la función de coste[4]. El algoritmo se divide en dos fases, como se describen en [21]:

Propagación hacia adelante: Las señales se propagan desde la capa de entrada hasta la capa de salida, generando la salida y el error cometido por la red; al comparar la salida obtenida con la esperada que se le facilita a la red durante el entrenamiento.

Propagación hacia atrás: De acuerdo a los errores cometidos en la capa de salida, Backpropagation se encarga de corregir el valor de los pesos entre las conexiones de las neuronas mediante la retropropagación del error desde la capa de salida hacia la capa de entrada a través de las capas ocultas.

Una neurona de la capa oculta se conecta con otra de la capa de salida como se muesta en la ilustración; {x1…..xn} son las entradas a la capa oculta, x_0 es la entrada que dependiendo si es 1 será el bias,[5] por el contrario si es -1 será el umbral[6]. Los

[4] Se denomina función de coste al valor absoluto del error [21].

[5] Backpropagation en cualquiera de sus capas ocultas utiliza el bias que presentan un nivel de activación de valor 1. Su objetivo es mejorar la convergencia de la red y ofrecer un nuevo efecto umbral sobre la unidad que opera [21].

[6] El umbral representa la mínima entrada total ponderada necesaria para provocar la activación de la neurona [21].

valores de $\{w_{m1}, \ldots, w_{mn}\}$ son los pesos que conectan las entradas con la neurona oculta m y w_{m0} es el peso sináptico correspondiente a x_0:

$$v_m(t) = \sum_{i=0}^{n} w_{mi} * x_i(t)$$

Ecuación 2. 14

El índice m indica la neurona y el índice t indica el número de iteración, si a la función de activación se le denota por φ; entonces la salida de la neurona es:

$$y_m(t) = \varphi_m(v_m(t))$$

Ecuación 2. 15

El conjunto y_m contiene las salidas de las neuronas de la capa oculta, es decir las entradas de la capa de salida; los pesos que conectan estas entradas con la neurona p (de salida) serán $\{h_{p1,\ldots}h_{pr}\}$, p es la neurona y r el número de neuronas ocultas, ahora el sesgo es h_{p0}:

$$z_p(t) = \sum_{j=0}^{r} h_{pj} * y_j$$

Ecuación 2. 16

Si φes la función de activación, la salida de la neurona de salida es:

$$o_p(t) = \varphi_p(z_p(t))$$

Ecuación 2. 17

Si al valor de la salida deseada se denota d_p, el error queda definido por:

$$e_p = d_p - o_p$$

Ecuación 2. 18

En la retropropagación, los primeros pesos que se actualizan son los de la capa de salida, y si existen más capas ocultas el proceso es el mismo, se atraviesan todas las capas hasta llegar a la entrada.

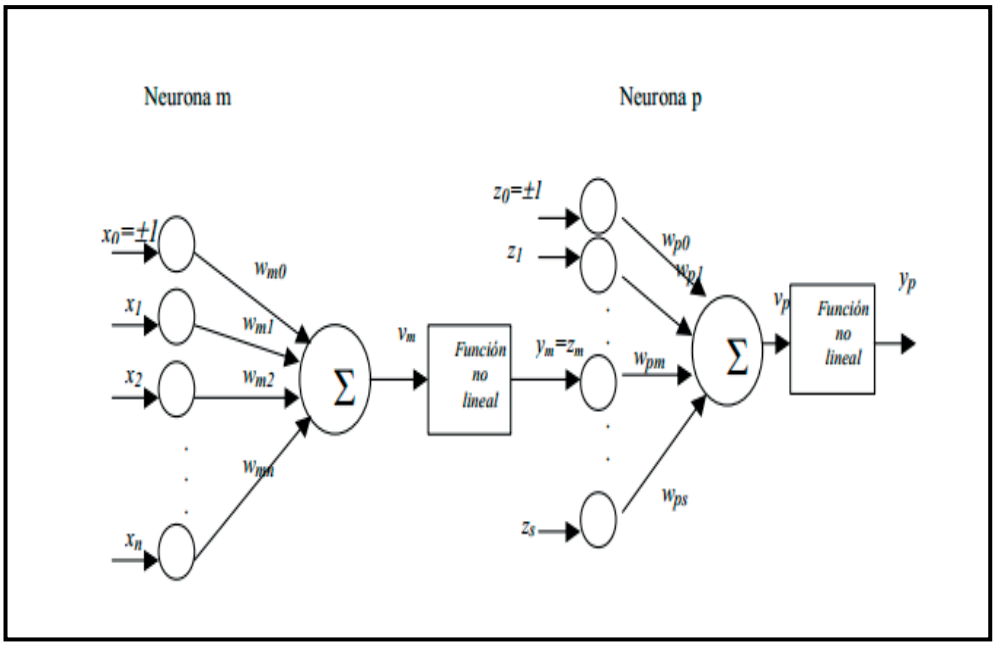


Ilustración 2. 9: Una sola neurona oculta conectada con la neurona de la capa de salida [21].

Proceso de aprendizaje de un Perceptrón Multicapa

Como se describe en [78], el proceso de entrenamiento de una red es:

- Se inician los pesos con valores aleatorios próximos a 0.
- Se presenta un patrón n de entrenamiento (X(n), S(n)), se propaga hacia la salida y se obtiene la respuesta de la red Y(n).
- Se evalúa el error que comete la red para cada neurona.
- Se aplica la regla delta generalizada para modificar los pesos de la red, empezando por la capa de salida hasta llegar a la capa de entrada.
- Se repite el proceso hasta alcanzar un mínimo de error.

Máquina de Boltzmann

Son capaces de representar y resolver complicados problemas de combinatoria y muy útiles para el reconocimiento de patrones, pues tratan de completar partes que no se conocen [46].

Su aprendizaje y funcionamiento se basa en una técnica conocida como el enfriamiento simulado (simulated annealing). Las neuronas de este tipo de red se

conectan entre sí mediante conexiones bidireccionales simétricas: con pesos iguales entre ellas, y tienen salidas binarias; además se distinguen dos grupos, como se menciona en [46][64]:

Neuronas visibles.- Que constituyen la interfaz entre la red y el entorno en el que operan, es decir recogen la información suministrada. Durante el entrenamiento, los estados de estas neuronas ya sean de salida o entrada se ajustan a estados específicos establecidos por los patrones de entrenamiento.

No visibles u ocultas.- Sirven principalmente para mejorar el desempeño de la red y siempre operan libremente.

Como se menciona en [46], existen dos arquitecturas principales para la máquina de Boltzmann, y la diferencia radica en la capa visible:

Completación de Boltzmann: En esta arquitectura existe únicamente un tipo de neuronas visibles y todas están conectadas de forma bidireccional con los pesos simétricos, incluso con las no visibles.

Red de Boltzmann de entrada-salida: En esta arquitectura, las neuronas visibles se dividen en las de entrada y en las de salida; dónde las de entrada se conectan unidireccionalmente con la capa no visible y las neuronas de salida, y las de salida se conectan con todas las demás excepto con las de entrada de forma bidireccional.

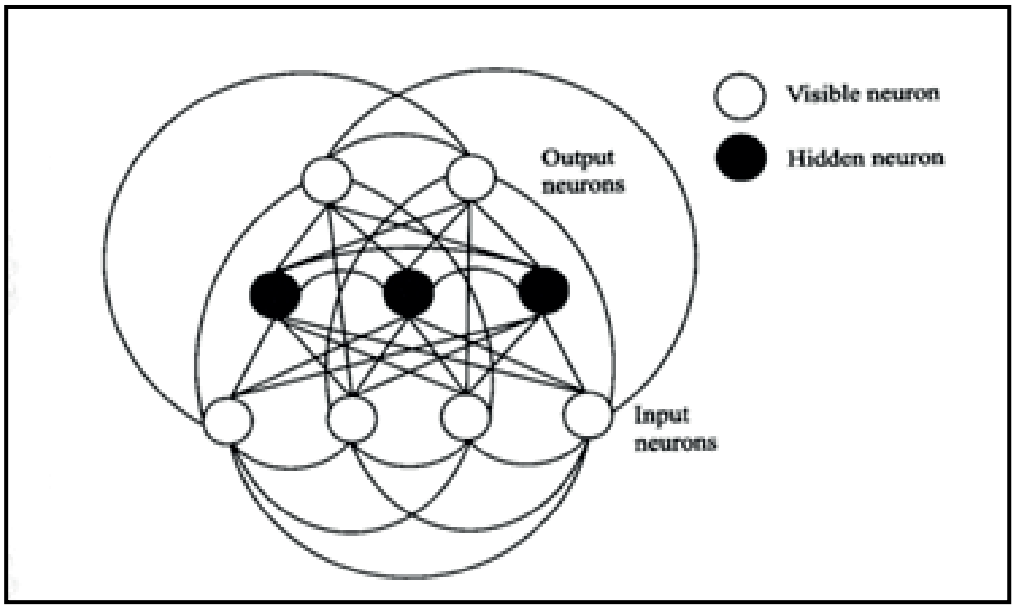


Ilustración 2. 10: **Arquitectura de la máquina de Boltzmann [46].**

Máquina de Cauchy

La máquina de Cauchy es una versión mejorada de la máquina de Boltzmann, pues considera funciones alternativas de probabilidad y ajuste de temperatura; aunque las dos presentan una arquitectura y funcionamiento idénticos. La principal ventaja de la máquina de Cauchy radica en su rapidez de convergencia, se ha demostrado además que combinando las funciones de probabilidad y ajuste de temperaturas previas, alcanza siempre el mínimo global de energía [54].

La máquina de Cauchy se utiliza en diferentes aplicaciones, como se menciona en [54], tenemos:

- Reconocimiento de patrones.
- Procesamiento de voz y de imágenes.
- Procesamiento de conocimiento
- Resolución de problemas de optimización.
- Reconocimiento de dígitos manuscritos
- Optimización de rutas de transporte
- Optimización presupuestaria

Sin embargo su uso no es tan requerido, debido a que durante la etapa de aprendizaje y funcionamiento necesitan demasiado tiempo, además su funcionamiento es muy complejo y mucho menos intuitivo que el de otras redes.

Red de Aprendizaje Asociativo.- Estas redes con aprendizaje no supervisado, conocido también como auto-supervisado no necesitan ninguna influencia del exterior para ajustar los pesos de sus conexiones. Se dice que estas redes son auto organizado, debido a que la red no recibe ninguna información que le indique si la respuesta generada en relación a la entrada de la red, es o no correcta.

> *"Estas redes deben encontrar las características, regularidades, correlaciones o categorías que se puedan establecer entre los datos que se presenten en su entrada; puesto que no hay supervisor que indique a la red la respuesta que debe generar ante una entrada concreta"* [23].

Al no tener un supervisor, estas redes generan varias salidas con distintas posibilidades de interpretación, dependiendo de la estructura y del algoritmo de aprendizaje que se haya empleado.

Mapas auto organizados 'Redes de Kohonen' (redes competitivas).- Fueron inventados por Teuvo Kohonen en 1984, estos mapas poseen aprendizaje competitivo no supervisado.

La arquitectura de las redes de Kohonen es simple, posee dos capas:

Capa de entrada.- Conocida también como sensorial, tiene n neuronas, una por cada variable de entrada.

Capa de competición.- Formada por m neuronas, en esta capa se realiza el procesamiento y es donde se forma el mapa de rasgos de dos dimensiones. Estas neuronas no están conectadas, sin embargo se puede decir que existen conexiones laterales de excitación e inhibición pues cada una tiene cierta influencia sobre sus vecinas.

Las conexiones entre las neuronas de la red son hacia adelante, desde la capa de entrada hacia la capa de salida, cada conexión posee un peso; y es así como las neuronas de salida tienen asociado un vector de pesos conocido como vector de referencia, pues constituye el vector prototipo de la clase representada por la neurona de salida [79].

Aprendizaje

En la fase de entrenamiento se recibe un vector de entrada para cada neurona, éste se propaga por las conexiones hasta la capa de competición, donde cada neurona genera una salida al comparar la entrada con los pesos; aquella célula que genere la salida más pequeña será la ganadora, y así se modifican los pesos de ella y sus vecinas; se usa la distancia euclídea para medir esa similitud [41][79].

Su forma de operar es de dos modos:

Entrenamiento.- Durante esta etapa se construye el mapa usando ejemplos entrenantes.

Mapeo.- Clasifica una nueva entrada.

Algoritmo

El método se resume en algunos pasos; como señala [41], son:

- Se inicializan los pesos con valores pequeños aleatorios.
- Presentar una nueva entrada cada cierto período, hasta que la red converja.
- El patrón de entrada se debe propagar hasta la capa de competición, donde la red genera los valores de salida de las neuronas que pertenecen a esa capa, para ello usamos la distancia euclídea.
- Se obtiene una célula ganadora C, que posee la menor salida,
- Se actualiza las conexiones entre la capa de entrada y la neurona ganadora C, conjuntamente con su vecindad, dependiendo de su grado de vecindad.
- Si los datos que se presentaron en la entrada son mayores que el umbral, se debe volver al paso 2, caso contrario se finaliza el método.

La red debe ser entrenada con un gran número de ejemplos

LVQ 'Learning Vector Quantization' (Red competitiva)

Su arquitectura es como la de un mapa organizado con la única diferencia que su aprendizaje es supervisado; existen algunas versiones de este algoritmo, mas aún la que se utiliza con mayor frecuencia y la que mejores resultados presenta es LVQ1.

Lvq 1

De una secuencia de patrones conocida como observaciones vectoriales se selecciona un conjunto inicial de prototipos o codebook conocidos también como vectores de referencia; de forma iterada se selecciona una observación X y se actualiza el conjunto de prototipos para que se asemeje mas a X; cada neurona aprende este vector de prototipos o codebooks, el cual permite a la neurona clasificar una región del espacio de entrada calculando la distancia entre el vector entrada y el vector de referencia directamente, mediante la regla del vecino más cercano. Cuando ya se ha construído el vector de prototipos o codebook quiere decir que la red se ha entrenado; y es posible empezar con las pruebas de la red [61] [82].

Redes recurrentes

Este tipo de redes posee diferentes conexiones, como se menciona en [68]:

- Entre neuronas de una misma capa.

- Entre neuronas de una capa a una capa anterior.

A este tipo de redes neuronales las podemos definir como sistemas que tienen conexiones que van desde los nodos de salida hacia los nodos de entrada, además de conexiones arbitrarias entre cualquiera de sus nodos; "de esta forma el estado interno de la red se modificará a medida que se le presenten datos, simulando una memoria"[25].

> *"El entrenamiento en este tipo de redes se realiza conectando su salida a la entrada, y procesando los datos hasta llegar a un estado de equilibrio, en el que las salidas mantienen su valor"* [24].

Las redes recurrentes más importantes son: la red de Elman y la red de Hopfield que se describe a continuación.

Redes de Elman.- Este tipo de redes está formada por algunas capas: una capa de entrada, dos capas intermedias (una de neuronas oculta y otra de unidades de contexto) y una capa de salida. Las neuronas de las capas de entrada y salida recogen información del entorno; las unidades de salida reciben las salidas de las neuronas de la capa oculta ponderadas por los correspondientes pesos sinápticos [60].

La característica que diferencia este modelo, son las unidades de la capa de contexto, como se menciona en [60], se utilizan para memorizar las salidas de las unidades ocultas en la etapa anterior, de manera que cada unidad de contexto tiene como salida, la salida de la unidad oculta correspondiente en la etapa anterior. Las neuronas ocultas reciben como entrada las salidas de las unidades de contexto y de las de entrada ponderada por sus pesos sinápticos. Por lo que la salida de la red depende de algunos patrones: de los patrones de entrada actual y de los anteriores a través de las unidades de contexto.

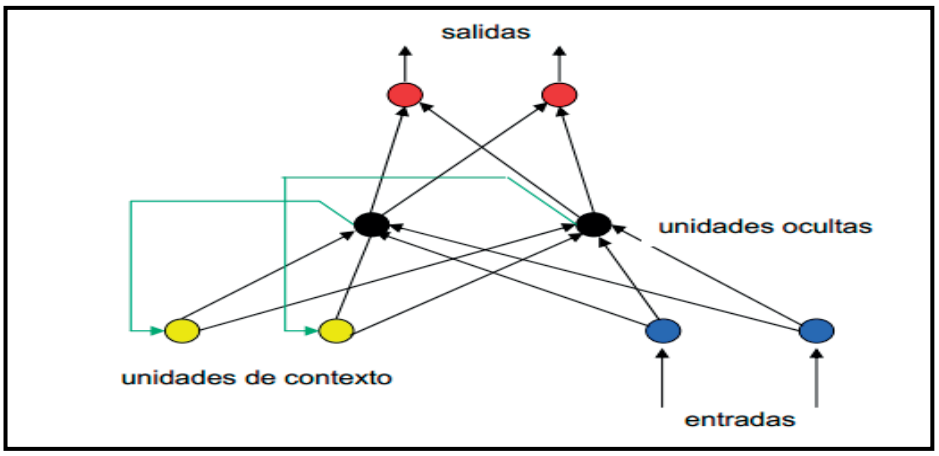


Ilustración 2. 11 **Arquitectura de una red de Elman [60].**

Redes de Hopfield

Fue propuesta en 1982 por el físico John Hopfield. Se basa fundamentalmente en estos aspectos novedosos, que se mencionan en [21]:

- Planteamiento de una memoria asociativa[7], es decir permite recuperar patrones a partir de información incompleta.
- Todas las neuronas están conectadas con todas las demás.

La red de Hopfield basa su funcionamiento en almacenar información a manera de memoria asociativa. Como se describe en [21], la memoria asociativa parte de un estado inicial llamado información de partida, después se deja evolucionar al sistema hasta que llegue a un estado estable, siendo este estado estable el patrón que más se parece a la información inicial. Se debe mencionar que el patrón inicial puede ser una versión deteriorada o incompleta del patrón que se desea obtener.

De los patrones almacenados inicialmente, la red encontrará aquel que más se parezca al presentado en la entrada planteada.

Arquitectura de la red de Hopfield

El modelo original de Hopfield consta de N neuronas binarias, donde cada célula se conecta con todas las demás mediante conexiones laterales, pero no consigo misma [62][68].

La matriz de pesos $W = (w_{ij})$ de $n * n$, donde w_{ij} representa el peso de la conexión de la neurona i a la j. Esta matriz posee algunas características, como se menciona en [62]:

Es una matriz simétrica, es decir $w_{ij} = w_{ji}, \forall\, i, j = 1,2, \ldots\ldots, n$. Es decir el peso de la conexión entre dos neuronas, tiene el mismo valor en ambos sentidos.

[7] **Memoria Asociativa.-**Es el almacenamieto y recuperación de información por asociación con otras informaciones. Permite recuperar información a partir de conocimiento parcial de su contenido, sin saber su localización de almacenamiento [77].

Los elementos de la diagonal de la matriz son iguales a 0. Esto quiere decir que $w_{ii} = 0 \ \forall \ i = 1,2, \ldots \ldots, n$; debido a que no existen conexiones de una neurona a sí misma.

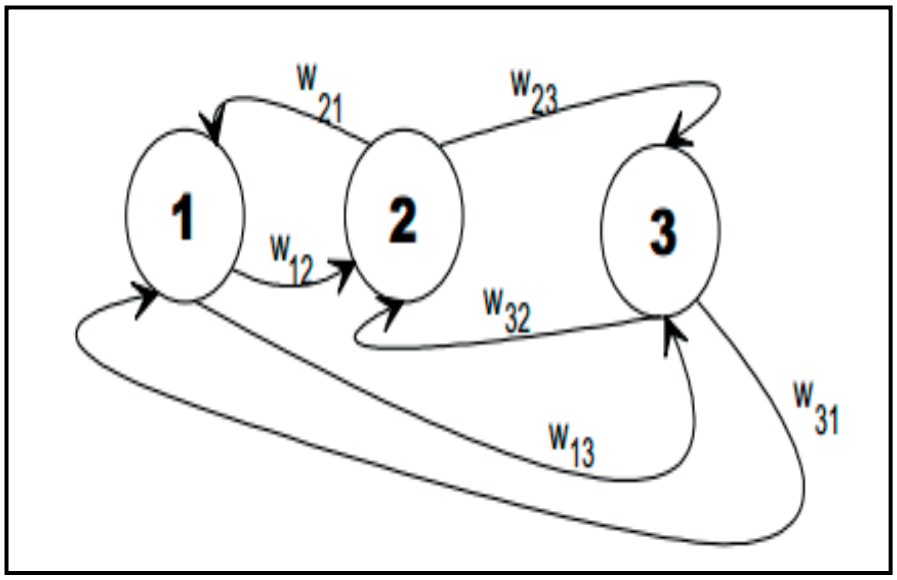


Ilustración 2. 12: Representa el esquema de la red de Hopfield, con sus pesos simétricos [58**].**

Las neuronas son binarias con dos estados generalmente -1 y 1, determinadas por el nivel de activación que recibe la neurona. El estado de la neurona i en t+1 está dada por:

$$s_i(t+1) = sgn\big(v_i(t+1)\big) para\ i = 1,2, \ldots . , n$$

Ecuación 2. 19

$$sgn\big(v_i(t+1)\big) = \begin{Bmatrix} +1\ si\ v_i(t+1) > 0 \\ -1\ si\ v_i(t+1) < 0 \end{Bmatrix}$$

Ecuación 2. 20

Donde $v_i(t+1)$ es el estado de activación de la neurona i, que se ha calculado como indica [68]:

$$v_i(t+1) = \sum_{j=1}^{n} w_{ji} s_j(t) - u_i\ para\ i = 1,2, \ldots , n$$

Ecuación 2. 21

Dónde $s_j(t)$es el estado de la neurona j en el instante anterior t y u_i es un umbral fijo aplicado a la neurona i.

En este tipo de redes hablamos de estados en un instante t, por ejemplo si el $v_i(t+1=0$, se considera que el estado de i no cambia.

Para una red que tenga n neuronas, el estado en (t+1), estaría dado por:

$$s(t+1) = [s_1(t+1), s_2(t+1), \ldots., s_n(t+1)]$$

Ecuación 2. 22

El estado S representa un objeto binario con n bits de información.

Funcionamiento

El primer paso para trabajar con una red de Hopfield es codificar y presentar la información en forma de vector, la codificación será binaria. El vector de entrada X debe tener el mismo número de componentes que el número de neuronas en la red. En t=0 la entrada es aplicada a la única neurona que tiene la red, obteniendo así la salida $v_j(0)$, que luego se convierten en las nuevas entradas a la red debido a las realimentaciones. La primera salida es tomada como entrada en el ciclo siguiente, produciendo una nueva salida; por lo que el aprendizaje es distinto y se basa en encontrar los pesos en función de un problema y no de ir ajustando los pesos, ya que estos permanecen estables desde el comienzo. Además existe una nueva función denominada, Función de Energía que puede representar el conjunto total del sistema [62].

En este tipo de red neuronal se distinguen dos fases de operación, que se mencionan en [62]:

Fase de almacenamiento.- Donde se va a establecer los valores que deben tener los pesos para almacenar un conjunto de patrones.

Fase de recuperación.- Que describe el proceso a seguir para recuperar información almacenada a partir de datos incompletos.

Fase de almacenamiento.- Sea $\{x(k) = (x_1(k), x_2(k), \ldots\ldots., x_n(k))\}_{k=1,\ldots..p}$ el conjunto de p patrones que se desea almacenar, donde cada patrón x(k) es un vector

n-dimensional donde sus componentes toma valores binarios -1 ó 1. Para almacenar patrones, el peso de la conexión de la neurona *j* a la *i*, como se menciona en [62] es:

$$w_{ij} = \sum_{k=1}^{p} x_j(k)x_i(k) \quad \forall i \neq j$$

Ecuación 2. 23

Esta ecuación cumple la regla de Hebb, debido a que si $x_i(k)$ y $x_j(k)$ son iguales, el peso de la conexión de i a j se incrementa en una unidad; y si son distintas, se reducen en 1.

Fase de recuperación.- Sea x = $(x_1, x_2, \ldots, x_n)$ un patrón de prueba diferente a uno de los patrones almacenados anteriormente. Dicho patrón suele ser uno de los vectores x(k) que se almacenó en la etapa anterior con información incompleta o que tiene ruido. En esta fase la red recuperará el patrón almacenado más próximo al patrón de prueba X; el procedimiento que se sigue es el que se indica en [62]:

Se inicializan los estados de las n neuronas de la red utilizando el patrón x:

$$s_i(0) = x_i \quad para\ i = 1,2, \ldots, n$$

Ecuación 2. 24

Se debe calcular los estados de la red y se espera a que la red alcance un estado estable, se refiere a que el estado de los elementos de la red permanezca invariante en el tiempo, utilizando la ecuación:

$$s_i(t+1) = s_i(t) \quad \forall i = 1,2, \ldots, n$$

Ecuación 2. 25

Entonces, este estado estable representa el patrón recuperado a partir del patrón de prueba x.

Algunas veces sucede que la red llega a un estado estable que no corresponde con el patrón almacenado, esto puede ocurrir cuando se almacena un excesivo número de patrones [68].

Función de energía o de error.- Como ya se ha explicado, en esta red los pesos se calculan por adelantado y no forman parte de un sistema dinámico[8]. Para cada estado de la red, Hopfield demostró que se puede definir una función de energía:

$$E = -\frac{1}{2}\sum_{i=1}^{n}\sum_{j\neq 1}^{n} w_{ij}s_i s_j + \sum_{i=1}^{n} U_i s_i$$

Ecuación 2. 26

2.4 APLICACIONES DE UNA RED NEURONAL

La computación neuronal tiene ciertas ventajas con respecto a los métodos tradicionales, por ejemplo pueden funcionar razonablemente incluso cuando se tiene entradas incompletas o con ruido. Por tal motivo se puede aplicar en una extensa variedad de aplicaciones, como las siguientes:

Conversión texto a voz: La iniciativa principal dentro de esta área es de Terrence Sejnowsk, junto con Rosemberg presentaron un sistema llamado NetTalk, el cual convierte un texto en fonemas y con la ayuda de un sintetizador de voz, Dectalk genera voz a partir de un texto escrito. Dentro de esta área conversión texto-voz, las tecnologías neuronales tiene gran ventaja con respecto a la computación tradicional, debido a que se elimina la necesidad de programar un complejo conjunto de reglas de pronunciación en el ordenador [21].

Procesado natural del lenguaje: Los científicos Rumelhart y McClelland han desarrollado una red neuronal de proceso natural del lenguaje, esta aplicación ha aprendido el tiempo verbal past tense de los verbos en inglés; como indica [21], su funcionamiento consiste en:

> *"Las características propias de la computación neuronal como la capacidad de generalizar a partir de datos incompletos y la capacidad de abstraer, permiten al sistema generar buenos pronósticos para verbos nuevos o verbos desconocidos".*

[8] Estado dinámico se refiere a que cambian en función del tiempo.

Compresión de imágenes: Los científicos Cottrel, Munro y Zisper de la Universidad de San Diego y Pisttburgh, han desarrollado un sistema para la compresión de imágenes que utiliza una red neuronal, la compresión es de 8 a 1 [21].

Reconocimiento de caracteres: Los investigadores de Nestor Inc., han desarrollado un sistema que es capaz de reconocer caracteres que no ha visto antes, después de un entrenamiento con un conjunto de tipos de caracteres de letras [21].

Problemas de combinatoria: Para resolver estos problemas, la programación tradicional requiere "un tiempo de proceso que es exponencial con el número de entradas"; estos inconvenientes de combinatoria han sido resueltos con éxito por Hopfield, desarrollando una red neuronal artificial que ofrece excelentes resultados [21].

Procesado de la señal: Esta aplicación se divide en tres procesos distintos que utilizan una red neuronal artificial para cada proceso:

Predicción: Para poder predecir algún fenómeno, en muchos casos, nos basamos en una serie temporal de datos, los cuales nos permiten conocer su comportamiento. "Lapedes y Farber del Laboratorio de Investigación de los Álamos, han demostrado que la red backpropagation supera en un orden de magnitud a los métodos de predicción convencionales" [21].

Modelado de Sistemas: Una red neuronal es capaz de aprender una función de transferencia y comportarse de manera correcta como el sistema lineal que se puede estar modelando.

Filtro de ruido: Las redes neuronales son capaces de "mantener en un alto grado las estructuras y valores de los filtros tradicionales," se utilizan para eliminar el ruido de una señal [21].

Servo Control: Una RNA es capaz de predecir el error que se produce en la posición final de un robot; muchas redes han sido entrenadas para lograr este objetivo. Cuando se conoce el error, éste es combinado con la posición deseada, y así es posible indicar una corrección para mejorar la exactitud de la posición final [21].

Reconocimiento de voz: Una red neuronal artificial presenta un alto grado de aprendizaje en las señales de entrada, logrando de esta manera niveles altos de

adaptabilidad en las variantes que presenta la voz. El tipo de red que presenta los mejores resultados en este reconocimiento es la red híbrida, que posee dos componentes: uno de aprendizaje supervisado y otro auto organizado, el sistema toma como entrada un espectrograma[9]; la fase de entrenamiento puede ser costosa en tiempo de CPU, sin embargo una vez ajustados todos los pesos, el sistema ofrece una salida en tiempo real [30] [35].

Entre otras aplicaciones de una red neuronal, en diversas áreas; como señala [26], se puede mencionar:

"Biología:

Aprender más acerca del cerebro y otros sistemas.

Obtención de modelos de la retina.

Empresa:

Evaluación de probabilidad de formaciones geológicas y petrolíferas.

Identificación de candidatos para posiciones específicas.

Explotación de bases de datos.

Optimización de plazas y horarios en líneas de vuelo.

Optimización del flujo del tránsito controlando convenientemente la temporización de los semáforos.

Reconocimiento de caracteres escritos.

Modelado de sistemas para automatización y control.

Medicina:

Analizadores del habla para ayudar en la audición de sordos profundos.

Diagnóstico y tratamiento a partir de síntomas y/o de datos analíticos (electrocardiograma, encefalogramas, análisis sanguíneo, etc.).

[9] Información de la evolución del contenido frecuencial de la señal a lo largo del tiempo que dura la pronunciación [35].

Monitorización en cirugías.

Predicción de reacciones adversas en los medicamentos.

Entendimiento de la causa de los ataques cardíacos.

Campo militar

Clasificación de las señales de radar.

Creación de armas inteligentes.

Optimización del uso de recursos escasos.

Reconocimiento y seguimiento en el tiro al blanco".

Todas estas aplicaciones basan su funcionamiento en el reconocimiento de patrones, como señala [26]:

Buscan un patrón en una serie de ejemplos, clasifican patrones, completan una señal a partir de valores parciales o reconstruir el patrón correcto partiendo de uno distorsionado.

Hoy en día las aplicaciones de una red neuronal artificial han crecido considerablemente; como se mencionó, actúa en campos como la medicina, ayudando a descubrir y a tratar enfermedades.

Brittany Wegner desarrolló un sistema de red neuronal llamado Fine Needle Aspirate con la capacidad de diagnosticar el cáncer de mama. Como se explica en [27], el funcionamiento consiste en:

> *"Utiliza programas de computación que trabajan como si fuera neuronas en un cerebro a través de un servicio de nube. Gracias a estos mecanismos, el 'cerebro artificial' es capaz de detectar patrones complejos que se van desarrollando".*

Se ha probado el sistema obteniendo como resultado una tasa de éxito del 99.1%; el proyecto aún sigue en construcción y se podría implementar en hospitales y hacerlo extensivo a otros tipos de cáncer.

Google mantiene una división llamada Google X, que últimamente ha presentado una red neuronal capaz de distinguir imágenes, "ha sido capaz de distinguir imágenes de gatos sin necesidad de haberle enseñado a distinguirlos", para lograrlo los ingenieros del proyecto unieron 1000 computadores para tener un conjunto de 16.000 procesadores con los que formaron una gran red neuronal en la que se establecieron más de mil millones de conexiones entre neuronas [29].

Esta red neuronal, pretendía reconocer patrones e imágenes, se le presentó videos de Youtube con el objetivo de identificar rasgos de gatos en las más de 10 millones de imágenes que se le pasaban. La idea era que fuese una red totalmente autónoma, que el aprendizaje no fuese guiado por ningún humano, y los resultados han sido sorprendentes, pues es la primera vez que un sistema es capaz de distinguir imágenes sin haber recibido un feedback externo, es decir no recibió ayuda, y obtuvo un acierto del 15.8% en una muestra de 20000 imágenes aleatorias [28].

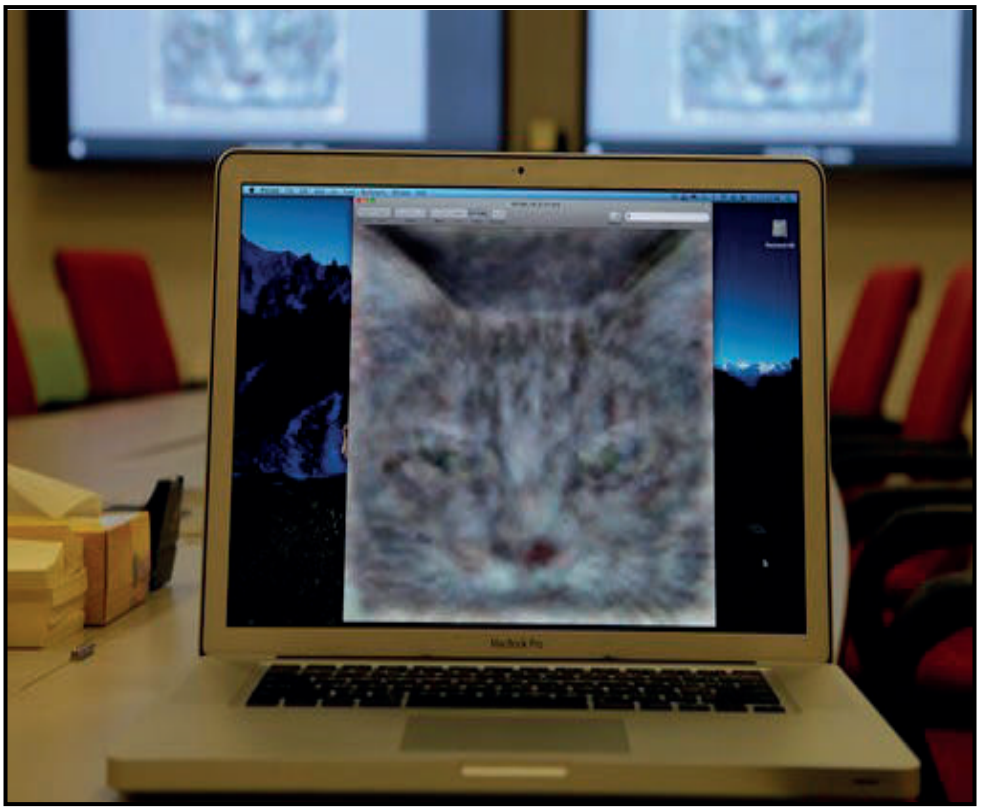


Ilustración 2. 13: Imagen obtenida, sin recibir un feedback externo, y después de mostrarle 20000 imágenes aleatorias. [28]

2.5 VENTAJAS Y DESVENTAJAS DE LAS REDES NEURONALES

Ventajas de una RNA

Las redes neuronales artificiales poseen características similares a las del cerebro, pueden realizar por lo tanto algunas actividades propias del hombre: logran aprender de la experiencia, son capaces de generalizar de casos anteriores a nuevos casos, abstraen características esenciales a partir de entradas con información irrelevante [25].

Dentro de las ventajas, se menciona en [25] las más importantes:

Aprendizaje: Una red neuronal artificial es capaz de aprender, esto se da a partir de un conjunto de datos de entrada que se presenta a la red; mediante un entrenamiento o una experiencia inicial.

Auto organización: Una red neuronal puede tener su propia representación de la información en su interior.

Tolerancia a fallos: Una RNA es muy robusta, pues a pesar de estar dañada parcialmente puede seguir respondiendo de forma aceptable, esto se debe a que almacena la información de forma redundante.

Flexibilidad: Cuando los datos de entrada en una red presentan cambios no muy significativos como ruido, la RNA puede manejar estos cambios adecuadamente.

Tiempo real: La respuesta de una red puede darse en tiempo real, debido a que la estructura de una RNA es en paralelo. Cuando se implementa en dispositivos electrónicos como en una computadora las respuestas pueden ser inmediatas.

Desventajas de una RNA

A pesar de las útiles y variadas aplicaciones de una red neuronal, también tienen ciertas desventajas en su mayoría superables; como señala [29], estas son:

Complejidad de aprendizaje para grandes tareas: Cuando una red neuronal tiene que realizar muchas tareas, también tiene que aprender a realizarlas, entonces enseñarle a la red es un trabajo más complicado.

Tiempo de aprendizaje elevado: Cuando una RNA tiene que reconocer o clasificar mayor cantidad de patrones, o si existe patrones muy similares que la red debe identificar, se tiene que invertir más tiempo en lograr unos pesos adecuados que representen lo que se quiere enseñar a la red neuronal.

No permite interpretaciones: La salida de la red es únicamente un número que no puede ser interpretado por ella misma; depende de la aplicación y del programador darle un significado a dicho valor.

Elevada cantidad de datos para el entrenamiento: Cuando se desea que una red aprenda de forma correcta a identificar patrones, se debe enseñar mayor cantidad de información; esto implica varias horas de consumo de CPU, además que un entrenamiento es para un solo problema.

CAPÍTULO III

CLASIFICACIÓN DE LAS REDES NEURONALES POR TIPO DE APRENDIZAJE

CAPITULO III

CLASIFICACIÓN DE LAS REDES NEURONALES POR TIPO DE APRENDIZAJE.

Las redes neuronales artificiales utilizan algoritmos de aprendizaje o de entrenamiento que permitan encontrar valores adecuados para unos parámetros denominados pesos sinápticos; existen dos conceptos fundamentales en el aprendizaje que se explica posteriormente [38].

Esta es la característica más importante de una RNA, debido a que mediante este proceso la red va ajustando internamente los pesos asociados a cada rama para obtener la salida esperada y así pueda responder después por sí sola a situaciones diferentes a las aprendidas [44].

Aprendizaje de las RNAs

Para que una red aprenda, es necesario que siga un proceso que consiste en modificar los pesos de las conexiones en respuesta a una información de entrada, el aprendizaje se encuentra representado en estos pesos. Las conexiones entre las neuronas pueden destruirse, modificarse o crearse en función al valor de sus pesos; es decir si el peso de la conexión toma el valor de 0, la conexión se destruye; cuando el peso toma un valor distinto de 0, la conexión se crea o modifica; y cuando los pesos de las conexiones permanecen estables se considera que el proceso de aprendizaje ha terminado [20].

Existe un aspecto muy importante en el aprendizaje de las redes: conocer como se modifican los pesos, es decir cuáles son los criterios que se siguen para cambiar el valor asignado a las conexiones cuando la red tiene que aprender nueva información; como se mencionó anteriormente existen dos conceptos fundamentales en el aprendizaje, se describe en [20][53][55][59]: Paradigma de aprendizaje y Regla de aprendizaje.

PARADIGMA DE APRENDIZAJE

Se refiere a la información de la que dispone la red. Dentro de este concepto tenemos:

- Aprendizaje online
- Aprendizaje offline
- Aprendizaje supervisado.
    - Aprendizaje por corrección de error
    - Aprendizaje por refuerzo
    - Aprendizaje estocástico
- Aprendizaje no supervisado
    - Aprendizaje Hebbiano
    - Aprendizaje competitivo y cooperativo

Aprendizaje On Line

La red puede aprender durante su funcionamiento; no se distinguen fases de entrenamiento y de operación, por lo que los pesos de las conexiones varían de forma dinámica cada vez que se presente una nueva información al sistema. Como presentan un carácter dinámico, en este tipo de redes es importante analizar la estabilidad.

Aprendizaje Off Line

La red requiere una fase previa de aprendizaje y otra fase de operación, por lo que debe existir un conjunto de datos de entrenamiento y un conjunto de datos de prueba. Los pesos de las conexiones se mantienen fijos después de terminar la etapa de entrenamiento de la red. Este tipo de redes presentan un carácter estático, por lo que los sistemas no muestran problemas de estabilidad durante su funcionamiento.

3.1 APRENDIZAJE SUPERVISADO.

A la red se le proporciona los valores de entrada y los valores de salida deseados, y ésta aprende a asociarlos; el proceso de aprendizaje consiste en un entrenamiento supervisado por un agente externo quien determina la respuesta en base a las entradas que se presentaron a la red; cuando se obtiene una salida, el agente la compara con la esperada y si éstas no coinciden se procede a modificar los pesos de las conexiones hasta lograr la respuesta deseada [36].

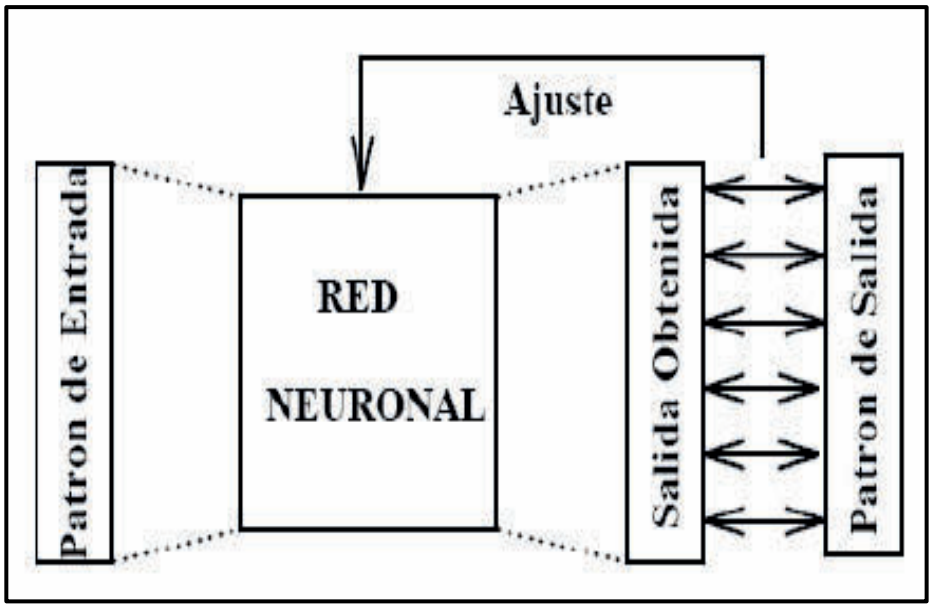


Ilustración 3. 1: Estructura de un Aprendizaje Supervisado, se observan las salidas: esperada y la obtenida, con el ajuste para lograr la deseada [40].

Dentro del aprendizaje supervisado, existen tres formas de entrenamiento: aprendizaje por corrección de error, aprendizaje por refuerzo y aprendizaje estocástico.

3.1.1 APRENDIZAJE POR CORRECCIÓN DE ERROR

Cuando una RNA genera una salida, ésta es comparada con la que se espera; la diferencia entre estos valores se utiliza para el aprendizaje por corrección de error. Los pesos de las conexiones se ajustan de acuerdo a esta diferencia, es decir en función al error cometido en la salida y van cambiando hasta lograr que la respuesta que generó la red sea la que se desea. La fórmula para la corrección de los pesos, como señala [36], podría ser la siguiente:

$$Incr\ (wji) = \ \beta\ yi\ (dj - yj)$$

"Siendo:

Incr (wji): Variación en el peso de la conexión entre las neuronas i y j

yi : Salida de la neurona i

dj : Valor de salida deseado para la neurona j

yj : Valor de salida obtenido en la neurona j

ß : Factor de aprendizaje (0 < ß <= 1) que regula la velocidad del aprendizaje".

La idea principal de este aprendizaje es minimizar el error entre la salida esperada y la que se obtiene, un ejemplo claro dentro de este tipo de aprendizaje, es el Perceptrón.

El método se puede describir como menciona [37], en los siguientes pasos:

1. *"Inicializar aleatoriamente los pesos.*
2. *Presentación de conjunto de entrenamiento (CE)*
3. *Obtención de las salidas para el CE.*
4. *Comparación de salidas deseadas con las actuales.*
5. *Si se verifica el criterio de finalización ir al siguiente paso, si no ir al paso 2.*
6. *Fin".*

3.1.2 APRENDIZAJE POR REFUERZO

Este tipo de aprendizaje se considera más lento que el anterior, y no dispone de un ejemplo completo que indique el comportamiento deseado de la red, de igual manera se desconoce la salida exacta para cada entrada; únicamente existe un procedimiento general para las diferentes entradas de una RNA. El proceso que siguen estas redes se denomina éxito o fracaso y se basa en la relación entrada-salida de la red, genera una señal "Señal de refuerzo (éxito = +1 o fracaso = -1), que mide el buen funcionamiento del sistema", siendo esta señal utilizada por el supervisor, quien únicamente opina sobre la respuesta que generó la red, para indicar si la salida se ajusta a la deseada; y en función de ello los pesos se cambian, utilizando probabilidades, como se explica en [37]:

"Si una acción tomada por el sistema de aprendizaje es seguida por un estado satisfactorio, entonces la tendencia del sistema a producir esa particular acción es reforzada. En otro caso, la tendencia del sistema a producir dicha acción es disminuida".

Es decir, un aprendizaje por refuerzo tiene un enfoque distinto, trata de aprender de la experiencia y no de un conjunto de ejemplos. Entre las características más importantes de este tipo de aprendizaje, como se señala en [43], tenemos:

- ✓ Está dirigido por objetivos; el objetivo es cada acción que se realiza sobre un entorno y por el cual devuelve una recompensa, la salida que debe generar el sistema no se conoce, únicamente se tiene en cuenta que dicha salida debe generar un efecto que maximice la recompensa recibida.
- ✓ El entorno mantiene un comportamiento desconocido y obedece a una cierta función de probabilidad que indica la evolución del entorno y la recompensa que se debe generar.
- ✓ La recompensa podría retardarse, debido a que los beneficios de las acciones que se ha realizado en un sistema, puede reflejarse después de algunas evaluaciones.
- ✓ El comportamiento del entorno no se conoce, por lo tanto para lograr un aprendizaje correcto, se requiere una fuerte carga de ensayo y error.

Dentro de este tipo de aprendizaje existen elementos necesarios para el entrenamiento; se menciona en [43], a continuación:

Agente: Es fundamental en este tipo de aprendizaje; lee el estado del entorno, realiza acciones sobre el entorno y lee las recompensas que generan estas acciones.
Entorno: Es el medio donde el agente realiza las diferentes acciones, el entorno las recibe y evoluciona. El funcionamiento es desconocido y responde a funciones de probabilidad, se encarga de generar las recompensas de acuerdo a las acciones y los cambios de estado.

Política: Son las reglas de asociación existentes entre el agente, el entorno y las acciones a tomar.

Función de refuerzo: Es el que se encarga de establecer la recompensa, de acuerdo al estado del entorno y la acción que se genera sobre éste.

Función de evaluación: Estima la recompensa que se va a recibir, basándose en un estado y siguiendo una política. Esta función nos ayuda a elegir la acción que se va a

realizar, de acuerdo al estado que obtenga un mayor valor. “El objetivo de los algoritmos de aprendizaje por refuerzo es construir esta función”

Modelo del entorno: Predice el comportamiento del entorno, así ayudando a prevenir o corregir errores.

3.1.3 APRENDIZAJE ESTOCÁSTICO

El aprendizaje estocástico consiste en ir cambiando aleatoriamente los valores de los pesos, para luego observar los resultados y evaluarlos de acuerdo a la respuesta deseada. Con las redes que utilizan este tipo de aprendizaje se suele hacer analogías y asociarlas con un sólido físico que posee cierto estado energético; en el caso de la red, la energía representaría el grado de estabilidad, de aquí se obtiene el “estado de mínima energía: valores de pesos con los que la estructura se ajusta al objetivo deseado” [37].

El proceso a seguir dentro de este aprendizaje se resume en [37], a continuación:

- *“Se realiza un cambio aleatorio en los pesos.*
- *Se determina la nueva energía de la red.*
    - *Si la energía decrece: se acepta el cambio.*
    - *Si la energía no decrece: se aceptaría el cambio en función de una determinada y preestablecida distribución de probabilidades”.*

3.2 APRENDIZAJE NO SUPERVISADO

En este tipo de aprendizaje se presenta a la red los valores de entrada, pero no se conoce las salidas que debería generar. La RNA no recibe ninguna información del entorno que le permita determinar si la salida con respecto a una entrada es correcta o no; “deben encontrar las características, regularidades, correlaciones o categorías que se pueden establecer entre los datos de la entrada” [36].

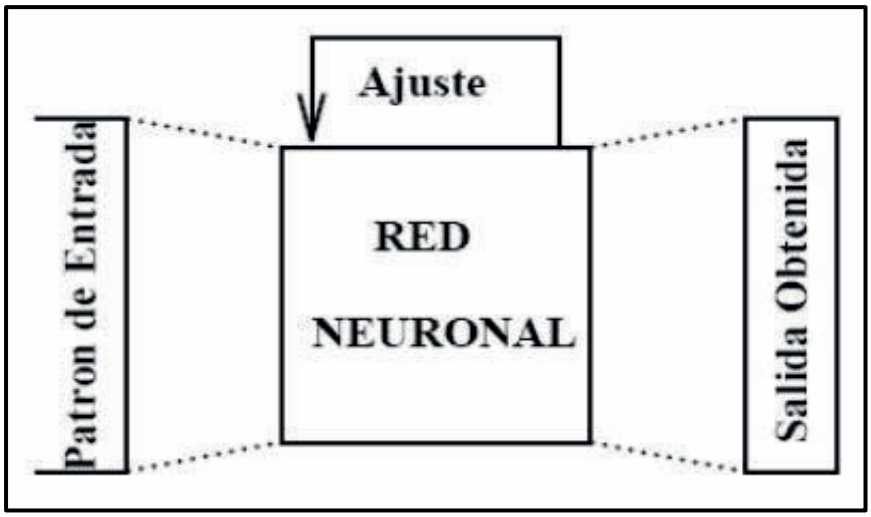


Ilustración 3. 2: Estructura del Aprendizaje no supervisado, como se observa no existe una salida que indique lo que se espera [40].

En cuanto a la salida de la red, existen varias posibilidades de interpretación, como se explica en [36]:

- Puede representar el grado de familiaridad entre la información que se le está presentando en la entrada y la información que se le ha presentado en el pasado.
- Se establece categorías, la misma red indica a la salida a que categoría pertenece la información presentada en la entrada; además se encarga también de encontrar las categorías apropiadas de acuerdo a las correlaciones realizadas con la información de entrada.
- Se codifica la información de entrada, por lo que la salida tiene datos codificados con menos bits, manteniendo la información importante de los datos.

Las características generales y principales de un aprendizaje no supervisado, como señala [40], se describen a continuación:

- No es necesaria la participación de un profesor externo.
- Estas redes pueden auto-organizarse.
- Una RNA descubre por si sola datos importantes para el funcionamiento, en su entrada, información como características, correlaciones y categorías.

- Para su entrenamiento requieren menor tiempo que las supervisadas.
- Poseen una arquitectura simple.

Arquitectura de un Aprendizaje no supervisado

El aprendizaje no supervisado posee una arquitectura con ciertas características; como señala [41], las más importantes son:

- Las células que se encuentran en un entorno cercano se conectan de forma excitatoria.
- Aquellas neuronas que están conectadas con un entorno menos cercano, se conectan de forma inhibitoria.
- Entre las células que se encuentran alejadas, no existe conectividad.
- La distancia entre las células, disminuye la intensidad de la conexión.

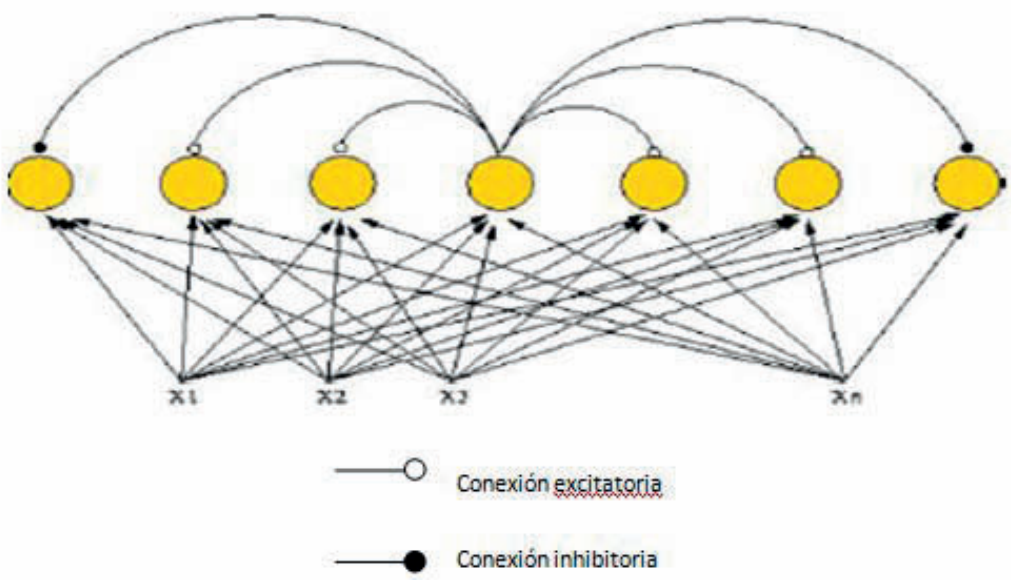


Ilustración 3. 3: Arquitectura de aprendizaje no supervisado, con sus conexiones excitatorias e inhibitorias, dependiendo de la cercanía de conexión [41].

3.2.1 APRENDIZAJE HEBBIANO

En 1949 Hebb postuló este aprendizaje que es la base para muchos otros, y consiste en medir la familiaridad o extraer características de la información de

entrada, ajusta los pesos de las conexiones de acuerdo con la correlación[10] entre los valores de entrada y salida de cada neurona.

Se fundamenta básicamente en: "si dos neuronas Ni y Nj toman el mismo estado simultáneamente (ambas activas o ambas inactivas), el peso de la conexión entre ambas se incrementa" [26].

La idea de Hebb consiste: si las dos unidades son activas, se refuerza la conexión, y por el contrario cuando la una es activa y la otra pasiva, la conexión se debilita [39].

Esta regla es un aporte muy significativo, pues a partir de ella se puede decidir "cómo y en qué grado modificar las conexiones sin depender de factores externos" [41].

3.2.2 APRENDIZAJE COMPETITIVO Y COOPERATIVO.

Dentro de este tipo de redes, se dice que las neuronas compiten y cooperan unas con otras para lograr una tarea dada; es decir cuando a una red se le presenta información de entrada, únicamente una neurona o una por grupo de neuronas es la que presenta la salida o se activa, por lo tanto todas compiten para lograr una vencedora, o una por grupo, que será la neurona que se active, dejando a las otras anuladas con su valor de respuesta mínimo; cabe mencionar que esta competición se realiza en todas las capas de la red. Las conexiones entre neuronas se realizan en todas las capas de la red, y pueden ser inhibitorias (con signo negativo) y excitatorias, entre neuronas vecinas [36].

> *"El objetivo de este aprendizaje es categorizar los datos que se introducen en la red, de esta forma las informaciones similares son clasificadas formando parte de la misma categoría y por tanto deben activar la misma neurona de salida"* [23].

El funcionamiento consiste principalmente en categorizar los datos de la entrada, así la información similar es clasificada dentro de la misma categoría, por lo tanto la neurona de salida que se activa, será la misma. Si un nuevo patrón que se ingresa no pertenece a una categoría reconocida anteriormente, los pesos de la red serán

[10] Hace referencia a la correspondencia o relación recíproca que existe entre los datos de entrada y salida.

ajustados para reconocer una nueva categoría. El peso de una conexión entre dos neuronas i, j será nula si la neurona j no es excitada por la neurona i; y se modificará si la neurona j recibe excitación por parte de la neurona i. La misma red es la que crea las categorías, pues se trata de un aprendizaje no supervisado [36].

Su arquitectura básica consiste en dos capas, como se describe en [21]:

- **F1**.- Es la que recibe los estímulos o entrada que provienen del entorno, y cada neurona está conectada con cada neurona de la capa F2 a través de pesos adaptativos, es decir que son modificados mediante el aprendizaje.
- **F2**.- Es la capa de competición y la que produce la salida de la red; posee conexiones laterales inhibitorias con el resto de neuronas de esa capa y una conexión excitatoria consigo misma. Estas conexiones son fijas y permiten que la neurona que tiene mayor excitación se refuerce más a sí misma a través de autoconexión excitatoria y también lograr que inhiba con más fuerza al resto de neuronas de la capa. Por lo que se dice que existe un proceso de competición en el que cada neurona de la capa compite por aumentar su nivel de activación y reducir el nivel de activación de las restantes.

Las neuronas se refuerzan a sí mismo, las neuronas de la capa F2, una vez que han sido activadas por las entradas de la capa F1, propagan la señal a lo largo de toda la capa; las neuronas compiten entre sí e intentan impedir que las demás tengan un valor alto, este proceso es conocido como conexión inhibitoria, intentan ellas mismas tener un valor alto, lo que se denomina conexión excitatoria [42].

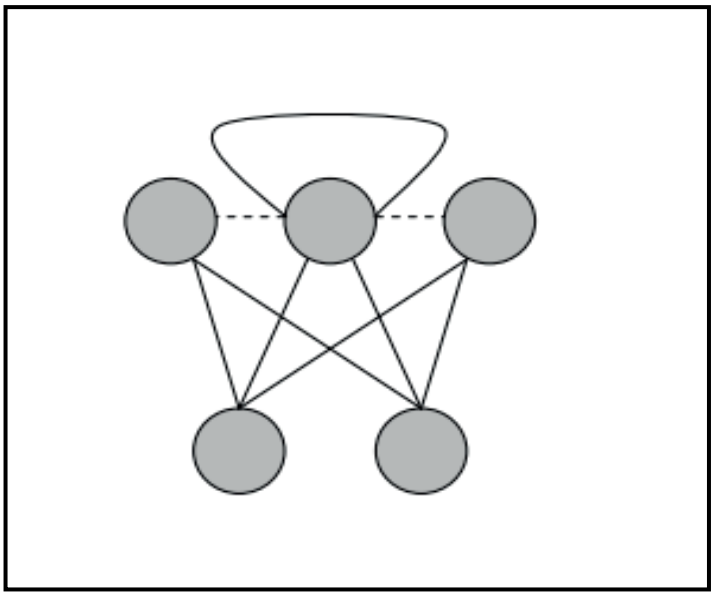


Ilustración 3. 4: Estructura competitiva, con sus dos capas F1 de entrada. F2 de competición y salida con conexiones laterales inhibitorias y conexión excitatoria consigo misma [21].

Algoritmo de aprendizaje

El aprendizaje se describe en [21]:

- Se recibe el estímulo en la capa F1.
- La señal se propaga hasta la capa F2 a través de los pesos adaptativos entre F1 y F2. Cada una de las neuronas F2 se debe calcula su nivel de excitación.
- Comienza el proceso de competición mediante inhibición lateral y autoexcitación.
- Cuando finaliza la competición, se realiza la modificación de los pesos adaptativos que están asociados a la neurona ganadora.

Después de la competición la neurona ganadora es la que mejor se corresponde con el estímulo de entrada; el aprendizaje pretende reforzar esta correspondencia modificando los pesos entre F1 y la neurona ganadora, y así sea más fácil para la neurona ganadora 'reconocer' el mismo estímulo o estímulos parecidos en una entrada posterior [21].

REGLA DE APRENDIZAJE

Existen cuatro tipos principales: minimización del error, Boltzmann, Hebb y competitivo.

Minimización del error.- El ajuste de los pesos trata de que el error que se comete sea mínimo. Dentro de esta clasificación se encuentran aprendizajes como:

Regla del Perceptrón

Método del Descenso por Gradiente

Algoritmo Backpropagation (Aprendizaje offline)

Máquina de Boltzmann.- Se contemplan parámetros aleatorios.

Aprendizaje de Hebb.- Trata sobre la activación de una neurona, cuando una célula dispara a otra y la activa, el peso de la conexión entre ambas tiende a reforzarse.

Competitivo y Cooperativo.- Únicamente aprenden las neuronas que se acercan más a la salida deseada.

Tratamiento de los datos

El tratamiento de los datos es muy importante antes de empezar con el entrenamiento de una red neuronal artificial, debido a que como se presenten los datos a la red, éstos influyen en su respuesta. Existen algunos procedimientos para el tratamiento de los datos.

Las entradas de los datos pueden tener diferentes variaciones de valores viéndose la red influenciada, pues el incremento de los pesos en una neurona es proporcional a su entrada. La normalización ayuda a tener valores parecidos en las entradas, existen algunos métodos para lograrlo, se menciona en [74]:

Método 1: MÁXIMO

Si se tiene un vector DATOS (i), con i=1,……n. El proceso que se debe seguir es el siguiente:

1.- Se debe buscar el máximo del vector DATOS (i).

2.- Normalizamos los datos según la relación:

$$DATOS_{norm(i)} = \frac{DATOS(i)}{Max}$$

Ecuación 3. 1

Cuando los datos originales son positivos, luego de haber normalizado caen dentro del intervalo {0,1}, y si son negativos caen dentro del intervalo {-1,1}.

Método 2: MÍNIMO – MÁXIMO

El método a seguir es el siguiente:

1.- Se busca el máximo y el mínimo del vector DATOS(i)

2.- Los datos son normalizados según la relación:

$$DATOS\_norm(i) = \frac{DATOS(i) - Min}{Max - Min}$$

Ecuación 3. 2

Sin importar cuales sean los datos originales, los datos normalizados caen dentro del intervalo {0,1}. Es importante mencionar que en los datos normalizados siempre habrá un valor que será 0 y otro que será 1.

Método 3: MEDIA - DESVIACIÓN TIPO

Utilizando este método se obtienen datos normalizados dentro del intervalo {-1,1}o dentro del intervalo {0,1}. Inicialmente se calcula:

$$DATOS\_norm(i) = \frac{DATOS(i) - \mu}{\sigma}$$

Ecuación 3. 3

Los datos normalizados caen dentro del intervalo {-k1,k2}, siendo k1 y k2 números reales positivos y μ, σ son la media y la desviación tipo de los datos de entrada, respectivamente. Si es necesario que los datos caigan dentro del intervalo {-1,1} se dividen para el mayor de k1 o k2.

Para que los datos normalizados caigan dentro del intervalo {0,1} se hace lo siguiente:

A los datos normalizados se les suma el mayor de k1 y k2. Entonces estos datos caen dentro del intervalo {0,k3}, donde k3 = K1+K2.

Los datos normalizados se dividen por k3 y así se obtiene el intervalo {0,1}.

Codificación de los datos

En los datos de entrada numéricos por ejemplo longitud, peso, no existe inconveniente ya que se podría normalizar; ahora cuando los datos son discretos como el sexo por ejemplo, no se recomienda codificar hombre/mujer a 0/1 debido a que la actualización de los pesos depende de estas entradas y con estos valores no existiría actualización. Entonces lo más adecuado sería utilizar codificaciones como +1/-1 [21].

Las salidas se deben codificar y es mucho más recomendable que hacerlo en las entradas; por ejemplo: si es necesario clasificar patrones en dos clases podemos colocar una neurona a la salida y codificar como si perteneciera a una clase o a otra dependiendo si la salida tiene valores cercanos a 1,0 ó +1,-1 dependiendo de la función de activación que se use (sigmoide o tangencial).

Información en los patrones

- Pueden faltar datos de entrada a la red, entonces se puede realizar:
- Eliminar los datos no completos.
- Dar un valor a los datos que faltan, similares a los del resto.

Extracción de características

- Selecciona las entradas realmente necesarias, para ello se debe realizar dos tareas:
- Extracción de las características más relevantes.
- Eliminación de redundancia entre datos.

Consistencia de los datos

Existen tres formas de eliminar las inconsistencias, como se describe en [21]:

- Mediante inspección visual si el problema no es muy complejo.
- Mediante histogramas.
- Mediante la colocación de umbrales máximo y mínimo que invaliden cualquier dato fuera de ellos.

CAPÍTULO IV

TIPOS DE HERRAMIENTAS

CAPITULO IV

TIPOS DE HERRAMIENTAS.

Los simuladores permiten observar el funcionamiento de una red neuronal artificial sin tener que realizar la programación de algoritmos de cálculo. Mediante interfaz gráfica o línea de comando se elige la red a construir: el número de capas, el algoritmo de aprendizaje que se crea conveniente y el tipo de neuronas; el simulador realiza todo el trabajo de inicio y entrenamiento de la red [52].

4.1 FUNEGEN.

Es una herramienta basada en sistemas neurodifusos[11], que permite generar clasificadores difusos desde muestras de datos (archivos de entrenamiento) sin ayuda experta, para aplicaciones del mundo real; es capaz de manejar cualquier número de entradas y salidas, y si existe entradas redundantes permite eliminarlas de forma automática, esto quiere decir puede reducir la función de entrada posterior. El rendimiento del sistema difuso extraído se puede indicar mediante el uso de una nueva muestra de datos (archivos de prueba) [45].

El sistema extrae una base de conocimientos difusa en la primera fase; en la segunda fase el sistema difuso extraído es puesto a punto. El número máximo de reglas a ser considerado para la solución final puede estar limitado por el usuario; si este es menor que el número extraído de reglas puestas a punto en la segunda fase los nodos adicionales de las reglas más débiles son eliminados.

Con la herramienta Funegen es posible generar programas correspondientes al lenguaje de programación C, y su aprendizaje puede ser realizado mediante la aplicación de patrones y épocas [47].

Desarrollador: Darmstadt University of Tech [45].

[11] Los sistemas neurodifusos son una combinación de redes neuronales, que buscan emular el funcionamiento del cerebro humano; y sistemas difusos que expresan el conocimiento de un humano, mediante reglas if-then simples

4.2 LVQPAK.

Learning Vector Quantization, aprendizaje de cuantificación vectorial fue desarrollado por Teuvo Kohonen, es un paquete o un conjunto de métodos que se utiliza para la clasificación estadística y reconocimiento de patrones de muestras estocásticas vectoriales, en los cuales las clases son descritas por un número relativamente pequeño de vectores código.

Existen tres principales variantes de este paquete LVQ1, LVQ2 y LVQ3. Es recomendable para el aprendizaje que se inicie siempre con el algoritmo optimizado LVQ1 que tiene una convergencia muy rápida y a menudo su fase de aprendizaje puede ser suficiente para las aplicaciones prácticas. Sin embargo si se desea mejorar la precisión del reconocimiento se puede continuar con el LVQ2 o el LVQ3 utilizando un valor inicial bajo de tasa de aprendizaje, que sería el mismo para todas las clases [68].

Un entrenamiento en un paquete LVQ es mucho más rápido que el que se realiza con otras redes y alcanza el 100% en una tasa de reconocimiento después de un número considerado de iteraciones. Sin embargo este paquete no ofrece salidas que permitan rastrear el error durante el entrenamiento.

Para descargarse el programa se lo puede realizar desde la consola de Linux; todos los programas y la documentación están almacenados en el directorio /pub/lvq_pak, los archivos se encuentra en varios formatos para facilitar su descarga y compilación. El directorio /pub/lvq_pak contiene los siguientes archivos:

README.- Contiene una descripción corta de lvq_pak
Lvq_doc.ps.- Este documento está en formato PostScript
Lvq_doc.ps.Z.- El mismo formato anterior pero comprimido.
Lvq_p3r1.exe.- Para el sistema operativo MSDos
Lvq_pak-3.1.tar.- Para el sistema operativo Unix.
Lvq_pak-3.1.tar.Z.- El mismo archivo que el anterior pero comprimido.

El archivo .tar contiene el código fuente, archivos de compilación y los conjuntos de datos de ejemplo del paquete.

Los parámetros más utilizados, como se menciona en [68], son:

- -noc: Número de vectores del libro de códigos.
- -din: nombre del archivo de entrada de datos.
- -dout: nombre del archivo de datos de salida.
- -cin: nombre del archivo desde el que los vectores del libro de códigos se leen.
- -cout: nombre del archivo en el que los vectores del libro de código se almacenan.
- -rlen: tamaño de ejecución (número de pasos) en el entrenamiento.
- -alpha: tasa inicial de aprendizaje.
- -knn: número de vecinos usado en la clasificación knn.
- -cfout: nombre del archivo de información de la clasificación.

Existen versiones para UNIX y MS-DOS
Desarrollador: Teuvo Kohonen, Helsinki University of Technology; Finlandia [45].

4.3 NEUROFORECASTER/GA.

Es una red neuronal de 32 bits y algoritmos genéticos; avanzada, fácil de usar e inteligente herramienta de pronóstico para redes neuronales. Es orientada a las finanzas y los negocios, pues está basada en programas de predicción. Es un simulador comercial. Como se menciona en [66], sobresale en las siguientes aplicaciones:

- Pronóstico tiempo-serie, por ejemplo pronósticos de valores y monedas de mercado, previsión del PIB.
- Clasificación, por ejemplo selección de valores, índice de bonos, asignación de créditos, valoración de la propiedad.
- Análisis de Indicadores: identificación de indicadores de aportaciones útiles.

Características de Neuro Forecaster/GA

Las características principales de este simulador, se menciona en [66]:

- Aplicación completa para Windows 32 bits, es in simulador comercial.
- Para propósitos generales de negocios y previsiones financieras.
- Realiza análisis de series temporales y de indicadores.

- Doce paradigmas de redes neuronales.
- Posee las funciones AutoTest, AutoSave y AutoStop.
- Construido en hoja de cálculo compatible con Excel para la fácil manipulación de datos.
- Lee algunos formatos de datos como en ASCII y en Excel.
- Fácil de instalar y ejecutar.
- Viene con algunos ejemplos proporcionados.

Desarrollador: NIBS Inc., [45].

4.4 NN/XNN.

NN es un lenguaje de alto nivel para la especificación de las redes neuronales: diseño, estudio y simulación. Su compilador permite generar código en C. La programación no es necesaria si se utiliza los modelos que se incluyen en el sistema.

Se compone de tres partes, como se menciona en [67]:

- nn, un lenguaje de alto nivel para la especificación de redes neuronales.
- xnn, una interfaz gráfica,
- Netpack, un número de algoritmos de red implementados en nn.

Algoritmos: Madaline, Backpropagation, ART1, counterpropagation, Elman, GRNN, Hopfield, Jordan, LVQ, Perceptron, Redes de base radial, Mapas de Kohonen.
Su licencia es tipo GPL Desarrollador: Neureka ANS, Solheimsviken, Norway [45].

Es ejecutable en los sistemas UNIX y MS-DOS, gratis los 30 primeros días, con todas sus características.

4.5 MÁQUINAS DE SOPORTE VECTORIAL

Las máquinas de soporte vectorial son un conjunto de algoritmos de aprendizaje supervisados desarrollado por Vladimir Vapnik y su equipo en los laboratorios AT&T [70].

Es un método utilizado para la clasificación; su objetivo principal consiste en encontrar un hiperplano que divida los datos en dos clases, y luego en la clasificación observar en que parte se encuentra la mitad de los datos que puedan representar una nueva muestra no conocida anteriormente, si las muestras tienen una buena

separación, permitirá una clasificación correcta. Un ejemplo de red a entrenar en este tipo de herramientas es el Perceptrón [49].

Están consideradas como clasificadores lineales y son aplicadas en un sinnúmero de activiades, dentro de la Visión Artificial se usa para el reconocimiento de caracteres, rostros, matrículas y objetos [25].

Cuando las clases no se pueden separar, se utilizan cotas de error para identificar las muestras de entrenamiento como pertenecientes a la clase incorrecta; y en otros casos, si las muestras no son linealmente separables, se introduce la utilización de kernels con el objetivo de transformar el conjunto de datos a un espacio donde sea separable o se encuentre bajo una cota de error aceptable; y así poder realizar la clasificación [49].

Para descargarse el paquete SVM para Linux, se lo puede hacer desde el siguiente enlace [71]. Consta de un módulo de aprendizaje (svm_learn) y un módulo de clasificación (svm_classify), el módulo de clasificación se puede utilizar para aplicar el modelo aprendido a nuevos ejemplos.

El módulo svm_learn es llamado con los siguientes parámetros:
svm_learn [options] example_file model_file. Existen diferentes opciones que se describen en [72]. El archivo de entrada example_file contiene los ejemplos para el entrenamiento, la primera línea puede contener comentarios y son ignorados si ellos empiezan con # . Los ejemplos de prueba en example_file están en el mismo formato que los archivos de entrenamiento.

4.6 FANN / JAVANNS

FANN Fast Artificial Neural Network, es una librería de código abierto para trabajar con redes neuronales artificiales multicapa y totalmente conectadas; es fácil de usar, rápida, muy bien documentada y está disponible en algunos lenguajes de programación como C y Python. Fue desarrollada por Steffen Niessen en el año 2003 en la Universidad de Copenhague [50].

Características de FANN

El simulador FANN posee algunas características importantes, como se menciona en [65]:

- Biblioteca en C para redes neuronales artificiales multicapa.
- Entrenamiento Backpropagation.
- Fácil de usar: crea, entrena y ejecuta una RNA
- Rápido, ejecución de hasta 150 veces más rápido que otras bibliotecas.
- Versátil, puede ajustar muchos parámetros y características.
- Multiplataforma, scripts de configuración para linux y unix, archivos DLL para Windows, archivos de proyecto para MSVC ++ y compiladores de Borland.
- Varias funciones de activación diferente implementadas.
- Fácil de usar y cargar RNAs enteros.
- Varios ejemplos fáciles de usar.

Es posible descargarse FANN desde [65].

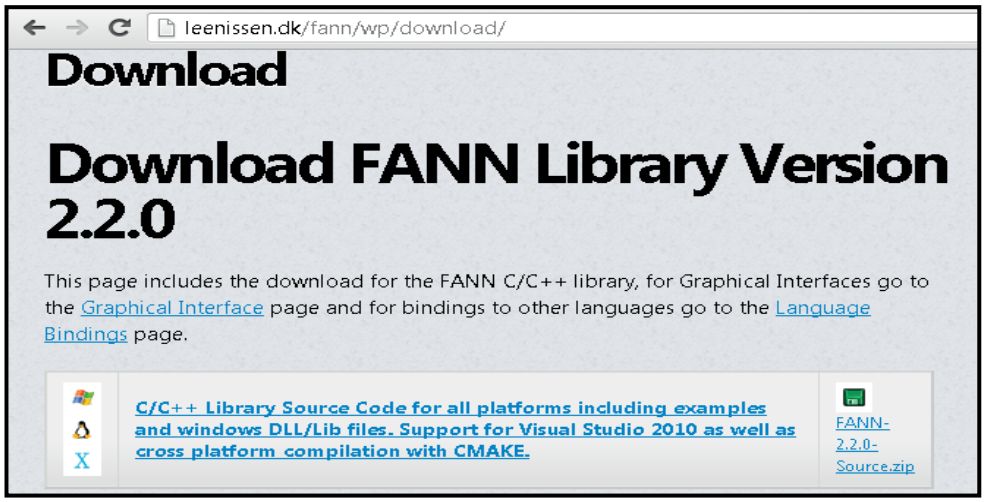


Ilustración 4. 1: Sitio de descarga de la librería FANN [65].

Instalación de FANN

- Copiar el instalador FANN.
- FANN está disponible bajo los términos de la licencia GNU (Licencia Pública General)

- El sistema de instalación FANN está basado en CMAKE, una vez instalado es posible ejecutar los siguientes comandos en el directorio FANN, para compilarlo e instalarlo:

  cmake .

  sudo make install

Una vez instalado, se puede probar la librería utilizando los ejemplos; antes debemos compilarlos escribiendo make en su directorio.

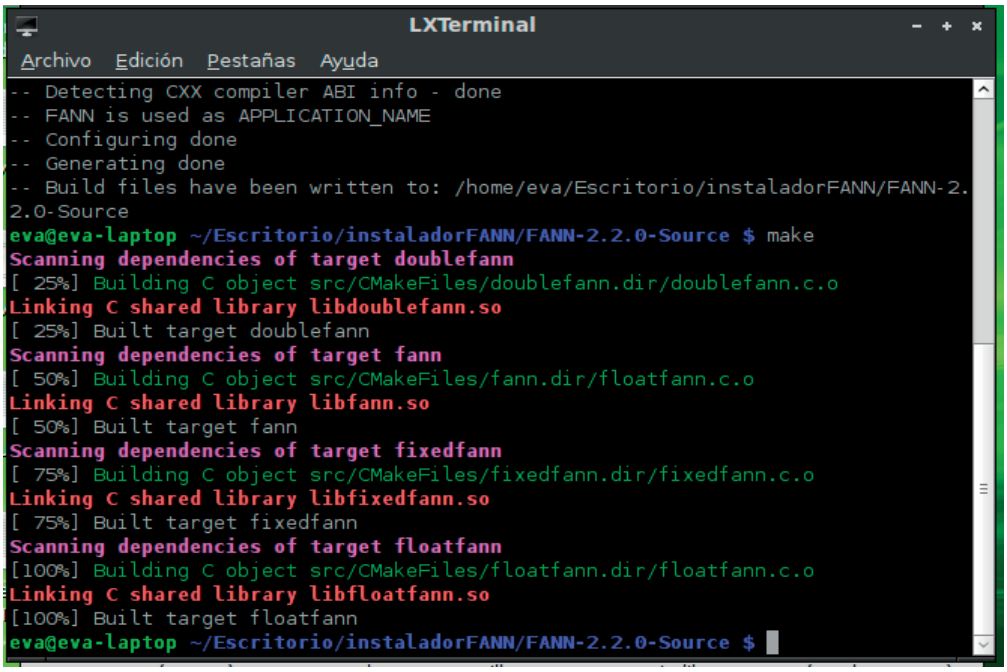


Ilustración 4. 2: Ventana que muestra la compilación del código fuente de FANN.

Una red neuronal se suele ejecutar en dos modos: un modo de entrenamiento y otro de ejecución. A continuación se presenta un codigo que entrena la red con un conjunto de datos, como se muestra en [65]:

Ejemplo de entrenamiento

```
“#include "fann.h"
int main()
{
    const unsigned int num_input = 2;
    const unsigned int num_output = 1;
    const unsigned int num_layers = 3;
```

```
    const unsigned int num_neurons_hidden = 3;
    const float desired_error = (const float) 0.001;
    const unsigned int max_epochs = 500000;
    const unsigned int epochs_between_reports = 1000;
    struct fann *ann = fann_create_standard(num_layers, num_input,
        num_neurons_hidden, num_output);
    fann_set_activation_function_hidden(ann, FANN_SIGMOID_SYMMETRIC);
    fann_set_activation_function_output(ann, FANN_SIGMOID_SYMMETRIC);
    fann_train_on_file(ann, "xor.data", max_epochs,
        epochs_between_reports, desired_error);
    fann_save(ann, "xor_float.net");
    fann_destroy(ann);
    return 0;
}"
```

El ejemplo también nos muestra un ejemplo de archivo que se usa para el entrenamiento de la red:

```
4 2 1
-1 -1
-1
-1 1
1
1 -1
1
1 1
-1
```

La primera línea consta de tres números: el primero es el número de pares de entrenamiento en el archivo, el segundo valor es el número de entradas y la tercera es el número de salidas. El resto del archivo son los datos de entrenamiento que consiste en una línea de entrada y otra línea de salida.

Ejemplo de Ejecución de FANN

```
“#include <stdio.h>
#include "floatfann.h"
int main()
{
   fann_type *calc_out;
   fann_type input[2];
   struct fann *ann = fann_create_from_file("xor_float.net");
   input[0] = -1;
   input[1] = 1;
   calc_out = fann_run(ann, input);
   printf("xor test (%f,%f) -> %f\n", input[0], input[1], calc_out[0]);
   fann_destroy(ann);
   return 0;
}”
```

JavaNNS

JavaNNS cuenta con un núcleo implementado en C, es un simulador de redes neuronales artificiales desarrollado por la Universidad de Tübingen, siendo evolución de la anterior herramienta SNNS [51].

El simulador es software libre y se distribuye en forma de fichero comprimido tar.gz para Linux; una vez que se descomprime genera algunos ficheros y carpetas, como se menciona en [52], son:

JavaNNS.jar: es un archivo java que contiene las clases de interfaz de usuario.
Examples: carpeta de ejemplos.
Manual: carpeta con el manual.

Instalación de JavaNNS

Para poder utilizar JavaNNS es necesario tener instalado el JDK. El paquete JavaNNS.jar contiene las clases de usuario, interfaz gráfica; la carpeta de ejemplos que además tiene patrones y la carpeta que contiene el manual.

Cuando se ejecuta por primera vez JavaNNS, se crea una copia del núcleo del simulador en la carpeta que el usuario elija; la dirección de la librería se almacena en el fichero JavaNNS.properties en el directorio personal del usuario. Esta operación únicamente se realiza la primera vez, y luego el sistema operativo arranca la interfaz gráfica basada en Java, una ventana con las herramientas necesarias que contendrá las redes neuronales.

Para correr JavaNNS solo se necesita escribir dentro del directoio donde se encuentra el JavaNNS: java –jar JavaNNS.jar

Creación de una red neuronal

En JavaNNS, la operación crear una red, consiste en especificar su arquitectura: seleccionar el número de capas y neuronas, la conectividad entre capas; este proceso se lo hará mediante la ventana Create Layers que se encuentra en la pestaña Tools, en esta misma ventana podemos seleccionar la anchura y la altura de cada capa de red, el tipo de neuronas que la componen la función de activación y de salida [52]. Además es necesario observar el progreso de entrenamiento a través de la ventana Error Graph, información numérica mediante el Log Window que se encuentran en el menú View.

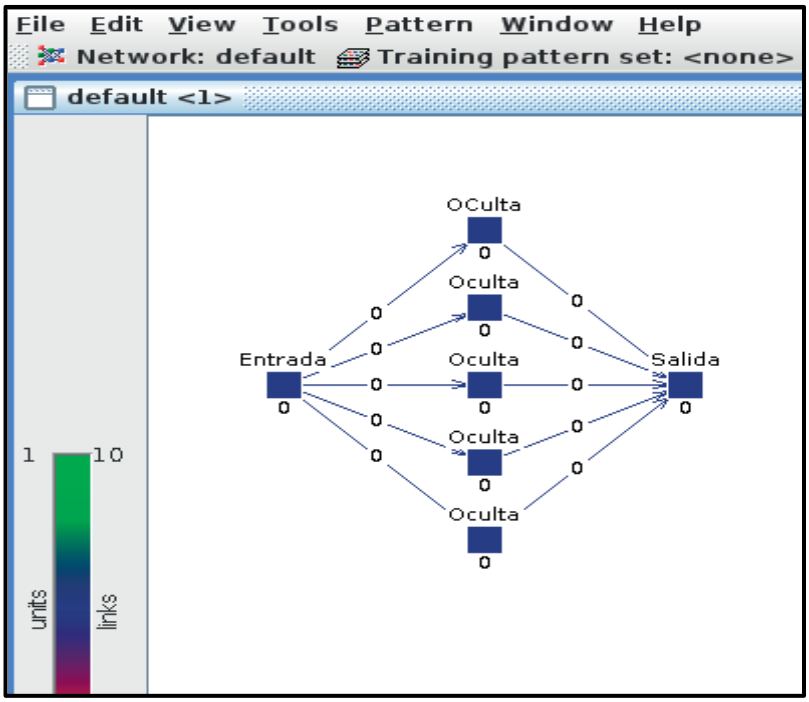


Ilustración 4. 3: Red de tres capas creada en JavaNNS con todas sus conexiones.

Entrenamiento de la red

Para comenzar con el aprendizaje de la red, antes es necesario cargar el conjunto de entrenamiento; seleccionamos la opción Open dentro del menú File, se escoge el directorio examples donde el fichero letters.pat podría ser utilizado; se debe tomar especial atención a las especificaciones, en especial aquellas que se encuentran en la cabecera, pues como se menciona en [52], nos indican: el número de patrones que contiene el fichero, el número de entradas y de salidas; posteriormente se ubican de forma alternada los valores de entrada y salida. Cuando ya han sido cargados los patrones, se selecciona el algoritmo de entrenamiento que se desea, dentro de la opción panel de control en la pestaña tools.

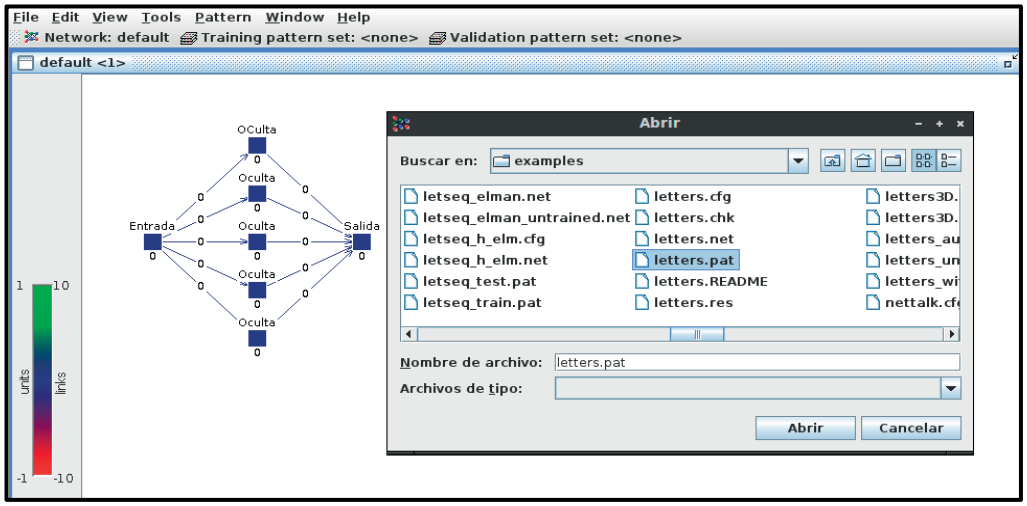


Ilustración 4. 4: Ventana de JavaNNS cargando datos para el entrenamiento.

El panel de control consta también de seis pestañas que pueden estar o no activas, como indica [52], son: Initializing permite elegir el modo de inicialización de los pesos; Updating que define el orden de actualización de los pesos; Learning selecciona el algoritmo de aprendizaje y los parámetros que lo caracterizan; Pruning permite eliminar selectivamente sinapsis; Patterns nos ayuda a definir de los ficheros de patrones que hemos abierto, cuales se van a emplear como conjunto de entrenamiento o validación; Subpatterns que se activará cuando se ha definido subpatrones en el conjunto de entrenamiento.

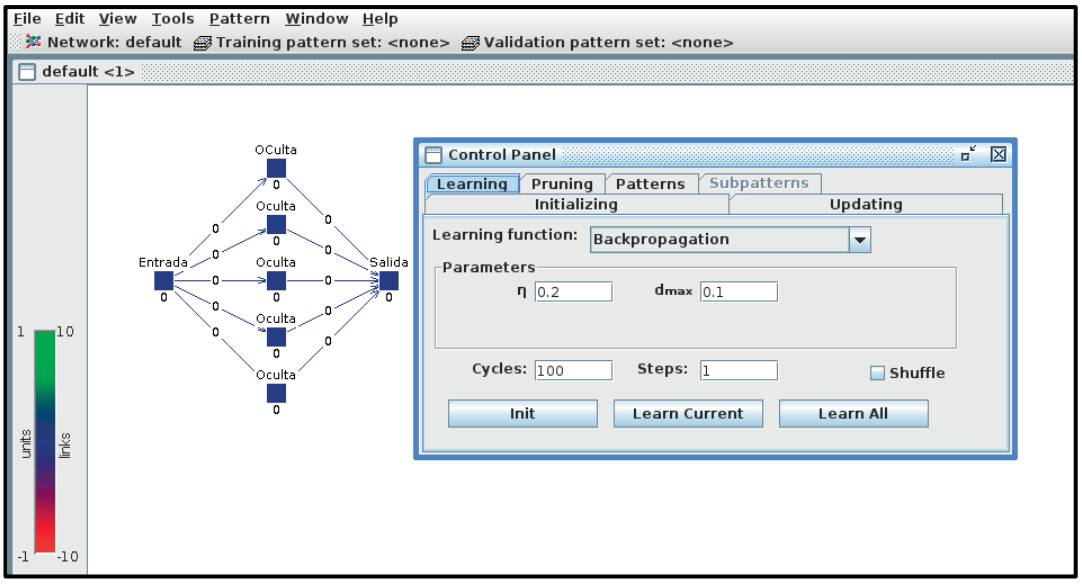


Ilustración 4. 5: Ventana que indica el Control Panel de JavaNNS.

Análisis de la red

Para el análisis y rendimiento de la red existen herramientas como: Analyzer en el menú Tools y Projection y Weights en el menú View.

4.7 OTRAS HERRAMIENTAS

LIBRERÍA NEUROLAB

Es una herramienta sencilla y potente de redes neuronales para python.

Entre las principales características, como se describe en [85] tenemos:

Python puro mas la librería numpy[12].

Configuraciones flexibles de red y algoritmos de aprendizaje.

En esta herramienta contiene las arquitecturas de red básicos:

Tipo de **Red**	**Función**
Perceptrón simple	newp
Perceptrón multicapa	newff
Redes competitivas	newc
LVQ	newlvq
Elman	newelm
Hopfield	newhop

Tabla 4. 1: Contiene las diferentes redes con su función para la librería NeuroLab. [85]

[12] Es es una extensión de Python que permite manipular de manera rápida y eficiente arreglos numéricos, tales como vectores y matrices, así como arreglos multidimensionales de rango arbitrario [86].

Por ejemplo para crear la red de Hopfield se tendría lo siguiente:

neurolab.net.newhop(target, transf=None)

Parámetros:

Target: es un arreglo de patrones de entrenamiento.

Trans: Función de activación, por defecto es Hardlims.

OPEN CV

Open Source Computer Vision, es una librería de libre distribución para el procesamiento de imágenes. Esta herramienta se ejecuta en diferentes plataformas como: Microsoft Windows, GNU/Linux y soporta algunos lenguajes de programación, entre ellos C, C++ y Python.

Es de gran utilidad en cuanto al tratamiento de imágenes, debido a que posee mas de 2000 algoritmos estandarizados para realizar distintas operaciones sobre las imágenes [84].

Como se menciona en [84], la estructura de Open está basada en 5 librerías que se detallan a continuación:

CXCORE: Posee las estructuras y algoritmos básicos que usan las demás funciones.

CV: Implementa las funciones principales de procesamiento.

HighGUI: Sirve para realizar la interfaz gráfica e interacción con archivos de imágenes y videos.

ML: Posee algoritmos de aprendizaje y clasificadores.

CVAux: Admite algunos algoritmos experimentales.

La librería OpenCV permite trabajar con varios formatos de imágenes: BMP, JPEG, JPG, JPE, PNG, PBM, PGM, PPM, SR, RAS, TIFF, TIF

Esta herramienta se ha utilizado para el tratamiento de imágenes necesario antes de aplicar la red neuronal Hopfield.

Función Dilate

Esta función provoca en imágenes blanco y negro el crecimiento de las regiones de píxeles negros, es decir engrosa las áreas negras de acuerdo a cierto patrón [83].

MATLAB Neural Network Toolbox

Es considerada una herramienta muy confiable, capaz de acelerar las investigaciones, reducir costos de proyectos, y producir soluciones eficaces; además que su ambiente fomenta la creatividad, permite probar y comparar múltiples alternativas.

Las funciones incorporadas de matemáticas y gráficos, la interfaz y su lenguaje hacen de Matlab una plataforma preferida por los usuarios [46].

Características

El software posee ciertas características que lo hacen muy utilizado en distintas áreas como la ingeniería y la ciencia; a continuación en [46], se detallan las más importantes:

Matlab integra: procesador matemático, la visualización y un lenguaje técnico de gran alcance.

El procesador numérico permite obtener los resultados rápidos y exactos.

Mediante el uso de gráficos se puede observar y analizar los datos.

Permite crear interfaces a los lenguajes externos como: C, C++, Fortran y Java.

Matlab es un lenguaje de alto rendimiento e iterativo que integra el computo, la visualización y la programación en un ambiente fácil de utilizar.

Sistema de Matlab

El sistema de Matlab está compuesto por cinco partes principales, que se describen en [46]:

Ambiente de Desarrollo: Aquí encontramos las herramientas y las instalaciones que ayudan a utilizar funciones y archivos de Matlab.

La Biblioteca matemática de la función de Matlab: Es un conjunto extenso de algoritmos que contiene funciones elementales como la suma, seno, coseno y funciones más sofisticadas como la transformada rápida de Fourier.

El Lenguaje de Matlab: Es un lenguaje de alto nivel que posee algunas características importantes como: declaraciones de control del flujo, las estructuras de datos, la entrada-salida, y la programación orientada a objetos.

Manejador Graphics: Contiene los gráficos de Matlab; los comandos de alto nivel para visualizar datos, procesamiento de la imagen, la animación, gráficos de dos y tres dimensiones; como los comandos de bajo nivel que permiten que se pueda modificar gráficos de acuerdo a requisitos particulares.

El Application Program Interface de Matlab (Api): Es una biblioteca que permite escribir programas de C y Fortran, los cuales trabajan recíprocamente con Matlab.

Caja de Herramientas de Redes Neuronales

Matlab posee una importante Caja de Herramientas de Redes Neuronales (Neural Network Toolbox), que extiende el ambiente de aplicaciones, y permite diseñar, poner en práctica, visualizar y simular redes neuronales. Esta herramienta proporciona ayuda mediante una interfaz gráfica simple para diseñar y manejar una RNA; además posee un diseño modular abierto y extensible que facilita la creación de funciones y de redes con requisitos particulares [46].

Características de Neural Network Toolbox

Las principales características de la caja de herramientas se describen a continuación en [46]:

Posee una interfaz gráfica GUI para crear, entrenar y simular redes neuronales.

Información sobre arquitectura de red supervisada y no supervisada de ayuda para el usuario.

Sistema de entrenamiento y de funciones de aprendizaje.

Algunos bloques de Simulink[13], documentación y demostración de los usos del sistema.

Generación automática del Simulink, modelando una RNA.

Representación modular de la red, con un número ilimitado de entradas e interconexiones de la red.

Construcción de una Red Neuronal Artificial

Lo primero que se debe realizar es la creación de la RNA; la función newff crea una red con conexión hacia adelante, esta función necesita cuatro parámetros y devuelve un objeto del tipo red; como se describe en [46], estos parámetros son:

La primera entrada debe ser una matriz de (p-1) * 2 XR de mínimos y máximos valores por cada elemento (p-1) del vector de entrada.

La segunda entrada debe ser un arreglo con el tamaño de cada capa.

La tercera entrada es un arreglo que contiene los nombres de las funciones de transferencia que se van a utilizar en cada capa.

Y la última entrada contiene el nombre del algoritmo de entrenamiento que se va a utilizar.

[13] "**Simulink** es un entorno de programación visual, que funciona sobre el entorno de programación Matlab" [48].

A continuación se presenta un ejemplo: una red que tendrá dos capas, un vector de entrada con dos elementos p-1 = 2, tres neuronas en la primera capa L-1 = 3, y una neurona en la segunda capa, en este caso capa de salida m = 1. Las funciones de transferencia que se utilizarán son: en la primera capa será la tansig, y la función de la capa de salida será lineal. El primer elemento del vector de entrada va a contener valores entre el rango -1 y 2, el segundo elemento del vector tendrá valores entre 0 y 5; entonces la matriz XR quedaría de la forma: XR = [-1 2; 0 5]. Finalmente la función de entrenamiento es traingd [46].

El comando que crea la red es:

net = newff (XR, [L-1, m],'tansig','purelin','traingd')

Este código crea un objeto del tipo red, inicializa los pesos y el bias, usando por defecto el comando initnw, luego de esto la red está lista para ser entrenada. Si es necesario re-inicializar los pesos o cambiar la iniciación por defecto, se utiliza el comando init: $net = init\ (net)$ el cual toma como entrada un objeto del tipo red y devuelve un objeto del tipo red con los pesos y bias reinicializados [46].

Para simular una red neuronal artificial, se usa el comando sim, que trabaja con dos parámetros: la entrada de la red x, y el objeto de red net; devolviendo una salida de la red Y; como se muestra en [46], el comando sim utilizado para simular la red creada anteriormente para un vector de entrada simple:

$x = [1\ ;\ 2\ ]$ Ecuación 4. 1

$a = sim\ (net, x)$ Ecuación 4. 2

$a = -0.1011$ Ecuación 4. 3

CAPÍTULO V

DESARROLLO DE UN PROCESO DE EXPERIMENTACIÓN EMPLEANDO ALGUNOS TIPOS DE REDES NEURONALES

CAPÍTULO V
DESARROLLO DE UN PROCESO DE EXPERIMENTACIÓN EMPLEANDO ALGUNOS TIPOS DE REDES NEURONALES

5.1 SELECCIÓN DEL CORPUS

Para el desarrollo de este proceso de experimentación se han seleccionado algunos corpus que contienen información clasificada para el aprendizaje de las redes.

Corpus Iris.- Este corpus contiene 150 datos clasificados de acuerdo a tres tipos de flores, cada uno con 50 datos: setosas, versicolor y virginicas

La información contenida en este corpus representa:

- ✓ Largo y ancho de los pétalos: las dos primeras variables.
- ✓ Largo y ancho de los sépalos: las dos variables siguientes
- ✓ Tipo de flor: el último valor.

Antes de empezar con el aprendizaje de la red; los datos del corpus deben ser tratados, pues como estos se presenten a la red, influyen en su respuesta.

Codificación de salidas.- Para el corpus Iris se ha tenido que codificar los datos de salida con valores 0 y 1 los resultados se muestran en la tabla 5.1.

Setosa	0	0	1
Versicolor	0	1	0
Virgínica	1	0	0

Tabla 5. 1: Muestra los tipos de flores codificados.

El siguiente código se utilizó para codificar las salidas, desarrollado en lenguaje de programación Python.

```
corpus = open("iris2.data","r")
nuevocorpus = open("nuevoiris.data","w")
linea = corpus.readline()
```

```
valormaximo = 0;
numerolinea = 0; # Se usa para saber que cada 50 de estos cambiamos de flor.
while linea: #Mientras existan lineas en el archivo
valores = linea.split(",") # se corta cada línea que leo separado por las comas y se guarda en valores
print(valores)
for i in range(0,5): # Se recorre mediante un for cada uno de los valores
#print(valores[i])
if i <= 2:
nuevocorpus.write(valores[i]+",") #En el nuevo corpus se agrega los valores y se los vuelve a separar con comas
if i == 3:
nuevocorpus.write(valores[i]) #Si es el último valor ya no se pone la coma
if i == 4:
if(numerolinea > 0 and numerolinea < 50): #Si se encuentra en las líneas entre 0 y 50 pertenecientes al primer tipo de flores
nuevocorpus.write("\n0,0,1\n") #Entonces se agrega un enter, el código 0,0,1 y nuevamente un enter para seguir con los demas pares. Se hace lo mismo con los demás tipos de flores.
if(numerolinea > 50 and numerolinea < 100):
nuevocorpus.write("\n0,1,0\n")
if(numerolinea > 100):
nuevocorpus.write("\n1,0,0\n")

numerolinea = numerolinea + 1 # Actualiza el contador de número de linea en el que se encuentra
linea = corpus.readline() #Se lee la siguiente linea

print "termine y soy feliz"
print valormaximo
```

Normalización de entradas.- En cuanto a las entradas, se ha normalizado utilizando el método 2 que se describe en el capítulo III, el cual nos devuelve valores en un rango [0,1] debido a que la función de activación que se usó es Sigmoide.

Formato de los datos.- El formato de los datos es en pares de entrada y salida pues el simulador que se utiliza los requiere de esa forma, como se observa en la ilustración 5.1.

Ilustración 5. 1: Muestra los datos del corpus organizados en pares de entrada-salida. La primera fila indica el número de datos,el número de variables de entrada y el número de salida.

División del conjunto de entrenamiento y de prueba.- Se divide el conjunto de datos en dos subconjuntos: Entrenamiento y prueba; el primer subconjunto tendrá el 70% de los valores del conjunto de datos, el 30% restante para el subconjunto de prueba.

Este tratamiento es utilizado para todos los corpus que se entrenan en Perceptrón y en simulador FANN. En algunos casos, cuando los datos son pocos y se encuentran claramente separados por clases, no se ha realizado normalización, presentando buenos resultados.

Corpus Iris para el algoritmo LVQ.-

Se utiliza el mismo porcentaje de datos para prueba y entrenamiento. Para estos algoritmos es necesario que los datos se encuentren organizados: primero las entradas y al final la salida; al comienzo de cada archivo debe contener un valor que indique la dimensión de las variables de entrada; en la salida no es necesaria su binarización. A continuación se presenta un ejemplo:

```
4
5.1 3.5 1.4 0.2 Iris-setosa
4.9 3.0 1.4 0.2 Iris-setosa
4.7 3.2 1.3 0.2 Iris-setosa
4.6 3.1 1.5 0.2 Iris-setosa
7.0 3.2 4.7 1.4 Iris-versicolor
6.4 3.2 4.5 1.5 Iris-versicolor
6.9 3.1 4.9 1.5 Iris-versicolor
5.5 2.3 4.0 1.3 Iris-versicolor
7.9 3.8 6.4 2.0 Iris-virginica
6.4 2.8 5.6 2.2 Iris-virginica
6.3 2.8 5.1 1.5 Iris-virginica
6.1 2.6 5.6 1.4 Iris-virginica
```

Se debe realizar el mismo procedimiento para todos los corpus que se usen en el algoritmo LVQ.

Corpus Clases de Vinos.- Determina la cantidad de 13 componentes encontrados en cada uno de los tres tipos de vinos que existen. Está dividido en 3 clases diferentes:

- Clase 1: 59 vinos.
- Clase 2: 71 vinos.
- Clase 3: 48 vinos.

Las primeras trece variables son componentes de los vinos y la última indica la clase a la que pertenece, estos son:

- Variable 1: Alcohol
- Variable 2: Ácido málico
- Variable 3: Ceniza

- Variable : Alcalinidad de cenizas
- Variable 5: Magnesio
- Variable 6: Fenoles totales.
- Variable 7: Flavonoides
- Variable 8: Fenoles nonfavanoides
- Variable 9:Proantocianinas
- Variable 10: Intensidad del color
- Variable 11: Matiz
- Variable 12: Vinos diluidos
- Variable 13: Prolina
- Variable 14: Es la salida que representa la clase de vino a la que pertenece.

El tratamiento que se le ha dado al corpus es el explicado anteriormente en el caso del corpus Iris, pues se trata de entrenarlo en la red Perceptrón y el simulador FANN.

Corpus de imágenes Wang.- Es el mismo que se utilizó en la tesis del compañero Javier Poveda, el cual ya está cuantizado en espacio de color HSV y RGB; este corpus llamado Wang posee 1000 imágenes clasificadas en 10 clases con 100 imágenes cada una, estas clases son:

- Clase 0: África
- Clase 1: Playas
- Clase 2: Monumentos
- Clase 3: Buses
- Clase 4: Dinosaurios
- Clase 5: Elefantes
- Clase 6: Flores
- Clase 7: Caballos
- Clase 8: Montañas
- Clase 9: Comida

Este corpus se puede descargar desde [80].

Para el tratamiento de este corpus de imágenes, como se describe en [81] se ha seguido un procedimiento:

1. **Extracción de las características de las imágenes.-** Se extrae las características de las imágenes en espacio de color RGB, pues a partir este se convertirá a otros histogramas de color HSV.
2. **Convertir de espacio de color RGB a HSV.-** RGB tiene una estructura rojo, verde y azul por lo que puede formar una gran gama de colores, pero no tiene el nivel de precisión del ojo humano para detectar los colores de su medio ambiente, por lo que se ha desarrollado el histograma HSV que si se aproxima a la precisión del ojo humano; mediante las siguientes fórmulas:

$$H = \cos^{-1}\left\{\frac{0.5[(R-G)+(R-B)]}{\sqrt{(R-G)^2+(R-B)(R-G)}}\right\}$$

Ecuación 5. 1

$$s = 1\frac{3}{R+G+B}[min(R,G,B)]$$

Ecuación 5. 2

$$V = \frac{1}{3}(R+G+B)$$

Ecuación 5. 3

3. **Cuantización de las imágenes.-** En el caso de estos histogramas HSV y RGB se cuantizará a 256 valores cada uno.
4. **División del conjunto de entrenamiento y prueba.-** Se divide el 70% para el subconjunto de entrenamiento y el 30% restante para el subconjunto de prueba.

De este corpus se obtuvo cuatro archivos de datos, dos de cada histograma HSV y RGB.

- Un archivo de entrenamiento en HSV.
- Un archivo de prueba en HSV.
- Un archivo de entrenamiento en RGB.
- Un archivo de prueba en RGB.

Corpus LFWcrop face dataset.- Este corpus posee imágenes de rostros de 64X64 píxeles. El corpus seleccionado se encuentra en escala de grises. Se utilizará para entrenar la red de Hopfield.

Con la ayuda de la herramienta open cv, se ha realizado un tratamiento a la imagen, consiste básicamente en:

1. De escala de grises se convierte a un archivo binarizado, mediante el siguiente código:

```
	import cv2
nombres = open("nombreImagenes.txt","r")
linea = nombres.readline()
if linea:
	linea = (linea.split(".")[0])+".pgm"
	imagen = cv2.imread(linea, cv2.CV_LOAD_IMAGE_GRAYSCALE)
#cargo la imagen especificando que se trata de escala de
grises
	#binarizo y saco el threshold, 128 es el threshold
inicial que estamos poniendo la mitad
	(thresh, im_bw) = cv2.threshold(imagen, 128, 255,
cv2.THRESH_BINARY | cv2.THRESH_OTSU)
	print "thresh: "+str(thresh)
	im_bw = cv2.threshold(imagen, thresh, 255,
cv2.THRESH_BINARY)[1] #aqui es donde realmente binarizo con el
threshold obtenido
	#guardo el binarizado
	cv2.imwrite('./binarizadas/'+linea,im_bw)

	linea = nombres.readline()
```

2. Se transforma en 1 y 0, mediante el código:

```
//codigo tomado de:
http://stackoverflow.com/questions/13091224/c-converting-
binaryp5-image-to-asciip2-image-pgm
#include <iostream>
#include <fstream>
#include <sstream>
using namespace::std;

int usage(char* arg) {
```

```
// exit program
    cout << arg << ": Error" << endl;
    return -1;
}

int main(int argc, char* argv[]) {
    int rows, cols, size, greylevels;
    string filetype;

    // open stream in binary mode
    ifstream istr(argv[1], ios::in | ios::binary);
    if(istr.fail()) return usage(argv[1]);

    // parse header
    istr >> filetype >> rows >> cols >> greylevels;
    size = rows * cols;

    // check data
    cout << "filetype: " << filetype << endl;
    cout << "rows: " << rows << endl;
    cout << "cols: " << cols << endl;
    cout << "greylevels: " << greylevels << endl;
    cout << "size: " << size << endl;

    // parse data values
    int* data = new int[size];
    int fail_tracker = 0; // find which pixel failing on
    for(int* ptr = data; ptr < data+size; ptr++) {
        //char t_ch;
        // read in binary char
                                //istr.read(&t_ch,
sizeof(char));
     unsigned char t_ch = istr.get();
        // convert to integer
        int t_data = static_cast<int>(t_ch);
        // check if legal pixel
        if(t_data < 0 || t_data > greylevels) {
```

```
            cout << "Failed on pixel: " << fail_tracker <<
endl;
            cout << "Pixel value: " << t_data << endl;
            return usage(argv[1]);
        }
        // if passes add value to data array
        *ptr = t_data;
        fail_tracker++;
    }
    // close the stream
    istr.close();

    // write a new P2 binary ascii image
    //ofstream ostr("greyscale_ascii_version.pgm");
    ofstream ostr(argv[2]);

    // write header
    //ostr << "P2\n" << rows << " " << cols << "\n" <<
greylevels << endl;

    // write data
    int line_ctr = 0;

    ostr << "[";
    int i=0;
    for(int* ptr = data; ptr < data+size; ptr++) {
        //si es mayor a 0 entonces que sea 1
     if(*ptr > 0)
           *ptr = 1;

     //realizo una inversion porque los espacios en blanco
estan en 1 y los espacios en negro en 0. Entonces invierto
para que lo que es blanco sea 0 y lo negro 1.
     *ptr = 1 - *ptr;

     // print pixel value
```

```
      if (i < 4095){ //si es el ultimo dato no se agrega la
',' 64x64 = 4096 puntos en la imagen.
            ostr << *ptr << ",";

      }
      else
            ostr << *ptr;
        // endl every ~20 pixels for some readability....
Bueno yo no le pongo cada 20 le pongo cada 64 porque las
imagenes son de 64x64
        if(++line_ctr % 64 == 0) ostr << endl;
      i++;
    }
    ostr << "]";
    ostr.close();

    // clean up
    delete [] data;
    return 0;
}
```

3. Cuando el archivo está en 1 y 0 puede ser usado por la red de Hopfield.

5.2 SELECCIÓN DE REDES NEURONALES Y HERRAMIENTAS

Existen redes neuronales sencillas que tienen una sola capa oculta hasta redes con mayor grado de complejidad y varios niveles de capas que poseen distintos tipos de aprendizaje, el cual consiste en ir cambiando el valor de sus pesos hasta lograr una salida deseada, o bien interpretar una salida obtenida de acuerdo al entorno en el cual se utilizó la red.

Se va a probar tres distintas redes: Perceptrón Multicapa, Lvq y la red de Hopfield.

Perceptrón Multicapa.- Es la red más utilizada en la clasificación de patrones. Su aprendizaje es supervisado, quiere decir que la red conoce la salida que debe generar; este entrenamiento consiste en actualizar los pesos de sus conexiones cuando exista un error entre la salida obtenida y la salida deseada.

Los parámetros que se necesitan para entrenar la red son:

- Datos de entrenamiento.- Consiste en los distintos corpus de datos que deben estar tratados y ordenados de acuerdo a un formato ya establecido, para este caso en pares de entrada y salida; además las salidas deben estar codificadas (0,1).
- Función de activación.- La función que se utiliza es una no lineal FANN_SIGMOID_SYMMETRIC que trabaja con datos en un rango [0,1].
- Datos de prueba.- Estos datos nos sirven para testear la red y obervar que tan eficiente fue el entrenamiento.

LVQ (Learning Vector Quantization)

Esta red trabaja con aprendizaje competitivo supervisado, ha sido seleccionada por su rapidez durante el entrenamiento y su precisión al clasificar. Su entrenamiento consiste principalmente en crear un vector de prototipos o codebook a partir de una secuencia de entrada, cada neurona debe aprender este vector de referencia para lograr una clasificación calculando la distancia más corta entre un vector de entrada y el vector de referencia.

Los parámetros necesarios para el entrenamiento de la red son:

- Conjunto de datos de entrenamiento.- Para Lvq no es necesario que los datos estén normalizados, pero si ordenados en una sola fila: cada línea está reservada para una muestra, cada línea se compone de n números de punto flotante seguido por la etiqueta de clase que puede ser cualquier cadena.
- Número de vectores de código.- Este valor depende del tamaño y la cantidad de los datos de entrenamiento, y el número de clases disponible. El software nos sugiere valores por defecto que se pueden usar inicialmente.
- Número de iteraciones para el entrenamiento.- Se sugiere que sea 40 veces mayor al número de vectores de código ingresado anteriormente.
- Datos de test.- Deben estar en el mismo formato que los datos de entrenamiento.
- Creación del codebook.- Es el conjunto de vectores de referencia que se crea durante el entrenamiento.

Pruebas de la red

Para utilizar el software como un clasificador, una vez entrenada la red; se debe ingresar el comando: knntest –din archivo.dat –cin file.cod –knn 5 ; que nos permite probar archivos de test independientes; el valor de –knn 5 se lo puede ir probando, el sugerido es 5.

Red de Hopfield

Este tipo de red basa su funcionamiento en almacenar información a manera de memoria asociativa, es decir puede recuperar información a partir de datos incompletos. La parte matemática que maneja es sencilla, por lo que la red es muy utilzada. Para el funcionamiento de la red de Hopfield se requiere que los datos estén almacenados en forma de vector binario y trabaja en dos fases: fase de almacenamiento que establece los valores que deben tener los pesos para almacenar patrones; y la fase de recuperación dónde se obtiene información almacenada a partir de datos incompletos.

La información requerida por la red para su entrenamiento, consiste en:

- Datos de entrenamiento.- Es un conjunto binarizado de datos de imágenes; para el entrenamiento es necesario cambiar los valores de 0 por -1.
- La red es entrenada mediante la herramienta Neurolab de python.
- Archivo de prueba.- El cual es alterado para observar que devuelve la red, es decir que porcentaje de similitud tiene con el objetivo incial.

Herramientas

Para probar las redes neuronales artificiales estudiadas, se investigó algunos simuladores que realizan el proceso de aprendizaje y prueba de una red. Se tuvo en cuenta la compatibilidad de la red con el simulador.

Para este caso se han escogido tres entornos de desarrollo, en los cuales es posible entrenar diferentes redes; permiten ingresar los parámetros de entrenamiento de forma sencilla, compilar y ejecutar la red mediante líneas de código.

FANN.- Es una librería de código abierto que permite entrenar redes neuronales multicapa como el Perceptrón, además admite lenguajes de programación como

Python que se ha utilizado en este caso. El simulador trabajará con 4 corpus distintos en la red Perceptrón.

LVQ.- Actua como una interfaz interactiva simple para algoritmos Lvq; el entrenamiento en un paquete LVQ es mucho mas rápido que el que se realiza con otras redes y alcanza el 100% en una tasa de reconocimiento después de un número considerado de iteraciones.

NeuroLab.- Es una librería para python para redes neuronales, sencilla y potente que trabaja con distintos tipos de redes, entre ellos están: Hopfield, Perceptrón simple, multicapa, etc.

Se ha seleccionado estas herramientas por los beneficios que presenta cada una, para las distintas redes que se están simulando.

5.3 DISEÑO PLAN EXPERIMENTACIÓN

Objetivo del experimento

El objetivo de esta implementación es probar algunas redes neuronales con diferentes corpus de datos en distintas herramientas de simulación, observar el error que se da en cada entrenamiento de red con los datos que se utilice y verificar cual es la red con mayor precisión.

Identificación de las fuentes de variación

Durante el proceso de entrenamiento de las redes neuronales existen factores que varían en función de la red que se utiliza, entre ellos están:

- Archivo de entrenamiento.
- Número de neuronas de entrada.- Este valor depende del problema que se vaya a resolver.
- Número de neuronas de salida.- Depende del problema a resolver.

- Número de capas ocultas.- Depende del programador y de las pruebas que se realice con los datos.
- Número de neuronas en la capa oculta.- El entrenamiento se prueba con un número a ser considerado por el programador y de acuerdo a la experiencia del mismo.
- Valor del error esperado.- Este valor varía de acuerdo al problema que se va a resolver.
- Número de épocas o iteraciones.- Si se obtiene el error deseado el número de iteraciones es suficiente. De acuerdo al experimento realizado es mejor aumentar e número de neuronas en la capa oculta que el número de iteraciones.
- Función de activación.- La función de activación depende de la salida de la red, cuando la salida es binaria la función de activación debe ser una de tipo escalón; si la salida es analógica se debe utilizar una de tipo sigmoidal.

Unidades experimentales

Las unidades experimentales lo constituyen los diferentes corpus a entrenar, donde será evaluada la correcta clasificación, el porcentaje de precisión y el reconocimiento de rostros.

Procedimientos de la experimentación

La experimentación de las diferentes redes se ha realizado de la siguiente forma:

Redes supervisadas

- **Tratamiento de los datos.-** De los datos de entrada depende la salida que genere la red, por lo que se debe realizar algunos procedimientos al conjunto de datos:
    - **Normalización de las entradas.-** La función de activación que se utiliza es una de tipo sigmoidal [0,1], por lo que las entradas serán normalizadas en este rango.
    - **Codificación de las salidas.-** Las salidas serán codificadas a valores 0,1.
    - **Ordenar el corpus en pares.**- Los datos del corpus deben ser ordenados en pares de entrada – salida.

    - **División en conjuntos de entrenamiento y prueba.-** El 70% de los datos se utilizará para el entrenamiento y el 30% restante para las pruebas.
- **Entrenamiento de la red.-** Se ingresa los parámetros para el entrenamiento de la red, como: número de neuronas de entrada y de salida, número de capas ocultas, número de neuronas en la capa oculta, el número de iteraciones y la función de activación.
- **Pruebas de la red.-** Con el conjunto de datos de prueba, se verifica si la red aprendió correctamente, de acuerdo al porcentaje de datos clasificados correctamente.

Red no Supervisada

- **Tratamiento de los datos:** Los datos para esta red no supervisada lo constituyen un conjunto de imágenes, por lo que el tratamiento consiste en:
    - **Binarizar la imagen**: La imagen original se encuentra en escala de grises, por lo que es necesario binarizarla, utilizando la herramienta openCV.
    - **Convertir en 0 y 1:** La imagen binarizada debe ser transformada a 1 y 0.
- **Entrenamiento de la red:** La red de Hopfield trabaja a manera de memoria asociativa.
- **Prueba de la red:** Se procede a verificar si la red 'memorizó' correctamente los datos, probando con un nuevo conjunto de datos modificado.

5.4 ANÁLISIS DE RESULTADOS OBTENIDOS

Perceptrón en FANN

La primera red que se utilizó para el entrenamiento es la red Perceptrón con el corpus Iris en el simulador FANN.

El corpus se ha divido dos partes:

CORPUS IRIS	
DATOS DE ENTRENAMIENTO 70%	105 datos
DATOS DE PRUEBA 30%	45 datos

Tabla 5. 2: Muestra los datos para entrenamiento y prueba.

Entrenamiento de la red

En el entrenamiento de la red Perceptrón, se tuvo en cuenta el número de intentos y el error que se esperaba, siendo estos valores 2000 y 0.001000. Con cuatro entradas para la red y 2000 intentos realizados, se obtuvo el siguiente error para cada intento, logrando al final el error mínimo esperado. En la ilustración 5.2, se puede observar el entrenamiento de la red en el simulador FANN.

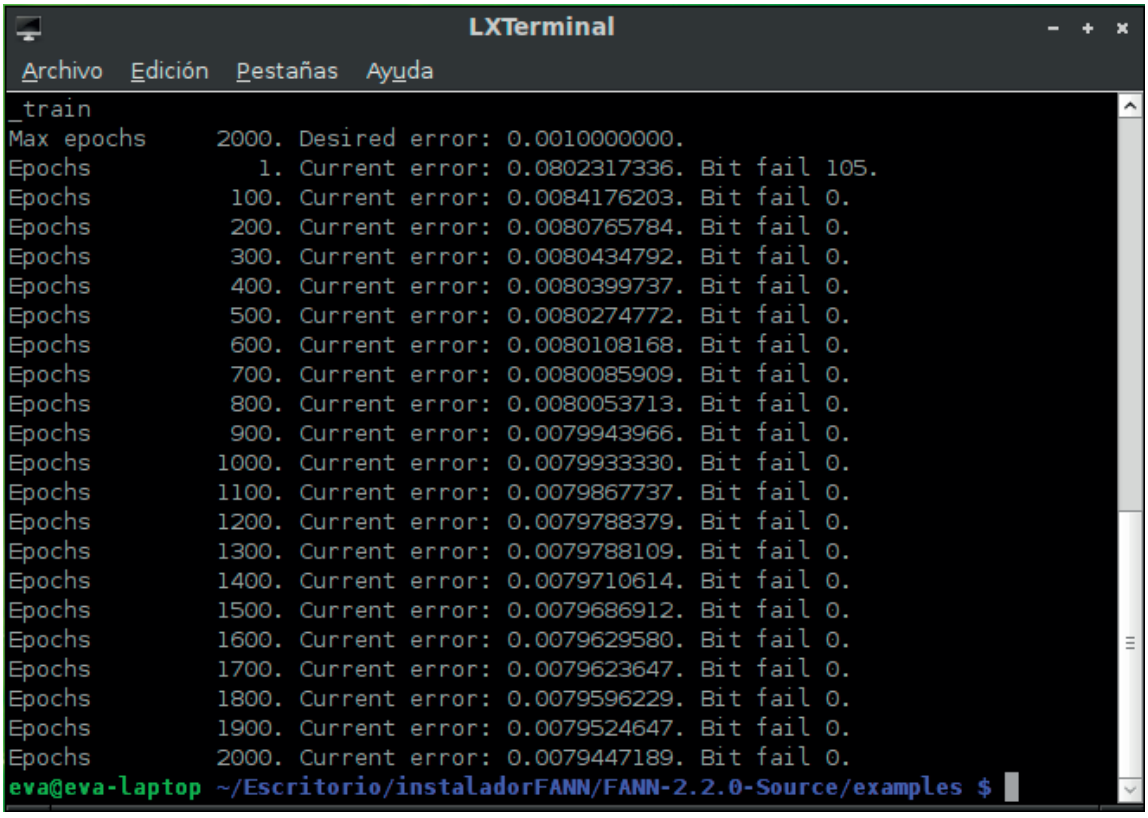


Ilustración 5. 2: Muestra el entrenamiento de la red con 2000 intentos y el error correspondiente.

EPOCHS	ERROR
1	0.0802317336
100	0.0084176203
200	0.0080765784
300	0.0080434792
400	0.0080399737
500	0.0080274772
600	0.0080108168
700	0.0080085909
800	0.0080053713
900	0.0079943966
1000	0.0079933330
1100	0.0079867737
1200	0.0079788379
1300	0.0079788109
1400	0.0079710614
1500	0.0079686912
1600	0.0079629580
1700	0.0079623647
1800	0.0079596229
1900	0.0079524647
2000	0.0079447189

Tabla 5. 3: Indica el número de intentos realizados con su correspondiente error obtenido.

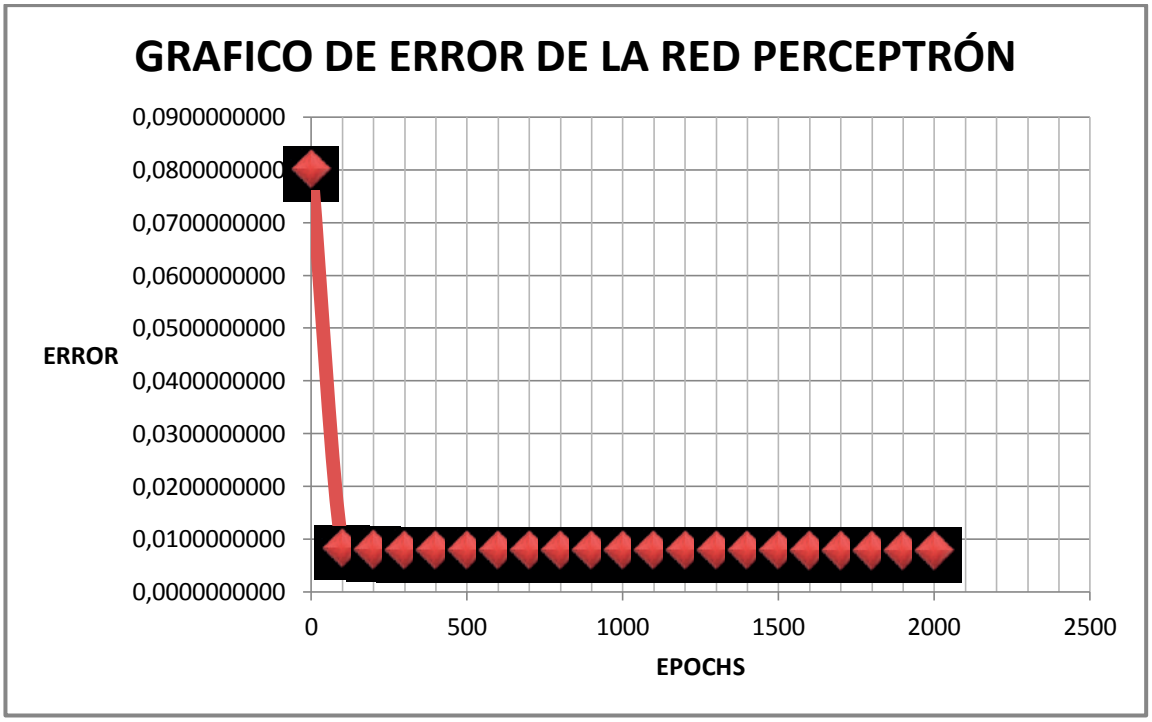


Ilustración 5. 3: Indica el error cometido por la red Perceptrón, durante los 2000 intentos realizados con el corpus Iris.

Prueba de la red

Los 45 datos de test han sido probados en el simulador FANN, obteniendo los siguientes resultados que se muestran en la ilustración 5.3.

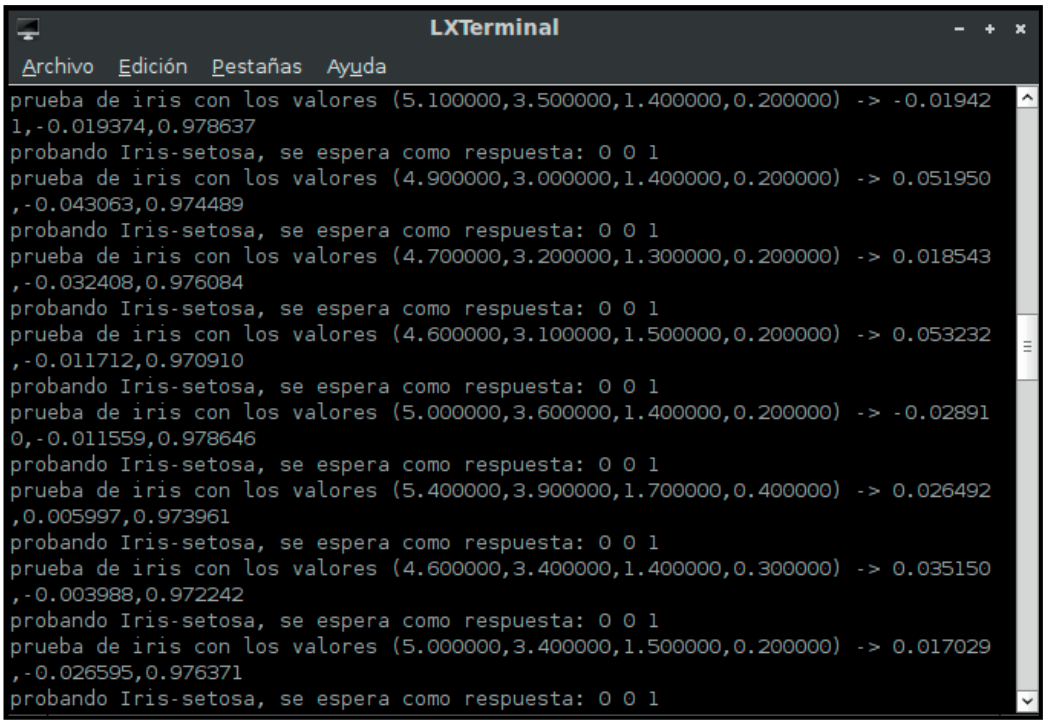


Ilustración 5. 4: Muestra los valores de entrada y los resultados obtenidos con el corpus Iris.

DATOS D ENTRADA				RESULTADOS OBTENIDOS		
5.1000000000	3.5000000000	1.4000000000	0.2000000000	-0.019421	-0.019374	0.978637
4.9000000000	3.0000000000	1.4000000000	0.2000000000	0.051950	-0.043063	0.974489
4.7000000000	3.2000000000	1.3000000000	0.2000000000	0.018543	-0.032408	0.976084
4.6000000000	3.1000000000	1.5000000000	0.2000000000	0.053232	-0.011712	0.970910
5.0000000000	3.6000000000	1.4000000000	0.2000000000	-0.028910	-0.011559	0.978646
5.4000000000	3.9000000000	1.7000000000	0.4000000000	0.026492	0.005997	0.973961
4.6000000000	3.4000000000	1.4000000000	0.3000000000	0.035150	-0.003988	0.972242
5.0000000000	3.4000000000	1.5000000000	0.2000000000	0.017029	-0.026595	0.976371
4.4000000000	2.9000000000	1.4000000000	0.2000000000	0.060911	-0.008312	0.969134
4.9000000000	3.1000000000	1.5000000000	0.1000000000	0.026535	-0.040297	0.976276
5.4000000000	3.7000000000	1.5000000000	0.2000000000	-0.044287	-0.005575	0.979811
4.8000000000	3.4000000000	1.6000000000	0.2000000000	0.034056	-0.011850	0.973365
4.8000000000	3.0000000000	1.4000000000	0.1000000000	0.025409	-0.044191	0.976428

4.3000000000	3.0000000000	1.1000000000	0.1000000000	-0.013581	-0.030862	0.977405
5.8000000000	4.0000000000	1.2000000000	0.2000000000	-0.216949	0.107320	0.983300
7.0000000000	3.2000000000	4.7000000000	1.4000000000	0.032501	0.832863	-0.012911
6.4000000000	3.2000000000	4.5000000000	1.5000000000	0.142295	0.800436	-0.025326
6.9000000000	3.1000000000	4.9000000000	1.5000000000	0.274521	0.753666	-0.028525
5.5000000000	2.3000000000	4.0000000000	1.3000000000	0.162248	0.787475	-0.027090
6.5000000000	2.8000000000	4.6000000000	1.5000000000	0.294837	0.742780	-0.029818
5.7000000000	2.8000000000	4.5000000000	1.3000000000	0.224915	0.768371	-0.033093
6.3000000000	3.3000000000	4.7000000000	1.6000000000	0.321897	0.732323	-0.033269
4.9000000000	2.4000000000	3.3000000000	1.0000000000	-0.291879	0.886426	0.092703
6.6000000000	2.9000000000	4.6000000000	1.3000000000	0.067156	0.821520	-0.015490
5.2000000000	2.7000000000	3.9000000000	1.4000000000	0.137754	0.796582	-0.030027
5.0000000000	2.0000000000	3.5000000000	1.0000000000	-0.145553	0.861481	0.030285
5.9000000000	3.0000000000	4.2000000000	1.5000000000	0.151715	0.795382	-0.027444
6.0000000000	2.2000000000	4.0000000000	1.0000000000	-0.142980	0.863975	0.035910
6.1000000000	2.9000000000	4.7000000000	1.4000000000	0.305328	0.737338	-0.033449
5.6000000000	2.9000000000	3.6000000000	1.3000000000	-0.208683	0.879023	0.037726
6.3000000000	3.3000000000	6.0000000000	2.5000000000	0.955694	-0.465280	0.047110
5.8000000000	2.7000000000	5.1000000000	1.9000000000	0.820351	0.173697	-0.008726
7.1000000000	3.0000000000	5.9000000000	2.1000000000	0.886279	-0.045223	0.014939
6.3000000000	2.9000000000	5.6000000000	1.8000000000	0.821979	0.172599	-0.005235
6.5000000000	3.0000000000	5.8000000000	2.2000000000	0.917484	-0.202170	0.023734
7.6000000000	3.0000000000	6.6000000000	2.1000000000	0.930775	-0.278289	0.034632
4.9000000000	2.5000000000	4.5000000000	1.7000000000	0.709504	0.395740	-0.030767
7.3000000000	2.9000000000	6.3000000000	1.8000000000	0.867893	0.029025	0.009750
6.7000000000	2.5000000000	5.8000000000	1.8000000000	0.865020	0.035278	0.005025

7.2000000000	3.6000000000	6.1000000000	2.5000000000	0.931335	-0.281035	0.035597
6.5000000000	3.2000000000	5.1000000000	2.0000000000	0.734409	0.367265	-0.015384
6.4000000000	2.7000000000	5.3000000000	1.9000000000	0.813545	0.195267	-0.006430
6.8000000000	3.0000000000	5.5000000000	2.1000000000	0.849495	0.092765	0.003954
5.7000000000	2.5000000000	5.0000000000	2.0000000000	0.857653	0.056762	-0.001953
5.8000000000	2.8000000000	5.1000000000	2.4000000000	0.921407	-0.228313	0.021759

Tabla 5. 4: Contiene los datos de entrada y los resultados obtenidos con el corpus Iris.

La siguiente tabla 5.5, indica el conteo de los valores que ha clasificado la red.

	Setosa	Versicolor	Virginica
Setosa 0 0 1	15		
Versicolor 0 1 0		15	
Virginica 1 0 0			15

Tabla 5. 5: Muestra el resultado de la clasificación de la red neuronal Peceptrón.

La red ha clasificado los datos con el 100% de precisión, como vemos en la ilustración 5.4 estos son los datos de los resultados que se obtuvieron; los tipos de flores se encuentran dispersos entre ellos.

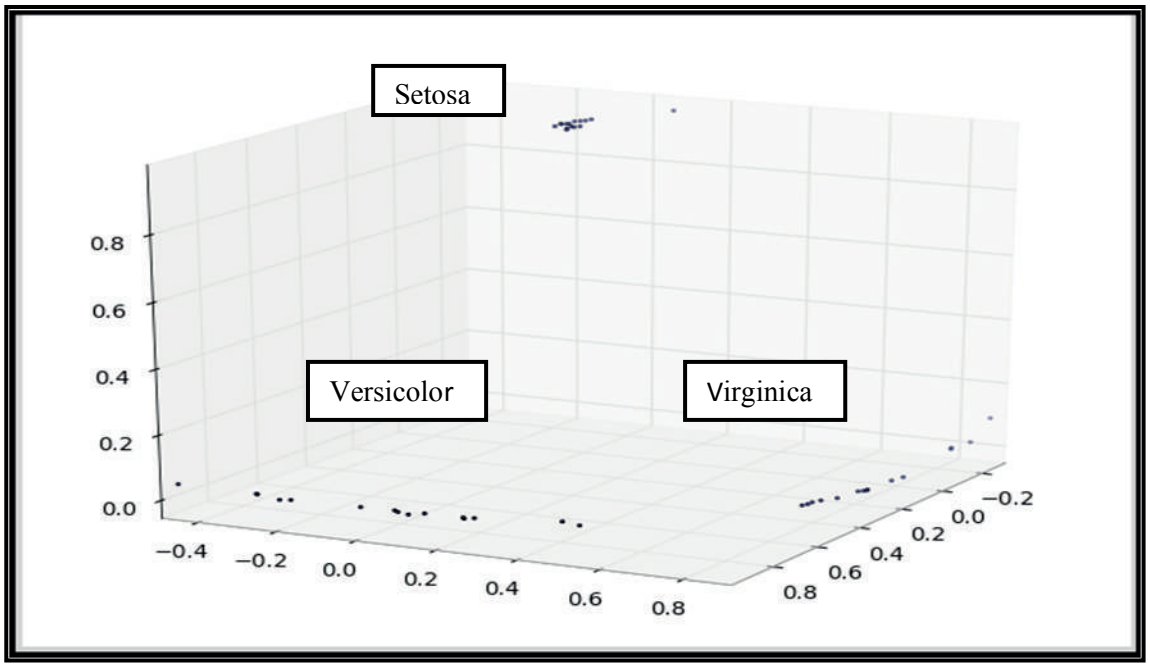


Ilustración 5. 5: Los datos clasificados correctamente, los tipos de flores se encuentra dispersas entre si. Autor de la tesis.

Corpus Clases de Vinos

Este corpus se entrenó con la misma red Perceptrón multicapa en el simulador FANN.

El corpus se ha divido dos partes:

CORPUS CLASES DE VINOS: 178 datos Clase 1: 59 Clase 2: 71 Clase 3: 48	
DATOS DE ENTRENAMIENTO 70% = 123 datos	Clase 1: 41 Clase 2: 49 Clase 3: 33
DATOS DE PRUEBA 30% = 55 datos	Clase 1: 18 Clase 2: 22 Clase 3: 15

Tabla 5. 6: Muestra los datos para entrenamiento y prueba.

Entrenamiento de la red

Durante el entrenamiento de la red Perceptrón con el corpus clases de vinos, se tuvo en cuenta:

- El número de intentos: 2500
- Error esperado: 0.0010000000

El error obtenido en cada época de entrenamiento se muestra en la siguiente ilustración:

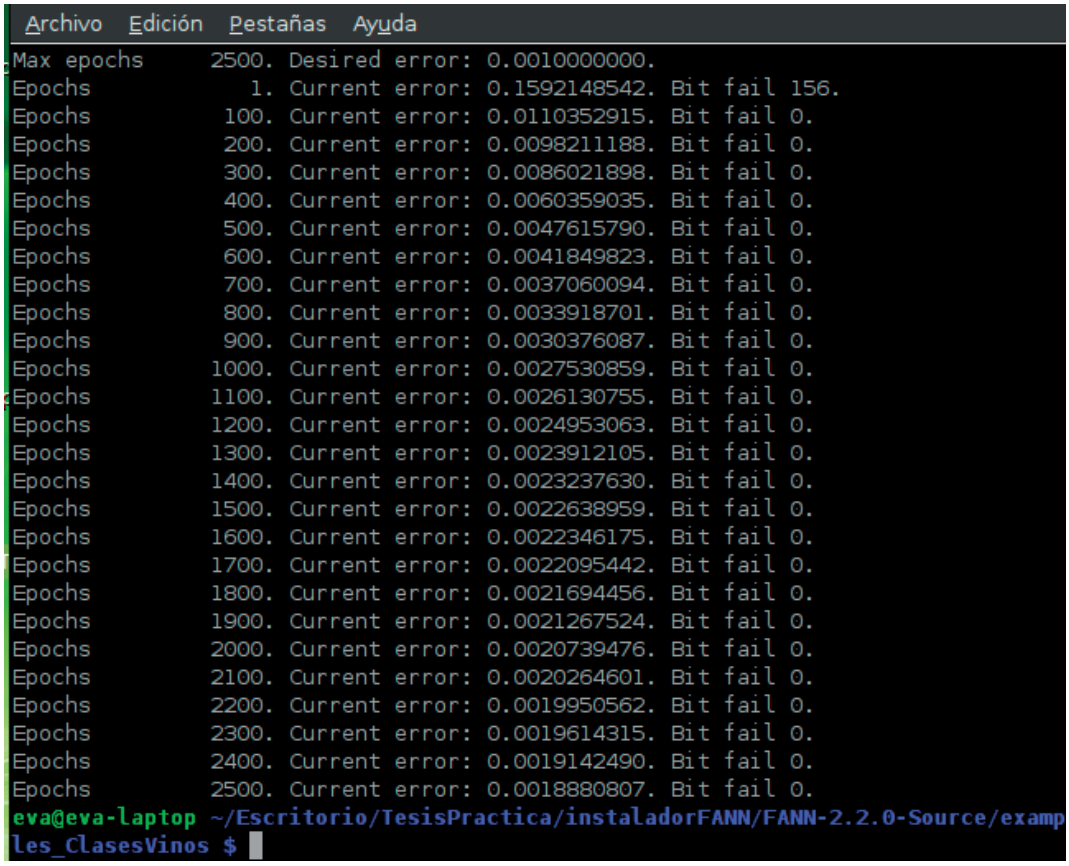


Ilustración 5. 6: Muestra el entrenamiento de la red Perceptrón con el corpus clases de Vinos.

EPOCHS	ERROR
1	0.1592148542
100	0.0110352915
200	0.0098211188
400	0.0060359035
600	0.0041849823
800	0.0033918701
1000	0.0027530859
1500	0.0022638959
2000	0.0020739476

2500	0.0018880807

Tabla 5. 7: Indica el número de intentos realizados con su correspondiente error obtenido.

En la siguiente ilustración se puede observar el error obtenido durante el entrenamiento del copus clases de vinos en la red Perceptrón.

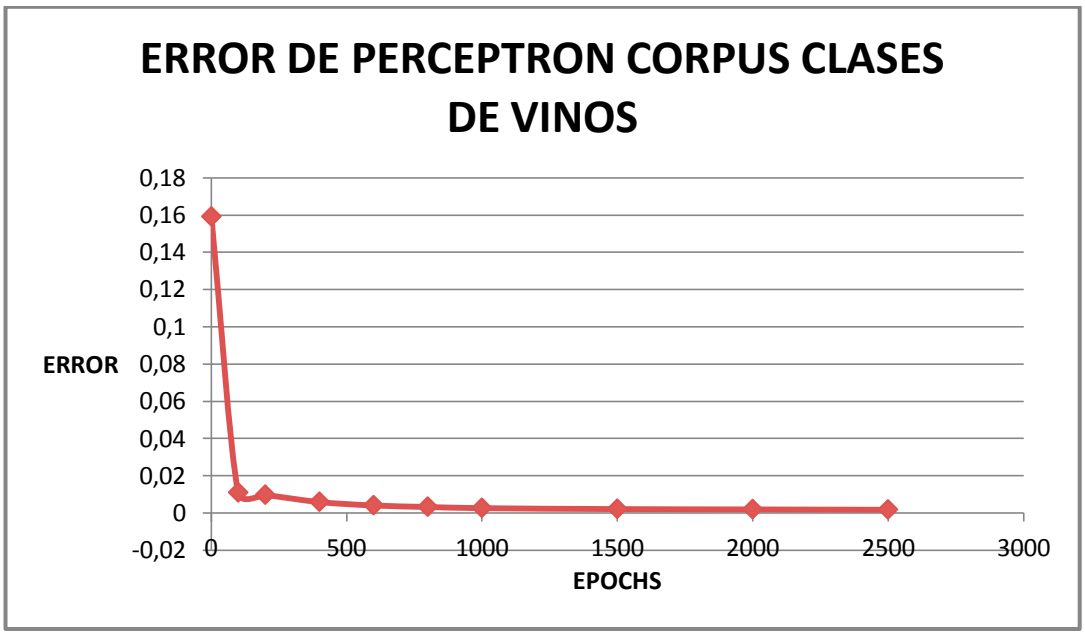


Ilustración 5. 7: Muestra el error cometido durante el entrenamiento de la red con el corpus Clases de Vinos.

Prueba de la red

Con el archivo de prueba del corpus clases de vinos, la red ha obtenido los siguientes resultados que se muestran en la ilustración.

```
eva@eva-laptop ~/Escritorio/TesisPractica/instaladorFANN/FANN-2.2.0-Source/examp
les_ClasesVinos $ python probador.py
probando Vino de clase 1, se espera como respuesta: 0 1
-0.208135,0.995247
Segmentation fault
probando Vino de clase 1, se espera como respuesta: 0 1
0.091784,0.995238
Segmentation fault
probando Vino de clase 1, se espera como respuesta: 0 1
-0.209876,0.997313
Segmentation fault
probando Vino de clase 1, se espera como respuesta: 0 1
0.055989,0.999995
Segmentation fault
probando Vino de clase 1, se espera como respuesta: 0 1
0.429347,0.656245
Segmentation fault
probando Vino de clase 1, se espera como respuesta: 0 1
0.089493,0.999980
Segmentation fault
probando Vino de clase 1, se espera como respuesta: 0 1
0.022246,0.999607
Segmentation fault
probando Vino de clase 1, se espera como respuesta: 0 1
```

Ilustración 5. 8: Muestra los resultados obtenidos con el corpus clase de vinos.

La siguiente tabla muestra los resultados de clasificación durante la prueba realizada a la red Perceptrón utilizando el corpus clases de vinos.

	Aciertos
Clase 1: 18	18
Clase 2: 22	18
Clase 3: 15	19

Tabla 5. 8: Resultado de la clasificación de la red neuronal Peceptrón, utilizando el corpus clases de vinos.

CorpusWang de imágenes en histograma HSV

Este corpus se entrenó con la misma red Perceptrón multicapa en el simulador FANN.

El corpus de 1000 imágenes se ha divido dos partes:

CORPUS WANG EN HSV: 1000 datos.

DATOS DE ENTRENAMIENTO 70%	700 datos
DATOS DE PRUEBA 30%	300 datos

Tabla 5. 9: Indica la cantidad de datos para el entrenamiento y prueba.

Entrenamiento de la red

Durante el entrenamiento de la red Perceptrón con el corpus de imágenes Wang en HSV, se tuvo en cuenta:

- El número de intentos.- 2500
- Error esperado.- 0.0010000000

El error obtenido en cada época de entrenamiento se muestra en la siguiente ilustración:

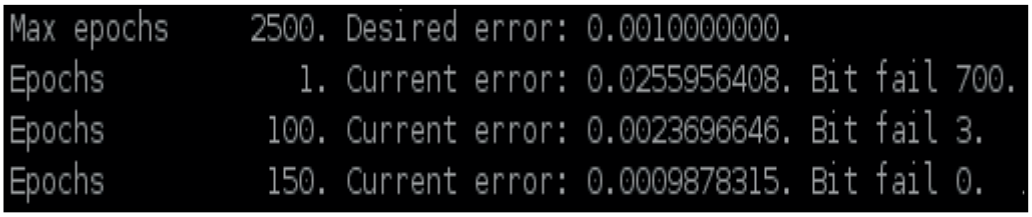


Ilustración 5. 9: Muestra el entrenamiento de la red, con el corpus de imágenes Wang en HSV.

EPOCHS	ERROR
1	0.0255956408
100	0.0023696646
150	0.0009878315

Tabla 5. 10: Indica el número de intentos realizados con su correspondiente error obtenido.

En la siguiente ilustración se puede observar el error obtenido durante el entrenamiento del copus de imágenes Wang en HSV en la red Perceptrón.

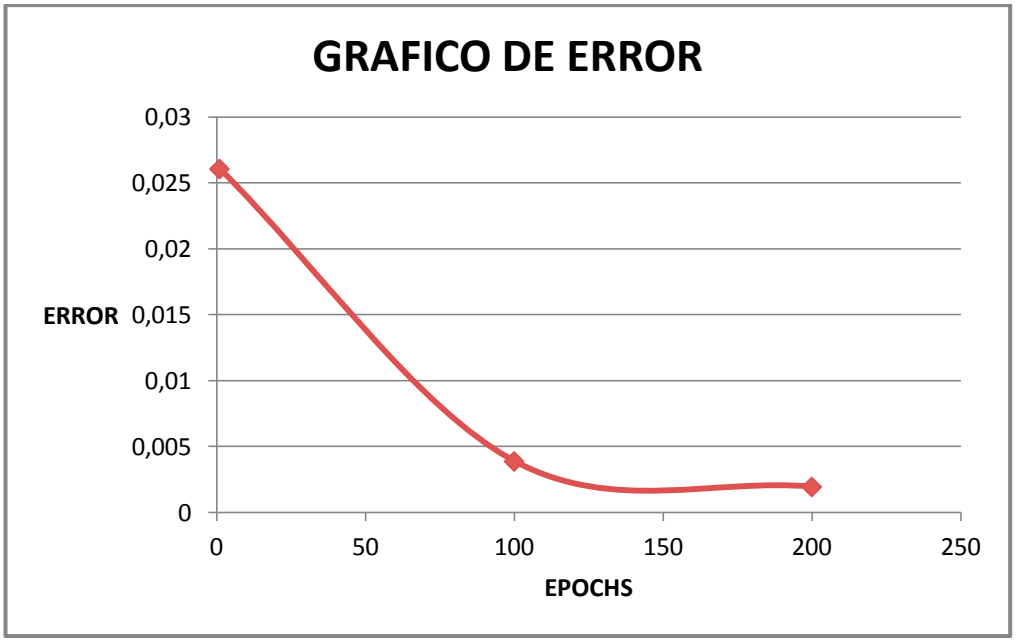


Ilustración 5. 10: Muestra el error generado por la red, en cada época de entrenamiento, con el corpus de imágenes Wang en HSV.

Prueba de la red

Para la prueba de la red es necesario especificar que se tiene 10 salidas, cada una representa la clase a la que corresponde cada imagen.

Clase	SALIDAS BINARIZADAS									
0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	1
2	0	0	0	0	0	0	0	0	1	0
3	0	0	0	0	0	0	0	0	1	1
4	0	0	0	0	0	0	0	1	0	0
5	0	0	0	0	0	0	0	1	0	1
6	0	0	0	0	0	0	0	1	1	0
7	0	0	0	0	0	0	0	1	1	1
8	0	0	0	0	0	0	1	0	0	0
9	0	0	0	0	0	0	1	0	0	1

Tabla 5. 11: Muestra las salidas binarizadas del corpus Wang en HSV.

La red ha obtenido las siguientes 10 salidas que se muestran en la ilustración.

```
0.003064,0.061841,-0.114736,-0.034196,0.209439,0.051064,0.032146,0.241728,0.2010
49,0.864915
0.056261,0.225981,0.378628,0.078114,-0.500583,-0.022797,0.170238,0.136657,0.0925
75,0.684731
-0.126408,-0.001802,-0.040834,0.045273,-0.095244,0.115506,0.024768,0.570468,-0.0
07036,0.915177
0.305897,0.016725,0.470291,0.066049,-0.494570,0.120122,0.154383,0.667319,0.08926
6,0.742372
-0.022444,0.160100,0.340075,0.046255,-0.509690,-0.009031,0.097636,0.140219,0.017
599,0.729538
-0.088037,0.065481,-0.041416,-0.051147,-0.050538,-0.036747,0.143782,0.056941,-0.
030708,0.908008
0.073656,0.154415,0.403986,0.092055,-0.414989,-0.024333,0.043373,-0.022321,0.017
496,0.569384
-0.141741,0.081904,-0.060327,-0.046107,0.111588,0.018151,0.407051,-0.074419,-0.0
72146,0.879009
0.096967,0.425477,0.427542,-0.177048,-0.393731,-0.068261,-0.008804,0.673695,0.05
5276,0.345699
0.806780,0.177340,-0.012196,0.038431,-0.195815,0.042343,0.183462,0.079841,-0.019
135,0.193567
-0.093240,0.019132,-0.011399,-0.010901,0.032301,0.018110,0.214397,0.088743,-0.01
0341,0.922886
-0.046991,-0.122430,-0.040979,-0.066974,-0.027252,-0.031475,0.124623,0.108521,0.
```

Ilustración 5. 11: Muestra las salidas obtenidas por la red con el corpus de imágenes en HSV.

Se muestra los resultados de la clasificación durante la prueba realizada en la red Perceptrón utilizando el corpus de imágenes Wang en HSV. Cada clase ha sido probada con 30 datos, siendo el porcentaje de acierto de 57.67%.

```
exitos de la clase 0: 17 de 30
exitos de la clase 1: 5 de 30
exitos de la clase 2: 14 de 30
exitos de la clase 3: 21 de 30
exitos de la clase 4: 29 de 30
exitos de la clase 5: 16 de 30
exitos de la clase 6: 22 de 30
exitos de la clase 7: 22 de 30
exitos de la clase 8: 14 de 30
exitos de la clase 9: 13 de 30.
```

Ilustración 5. 12: Muestra la clasificación obtenida por la red, con el corpus de imágenes en HSV.

Clases	Número de datos de acierto
Clase 0	17
Clase 1	5
Clase 2	14
Clase 3	21
Clase 4	29
Clase 5	16
Clase 6	22
Clase 7	22
Clase 8	14
Clase 9	13

Tabla 5. 12: Muestra la clasificación realizada con el corpus de imágenes en HSV.

Corpus Wang de imágenes RGB

Este corpus se entrenó con la misma red Perceptrón multicapa en el simulador FANN.

El corpus se ha divido dos partes:

CORPUS IMÁGENES WANG EN RGB	
DATOS DE ENTRENAMIENTO 70%	700 datos
DATOS DE PRUEBA 30%	300 datos

Tabla 5. 13: Muestra los datos para entrenamiento y prueba.

Entrenamiento de la red

Durante el entrenamiento de la red Perceptrón con el corpus imágenes en RGB, se tuvo en cuenta:

- El número de intentos.- 2500 iteraciones.
- Error esperado.- 0.0010000000.

El error obtenido en cada época de entrenamiento se muestra en la siguiente ilustración:

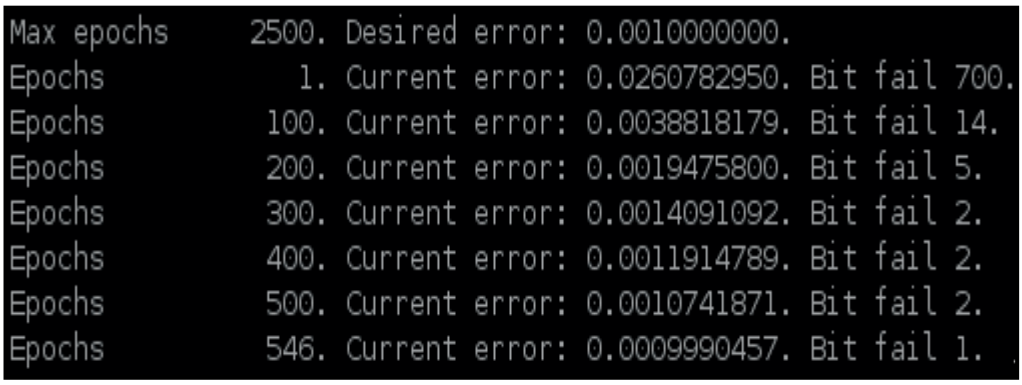


Ilustración 5. 13: Muestra el número de épocas y el error que generó la red, con el corpus de imágenes en RGB.

EPOCHS	ERROR
1	0.0260782950
100	0.0038818179
200	0.0019475800
300	0.0014091092
400	0.0011914789
546	0.0009996457

Tabla 5. 14: Indica el número de intentos realizados con su correspondiente error obtenido.

En la siguiente ilustración se puede observar el error obtenido durante el entrenamiento del corpus de imágenes Wang en RGB, en la red Perceptrón.

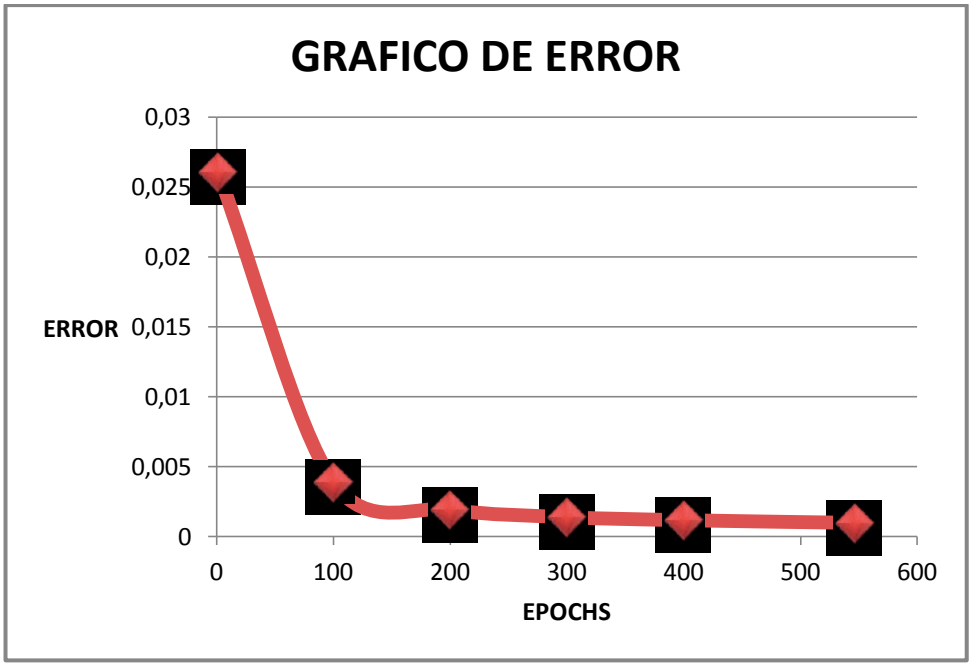


Ilustración 5. 14: Muestra el error generado por la red, con el corpus de imágenes en RGB.

Prueba de la red

Para la prueba de la red es necesario especificar que se tiene 10 salidas, cada una representa la clase a la que corresponde cada imagen.

Clase	SALIDAS BINARIZADAS									
0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	1
2	0	0	0	0	0	0	0	0	1	0
3	0	0	0	0	0	0	0	0	1	1
4	0	0	0	0	0	0	0	1	0	0
5	0	0	0	0	0	0	0	1	0	1
6	0	0	0	0	0	0	0	1	1	0
7	0	0	0	0	0	0	0	1	1	1
8	0	0	0	0	0	0	1	0	0	0
9	0	0	0	0	0	0	1	0	0	1

Tabla 5. 15: Muestra las salidas binarizadas, cada una es una clase.

La red ha obtenido las siguientes salidas que se muestran en la ilustración.

```
0.014282,0.021753,0.108410,-0.119084,0.004326,-0.032524,-0.021423,-0.153296,-0.0
43580,0.835738
0.197455,0.453766,-0.086163,-0.162796,-0.123044,0.082893,0.388283,-0.205270,-0.5
21567,0.762676
0.035493,0.062586,0.064085,-0.037705,0.020483,-0.002827,0.008655,-0.030350,-0.05
0159,0.808169
-0.241268,-0.359831,-0.227228,0.028395,-0.051952,-0.040373,0.075837,0.224728,0.3
73529,0.887707
-0.163522,0.034835,-0.056543,0.068669,0.006077,-0.006864,-0.040715,0.067946,0.13
5123,0.891628
0.629048,0.210556,-0.162073,0.171184,0.043752,0.021523,-0.270952,0.049779,-0.127
549,0.871652
0.046737,0.084121,-0.036994,0.117879,0.031800,0.004692,-0.011815,-0.007124,0.025
777,0.900652
-0.042246,-0.059032,-0.021489,-0.003855,0.002439,0.023396,0.166038,0.135565,0.16
2214,0.874258
0.003304,-0.095260,-0.029787,-0.002129,0.020088,0.006222,0.199381,-0.060416,0.37
5933,0.903407
-0.003907,0.119934,0.019782,-0.021824,0.004349,0.015562,0.111595,0.081705,-0.028
124,0.833815
0.719889,0.330513,-0.342913,0.294433,-0.065171,0.014197,-0.235495,-0.064177,-0.5
25475,0.828077
```

Ilustración 5. 15: Muestra las salidas de 10 valores que generó la red.

Se muestra los resultados de la clasificación durante la prueba realizada en la red Perceptrón utilizando el corpus de imágenes Wang en RGB. Cada clase ha sido probada con 30 datos, siendo el porcentaje de acierto 62%.

```
exitos de la clase 0: 19 de 30
exitos de la clase 1: 10 de 30
exitos de la clase 2: 12 de 30
exitos de la clase 3: 19 de 30
exitos de la clase 4: 28 de 30
exitos de la clase 5: 17 de 30
exitos de la clase 6: 22 de 30
exitos de la clase 7: 25 de 30
exitos de la clase 8: 11 de 30
exitos de la clase 9: 23 de 30
```

Ilustración 5. 16: Muestra la clasificación realizada por la red, con el corpus de imágenes en RGB.

La siguiente tabla muestra los datos clasificados durante la prueba realizada a la red Perceptrón utilizando el corpus de imágenes Wang en RGB. Cada clase contiene 30 datos.

Clases	Número de datos de acierto
Clase 0	19
Clase 1	10
Clase 2	12
Clase 3	19
Clase 4	28
Clase 5	17
Clase 6	22
Clase 7	25
Clase 8	11
Clase 9	23

Tabla 5. 16: Contiene la clasificación realizada por la red con el corpus de imágenes en RGB.

El porcentaje de acierto en el corpus Wang en RGB ha sido de un 62%.

RESULTADOS OBTENIDOS CON EL ALGORITMO LVQ

Corpus Iris

Se ha utilizado este corpus para entrenarlo con los algoritmos Lvq.

El corpus se ha divido en dos partes:

CORPUS IRIS 150	
DATOS DE ENTRENAMIENTO 70%	105 datos
DATOS DE PRUEBA 30%	45 datos

Tabla 5. 17: Muestra los datos para entrenamiento y prueba.

Entrenamiento de la red

Para el entrenamiento de la red, se requiere que los datos del corpus estén ordenados primero las entradas y la salida; en la cabecera debe tener un valor que indique el tamaño de las entradas.

El entrenamiento del algoritmo LVQ con el corpus Iris, requiere el ingreso de algunos parámetros:

- Archivo de entrenamiento: El cual es leído por el algoritmo Lvq.
- Número de vectores de código: Este número depende del número de clases existentes en el archivo de entrenamiento; sin embargo el programa sugiere un valor.
- Forma de inicialización de vectores de código: este valor depende de la cantidad de vectores de código que se asignó en el paso anterior.
- Número de iteraciones en el entrenamiento: el programa sugiere que sea 40 veces mayor al valor de vectores de libro de códigos.

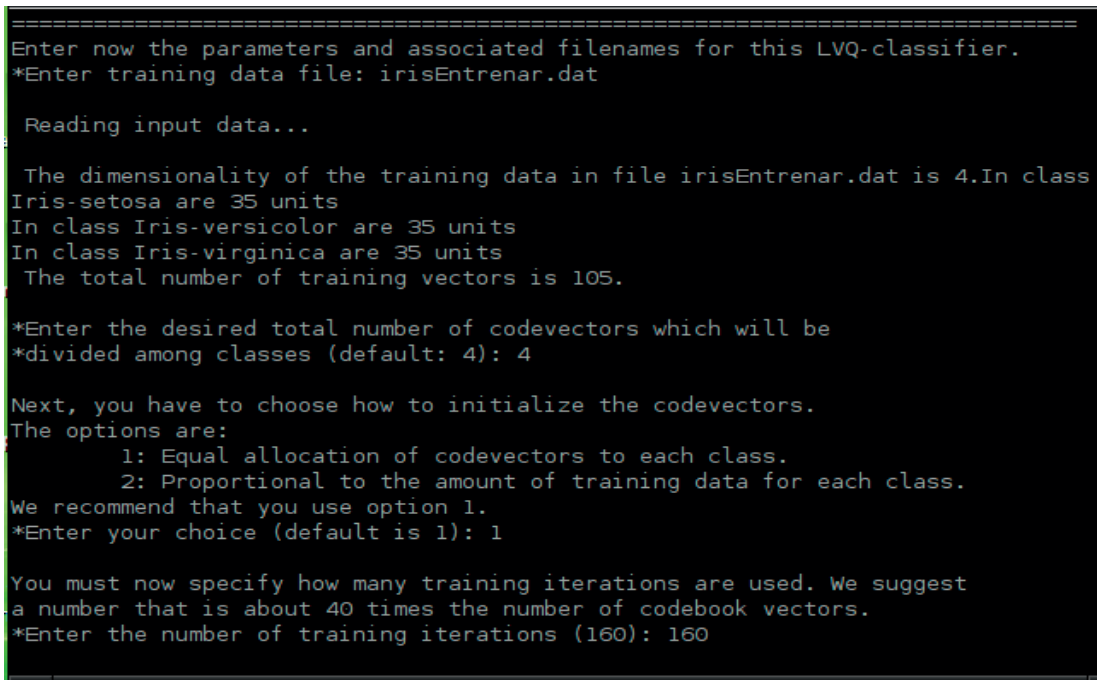


Ilustración 5. 17: Muestra los datos de entrenamiento necesarios para el algoritmo LVQ.

- Archivo de test.

- Archivo que contiene el vector de libro de códigos.

Durante el entrenamiento se puede modificar el númerode vectores de código, de tal forma que la mínima distancia entre los vectores de código dentro de cada clase sea balanceada. El algortimo se encarga del entrenamiento de la red.

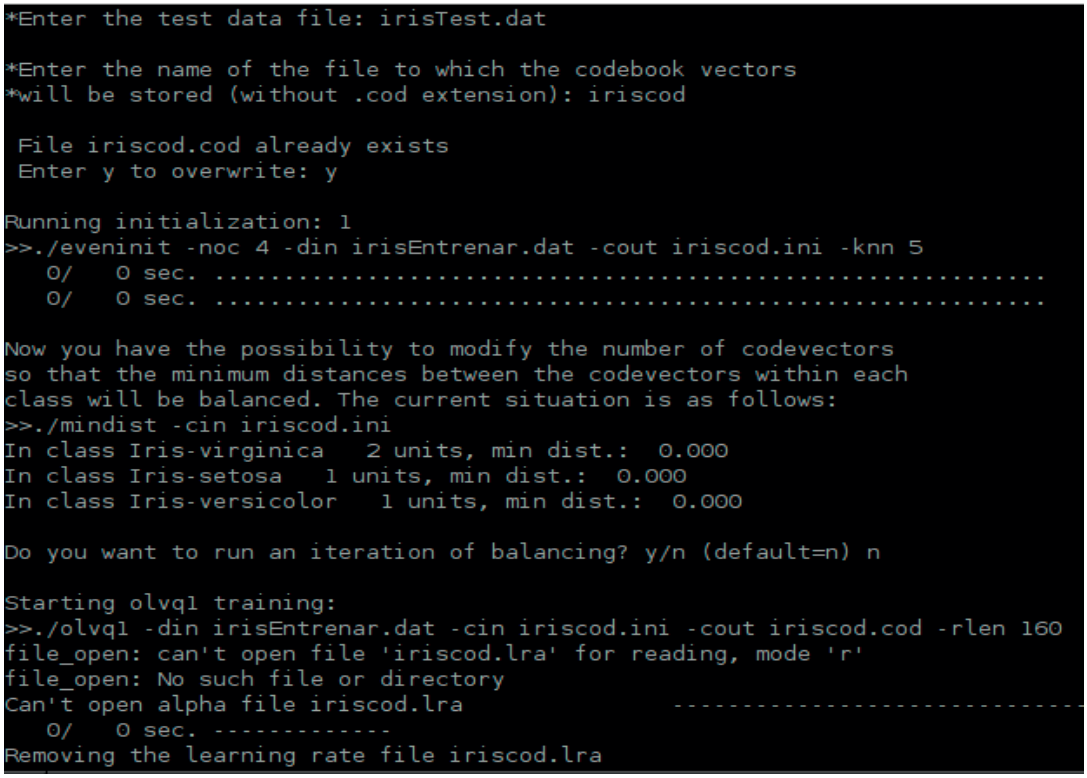


Ilustración 5. 18: Muestra el entrenamiento del algoritmo LVQ.

Prueba del algoritmo LVQ

Cuando la red ya está entrenada, se prueba para observar el porcentaje de aciertos en la clasificación, quedando lista para ser usada como un clasificador.

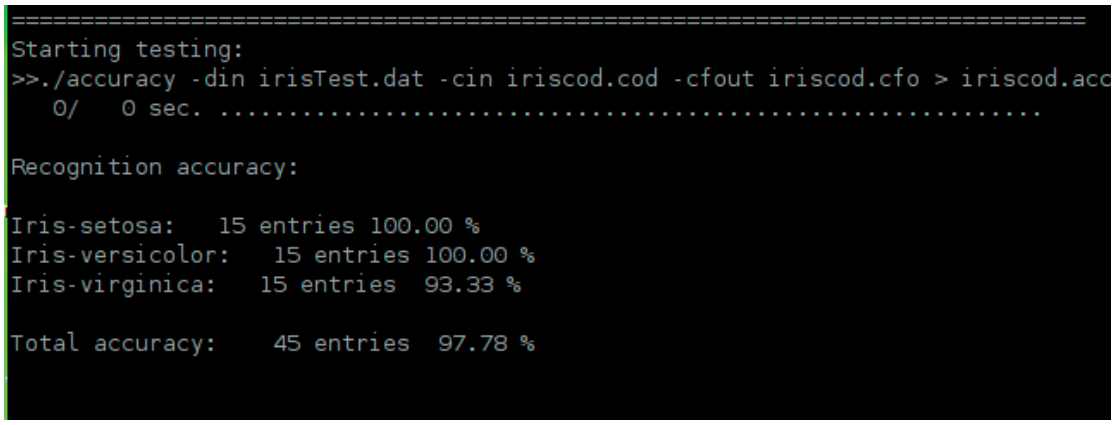


Ilustración 5. 19: Muestra el porcentaje de aciertos en la clasificación del corpus Iris.

La siguiente tabla muestra el porcentaje de clasificación durante la prueba realizada a la red Lvq utilizando el corpus Iris.

Tipo de Flor	Porcentaje de acierto
Setosa	100%
Versicolor	100%
Virginica	93.33%

Tabla 5. 18: Muestra el resultado de la clasificación de la red neuronal Lvq con el corpus Iris.

Estos algoritmos Lvq no permiten rastrear el error generado.

Prueba2

Una vez entrenada y probada la red, se puede usar como un clasificador, mediante el comando knntest –din pruebaIndependiente.dat –cin iriscod.cod –knn2

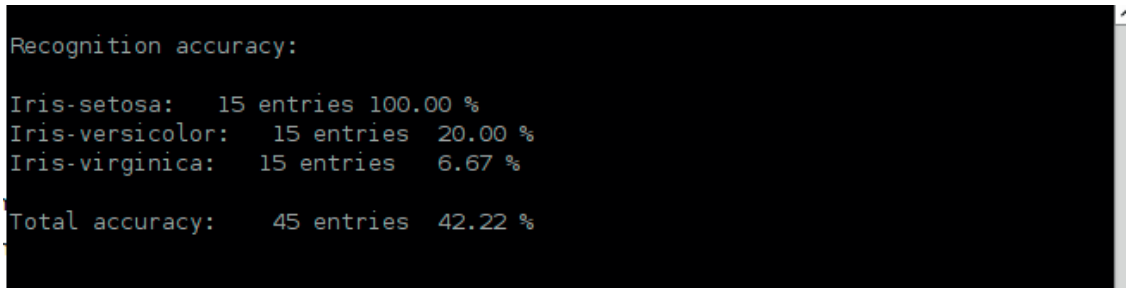


Ilustración 5. 20: Muestra el porcentaje de clasificación de un archivo de prueba independiente.

Tipo de Flor	Porcentaje de acierto
Setosa	100%
Versicolor	20%
Virginica	6.67%

Tabla 5. 19 Muestra el porcentaje de clasificación en la red LVQ.

Corpus Clases de Vinos

Se ha utilizado este corpus para entrenarlo con los algoritmos Lvq.
El corpus se ha divido en dos partes:

CORPUS CLASES DE VINOS: 178	
DATOS DE ENTRENAMIENTO 70%	123 datos
DATOS DE PRUEBA 30%	55 datos

Tabla 5. 20: Muestra los datos para entrenamiento y prueba.

Los datos del corpus con los que se ha probado el algoritmo LVQ son:

Clase	Cantidad
Clase 1	18
Clase 2	22
Clase 3	15

Tabla 5. 21: Indica la cantidad de datos usados para probar el algoritmo LVQ.

Una vez entrenado y probado el algoritmo LVQ con los parámetros adecuados para lograr un buen porcentaje de clasificación y utilizando los datos del corpus clases de vinos; se ha probado la red, obteniendo los siguientes resultados:

Ilustración 5. 21: Muestra el porcentaje de clasificación obtenido por el algoritmo LVQ.

Clase de vino	Porcentaje de acierto
1	94.44%
2	72.73%
3	80%

Tabla 5. 22: Muestra el porcentaje de aciertos en la clasificación con la red Lvq. Autor dela tesis.

Prueba2

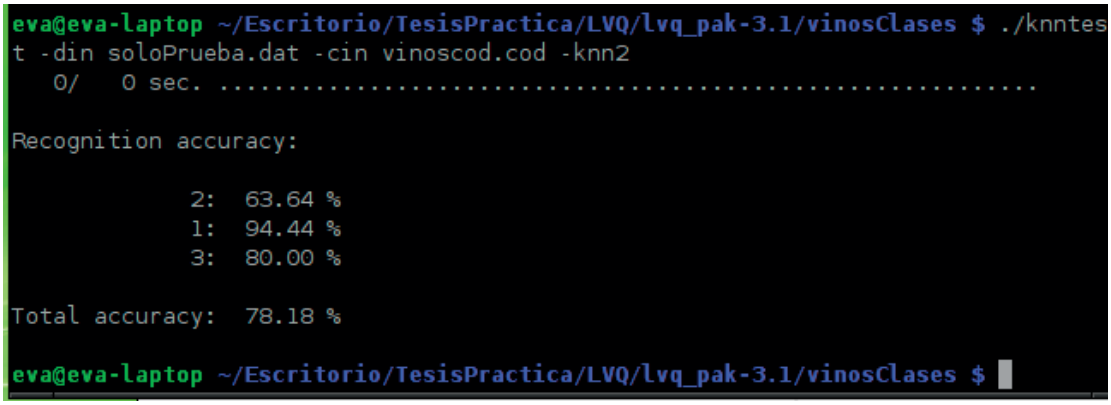


Ilustración 5. 22: Muestra el porcentaje de clasificación para un conjunto de datos modificado.

Clase de vino	Porcentaje de acierto
1	94.44%
2	63.64%
3	80%

Tabla 5. 23: Muestra el porcentaje de clasificación de un conjunto de datos modificado para la prueba. Autor de tesis.

RESULTADOS OBTENIDOS CON LA RED DE HOPFIELD

Módulo de reconocimiento de rostros

Para probar la red de Hopfield, se diseño un programa que permite realizar los diferentes procesos de binarización y dilatado de la imagen original y de prueba; para finalmente proceder con el reconocimiento de las imágenes mediante la red. En la ilustración 5.23, se observa el diseño de este módulo.

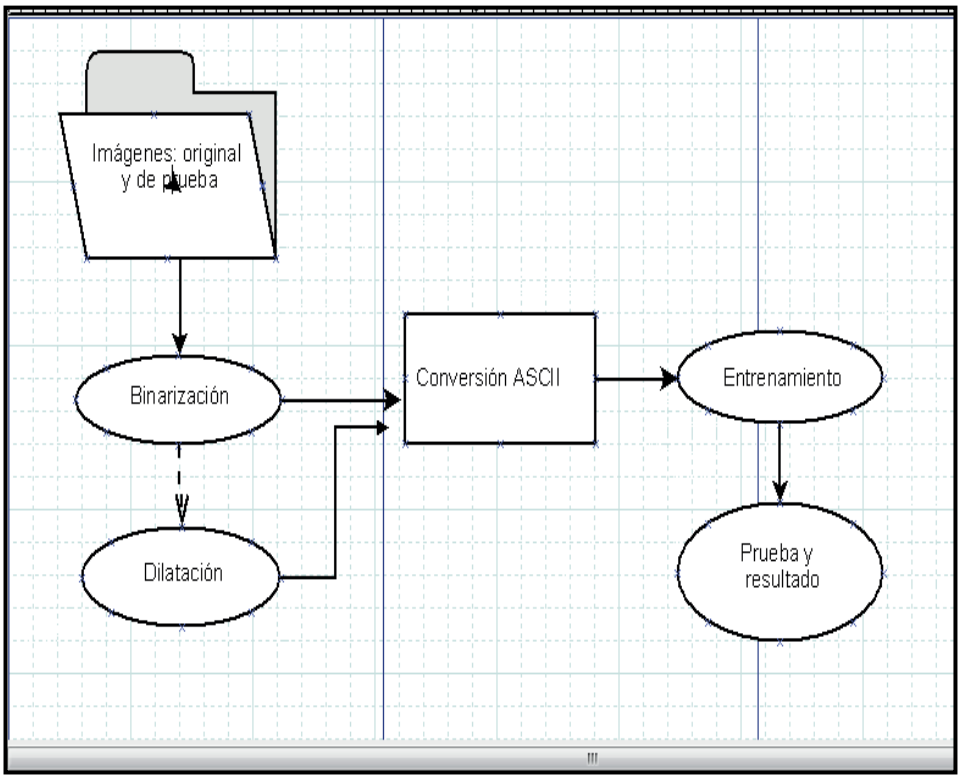


Ilustración 5. 23: Muestra el módulo de reconocimiento de rostros.

De forma detallada a continuación se describe la ejecución de este módulo:

La ejecución se realizará a través de interfaz gráfica, implementada en lenguaje de programación Python, por medio de ésta se carga la imagen original y de prueba.

Una vez cargadas las imágenes, son binarizadas.

La imagen original es almacenada a manera de memoria asociativa por la red de Hopfield.

La imagen de prueba es además modificada usando la función dilate de openCV.

Finalmente, se utiliza la red de Hopfield para el reconocimiento de los rostros.

Corpus de LFWcrop face dataset

Este corpus contiene imágenes .pgm (escala de grises), como se indicó anteriormente para la aplicación de la red de Hopfield, es necesario que las imágenes se encuentren en 1 y 0.

Prueba 1.

La prueba realizada es satisfactoria al reconocer el rostro de una misma persona en diferentes posiciones; la imagen a probar es modificada usando la función dilate de Neurolab; posteriormente son comparadas y la red reconoce los dos rostros como de la misma persona.

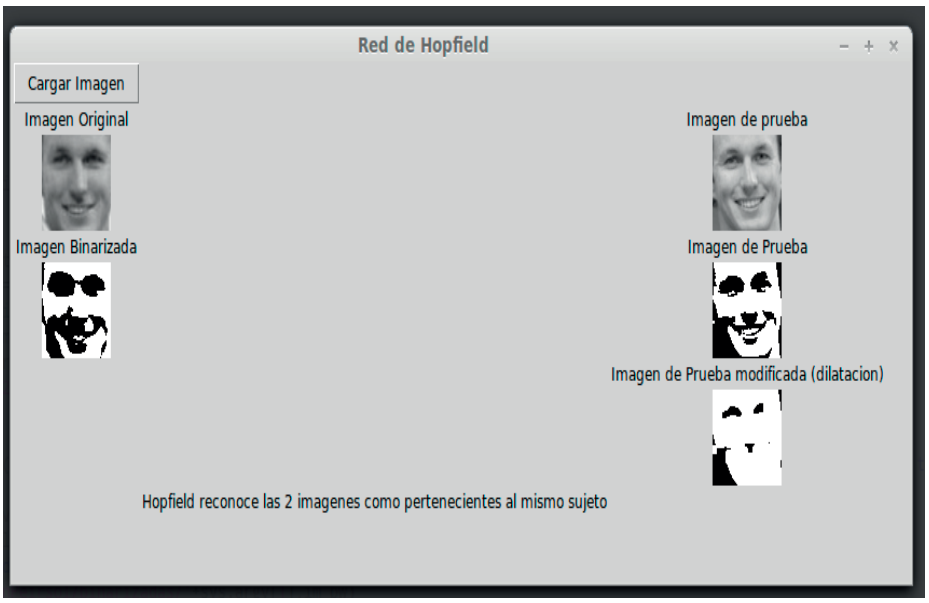


Ilustración 5. 24: Muestra el proceso de entrenamiento y prueba de una imagen en la red de Hopfield.

Prueba 2.

Se realiza una segunda prueba, la red de Hopfield reconoce ambos rostros como de una misma persona, a pesar de la modificación que se hace a la imagen.

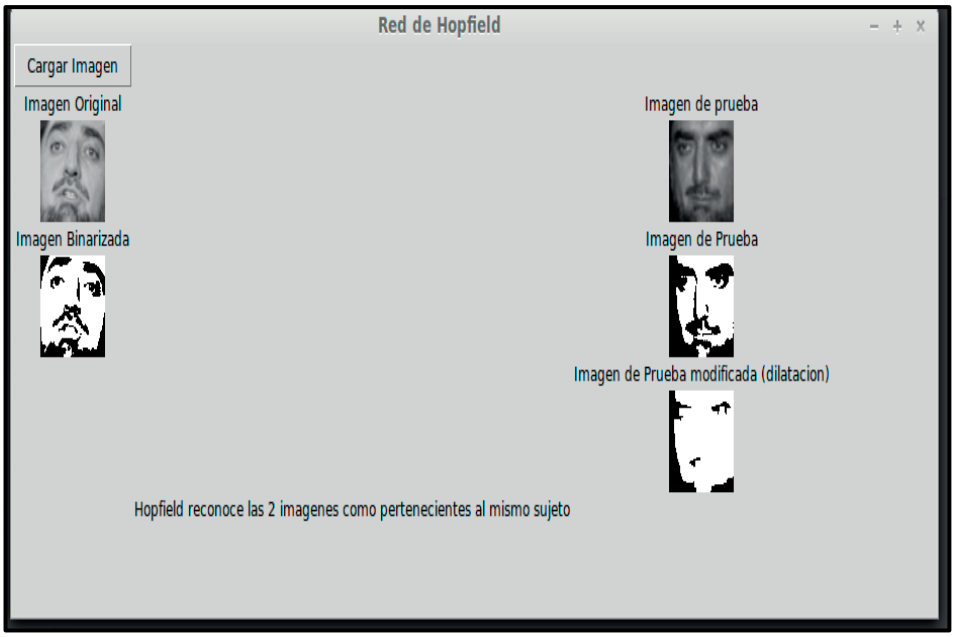


Ilustración 5. 25: Muestra el proceso que sigue una imagen en la red de Hopfield.

Prueba 3.

Otra imagen es reconocida por la red de Hopfield.

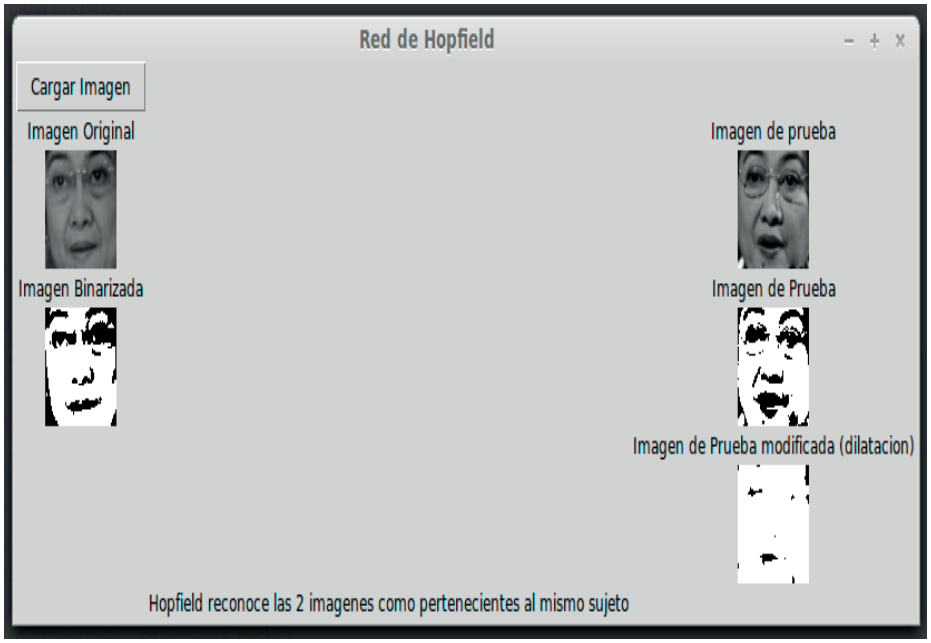


Ilustración 5. 26: Muestra el proceso que sigue una imagen en la red de Hopfield. Autor del a tesis.

Prueba 4.

En este caso la red de Hopfield no reconoce las imágenes como el rostro de una misma persona, se debe a que el rostro de prueba tiene gafas, al momento de binarizar los patrones cambian y ya no es reconocida por la red.

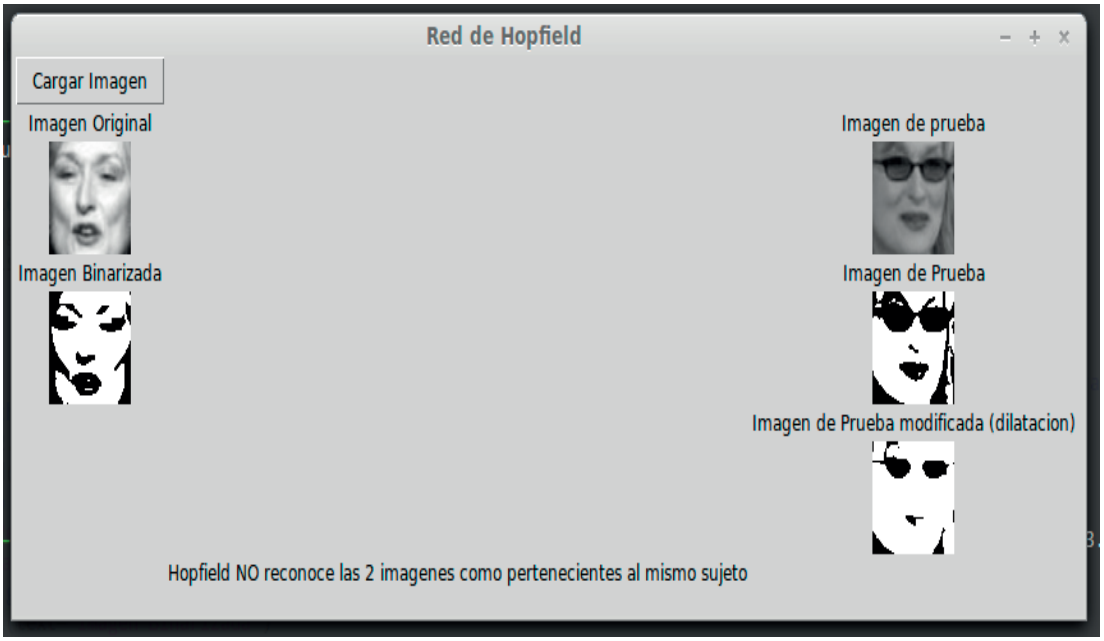


Ilustración 5. 27: Los dos rostros no son reconocidos como la misma persona, se debe a las gafas oscuras, al momento de binarizar cambian la imagen.

CONCLUSIONES

Y

RECOMENDACIONES

CONCLUSIONES

De acuerdo al estudio realizado, se puede decir, que en un futuro las redes neuronales artificiales no superarán la funcionalidad del cerebro humano, ni tan siquiera lo igualarán por la complejidad que éstos presentan.

Muchos autores señalan que las máquinas pueden evolucionar hasta convertirse en máquinas inteligentes que actúen por si mismas sin la participación del hombre; considero según lo investigado, que no es posible debido a que las máquinas no son independientes, actúan dentro de límites impuestos por una persona.

En lo que se refiere a la simulación de redes; para seleccionar la función de transferencia se debe tener en cuenta la salida que se espera, si son binarias o analógicas.

Para el diseño de una red neuronal, es muy importante determinar que el número de neuronas de entrada y de salida depende del problema que se vaya a resolver, y el número de capas ocultas y de neuronas en cada capa oculta se basan en la experiencia del programador y en ir probando la red a manera de ensayo – error.

La red Perceptrón simple es útil únicamente cuando se tiene patrones linealmente separables, es decir cuando las clases que se desean clasificar están claramente separadas unas de otras, por lo que para emplearlo en un problema real no resulta tan eficiente y eficaz como se esperara, sin embargo es la red mas antigua y con la que se empezó todo el desarrollo de una red neuronal artificial.

En la red Perceptrón multicapa el número de capas ocultas y de neuronas por capa no puede ser demasiado elevado, pues causa inconvenientes de tiempo, la red tarda demasiado en su etapa de entrenamiento.

De acuerdo al experimento realizado, una red Perceptrón se puede manejar con éxito utilizando dos capas ocultas y 150 neuronas en cada una de sus capas.

En cuanto al reconocimiento de rostros con la red de Hopfield ha sido necesario binarizar las imágenes a 1 y 0, debido a que esta red utiliza como función de transferencia una de tipo escalón.

No es aconsejable usar mas de dos capas ocultas en una red, pues aumenta de manera considerable el tiempo de entrenamiento, se ha optado por aumentar el número de neuronas en las capas, obteniendo buenos resultados.

Como se presenten los datos a la red, éstos influirán en su respuesta; por lo que la técnica de normalización de entradas se utilizará en todos los casos.

Existen las herramientas necesarias como FANN para la red Perceptrón; la librería Neurolab de Python que trabaja con la red de Hopfield; estas herramientas presentan un alto rendimiento y buen desempeño además de su libre acceso, por lo que han sido muy útiles y de gran ayuda para el desarrollo del experimento.

En cuanto a la efectividad de la red de Hopfield en el reconocimiento de rostros, se puede decir que los resultados varían de acuerdo a la binarización y posterior dilatación realizada a la imagen de prueba, la imagen dilatada tiene las áreas negras engrosadas lo que dificulta el reconocimento.

RECOMENDACIONES

Se recomienda realizar tratamiento de los datos: normalización de las entradas y codificación de las salidas, para que la clasificación sea mucho más efectiva.

Se debe tener en cuenta la función de salida para realizar la normalización de los datos de entrada.

En el diseño de la red neuronal se recomienda no utilizar demasiadas capas ocultas y neuronas en las capas, pues el tiempo y consumo de CPU aumentan de manera considerable.

En cuanto a la red de Hopfield se recomienda no cargar la red con demasiados patrones de entrenamiento.

Para el reconocimiento de rostros se aconsejable utilizar imágenes sin demasiados objetos como gafas, pues dificulta el reconocimiento.

BIBLIOGRAFÍA

[1] DAZA, Sandra, Redes Neuronales Artificiales, fundamentos, modelos y aplicaciones, fecha de recuperación: 18 de junio de 2012, http://www.monografias.com/trabajos12/redneur/redneur.shtml#MODELO

[2]El Prisma, Redes Neuronales, fecha de recuperación: 18 de junio de 2012, http://www.elprisma.com/apuntes/ingenieria_de_sistemas/redesneuronales/default.asp

[3] Universidad de Guadalajara, Características Principales de las Redes Neuronales, fecha de recuperación: 19 de junio de 2012, http://proton.ucting.udg.mx/posgrado/cursos/idc/neuronales2/RNArtificial.htm

[4] Wikipedia, Inteligencia Artificial, fecha de recuperación: 20 de junio de 2012, http://es.wikipedia.org/wiki/Inteligencia_artificial

[5] PINO, Raúl, y otros, *INTRODUCCIÓN A LA INTELIGENCIA ARTIFICIAL: Sistemas Expertos, Redes Neuronales Artificiales y Computación Evolutiva,* Servicio de publicaciones Universidad de Oviedo, Oviedo, 2001.

[6]Universidad Politécnica de Cataluña, Introducción a la Inteligencia Artificial, fecha de recuperación: 21 de junio de 2012, http://www.lsi.upc.edu/~bejar/ia/transpas/teoria/1-IA-introduccion.pdf

[7]Observatorio para la cibersociedad, La relación hombre-máquina y la ideología de una posthumanidad, fecha de recuperación: 22 de junio de 2012, http://www.cibersociedad.net/congres2006/gts/comunicacio.php?id=619

[8]International Business Times, Científico asegura Inteligencia Artificial alcanza Nivel Humano, fecha de recuperación: 25 de junio de 2012, http://mx.ibtimes.com/articles/4627/20100902/inteligencia-artificial-supera-hombrecientifico-google-nasa-unam.htm

[9] Holacape, Asimo el robot más avanzado del mundo: Características, fecha de recuperación: 26 de junio de 2012, http://www.holacape.com/2012/01/asimo-el-robot-mas-avanzado-del-mundo.html

[10] Neoteo, el Robot Asimo, fecha de recuperación: 27 de junio de 2012, http://www.neoteo.com/el-robot-asimo

[11]PALACIOS, Francisco, Redes Neuronales con GNU/Linux, fecha de recuperación: 28 de junio de 2012, http://softwarelibre.unsa.edu.ar/docs/descarga/2003/curso/htmls/redes_neuronales/x24.html

[12] Engadget, Honda nos presenta a Miimo, el jardinero de Asimo, fecha de recuperación: 25 de octubre de 2012, http://es.engadget.com/tag/HONDA/

[13] SAYA COMUNICACIONES S.A.C. – IDG COMUNICACIONES, Watdon: La nueva herramienta contra el cáncer, fecha de recuperación: 25 de octubre de 2012, http://www.cioperu.pe/articulo/11451/watson-la-nueva-herramienta-contra-el-cancer/

[14] Definición.de, Cognitivo, fecha de recuperación: 25 de octubre de 2012, http://definicion.de/cognitivo/

[15]Ecointeligencia, El sistema experto Watson de IBM suplanta a los médicos en hospitales top USA, fecha de recuperación: 25 de octubre de 2012, http://www.media-tics.com/noticia/2525/Dircom-2.0/sistema-experto-watson-ibm-suplanta-m%C3%A9dicos-hospitales-top-usa.html

[16]Total Publishing Network S.A., IBM Watson, superordenador capaz de entender y responder preguntas, fecha de recuperación: 25 de octubre de 2012, http://www.muycomputerpro.com/2011/01/14/ibm-watson-superordenador-capaz-de-entender-y-responder-preguntas/

[17] MASSARE, Bruno, Tal vez dejemos de preguntarnos qué es inteligencia artificial, fecha de recuperación: 29 de octubre de 2012, http://www.kcl.ac.uk/nms/depts/informatics/news/docs/Nota-de-tapa-IA-4.pdf

[18] SOMOLINOS, José, *Avances en Robótica y Visión por computador,* Ediciones de la Universidad de Castilla-La Mancha, Cuenca-España, 2002

[19] .Seguridad, Gestión de incidentes de seguridad informática con agentes inteligentes, fecha de recuperación: 29 de octubre de 2012, http://revista.seguridad.unam.mx/numero-14/gesti%C3%B3n-de-incidentes-de-seguridad-inform%C3%A1tica-con-agentes-inteligentes

[20] Universidad de Windsor, Inteligencia Artificial. Redes Neuronales, fecha de recuperación: 24-julio 2012, http://www.itnuevolaredo.edu.mx/takeyas/Apuntes/Inteligencia%20Artificial/Apuntes/tareas_alumnos/RNA/Redes%20Neuronales2.pdf

[21] BASOGAIN OLABE, Xabier, Redes Neuronales Artificiales y sus Aplicaciones, fecha de recuperación: 30 de julio de 2012, http://ocw.ehu.es/ensenanzas-tecnicas/redes-neuronales-artificiales-y-susaplicaciones/contenidos/pdf/libro-del-curso/

[22]OCHOA, Susana, Características de una Red Neuronal Artificial, fecha de recuperación: 30 de julio de 2012, http://fluidos.eia.edu.co/hidraulica/articuloses/flujoentuberias/neuronal/neuronal_archivos/page0002.htm

[23] Facultad de Ingeniería Eléctrica UTP, Principales Tipos de Redes Neuronales, fecha de recuperación: 8 de agosto de 2012, http://proton.ucting.udg.mx/posgrado/cursos/idc/neuronales2/General2.htm

[24] CONSTELA, Gabriel Roberto, Reconocimiento Óptico de Dígitos con Redes Neuronales, fecha de recuperación: 8 de agosto de 2012, http://es.scribd.com/doc/76581229/29/Redes-Neuronales-Recurrentes

[25] EcuRed: Enciclopedia cubana, Redes Neuronales Artificiales, fecha de recuperación: 9 de agosto de 2012, http://www.ecured.cu/index.php/Redes_neuronales_artificiales

[26]MATICH, Damian Jorge, Redes Neuronales: Conceptos Básicos y Aplicaciones, fecha de recuperación: 21 de agosto de 2012, http://www.frro.utn.edu.ar/repositorio/catedras/quimica/5_anio/orientadora1/monografias/matich-redesneuronales.pdf

[27]Emol: Tecnología, Joven construye red neuronal que detecta el cáncer de mama, fecha de recuperación: 28 de agosto de 2012, http://www.emol.com/noticias/tecnologia/2012/07/25/552386/joven-construye-red-neuronal-artificial-que-detecta-el-cancer-de-mama.html

[28]Experiensense, Google implementa una red neuronal de 16000 procesadores capaz de reconocer felinos, fecha de recuperación: 28 de agosto de 2012, http://www.experiensense.com/google-implementa-una-red-neuronal-de-16-000-procesadores-capaz-de-reconocer-felinos/

[29]HERNÁNDEZ HERNÁNDEZ Leticia, HERNÁNDEZ ESPINOZA Agustín, Introducción a las Redes Neuronales, fecha de recuperación: 30 de agosto de 2012, http://www.uaeh.edu.mx/docencia/P_Presentaciones/huejutla/sistemas/redes_neuronales/introduccion.pdf

[30] VARIOS AUTORES, Reconocimiento de voz con redes neuronales, DTW y Modelos ocultos de Markov, fecha de recuperación: 18 de octubre de 2012, http://redalyc.uaemex.mx/pdf/944/94403203.pdf

[31] Wikipedia, Proceso estocástico, fecha de recuperación: 18 de octubre de 2012, http://es.wikipedia.org/wiki/Proceso_estoc%C3%A1stico

[32] VARIOS AUTORES, Reconocimiento de voz mediante Modelos ocultos de Markov, fecha de recuperación: 18 de octubre de 2012, http://abbecerra.files.wordpress.com/2009/08/pdf.pdf

[33]Wikipedia, Densidad Espectral, fecha de recuperación: 18 de octubre de 2012, http://es.wikipedia.org/wiki/Densidad_espectral

[34] SALCEDO CAMPOS Francisco Javier, Modelos Ocultos de Markov. Del Reconocimiento de voz a la música, fecha de recuperación: 19 de octubre de 2012, http://books.google.com.ec/books?id=L2d0eVFr004C&pg=PA94&lpg=PA94&dq=modelo+oculto+de+markov+erg%C3%B3dico&source=bl&ots=PkleiPEyPj&sig=1zzn3vaBK8HAe67SqnzfzxQkZkg&hl=es&sa=X&ei=GpiBUNSFMYbe9ASGn4CoAw&ved=0CB8Q6AEwAA#v=onepage&q=modelo%20oculto%20de%20markov%20erg%C3%B3dico&f=false

[35] SANCHEZ LAZARO Ángel Luis*, Redes Neuronales Artificiales Aplicadas al Reconocimiento de palabras Independiente del Locutor,* Tesis Universidad de Valladolid, España 1993.

[36] Thales.Cica, Características de las redes neuronales, fecha de recuperación: 30 de agosto de 2012, http://thales.cica.es/rd/Recursos/rd98/TecInfo/07/capitulo3.html

[37] Varpa, Aprendizaje y Entrenamiento, fecha de publicación: 30 de agosto de 2012, http://www.varpa.org/~mgpenedo/cursos/scx/archivospdf/Tema2-0.pdf

[38] Monografías, Redes Neuronales, fecha de recuperación: 30 de agosto de 2012, http://www.monografias.com/trabajos12/redneuro/redneuro2.shtml

[39] LDI Universidad Carlos III de Madrid, Redes de Neuronas Artificiales, fecha de recuperación: 4 de septiembre de 2012, http://www.lab.inf.uc3m.es/~a0080630/redes-de-neuronas/kohonen-lvq.html

[40] DÍAZ MORENO Cristina, Clasificación no Supervisada, fecha de recuperación: 11 de septiembre de 2012, http://clustering.50webs.com/supervisadovsnosupervisado.html

[41]Universidad Carlos III de Madrid, Evolutionary Algorithms and Neural Networks and Artificial Intelligence, fecha de recuperación: 11 de septiembre de 2012, http://eva.evannai.inf.uc3m.es/et/docencia/rn-inf/documentacion/Tema5-AprendizajeNoSupervisado.pdf

[42] Universidad de Huelva, Aprendizaje por refuerzo, fecha de recuperación: 17 de septiembre de 2012, http://www.uhu.es/470004009/docs/tema6_bn.pdf

[43]VILLANUEVA ESPINOZA, María del Rosario, Las Redes Neuronales Artificiales y su importancia como herramienta en la toma de decisiones, fecha de recuperación: 17 de septiembre de 2012, http://sisbib.unmsm.edu.pe/bibvirtualdata/tesis/basic/Villanueva_EM/enPDF/Cap5.PDF

[44] VARIOS AUTORES, Aprendizaje con Redes Neuronales Artificiales, fecha de recuperación: 19 de septiembre de 2012, http://www.uclm.es/ab/educacion/ensayos/pdf/revista9/9_19.pdf

[45] Slide Share Inc., Simuladores de Redes Neuronales, fecha de recuperación: 1 de noviembre de 2012, http://www.slideshare.net/mentelibre/1-simuladores-rna

[46] Instituto Politécnico Nacional, Simuladores de Redes Neuronales, fecha de recuperación: 2 de noviembre de 2012, http://hugo-inc.net16.net/RNA/Unidad%206/6.1.html

[47] VARIOS EDITORES, Advances Concurrent Engineering, fecha de recuperación: 5 de noviembre de 2012, http://books.google.com.ec/books?id=Tyz9KulYeEsC&pg=PA739&lpg=PA739&dq=FuNeGen&source=bl&ots=Ynxd2uWZ4t&sig=IwyVuiVFzTrq81jCjD_TIrpSKM8&hl=es&sa=X&ei=8TFrUP-eOYeD0QH9yYDABg&ved=0CD4Q6AEwBA#v=onepage&q=FuNeGen&f=false

[48] Wikipedia, Simulink, fecha de recuperación: 5 de noviembre de 2012, http://es.wikipedia.org/wiki/Simulink

[49] KRIKORIAN Mauro, Reconocimiento de Dígitos Manuscritos Aplicando Transformadas Wavelet sin Submuestreo y Máquinas de Soporte Vectorial, fecha de recuperación: 6 de noviembre de 2012, http://www.dc.uba.ar/inv/tesis/licenciatura/2010/krikorian

[50] Linux-Magazine, Programación de redes neuronales con Libfann, fecha de recuperación: 7 de noviembre de 2012, http://www.linux-magazine.es/issue/35/056-060_RedesNeuralesLM35.crop.pdf

[51] AsturSolver, Simulador de redes neuronales JavaNNS, fecha de recuperación: 7 de noviembre de 2012, http://www.astursolver.net/blog/?p=79

[52] Universidad Nacional de Educación a Distancia, Sistemas Basados en el Conocimiento II: Introducción a la Neurocomputación, fecha de recuperación: 9 de noviembre de 2012, http://es.scribd.com/doc/53451515/30/Simulacion-con-JavaNNS#page=79

[53] DUQUE, Camilo, Redes Neuronales Artificiales, fecha de recuperación: 15 de novimebre de 2012, http://www.slideshare.net/camilorene/inteligencia-artificial-clase-3

[54] FLÓREZ LÓPEZ, Raquel, FENÁNDEZ, FERNÁNDEZ, José Miguel, Las Redes Neuronales Artificiales, Fundamentos Teóricos y aplicaciones prácticas, fecha de recuperación: 16 de noviembre de 2012, http://books.google.com.ec/books?id=X0uLwi1Ap4QC&pg=PA56&lpg=PA56&dq=red+maquina+de+cauchy&source=bl&ots=gLMzelpr3j&sig=7bOm9fYQj6gwP6zZs

xQr6FQXaFw&hl=es&sa=X&ei=BD6lUL3IHJD68QS5lIHoDw&ved=0CB8Q6AEwAA#v=onepage&q&f=false

[55] GONZÁLEZ MOLINA, F, Aplicación de Redes Neuronales en el cálculo de sobretensiones y tasa de contorneamientos, fecha de recuperación 20 de noviembre de 2012, http://www.tdx.cat/bitstream/handle/10803/6281/Capitulo_6a.PDF?sequence=9

[56] VILLANUEVA ESPINOZA, María del Rosario, Las Redes Neuronales Artificiales y su importancia como herramienta en la toma de decisiones, fecha de recuperación: 21 de noviembre de 2012, http://sisbib.unmsm.edu.pe/bibvirtualdata/tesis/basic/Villanueva_EM/enPDF/Cap3.PDF

[57] HERNANDEZ CARDOSA, Amaterazú, Capítulo 4: Algoritmos de Aprendizaje, fecha de recuperación: 23 de noviembre de 2012, http://catarina.udlap.mx/u_dl_a/tales/documentos/lem/oropeza_c_ca/capitulo4.pdf

[58] VARIOS AUTORES Universidad de Valencia, Redes Neuronales Artificiales, fecha de recuperación: 28 de noviembre de 2012, http://ocw.uv.es/ingenieria-y-arquitectura/1-2/libro_ocw_libro_de_redes.pdf

[59] DIAWEB, Redes Neuronales, fecha de recuperación: 29 de noviembre de 2012, http://avellano.usal.es/~lalonso/RNA/index.htm

[60] MUÑOZ, José, Redes Recurrentes para la Predicción, fecha de recuperación: 3 de diciembre de 2012, http://www.lcc.uma.es/~munozp/documentos/modelos_computacionales/temas/Tema8MC-05.pdf

[61] UNIVERSIDAD TECNOLÓGICA DE PEREIRA, Red de Elman, fecha de recuperación: 4 de diciembre de 2012, http://proton.ucting.udg.mx/posgrado/cursos/idc/neuronales2/EntrenamientoE.htm

[62] quegrande.org, Red de Hopfield y crecimiento de redes, fecha de recuperación: 5 de diciembre de 2012, http://quegrande.org/apuntes/EI/2/SC/teoria/red_de_hopfield_y_crecimiento_de_redes.pdf

[63]PERSO.WANADOo; Redes de Hopfield, fecha de recuperación: 7 de diciembre de 2012, http://perso.wanadoo.es/alimanya/hopfield.htm

[64] VARIOS AUTORES, Redes Neuronales-Máquina de Boltzmann, fecha de recuperación: 12 de diciembre de 2012, http://www.frlp.utn.edu.ar/materias/ia/IA2011-TrabajoTeorico-Grupo05.pdf

[65] NISSEN STEFFEN, FANN Fast Artificial Neural Network Library, fecha de recuperación 14 de ener de 2013, http://leenissen.dk/fann/wp/

[66] INSIDEHIGHERED.ORG, Neuroforecaster, fecha de recuperación: 17 de enero de 2013, http://www.insidehighered.org/is-nfga.htm

[67]ARTIFICIAL INTELLIGENCE, nn/xnn, fecha de recuperación: 19 de enero de 2012, http://www.sai.msu.su/sal/Z/3/NNXNN.html

[68] VARIOS AUTORES, LVQ_PAK The Learning Quantization Program Package, fecha de recuperación: 21 de enero de 2013, http://eva.evannai.inf.uc3m.es/et/docencia/rn-inf/documentacion/lvq_doc.pdf

[69] NIKOS DRAKOS, Lvq_pak, fecha de recuperación: 21 de enero de 2013, http://www.uow.edu.au/~markus/apods/doc/benchmark/TPbenchnode17.html

[70] WIKIPEDIA, Máquinas de Vectores de Soporte, fecha de recuperación: 21 de enero de 2013, http://es.wikipedia.org/wiki/M%C3%A1quinas_de_vectores_de_soporte

[71] THORSTEM Joachims, Support Vector Machine, fecha de recuperación: 21 de enero de 2013, http://download.joachims.org/svm_light/current/svm_light_linux.tar.gz

[72] THORSTEM Joachims, Support Vector Machine, fecha de recuperación: 21 de enero de 2013, http://svmlight.joachims.org/

[73] TANCO, Fernando, Introducción a las Redes Neuronales Artificiales, fecha de recuperación: 22 de enero de 2013, http://www.secyt.frba.utn.edu.ar/gia/RNA.pdf

[74] GONZALEZ, F, Aplicación de redes neuronales en el cálculo de sobretensiones y tasa de contorneamientos, fecha de recuperación: 24 de enero de 2013,

[75] VARPA, Aprendizaje y Entrenamiento, fecha de publicación: 29 de enero de 2013, http://www.varpa.org/~mgpenedo/cursos/scx/archivospdf/Tema4-0.pdf

[76] KINDELÁN, Ultano, Resolución de Sistemas lineales de ecuaciones: Método del gradiente conjugado, fecha de recuperación: 30 de enero de 2013, http://ocw.upm.es/matematica-aplicada/programacion-y-metodos-numericos/contenidos/TEMA_7/Apuntes/Tema7.pdf

[77] WIKIPEDIA, Memoria asociativa, fecha de recuperación: 31 de enero de 2013. http://es.wikipedia.org/wiki/Memoria_asociativa_(RNA)

[78] VALLE, José María, Perceptrón Multicapa, fecha de recuperación: 2 de febrero de 2013, http://eva.evannai.inf.uc3m.es/et/docencia/rn-inf/documentacion/Tema3-MLP.pdf

[79]W3C, Redes de Neuronas, fecha de recuperación: 1 de febrero de 2013, http://www.redesdeneuronas.com/mapas-de-kohonen.html

[80] Index of research/imagedatabase/groundtruth, fecha de recuperación: 4 de febrero de 2013, http://www.cs.washington.edu/research/imagedatabase/groundtruth

[81] POVEDA, Javier, E*squemas de votación borda aplicados a la clasificación de imágenes utilizando los histogramas de color RGB, HSV y el descriptor de distribución de color MPEG-7,* Universidad Politécnica Salesiana, Cuenca, enero de 2013.

[82] Elvex, clasificadores LVQ, fecha de recuperación: 4 de febrero de 2013, http://elvex.ugr.es/software/nc/help/spanish/nc/classification/lvq.html

[83]VARIOS autores, Desarrollo de una aplicación para trabajar con la técnica de "Stop Motion", fecha de recuperación 4 de febrero de 2013, http://www.fing.edu.uy/inco/pedeciba/bibliote/tgradoinf/tg-nsierra.pdf

[84] PAGUAY, Alba, URGILÉS, Pedro, *Recuperación de imágenes mediante extracción de blobs aplicando el operador laplaciano de Gauss y el kernel Gaussiano y desarrollo de un prototipo,* Universidad Politécnica Salesiana, Cuenca, febrero de 2012.

[85] NEUROLAB, fecha de recuperación: 4 de febrero de 2013, http://code.google.com/p/neurolab/

[86] SPIRALIA, Aprendiendo en red, fecha de recuperación: 4 de febrero de 2013, http://wiki.spiralia.net/index.cgi/Inicio/Cursos/PyNum/NumPy

[87] PDP8.CO, Index of, fecha de recuperación: 4 de febrero de 2013, http://pdp8.co.uk/library/www.ibiblio.org/neural-networks/programs/LVQPak/

ANEXOS

Anexo 1. Instalación Simulador FANN

FANN es un software de licencia GPL (General Public License), la versión que se va a utilizar es 2.2.0.

El simulador se ha descargado desde la página de internet **leenissen.dk/fann/wp/download** que será instalado en la plataforma LinuxMint.

La instalación es muy sencilla, como se indica a continuación:

1. El archivo que se descarga es FANN.2.2.0-Source.zip, por lo que debe ser descomprimido en alguna parte de LinuxMint.
2. Una vez hecho esto, dentro de la carpeta de FANN, en la pestaña Herramientas, escogemos abrir la carpeta actual en la terminal.

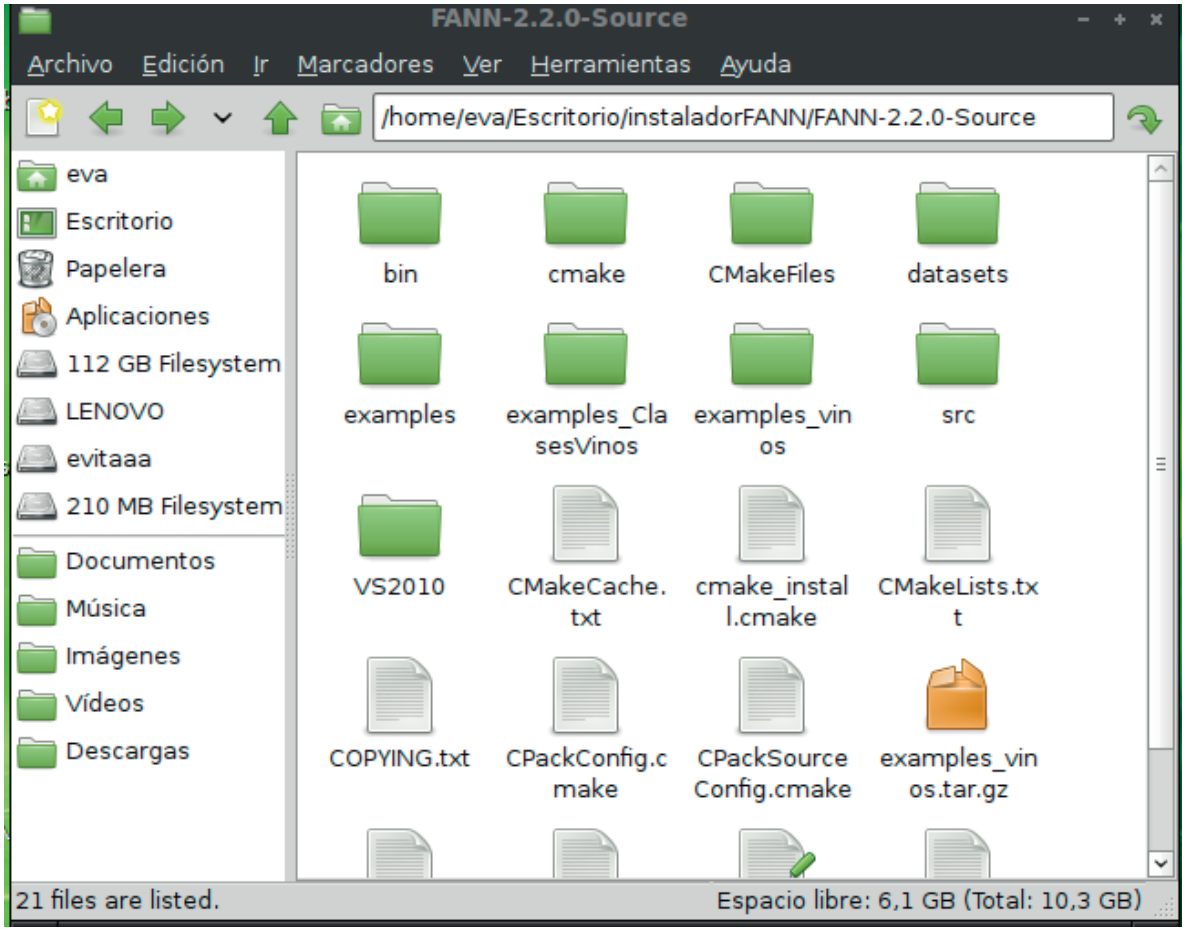


Ilustración A. 1: Muestra los archivos contenidos en la carpeta FANN.

3. Una vez en la terminal se prepara la compilación FANN, mediante la línea de código cmake.

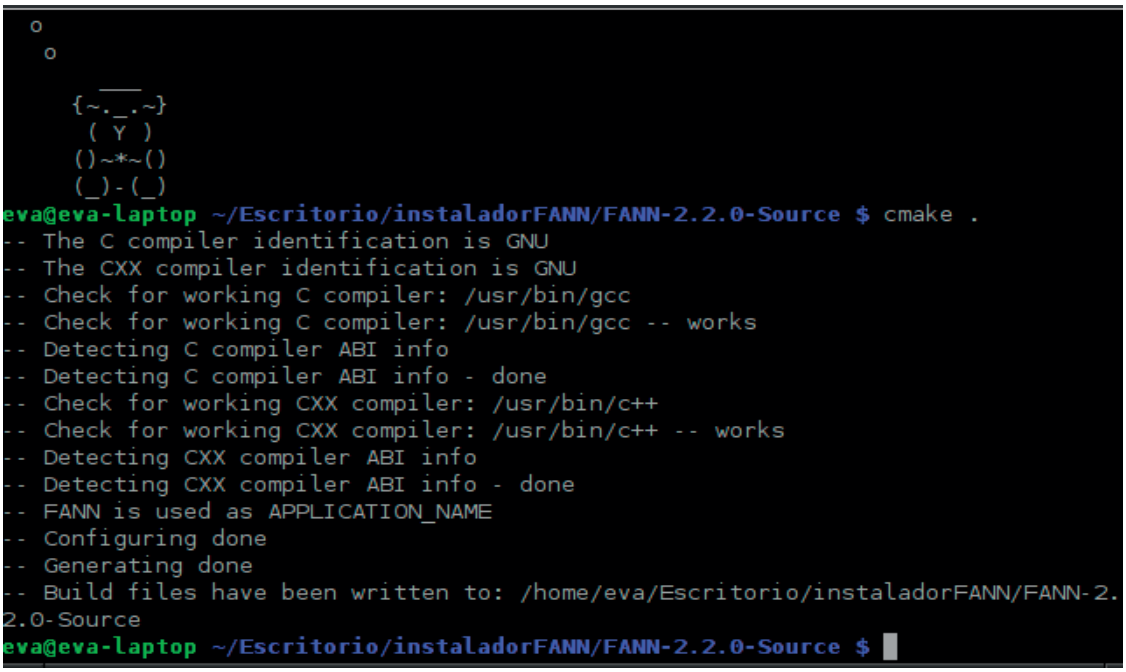


Ilustración A. 2: Se muesta la ejecución del archivo cmake.

4. Compilamos FANN, utilizando el código make.

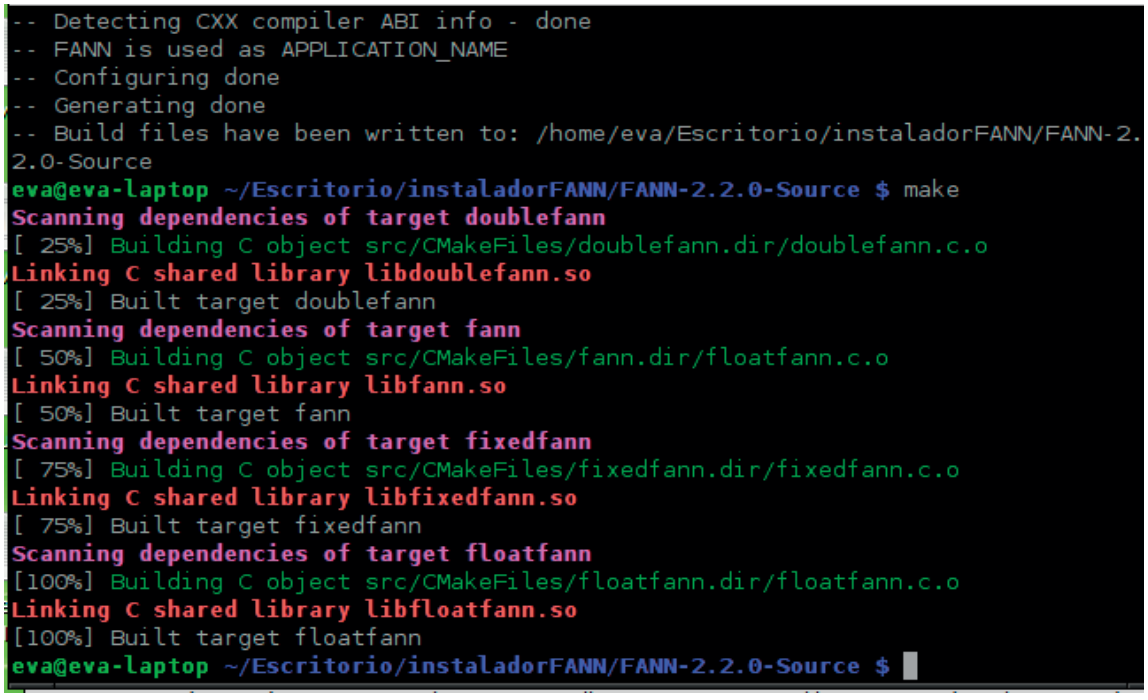


Ilustración A. 3: Muestra la compilación del programa mediante el comando make.

5. Se instala el simulador FANN, mediante la línea de código sudo-apt get install FANN.2.2.0-Source. Como indica la ilustración, la instalación es rápida y sencilla.

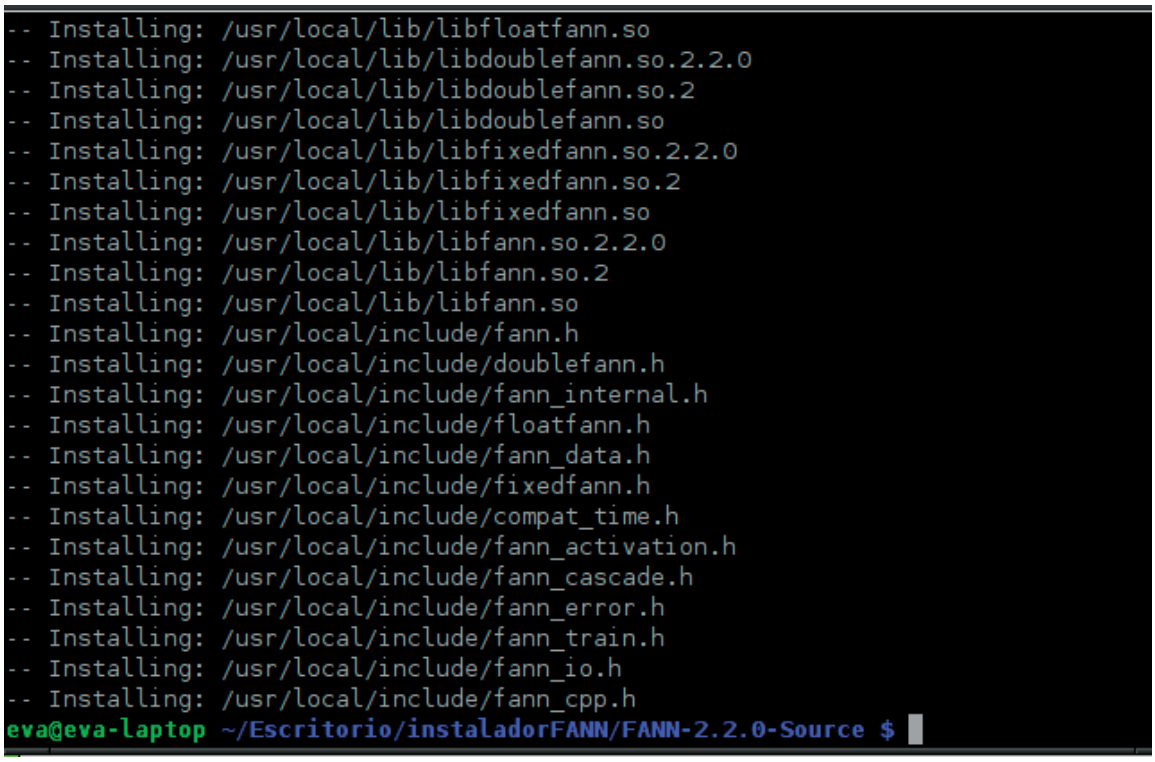


Ilustración A. 4: Muestra la instalación del programa FANN.

6. Una vez instalado el simulador FANN, podemos empezar a usarlo.

Anexo 2. Instalación simulador LVQ.

1. Se descarga el simulador Lvq desde [87].

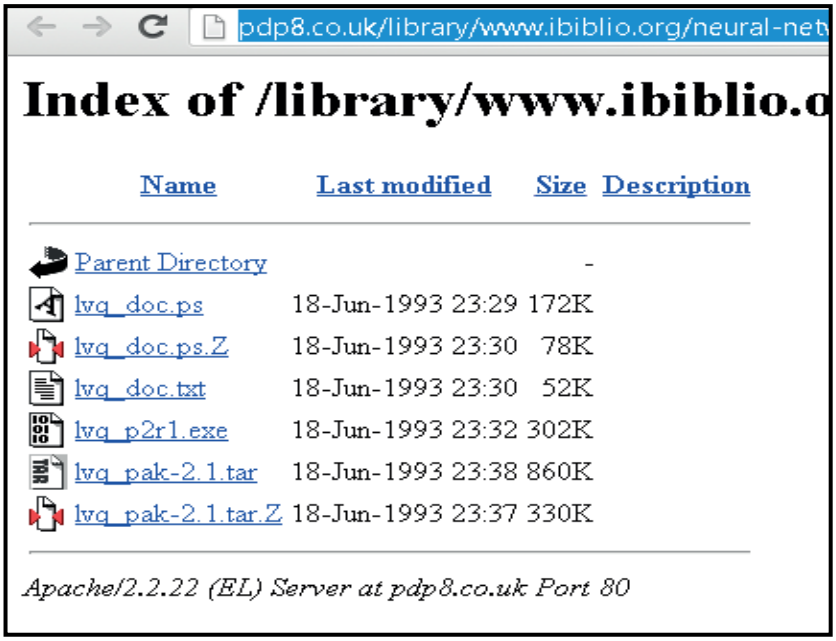


Ilustración A. 5: Muestra el enlace de descarga de LVQ_PK.

2. Se descomprime el archivo.

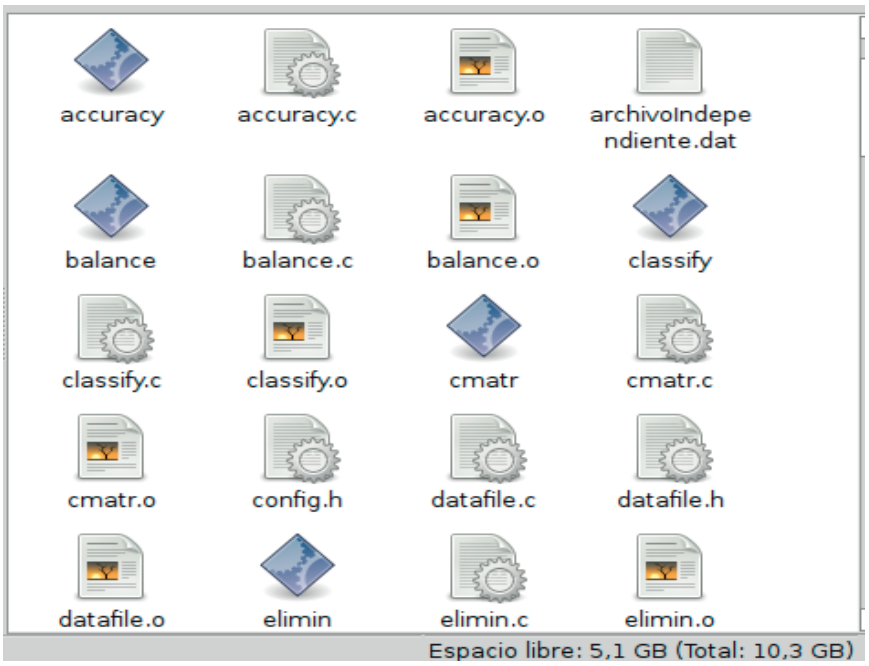


Ilustración A. 6: Muestra el contenido de la carpeta LVQ.

3. Se rea liza una copia del archivo makefile.unix a makefile.

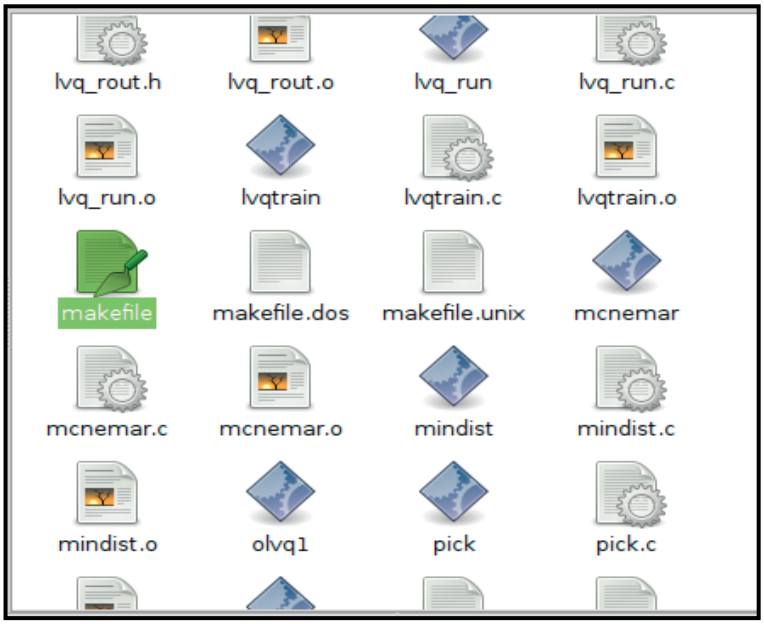


Ilustración A. 7: Muestra la copia del archivo makefile.unix a makefile.

4. Se ejecuta el make.

Anexo 3.

Instalación Python.

El lenguaje de programación Python ya viene instalado por defecto en el sistema operativo linux.

1. La librería Neurolab para trabajar con redes neuronales ha sido descargada desde [85].

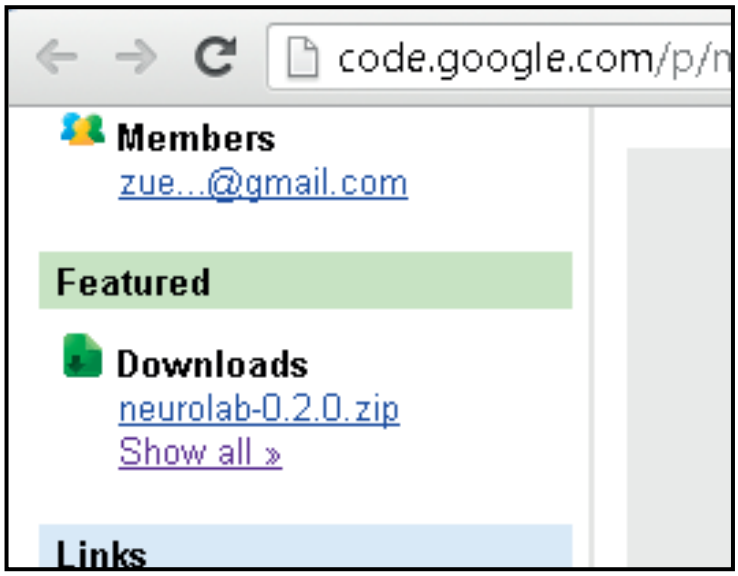


Ilustración A. 8: Muestra el sitio de descarga de la librería Neurolab.

2. Se descomprima la librería.

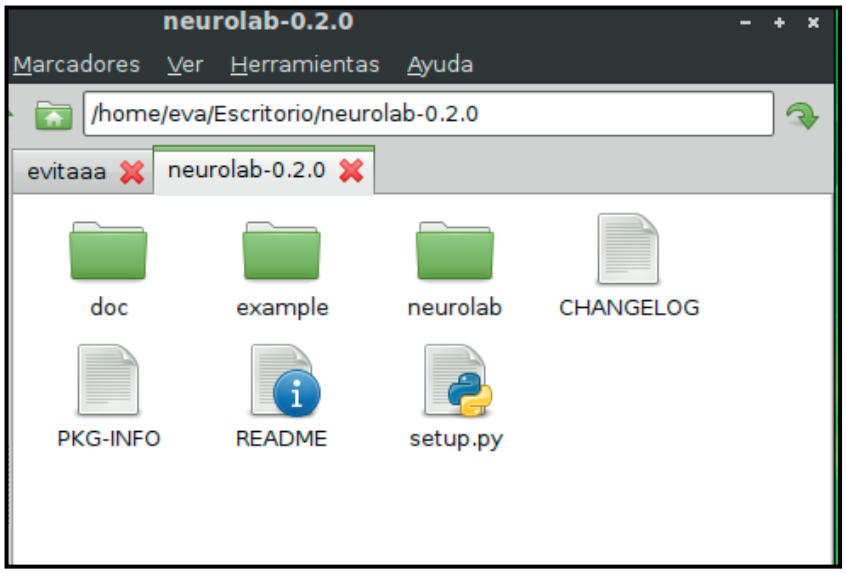


Ilustración A. 9: Muesta los archivos contenidos en la carpeta LVQ.

3. La librería está lista para ser utilizada, después de ejecutar: sudo python setup.py intall.

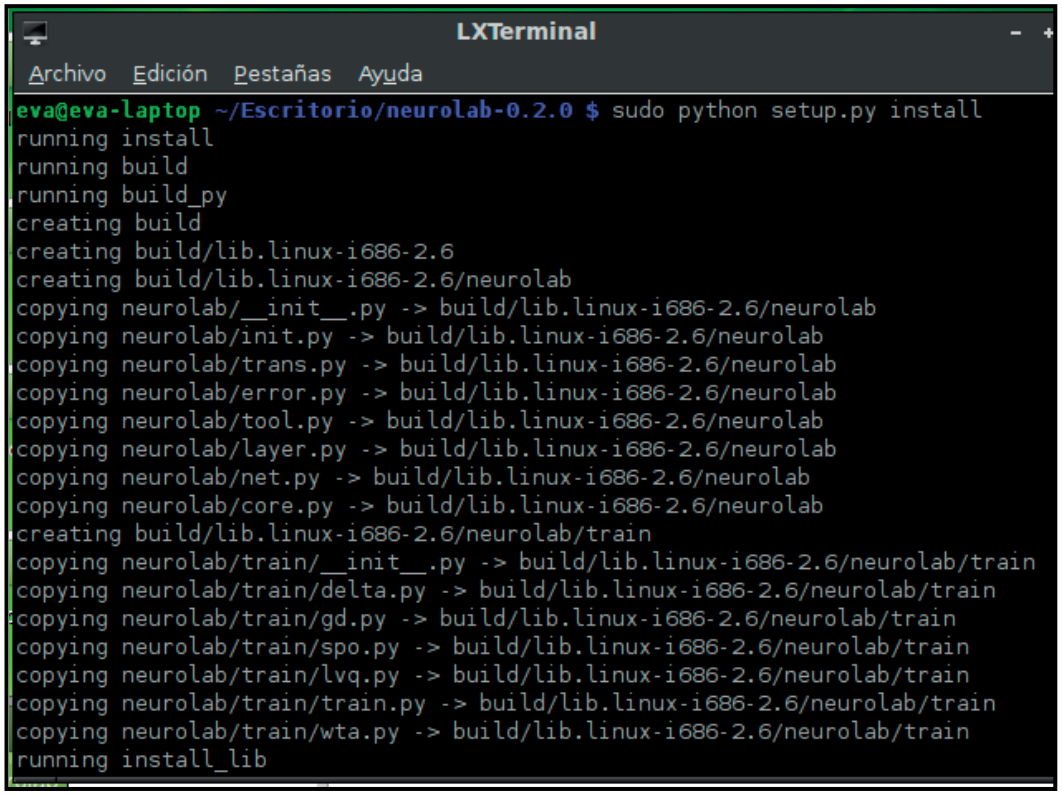


Ilustración A. 10: Muestra la ejecución del comando sudo python setup.py install.