UNIVERSIDAD NACIONAL HERMILIO VALDIZAN

ESCUELA DE POSTGRADO

USO DE REDES NEURONALES ARTIFICIALES PARA IDENTIFICACIÓN DE PERSONAS MEDIANTE EL RECONOCIMIENTO DE ROSTROS DE LA MUNICIPALIDAD DE LA VICTORIA – LIMA- 2015

TESIS PARA OPTAR EL GRADO ACADEMICO DE MAGISTER EN TECNOLOGÍAS DE LA INFORMACIÓN Y COMUNICACIÓN

TESISTA: FRANKLIN ARRIOLA RAMIREZ

LIMA, PERÚ

2015

DEDICATORIA

Dedico este trabajo a mi esposa Marianella y mis hijos Andrés Martín y Franklin quienes en todo momento me apoyaron con su paciencia y ánimo para llevar a cabo esta investigación.

AGRADECIMIENTOS

Agradezco al Dr. Pedro Villavicencio Guardia quien me asesoró en la elaboración de la presente investigación, a mis colegas la profesora Mg. Sally Torres Alvarado y la profesora Mg. Ivonne Ramsay Aniceto por compartir sus conocimientos en Redes Neuronales y Metodológicos.

RESUMEN

En el presente proyecto se aplicará reconocimiento de imágenes mediante redes neuronales para la identificación de rostros, lo cual será utilizado para el diseño de un prototipo que permita verificar que las personas que participen de las actividades organizadas por la Municipalidad de La Victoria-Lima sean solamente los beneficiarios debidamente registrados en los Programas Sociales. La identificación de rostros de personas será captada en tiempo real con una cámara web colocada en la laptop utilizada por el equipo de Trabajadores Sociales de la Municipalidad en el mismo lugar físico donde se realiza el evento, sin generar un costo adicional y/o instalación de equipos sofisticados.

En base a esta experiencia se planteará la solución al problema de identificación facial, basado en el problema de reconocimiento de patrones. Cabe mencionar que la presente investigación, además de resolver el problema de identificación facial también plantea crear un software en lenguaje Visual C++, que sirva como base para futuros desarrollos que se puedan realizar.

SUMMARY

In this project image recognition is implemented through neural networks for face identification, which will be used to implement a prototype to verify that the persons participating in the activities organized by the City of Victoria-Lima are only beneficiaries duly registered with the Social Programs. The identification of people's faces will be captured in real time with a web camera on the laptop used by the team of Social Workers of the Municipality in the same physical place where the event takes place without generating additional costs and / or implementation of sophisticated equipment.

Based on this experience, the solution to the problem of facial identification, based on pattern recognition problem will arise. It is noteworthy that the present investigation, in addition to solving the problem of facial identification also poses create software in Visual C ++ language, which serves as the basis for future developments that can be done.

INTRODUCCIÓN

Un gran número de investigadores ha centrado su atención en el desarrollo de nuevos sistemas de tratamiento de la información, que permitan solucionar problemas cotidianos, tal como lo hace el cerebro humano, desarrollando teoría de las Redes Neuronales Artificiales (RNA), las cuales emulan las redes neuronales biológicas. Una RNA, es un nuevo sistema para el tratamiento de la información, cuya unidad básica de procesamiento está inspirada en la célula fundamental del sistema nervioso humano, la neurona.

En el presente trabajo de investigación revisaremos el uso de RNA para el reconocimiento de rostros. Lo importante de la técnica de las RNA es su útil comportamiento al aprender, reconocer y aplicar relaciones entre objetos propios del mundo real. En este sentido, se utiliza las RNA como una herramienta para resolver problemas difíciles.

Hoy la tecnología de reconocimiento de rostros humanos está siendo utilizada para combatir el fraude de pasaportes, soporte al orden público, identificación de niños extraviados y minimizar el fraude en las identificaciones. Compara la imagen de una cara desconocida con todas las imágenes de caras conocidas que se encuentran en la base de datos para determinar su identidad. El sistema confirmará o rechazará la identidad de la cara.

En la Municipalidad de La Victoria – Lima Perú realiza el proceso de registro de los beneficiarios con sus datos personales, familiares, domiciliarios, entre otros. La Municipalidad organiza actividades generalmente en forma descentralizada, con la finalidad de favorecer a los beneficiarios en su misma zona de desarrollo. Para ello, siempre debe realizar un proceso de identificación para verificar que las personas que participen de estas actividades sean solamente

los beneficiarios debidamente registrados en los Programas Sociales. Para ello deben trasladar todos los archivos de datos o en todo caso, realizan una tarea previa para la generación de carnets de identificación. El riesgo de no realizar un proceso eficiente de identificación podría ser observado por los organismos reguladores del Estado (Contraloría General) por el uso indebido de los recursos de la Municipalidad en personas que no están consideradas en el grupo de Beneficiarios.

El reconocimiento facial haciendo uso de la RNA, será utilizado para la identificación de los beneficiarios a estos Programas Sociales durante las actividades que realiza la Municipalidad de La Victoria-Lima. Una de las características del presente proyecto, es el desarrollo de un sistema de bajo costo, por lo que se utiliza cámaras digitales externas o del mismo computador que se utilice para el procesamiento del prototipo, y un computador personal o computador portátil, por lo que el proyecto es viable económicamente. Como objetivos específicos tenemos instalar un hardware y elaborar un software para capturar imágenes de rostros, diseñar un prototipo para el reconocimiento de rostros utilizando redes neuronales artificiales, y diseñar un prototipo que identifique personas utilizando redes neuronales artificiales.

En el capítulo 1 se realiza el Planteamiento del Problema de investigación, con la definición del Problema Principal, Objetivos, Hipótesis planteadas, definición de las variables, justificación e importancia, viabilidad y limitaciones.

En el Capítulo 2 se desarrolla el Marco Teórico, mostrando antecedentes de la investigación, Bases Teóricas para entender los fundamentos de esta investigación y los temas que inspiraron el avance de las RNA, así como las características de las RNA.

En el Capítulo 3 se presenta la Metodología de Investigación utilizada así como el diseño de la solución, y la definición de los algoritmos utilizados para la construcción del prototipo para el reconocimiento de rostros. Se detalla la instalación del Hardware del sistema de reconocimiento de rostros, la construcción de software de captura de cuadros de imágenes, la construcción de pre-procesamiento de cuadros de imágenes capturadas, el diseño de la RNA para el reconocimiento de rostros, el diseño de algoritmo de aprendizaje de la RNA, la construcción y pruebas del software del algoritmos de aprendizaje, el diseño de algoritmo de funcionamiento de la RNA, y por último la construcción y pruebas del software del algoritmo de funcionamiento de la RNA.

En el Capítulo 4 se analizará los resultados obtenidos al ejecutar el prototipo para el reconocimiento de rostros con una muestra de fotografías.

Por último, se presentan las conclusiones y recomendaciones identificadas luego de lo desarrollado y revisado en el presente trabajo de investigación. En la parte final se detallan los anexos utilizados.

.

INDICE

CAPÍTULO I

EL PROBLEMA DE INVESTIGACIÓN

1.1. DESCRIPCIÓN DEL PROBLEMA

En la Municipalidad de La Victoria – Lima Perú se llevan a cabo Programas Sociales en beneficio de su comunidad. Para ello, la Municipalidad lleva a cabo un proceso de registro de los beneficiarios con sus datos personales, familiares, domiciliarios, entre otros.

Estos Programas Sociales organizan actividades generalmente en forma descentralizada, con la finalidad de favorecer a los beneficiarios en su misma zona de desarrollo.

Asimismo, para llevar a cabo estas actividades el equipo de Trabajadores Sociales de la Municipalidad de La Victoria, siempre deben realizar un proceso de identificación para verificar que las personas que participen de estas actividades sean solamente los beneficiarios debidamente registrados en los Programas Sociales.

El número de beneficiarios de estos Programas Sociales es voluminoso y complejo debido a que no solamente registra al Titular de una familia sino a todos sus miembros que estén bajo su entorno y que también se vean beneficiados con estos Programas Sociales. A su vez para ello deben trasladar todos los archivos de datos o en todo caso, realizan una tarea previa para la generación de carnets de identificación, lo cual en muchos casos no se puede llevar a cabo por la inasistencia de todos los beneficiarios para este proceso.

El riesgo de brindar un servicio o bien como resultado de estas actividades que organizan los Programas Sociales de la Municipalidad de La Victoria podría ser observado por los organismos reguladores del Estado (Contraloría General) por el uso indebido de los recursos de la Municipalidad en personas que no están consideradas en el grupo de Beneficiarios, solamente por el hecho de no haber podido realizar en forma eficiente la identificación de las personas que pudieron ser suplantadas de manera indebida.

Hoy la tecnología de reconocimiento de rostros humanos está siendo utilizada para combatir el fraude de pasaportes, soporte al orden público, identificación de niños extraviados y minimizar el fraude en las identificaciones. Compara la imagen de una cara desconocida con todas las imágenes de caras conocidas que se encuentran en la base de datos para determinar su identidad. El sistema confirmará o rechazará la identidad de la cara.

Lo cual también podría ser utilizado para la identificación de los beneficiarios a Programas Sociales durante las actividades que realiza la Municipalidad de La Victoria-Lima.

A su vez, el problema de reconocimiento de rostros humanos ha sido abordado, estudiado por una gran cantidad de investigadores[1] en los últimos años, los cuales han desarrollado distintas técnicas. Algunas de las técnicas necesitan un gran equipo (cámaras de alta resolución, sensores laser, etc.), cuyo alto costo impide su comercialización.

En el caso de las actividades que los Programas Sociales de la Municipalidad de La Victoria organizan en la comunidad se llevan a cabo de forma descentralizada en los mismos lugares donde se ubican los

beneficiarios, lo cual dificulta el uso de tecnologías de reconocimiento de rostros con equipos sofisticados. En el presente trabajo se estaría planteando una solución que no conlleve mayor gasto y permita la portabilidad de la tecnología necesaria para la identificación de los beneficiarios debidamente registrados en el Padrón del Programa Social.

1.2. FORMULACIÓN DEL PROBLEMA

1.2.1. PROBLEMA GENERAL

¿En qué medida las redes neuronales artificiales permiten la identificación de personas mediante el reconocimiento de rostros del Padrón de beneficiarios en los Programas Sociales de la Municipalidad de La Victoria, Lima - 2015?

1.2.2. PROBLEMAS ESPECÍFICOS

- ¿De qué manera se capturan imágenes de rostros?
- ¿De qué manera se reconocen rostros utilizando redes neuronales artificiales?
- ¿En qué medida se identifican personas utilizando redes neuronales artificiales?

1.3. OBJETIVOS DE LA INVESTIGACIÓN

1.3.1. OBJETIVO GENERAL

Diseñar un prototipo con el uso de redes neuronales artificiales que permitan la identificación de personas mediante el reconocimiento de rostros de la Municipalidad de La Victoria-Lima 2015

1.3.2. OBJETIVOS ESPECÍFICOS

- Instalar un hardware y elaborar un software para capturar imágenes de rostros
- Diseñar un prototipo para el reconocimiento de rostros utilizando redes neuronales artificiales
- Diseñar un prototipo que identifique personas utilizando redes neuronales artificiales

1.4. HIPÓTESIS DE LA INVESTIGACIÓN

1.4.1. HIPÓTESIS GENERAL

Las redes neuronales artificiales permiten la identificación de personas mediante el reconocimiento de rostros.

1.4.2. HIPÓTESIS ESPECÍFICAS

- Es posible diseñar la captura de imágenes utilizando hardware y software
- Es posible diseñar un prototipo para el reconocimiento de rostros utilizando redes neuronales artificiales
- Es posible diseñar un prototipo que identifique personas utilizando redes neuronales artificiales

1.5. VARIABLES E INDICADORES

1.5.1. VARIABLES INDEPENDIENTES

Redes neuronales artificiales

1.5.2. VARIABLES DEPENDIENTES

Identificación de personas mediante el reconocimiento de rostros.

1.5.3. VARIABLES INTERVINIENTES

No tiene variables intervinientes.

1.6. JUSTIFICACIÓN E IMPORTANCIA DE LA INVESTIGACIÓN

- Justificación teórica

  La presente tesis es de suma importancia en nuestro medio ya que permitirá profundizar el conocimiento de redes neuronales y su aplicación en el desarrollo de prototipos.

- Justificación metódica

  Se aplicarán metodologías de desarrollo de sistemas para el diseño, construcción de algoritmos de redes neuronales.

- Justificación práctica

  - La aplicación de tecnologías para el reconocimiento de rostros con bajo costo para su uso sin necesidad de afectar el objetivo principal de los Programas Sociales el cual es brindar mayores servicios o bienes a los beneficiarios.
  - La portabilidad de tecnologías, a bajo costo, para el reconocimiento de rostros en los diferentes puntos descentralizados donde se llevan a cabo las actividades de los Programas Sociales de la Municipalidad de La Victoria-Lima.

- Justificación legal

  Se apoyaría a la Municipalidad de La Victoria con la identificación oportuna de los beneficiarios registrados para participar en las actividades de los Programas Sociales, evitando que personas inescrupulosas que podrían suplantar a otras, podría conllevar a una falta grave por un mal uso de los recursos de la Municipalidad de La

Victoria-Lima, lo cual es observado por los organismos de control del Estado.

- Justificación científica

  Las Redes Neuronales Artificiales (RNA) constituyen una línea de investigación del movimiento científico de la cibernética que investigó sobre los temas de la Inteligencia Artificial (IA), la cual tiene como objetivo primario la construcción de máquinas inteligentes Este movimiento científico se articuló en torno a la idea de que el funcionamiento de muchos sistemas, vivos o artificiales, puede ser captado mejor por modelos basados en la transferencia de información que por modelos basados en la transferencia de energía.

- Justificación psicológica

  El aprendizaje es un aspecto de la psicología humana, lo cual es necesario entenderlo para aplicarlo en las Redes Neuronales Artifiicales (RNA).

- Justificación Epistemológica

  En el desarrollo del trabajo de investigación se tratará frecuentemente los temas de la generación y validación del conocimiento lo cual es un tema fundamental de la epistemología.

1.7. VIABILIDAD DE LA INVESTIGACIÓN

Una de las características del presente proyecto, es el desarrollo de un sistema de bajo costo, por lo que se utiliza cámaras digitales externas o del mismo computador que se utilice para el procesamiento del

prototipo, y un computador personal o computador portátil, por lo que el proyecto es viable económicamente.

Asimismo, se dispone de información teórica acerca de las redes neuronales, así como de técnicas de algoritmos lo cual va a ser utilizado en el presente proyecto. La intención de esta tesis es analizar uno de los modelos de redes neuronales que existen, utilizar su algoritmo, codificando y haciendo un análisis de sus resultados. A su vez, los instrumentos a utilizar son viables de obtener, debido a que se utilizarán fotografías ya tomadas con alguna cámara convencional que grabe la información en formato jpg o bmp.

1.8. LIMITACIONES

- Limitación teórica

  Existe información teórica de redes neuronales así como de los diferentes métodos de aplicación. Sin embargo, es limitada la información con respecto a cómo se podrían diseñar y usar estos métodos en algoritmos y código de programación.

- Limitación metodológica

  La codificación de este tipo de modelos de redes neuronales requiere un nivel de precisión para lo cual se deben ejecutar continuamente los algoritmos que se codifiquen hasta lograr un nivel de precisión. En este caso la limitación será el tiempo que se disponga para presentar el trabajo final.

- Limitación de recursos

  Otra limitación será que la investigación se restringirá a utilizar fotografías en blanco y negro.

Estas limitaciones son el empuje de esta investigación para poder brindar un avance en este ámbito de temas, y compartir en el ámbito académico y de investigación para avanzar en nuestro medio con respecto a la industria del software.

CAPÍTULO II

MARCO TEORICO

2.1. ANTECEDENTES DE LA INVESTIGACIÓN

A mediados de los años 60, se desarrolló el primer sistema semi automatizado de reconocimiento facial, el cual funcionaba midiendo de forma precisa distancias a rasgos como los ojos, las orejas, la nariz y la boca en las fotografías, y posteriormente buscando coincidencias, comparando con una base de datos. Es relativamente nuevo el concepto te concepto del reconocimiento facial computarizado.

En los años 70 llegó la automatización del reconocimiento facial, cuando se comenzó a utilizar características como el color del cabello o el grosor de los labios.

La biometría facial surge a partir de los años noventa, aunque recién en el 2001 se inició su uso práctico, con la celebración de la SuperBowl del NFL, en la cual se archivaron fotografías de los sistemas de vigilancia y luego se compararon con bases de datos digitales.

2.1.1. A NIVEL INTERNACIONAL

- Moreno Díaz, Ana Belén, en su Tesis Doctoral, “RECONOCIMIENTO FACIAL AUTOMATICO MEDIANTE TECNICAS DE VISION TRIDIMENSIONAL”, Universidad politécnica de Madrid-Facultad de Informática (2004).

Las imágenes no son afectadas por cambio de iluminación, debido a que se presenta como una imagen de nubes de puntos o una malla 3D, siendo una ventaja en este tipo de reconocimiento, dado que utiliza la técnica de visión tridimensional.

En este trabajo de tesis, se selecciona un conjunto de descriptores de forma 3D a partir de un análisis de relevancia haciendo uso de coeficientes de Fisher en diferentes regiones del rostro los cuales hacen parte de un modelo antropométrico del rostro. El rendimiento del sistema de reconocimiento se incrementa como resultado del análisis de relevancia.

- Jorge Rafael Valvert Gamboa, en su trabajo de graduación, "MÉTODOS Y TÉCNICAS DE RECONOCIMIENTO DE ROSTROS EN IMÁGENES DIGITALES BIDIMENSIONALES**",** Universidad de San Carlos de Guatemala-Facultad de Ingeniería (2006), para el reconocimiento de un rostro en una imagen digital bidimensional se describen las fases que se deben cubrir.

  Para el reconocimiento y aprendizaje de rostros humanos, se dan a conocer los métodos de inteligencia artificial (enfocado principalmente en redes neuronales y algoritmos genéticos), además de las técnicas de identificación de patrones, de reconocimiento de rostros humanos en una imagen a través de rasgos característicos aprendidos. Considera además una aplicación para el aeropuerto nacional de Guatemala y, a su vez, se describen dos aplicaciones comerciales existentes y sus características básicas.

- Henry Arguello Fuentes, en su trabajo de investigación “SISTEMAS DE RECONOCIMIENTO BASADOS EN LA IMAGEN FACIAL”, Universidad Industrial de Santander- Colombia (2011). Consolida las principales investigaciones que se están llevando a cabo en el área de los sistemas de reconocimiento a través de la imagen facial. Se describe las principales líneas de trabajo en los sistemas de identificación de personas haciendo uso de la imagen del rostro. Asimismo, se sintetiza las últimas técnicas matemáticas para realizar la extracción de características dentro de estos sistemas de identificación.
- Rodríguez y Sani, en su trabajo de graduación, “DISEÑO E IMPLEMENTACIÓN DE UN SISTEMA DE SEGURIDAD BASADO EN EL RECONOCIMIENTO DE CARAS HUMANAS, Escuela Politécnica Nacional de Quito (2004). El sistema diseñado identifica a una persona a partir de una imagen de su cara, para lograr esto la imagen es procesada con un banco de filtros de Gabor, el cual resalta los rasgos faciales más representativos de la cara (ojos, nariz, boca, etc.), los mismos que son utilizados en la comparación con otros rasgos faciales almacenados previamente. Para ello, utiliza la biométrica y métodos de procesamiento de imágenes,
- Cabello Pardos, Enrique, tesis doctoral, “Técnicas de reconocimiento facial mediante redes neuronales, Universidad Politécnica de Madrid- facultad de informática (2004).

  En esta Tesis se exploran soluciones a la verificación facial. Se han estudiado técnicas basadas en imágenes bidimensionales y se ha realizado un estudio inicial basado en un modelo tridimensional de

la cara. Se han comparado tres clasificadores (k vecinos más cercanos, redes neuronales del tipo funciones de base radial y máquinas de vector soporte), mostrando los resultados obtenidos con los diferentes datos de entrada. En dos dimensiones se han propuesto dos técnicas de reducción de información, basadas en la utilización de análisis de componentes principales (PCA) y el empleo de imágenes de baja resolución. En la verificación partiendo de imágenes se puede observar que el clasificador que mejores resultados proporciona es el basado en máquinas de vector soporte (SVM), y el mejor método de procesamiento es PCA utilizando una plantilla por sujeto y siendo esta plantilla una imagen del sujeto que se quiere verificar. Además se ha desarrollado un experimento inicial que permite disponer de una idea intuitiva sobre el funcionamiento de un sistema de verificación facial basado en datos tridimensionales. En esta Tesis se ha desarrollado un modelo de cara en tres dimensiones y se ha comprobado que es posible su ajuste a una nube de puntos poco densa con un error pequeño. Este tipo de datos permite obtener resultados más robustos frente a actitudes no colaborativas de los sujetos.

- Montaña, t. Yubraska L, Giuseppe d., tesis de grado, “Desarrollo de una aplicación para el reconocimiento facial mediante redes neuronales artificiales”, departamento de computación y sistemas-Universidad del Oriente, Cumana-Venezuela-2009. Basadas en niveles de gris mediante el uso de redes neuronales artificiales, se ha desarrollado una aplicación para el reconocimiento de imágenes faciales. A través de esta herramienta el usuario podrá indicar los

rostros que la aplicación debe memorizar durante el aprendizaje, también podrá seleccionar algunos parámetros de las redes neuronales, como son: el número de capas y el número de neuronas empleadas por cada capa y el factor de aprendizaje, una vez configurados todos estos parámetros se podrá dar inicio al entrenamiento de las redes neuronales perceptrón multicapa y función de base radial, una vez finalizado el entrenamiento se genera un archivo con todos los pesos, umbrales y demás resultados obtenidos durante el entrenamiento, necesarios para que las redes logren el reconocimiento, este archivo es denominado memoria y se actualiza cada vez que se encuentra una mejor solución a un problema, por lo tanto, se puede entrenar tantas veces como se desee en busca de una mejor solución. Una vez que las redes neuronales artificiales se encuentren entrenadas estas serán capaces de identificar o verificar con un alto grado de acierto los sujetos aprendidos.

- Gibran Fuentes Pineda, grado de maestro, “Reconocimiento de rostros mediante wavelets y redes neuronales”, instituto politécnico nacional- México- Facultad de ingeniería en microelectrónica (2008). Se propone un método para reconocer rostros a cambios de iluminación, postura y expresión facial. El método propuesto se encuentra dividido en dos etapas fundamentales:

1) Extracción de características faciales por medio de la transformada wavelet discreta (TWD)
2) Clasificación de patrones mediante la red neuronal perceptrón multicapa a partir de los vectores característicos extraídos.

La TWD es usada eficientemente por el algoritmo de la transformada rápida wavelet (TRF), el cual se lleva a cabo mediante la aplicación recursiva de una serie de filtros paso-bajo y paso-alto.

Para la etapa de clasificación de patrones se emplea, la red neuronal multicapa para resolver problemas de clasificación no lineal y generalizar a partir de un número limitado de ejemplos. La combinación de las redes neuronales y la TWD permite eliminar o al menos reducir algunos inconvenientes que presentan varios métodos propuestos de reconocimiento de rostros.

2.1.2. A NIVEL NACIONAL

- Jorge Gutiérrez Gutiérrez, trabajo de graduación, “DESARROLLO DE UN SISTEMA INTELIGENTE USANDO REDES HOPFIELD PARA EL RECONOCIMIENTO DE ROSTROS, Universidad Nacional de Trujillo- facultad de ciencias físicas y matemáticas- escuela académica profesional de informática (2013). En si trabajo de graduación para reconocimiento de rostros emplea Técnicas Para Extracción de Características, en la cual las imágenes faciales se seleccionan regiones de interés del rostro como ojos cejas y boca. Para el reconocimiento de rostros utiliza la red-hopfiel, Modelo construido en 1982 por el físico norteamericano J. Hopfield Hopfield redescubrió el mundo casi olvidado de las redes auto asociativas, caracterizadas por una nueva arquitectura y un nuevo funcionamiento, a las que se tuvo que añadir otro tipo de reglas de aprendizaje. Las consecuencias fueron redes con un

comportamiento diferente a las diseñadas con estructura progresiva hacia adelante (*feedforward*).

2.2. BASES TEÓRICAS

Los sistemas de computación secuencial desarrollados Von Neuman, utiliza el enfoque algorítmico para la resolución de problemas. Ello implica entender totalmente el problema en cuestión para poder construir un algoritmo que efectivamente lo pueda resolver. Este enfoque se ha venido aplicando con éxito en diversas áreas, sin embargo hoy en día existen problemas cuya complejidad, tales como reconocimiento de texto manuscrito, reconocimiento de voz, reconocimiento de rostros, control de procesos, etc., hace prácticamente imposible afrontarlos utilizando el enfoque algorítmico.

Esta dificultad de los sistemas de cómputo que trabajan bajo la filosofía de los sistemas secuenciales, desarrollados por Von Neuman, ha hecho que un gran número de investigadores centre su atención en el desarrollo de nuevos sistemas de tratamiento de la información, que permitan solucionar problemas cotidianos, tal como lo hace el cerebro humano, desarrollando teoría de las Redes Neuronales Artificiales (RNA), las cuales emulan las redes neuronales biológicas. Una RNA, el un nuevo sistema para el tratamiento de la información, cuya unidad básica de procesamiento está inspirada en la célula fundamental del sistema nervioso humano, la neurona.

2.2.1. NEURONAS BIOLÓGICAS

Las redes neuronales tienen sus comienzos a fines del siglo XIX, con el científico español Santiago Ramón y Cajal, en 1888, demostró que el

sistema nervioso está compuesto por una red de células individuales, las neuronas, ampliamente interconectadas entre sí. Estableció que la información fluye en la neurona desde las dendritas hacia el axón, atravesando el soma. Con este descubrimiento obtuvo el premio nobel de fisiología y medicina en 1906. [Bonifacio 1]

Las neuronas son las células que forman la corteza cerebral de los seres vivos. La capacidad de comunicarse es una de las características de las neuronas. Los tres componentes principales de las neuronas son los siguientes:

- Las dendritas
- El cuerpo o soma y un núcleo.
- El axón.

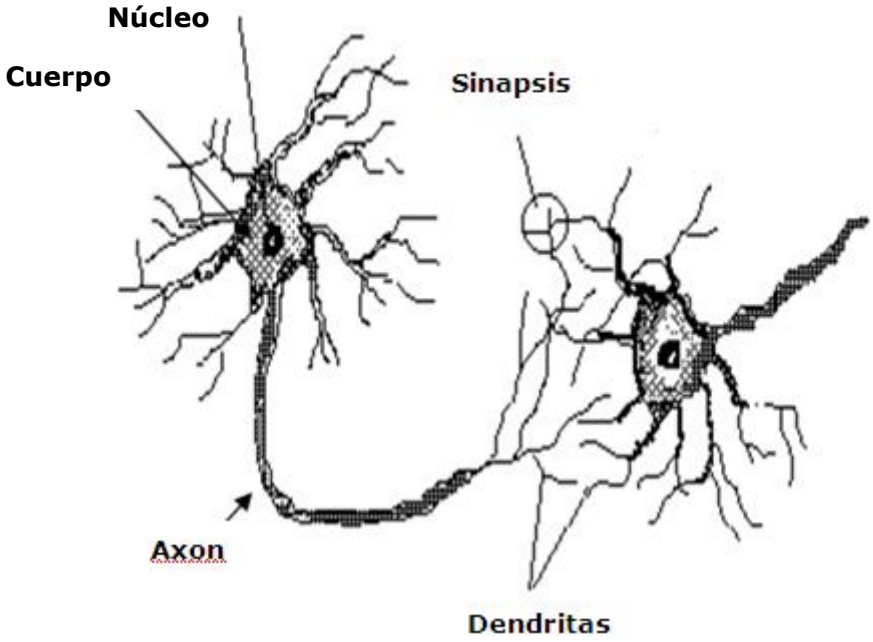


Figura 2.1 Estructura de una neurona biológica típica (Isasi Viñuela- GalvánLeón, 2004-p.4)

En la figura 2.1 se observa que el cuerpo de la neurona cuenta con un gran número de ramificaciones de entrada, las dendritas, que propagan la señal al interior de la neurona. El axón, es una ramificación de salida

de la neurona, a través de él se propaga una serie de impulsos eléctrico-químico. [Isasi 2]. La conexión entre el axón de una neurona y una dendrita de otra neurona recibe el nombre de sinapsis y determina la fuerza y tipo de conexión entre ellas [López 3]. El núcleo de la neurona procesa estímulos de entrada. [Isasi 2]

Las señales que utilizan las neuronas son de dos tipos: eléctrica y química. La señal generada por la neurona y transportada a lo largo del axón es un impulso eléctrico, en la sinapsis se produce una transformación del impulso eléctrico en un mensaje neuroquímico mediante la liberación de unas sustancias llamadas neurotransmisor.

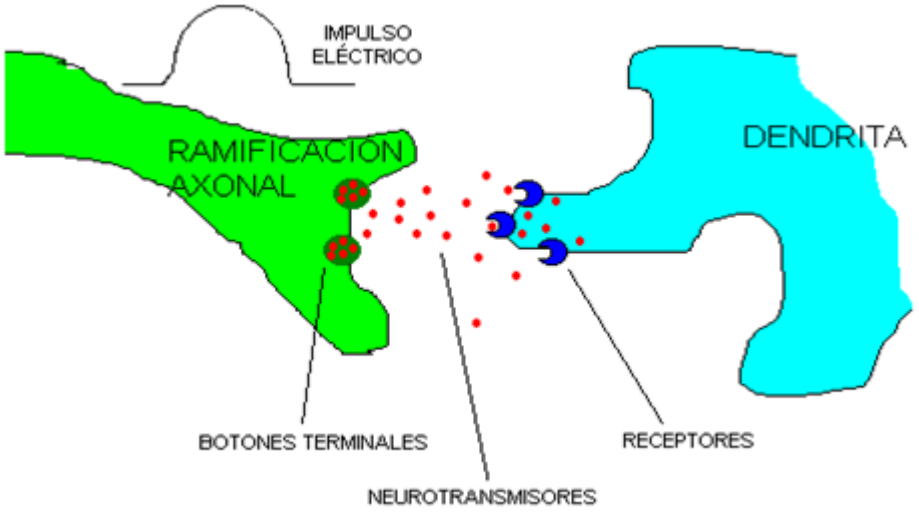


Figura 2.2. Detalle de una sinapsis
Fuente:http://magomar.webs.upv.es/rna/tutorial/RNA_marcos.html

Efecto de los neurotransmisores sobre la neurona receptora puede ser excitatorio (positivo) o inhibitorio (negativo). Las señales excitatorias e inhibitorias recibidas por una neurona se combinan, y en función de la estimulación total recibida, la neurona toma un cierto nivel de activación, que se traduce en la generación de breves impulsos nerviosos con una determinada frecuencia o tasa de disparo, y su

propagación a lo largo del axón hacia las neuronas con las cuales sinapta.

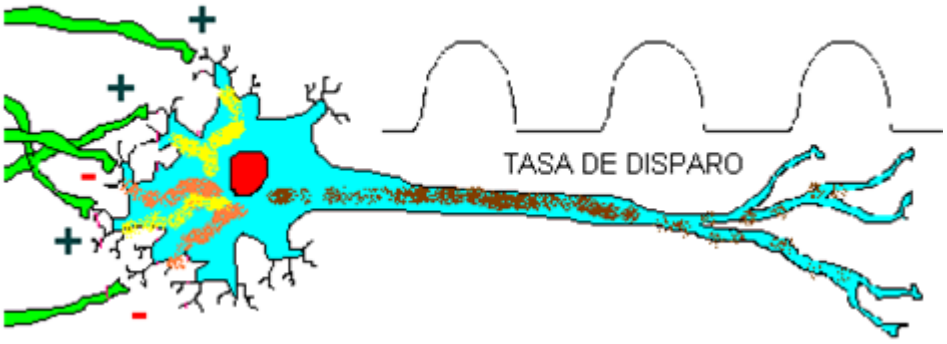


Figura 2.3.- Activación y disparo de una neurona.
Fuente: http://magomar.webs.upv.es/rna/tutorial/RNA_marcos.html

De esta manera el axón entrega su información como “señal de entrada” a una dendrita de otra neurona y así sucesivamente se va transmitiendo información de unas neuronas a otras y va siendo procesada a través de las conexiones sinápticas y las propias neuronas. Mediante la variación de la efectividad de las sinapsis se produce el aprendizaje de las redes neuronales.

El funcionamiento de la neurona es el siguiente: la sinapsis recoge información electro-química procedente de las células vecinas a la que está conectada, esta información llega al núcleo donde es procesada hasta generar una respuesta que es propagada por el axón, esta señal se ramifica y llega a las dendritas de otras neuronas, a través de lo que se denomina sinapsis, que es un espacio liquido donde existen determinadas concentraciones de elementos ionizados, normalmente de sodio y potasio. Estos iones hacen que el espacio intersinaptico posean determinadas propiedades de conectividad que activen o impidan el paso del impulso eléctrico. De esta forma la sinapsis se convierten en excitadores o inhibidores de la señal procedente de los

axones, actuando como aislantes o amplificadores a conveniencia. [Isasi 2]

El cerebro humano cuenta con millones de neuronas que se interconectan para formar "Redes Neuronales". Se calcula que hay aproximadamente 10 billones de neuronas en la corteza cerebral y 60 trillones de conexiones neuronales. [Haykin 4]. Estas redes ejecutan los millones de instrucciones necesarias para mantener una vida normal. Es difícil de fabricar, con la tecnología actual, un dispositivo con características similares.

Figura 2.4. Red neuronal biológica
Fuente: _http://logica-difusa.blogspot.com

2.2.2. FUNCIONAMIENTO DE UNA NEURONA BIOLÓGICA

La neurona es un procesador elemental. Una neurona recibe estímulos de entrada mediante las dendritas, estos estímulos son procesados mediante el núcleo de la neurona, para posteriormente emitir estímulos de salida mediante el axón.

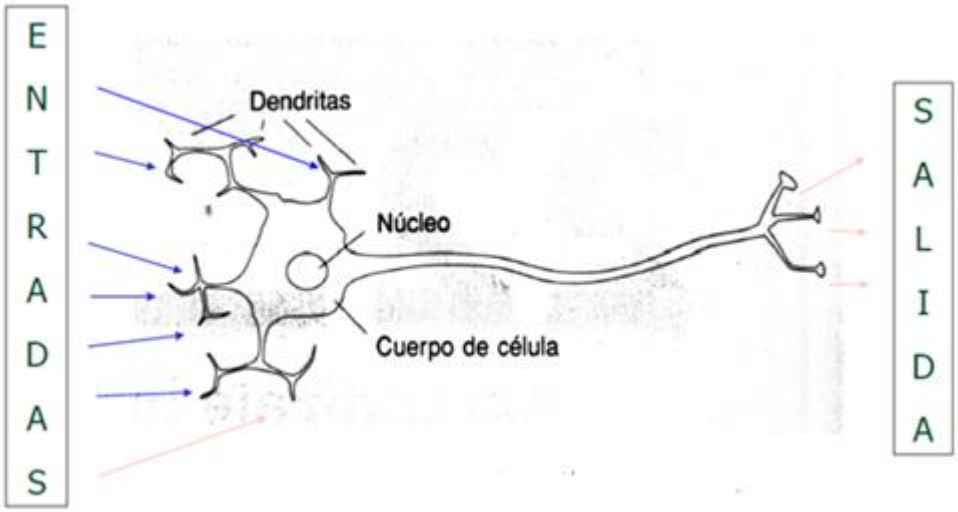


Figura 2.5. Funcionamiento de una neurona biológica (fuente propia)

2.2.3. RED NEURONAL ARTIFICIAL Y SU REPRESENTACIÓN

Una RNA según Haykin, es un procesador paralelo y distribuido que tiene la facilidad natural para almacenar conocimiento experimental y hacerlo útil para su uso, asemejando al cerebro en dos aspectos:

- El conocimiento es adquirido por la red a través de un proceso de aprendizaje
- La “fuerza” de las conexiones interneurona, conocida como pesos sinápticos son utilizados para almacenar el conocimiento. [Haykin 4]

Una RNA, según Freman y Skapura [Freman 5], es un sistema de procesadores paralelos conectados entre sí en forma de grafo dirigidos, que trata de emular el funcionamiento del cerebro. Esquemáticamente cada elemento de procesamiento llamados nodos representa a las neuronas y un conjunto de conexiones que

unen los nodos (representado en la figura 2.6) que llamaremos pesos. Lo importante de la técnica de las RNA es su útil comportamiento al aprender, reconocer y aplicar relaciones entre objetos propios del mundo real. En este sentido, se utiliza las RNA como una herramienta para resolver problemas difíciles.

Los nodos realizan la función de procesar la información que reciben, mediante la suma de sus entradas por sus respectivos pesos, y el resultado de esta es el argumento de una función de activación para producir una salida.

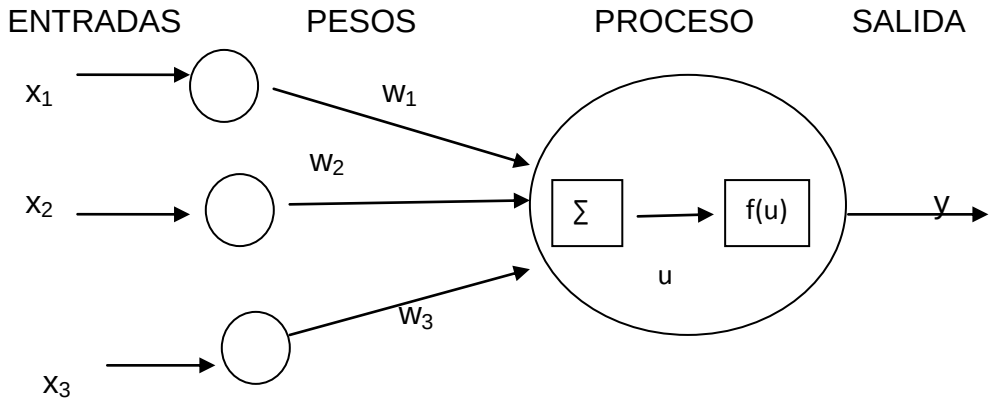


Figura 2.6. Modelo de una neurona artificial (fuente propia)

2.3. DEFINICIONES CONCEPTUALES

2.3.1. APRENDIZAJE

El proceso de aprendizaje consiste en hallar los pesos que codifican los conocimientos. Una regla de aprendizaje hace variar el valor de los pesos de una red hasta que estos adopten un valor constante, cuando esto ocurre se dice que la red ya "ha aprendido" [StatSoft 6]. El proceso de aprendizaje puede dividirse

en 2 grupos de acuerdo a sus características supervisado y no supervisado. [Anderson 7].

2.3.1.1. APRENDIZAJE SUPERVISADO

Parte de un vector de entrada del cual se conoce su salida, el hecho de conocer la salida, el proceso de aprendizaje se realiza mediante un entrenamiento controlado por un agente externo llamado supervisor o maestro. El supervisor comprueba la salida de la red y en el caso de que no coincida con la salida deseada, se procederá a modificar los pesos de las conexiones, con el fin de conseguir que la salida sea la deseada. Después de sucesivas presentaciones la red responde correctamente a todos los patrones de entrada, se puede considerar una red entrenada y dar por terminada la fase de aprendizaje. Ejemplos de este tipo de redes son: el perceptrón simple, la red Adaline, el perceptrón multicapa, red backpropagation, y la memoria asociativa bidireccional.

2.3.1.2. APRENDIZAJE NO SUPERVISADO

En el entrenamiento no supervisado, se desconoce la salida, únicamente se proporciona un vector de entrada para que la red pueda ajustar correctamente los pesos sinápticos. Muchos de los aprendizajes básicos que realizan los sistemas biológicos son de este tipo, por ejemplo los recién nacidos aprenden a organizar sus datos visuales sin ayuda de un maestro que les indique

para cada patrón de estímulos de entrada, cual es la organización-interpretación correcta de dichos estímulos. Ejemplos de este tipo de redes son: las memorias asociativas, las redes de Hopfield, la máquina de Boltzmann y la máquina de Cauchy, las redes de aprendizaje competitivo, las redes de Kohonen o mapas auto organizados y las redes de resonancia adaptativa (ART).

2.3.2. CARACTERISTICAS DE LAS REDES NEURONALES ARTIFICIALES

El cerebro humano, es completamente diferente al funcionamiento de un computador digital. El cerebro humano corresponde al de un sistema altamente complejo, no-lineal y paralelo, es decir que puede realizar muchas operaciones simultáneamente a diferencia de un computador convencional que son de tipo secuencial que realiza una sola operación a la vez. [Haykin 4]

Las RNA tienen 5 características que son citados por Hilera y Martinez [Hilera 8]:

- **Aprendizaje adaptativo.-** Es la capacidad de aprender a realizar tareas basadas en un entrenamiento o experiencias iníciales. Es decir permite adaptarse fácilmente a los nuevos ambientes, contrario a lo que sucede con el enfoque algorítmico el cual exige un nuevo diseño si las características del problema cambian.

- **Auto organización.**- Mientras que el aprendizaje es un proceso donde se modifica la información interna de la RNA, auto organización consiste en la modificación de la red completa con el fin de llevar a cabo un objetivo específico. Autoorganización significa generalización, de esta forma una red es capaz de responder de forma acertada frente a una entrada que nunca fue enseñada. Esta característica es útil sobre todo cuando la información de entrada es poca clara o se encuentra incompleta.
- **Tolerante a fallos.**- La eventual falla de una o varias neuronas no implican un fallo total en la red neuronal, es decir puede seguir funcionando, al igual como ocurre en los sistemas biológicos. Imagínese que sucedería si se quita una o más líneas a un programa en C++.
- **Operación en tiempo real.**- Las RNA son las más indicadas para el reconocimiento en tiempo real, debido a que trabajan en paralelo actualizando todas las instancias simultáneamente.
- **Fácil inserción en la tecnología existente.-** Es relativamente sencillo obtener chips especializados para redes neuronales que mejoren su capacidad en ciertas tareas. Ello facilita la integración modular en los sistemas existentes.

2.3.2.1. TOPOLOGÌA O ARQUITECTURA DE UNA RNA

La forma en que se disponen y conectan especialmente los nodos determina la arquitectura de la red [Barro 9].

Los parámetros fundamentales de la red son: el número de capas, el número de neuronas por capa, y el tipo de conexiones entre neuronas. Así se definen tres tipos básicos de redes:

- Monocapa. Compuesta por una única capa de neurona. Ejemplos: perceptrón, Adaline.

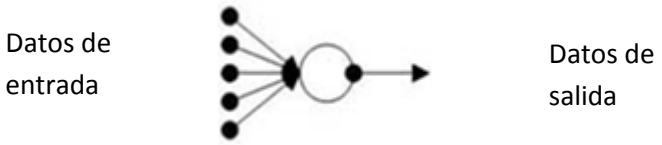


Figura 2.7 redes de una sola capa

Fuente: http://www.redes-neuronales.com.es/tutorial-redes-neuronales/clasificación-de-redes-neuronales-por-topología-arquitectura.htm

- Multicapa. Neuronas que se organizan en varias capas. Ejemplos: perceptrón multicapa.

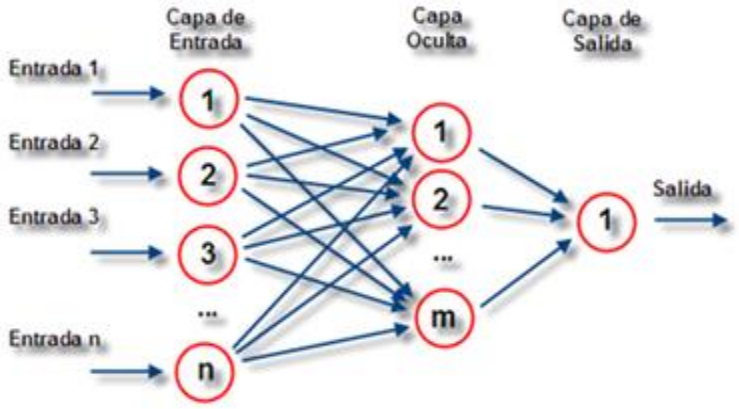


Figura 2.8 Redes de varias capas
Fuente:http://es.wikipedia.org/wiki/Red_neuronal_artificial

- Las redes recurrentes o realimentadas la información puede circular entre las capas en cualquier sentido, incluido el de salida-entrada. Ejemplos: Hopfield, máquina de Boltzmann.

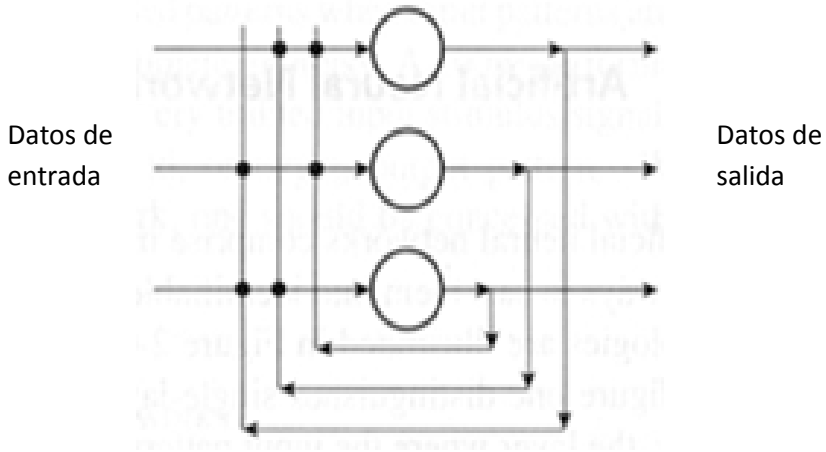


Figura 2.9 Redes recurrentes
Fuente:http://inteligenciaartificialalezzita.blogspot.com/2011/04/sistemas-inteligentes.html

2.3.2.2. TIPOS DE ENTRADA

Las RNA son capaces de procesar información de distinto tipo:

- Redes analógicas: procesan datos de naturaleza analógica, valores reales continuos, habitualmente, acotados en el intervalo [-1,1] o en el [0,1], para dar respuesta también valores reales continuos. Las RNA usa función de activación habitualmente lineal o sigmoidales. Ejemplos de este tipo de redes son: las redes de backpropagation, la red continúa de Hopfield, la red de Kohonen (mapas auto-organizados) y las redes de aprendizaje competitivo.
- Redes discretas (binarias): procesan datos de entrada de naturaleza discreta; habitualmente valores lógicos {0,1}, para dar respuesta también discreta. Entre estas

redes binarias destacan: la máquina de Boltzmann, la máquina de Cauchy, y la red discreta de Hopfield.

- Redes hibridas: estas redes procesan entradas analógicas para dar respuesta binaria. Entre estas redes destacan: el perceptrón, la red adaline y la red madaline.

2.3.2.3. APLICACIONES

Actualmente las RNA se emplean en diferentes campos, estos se agrupan según varios criterios, uno de ellos es propuesto por Deboeck [Deboeck 10], quien los agrupa en:

- Modelación financiera y económica.
- Perfiles de mercado y clientes.
- Aplicaciones médicas.
- Gerencia del conocimiento y “descubrimiento de datos”.
- Optimización de procesos industriales y control de calidad.
- Investigación científica.

2.3.3. FASE DE CREACIÓN Y DESARROLLO

El punto principal en el campo de la red neuronal artificial, es como definir su topología, y como asignar pesos a sus conexiones, de modo que realice la tarea que desea. Esta es la función de los algoritmos de aprendizaje, los cuales a partir de ejemplos cambian la disposición de las neuronas, las conexiones

y sus repectivos pesos, de tal forma que la red aprenda de dichos ejemplos. Comprende las siguientes fases: diseño, entrenamiento y validación. Diseño de la topología, en el diseño tenemos monocapa, multicapa, redes recurrentes. Entrenamiento de la red, tenemos entrenamiento supervizado y no supervizado. Validación de la red, se pedira a la red que responda a estimulos diferentes a los presentados durante la fase de entrenamiento. Gracias a los ejemplos aprendidos, la red debera de ser capaz de generalizar y dar respuestas correctas ante patrones de estimulos nuevos.

2.3.4. ANTECEDENTE HISTÓRICO

- 1943, el psicólogo Warren Sturgis McCulloch y Walter Pitts realizaron el primer modelo matemático de unas RNA. Está basado en la idea de que las neuronas operan mediante impulsos binarios, publicaban el artículo “Un cálculo lógico de la inminente idea de la actividad nerviosa”. [Isasi 2]
- 1949, Donald Hebb, sus estudios sobre las neuronas y las condiciones clásicas de aprendizaje se describen en su libro “La organización del comportamiento”, desarrolló posteriormente un procedimiento matemático de aprendizaje que lleva su nombre “aprendizaje hebbiano”. [Isasi 2]
- 1951 Marvin Minsky y Dean Edmons, fabrican una máquina con tubos, motores y relés, y pudo modelar con éxito el comportamiento de una rata buscando comida en un laberinto. Para ello se basaron en las ideas de McCulloch y Pitts. [Isasi 2]

- 1957 Frank Rosenblatt, “El Perceptrón”. Un sistema que permitía interpretar patrones tanto abstractos como geométricos. El perceptrón supone un gran avance en la IA y el inicio de las RNA.
- 1962 Bernad Widrow y Marcial Hoff, desarrollaron “Adaline y Madaline”. Fue la primera RNA, aplicable a un problema real (filtrados adaptativos para eliminar ecos de la línea telefónica), que se ha utilizado comercialmente durante varios años.
- 1967 Stephen Grossberg, desarrollo la red “Avalancha”. Que consiste en elementos discretos con actividad que varía en el tiempo que satisface ecuaciones diferenciales continuas, para resolver actividades como reconocimiento de habla y aprendizaje de los brazos de un robot.
- 1969 M. Minsky y S. Papert publicó el libro “Perceptrons”. en el que presentan el principal problema del perceptrón, el famoso problema del XOR o el Or exclusivo, la solución de este problema no es linealmente separable y el perceptrón no podía superarlo. Una red de perceptrones compuesta por varias capas podría solucionar el XOR, sin embargo eso no es posible porque no se sabe cuánto tienen que modificarse los pesos de la capa intermedia. Este libro desmoralizó a todos los investigadores y provocó un descenso del interés en las RNA. Además en este artículo se plantea “el problema de la asignación de créditos”.
- En la época posterior al libro de Minsky y Papert el interés generalizado por las RNA había desaparecido, sin embargo

algunos investigadores decidieron continuar con las investigaciones. En este periodo aparecen los modelos de Anderson, Kohonen. Grossberg, Hopfield.

- 1982 Rumelhart, hinton y William, exponen el modelo Back_Propagation que resolvía el problema de la asignación de créditos propuesto por Minsky. Después de este libro se produjo una explosión en las RNA apareciendo modelos, técnicas, campos de aplicación y fusiones híbridas de modelos.

2.3.5. CÉLULAS DE McCulloch-Pitts

Es el primer modelo neuronal propuesto por McCulloch-Pitts. Hace un modelo de una estructura y funcionamiento simplificado de las neuronas del cerebro, considerándolo como un dispositivo con solo dos estados posibles: apagado (0) y encendido (1).

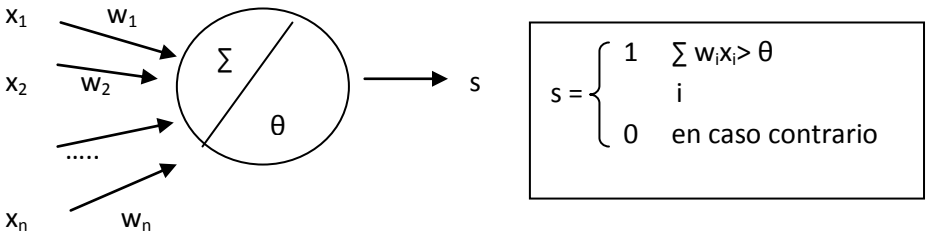


Figura 2.10 Esquema de la célula McCulloch-Pitts (Isasi viñuela-Galvan, 2004, p.23)

La célula de McCulloch-Pitts, recibe como entradas un conjunto de n valores binarios, $X = \{x_1, x_2, \ldots x_n\}$ procedentes de la salida de otras células, o de la entrada de la red y produce una única salida también binaria s.

- La forma de procesar la entrada es la siguiente: la célula se activara si la suma de las entradas multiplicadas por los pesos supera el umbral θ. [Isasi 2]

2.3.6. FUNCIÓN LÓGICA AND

La función and es verdadera si las dos entradas son verdaderas, en los demás casos es falso.

Se va incluir la tabla para las distintas entradas

Tabla 2.1 Función lógica AND

entradas		salidas
x_1	x_2	s
1	1	1
1	0	0
0	1	0
0	0	0

Aplicando las células de McCulloch-Pitts, Para la función lógica **and,** la célula tendrá dos entradas y una salida. El valor del umbral es 1 y de los pesos también. (x_1=x_2=1 y w_1=w_2=1)

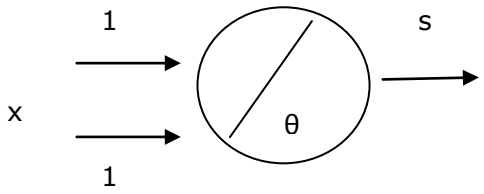


Figura 2.11 Función lógica AND (Isasi viñuela-Galvan, 2004, p.25)

La siguiente tabla muestra la salida de la célula para cada una de las entradas:

Tabla 2.2 Función lógica AND aprendizaje

entradas		proceso	Salidas
x_1	x_2	$\sum w_i x_i$	s
1	1	2	1
1	0	1	0
0	1	1	0
0	0	1	0

- La única vez que la sumatoria supera el umbral de 1 es cuando la sumatoria es 2. Este es el comportamiento de una

función lógica and, se aprecia como ambas tablas coinciden la salida. [Isasi 2]

2.3.7. PERCEPTRÓN UNICAPA

En 1957 Frank Rosenblat desarrollo el Perceptrón. El Perceptrón es un tipo de red de aprendizaje supervisado, es decir necesita conocer los valores esperados para cada una de las entradas presentadas. Se utiliza para clasificación de patrones linealmente separables. Un patrón es linealmente separable cuando trazamos una línea en la gráfica, a un lado de la línea quedan valores con 1 y en el otro lado valores con 0, es decir va a clasificar en dos clases diferentes.

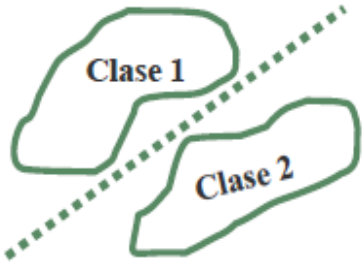


Figura. 2.12 Cada patrón se clasifica en dos clases diferentes (fuente propia)

La topología o arquitectura del perceptrón, es una red monocapa en la que recibe un conjunto de valores como entrada, según el problema, para dar respuesta una o varios valores de salida.

En la figura 2.13, las entradas son x_1, x_2, los pesos son w_1 y w_2, y la salida es y, además existe un parámetro adicional llamado umbral, denotado por θ. El umbral se utiliza como factor de comparación para producir la salida. [Isasi 2]

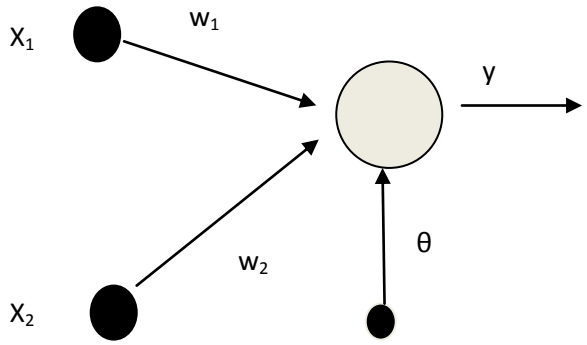


Figura 2.13 Arquitectura de un perceptrón con dos entradas y una salida (Isasi viñuela-Galvan, 2004, p.27)

Una neurona biológica no hace nada, a menos que la influencia colectiva de todas sus entradas alcance un nivel de umbral. Siempre que se alcanza tal umbral, la neurona produce una salida de potencia completa, que consiste en un pulso que se desplaza por el cuerpo de célula, pasa por el axón, hasta las ramas de éste, en este caso se dice que la neurona se dispara. Debido a que una neurona se dispara o no hace nada, se dice que este es un dispositivo de todo o nada.

Para que el perceptrón se dispare o active, la suma de las señales de entrada, multiplicadas por su respectivo peso debe de ser mayor al umbral, es decir: $\sum w_i x_i >$ umbral

Llamamos θ al umbral, y considerando 2 entradas y dos pesos obtenemos:

$$w_1x_1 + w_2x_2 > \theta \qquad \ldots\ldots\ldots (1)$$

Restando θ a ambos miembros de la ecuación 1 obtenemos:

$$w_1x_1 + w_2x_2 + (-\theta) > 0 \qquad \ldots. (2)$$

Esta ecuación equivale a introducir artificialmente un nuevo peso θ, que está conectada a una entrada ficticia con un valor constante de -1. Como función de activación usaremos la función escalón.

$$f(y) = \begin{cases} 1, & y > 0 \\ -1, & \text{en otro caso} \end{cases}$$

La salida es binaria, puede ser fácilmente traducible a una clasificación de 2 clases o categorías de la siguiente forma:

- Si la red produce salida 1, la entrada pertenece a la categoría A
- Si la red produce salida 0, la entrada pertenece a la categoría B. [2].

En el caso de dos dimensiones la ecuación 1 se transforma en:

$$w_1x_1 + w_2x_2 - \theta = 0$$

$$x_1 = -\frac{w_2x_2}{w_1} + \frac{\theta}{w_1}$$ (ecuación en su forma canónica)

Donde la pendiente es: $(-w_2/w_1)$ y la ordenada es (θ/w_1).

Consideremos un perceptrón con varias entradas

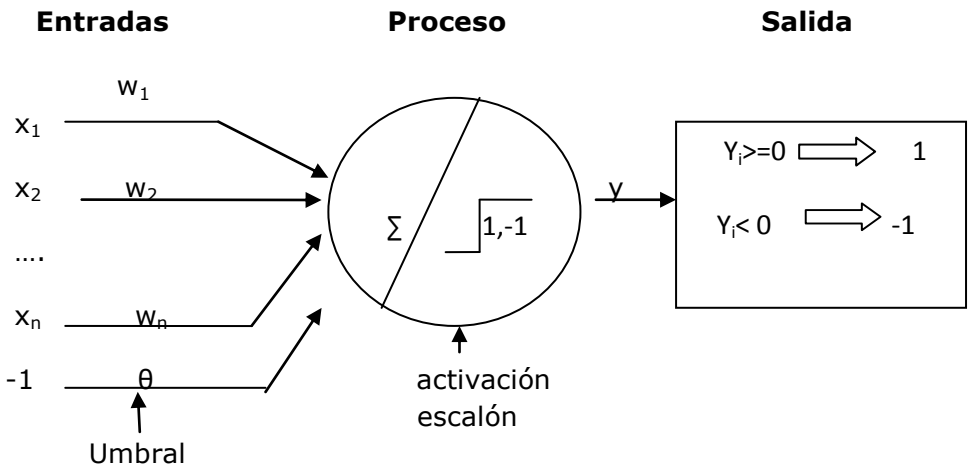


Figura 2.14. Perceptrón simple (fuente propia)

2.3.8. APRENDIZAJE HEBBIANO

El principal inconveniente del neurón umbral es que no aprende.

Donald Hebb (psicólogo) en 1949, propuso un mecanismo de aprendizaje para la neurona biológica, cuya idea básica consiste en que cuando un axón presináptico causa la activación de

cierta neurona postsináptica, la eficacia de la sinapsis que las relaciona se refuerza.

Se denomina aprendizaje hebbiano a aquellas formas de aprendizaje que involucra una modificación en los pesos Δw_{ij} proporcional al producto de una entrada j por la salida i de la neurona.

$$\Delta w_{ij} = \varepsilon\, y_i x_j$$

Siendo ε un parámetro denominado tasa de aprendizaje. El valor de ε varía entre 0 y 1. Un valor pequeño de ε tendremos un aprendizaje lento, requiriendo un período más grande de tiempo para completar el entrenamiento, un valor excesivamente grande puede conducir oscilaciones excesivas de los pesos, no es aconsejable en el proceso de entrenamiento.

Haciendo $t_i = y_i$, La regla de Hebb puede expresarse como:

$$\Delta w_{ij} = \varepsilon\, t_i x_j$$

Y por lo tanto:

$$w_{ij} = w_{ij} + \Delta w_{ij} = w_{ij} + \varepsilon t_i x_j$$

$$w_j = w_j + \varepsilon t_i x_j$$

Peso actual — Peso anterior

ε: factor de aprendizaje
t_i: valor que debe de aprender
x_j: entrada

El método de aprendizaje de Hebb permite que las neuronas aprendan a ajustar sus pesos en un entorno de aprendizaje.

Recordar que los pesos representan la memoria a largo plazo en las RN.

Consideremos las funciones lógicas:

Tabla 2.3 Funciones lógicas

Entradas		and	or	xor	nand
x_1	x_2	Salidas	Salidas	Salidas	salidas
1	1	1	1	0	0
1	0	0	1	1	1
0	1	0	1	1	1
0	0	0	0	0	1

La función lógica **and** es verdadera, si las dos entradas son verdaderas, en los demás casos es falso. La función lógica **or** es verdadera, si por lo menos una entrada es verdadera, en los demás casos es falso. La función lógica **xor** es verdadera, si sus dos entradas son diferentes, en los demás casos es falso. La función lógica **nand** es falsa, si las dos entradas son verdaderas, en los demás casos es verdadero.

Entrenamiento de la función lógica and.

Cambiando el 0 por -1 obtenemos

Tabla 2.4 Función or

entradas		salidas
x_1	x_2	t
1	1	1
1	0	0
0	1	0
0	0	0

Tabla 2.5 Función or

entradas		salidas
x_1	x_2	t
1	1	1
1	-1	-1
-1	1	-1
-1	-1	-1

Consideremos:

Los valores iníciales aleatorios serán asignado para los pesos:

w_1= 0.52593, w_2 = 0.07803, θ = 0.84093, ε=0.5.

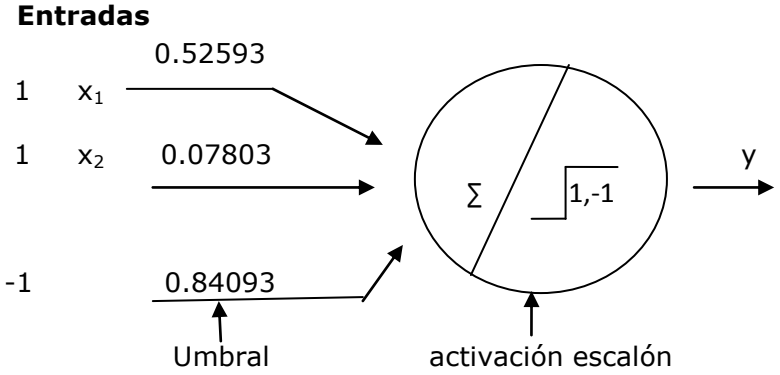


Figura 2.15 Perceptrón simple (fuente propia)

$yi=x_1w_1+ x_2w_2+ \theta i$

Para la entrada (1,1) salida 1

$y1=1*0.08064+ 1*0.18095 + -1*0.84093 = -0.12701 \Longrightarrow -1$

La clasificación es incorrecta (t=1, y=-1), los pesos serán modificados.

Se recalcula los pesos: $w_j= w_j+ \varepsilon t_i x_j$

$w_1 = 0.52593 + (0.5)*(1)*(1) = 1.02593$
$w_2 = 0.07803 + (0.5)*(1)*(1) = 0.57803$
$\theta = 0.84093 + (0.5)*(1)*(-1) = 0.34093$

Perceptrón con nuevos pesos:

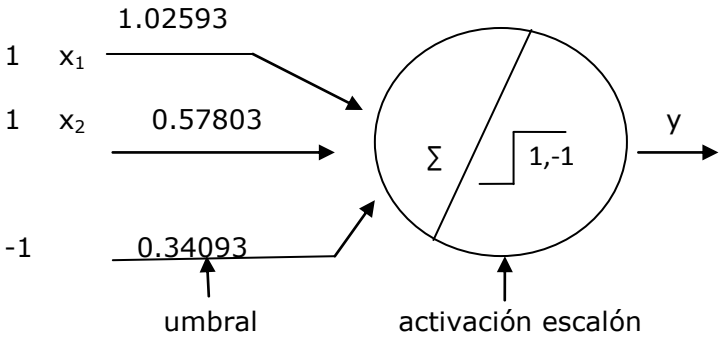


Figura 2.16. Actualización de pesos- Perceptrón simple (fuente propia)

Empezamos nuevamente desde el inicio
Para la entrada (1,1) salida 1

$y1=1*1.02593+ 1*0.57803+ -1*(0.34093)= 1.26303 \Longrightarrow 1$

La clasificación es correcta (t=1, y=1), los pesos no serán modificados.

Para la entrada (1,-1) salida -1

$y2=1*1.02593+ -1*0.57803+ -1*(0.34093)= 0.10697 \Longrightarrow 1$

La clasificación es incorrecta (t=-1, y=1), los pesos serán modificados.

Se recalcula los pesos: $wj= wj+ \varepsilon tixj$

$w1 = 1.02593 + (0.5)*(-1)*(1) = 0.52593$

w2 = 0.57803 + (0.5)*(-1)*(-1) = 1.07803

θ = 0.34093 + (0.5)*(-1)*(-1) = 0.84093

Perceptrón con nuevos pesos:

Entradas

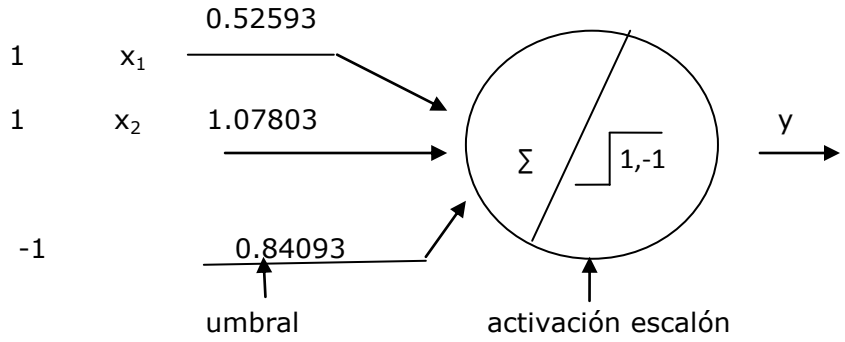


Figura 2.17. Actualización de pesos-Perceptrón simple (fuente propia)

Empezamos nuevamente desde el inicio

Para la entrada (1,1) salida 1

y1=1* 0.52593 + 1*1.07803 + 1*(-1)* 0.84093= 0.76303 ⟹ 1

La clasificación es correcta (t=1, y=1), los pesos no serán modificados.

Para la entrada (1,-1) salida -1

y2=1* 0.52593 + -1*1.07803 + 1*(-1)* 0.84093 = -1.39303 ⟹ -1

La clasificación correcta (t=-1, y=-1), los pesos no serán modificados.

Para la entrada (-1,1) salida -1

y3=-1* 0.52593 + 1*1.07803 + (-1)* 0.84093 = -0.2883 ⟹ -1

La clasificación correcta (t=-1, y=-1), los pesos no serán modificados.

Para la entrada (-1,-1) salida -1

y4=-1* 0.52593+ -1*1.07803+ (-1)* 0.84093 = -2.44483 ⟹ -1

La clasificación es correcta (t=-1, y=-1), los pesos no serán modificados.

Graficamos la ecuación de la recta.-

w_1 = 0.52593, w_2= 1.07803, θ = 0.84093

La ecuación:

$y = w_1x_1 + w_2x_2 - \theta$

Hacemos y=0, entonces

$w_1x_1 + w_2x_2 - \theta = 0 \Rightarrow x_1 = -\frac{w_2x_2}{w_1} + \frac{\theta}{w_1}$

Usando interceptos:

$x_1=0 \Rightarrow x_2 = \frac{\theta}{w_2} \Rightarrow x_2 = \frac{0.84093}{1.07803} = 0.7800 \Rightarrow (0,0.7800)$

$x_2=0 \Rightarrow x_1 = \frac{\theta}{w_1} \Rightarrow x_2 = \frac{0.84093}{0.52593} = 1.2472 \Rightarrow (1.2472,0)$

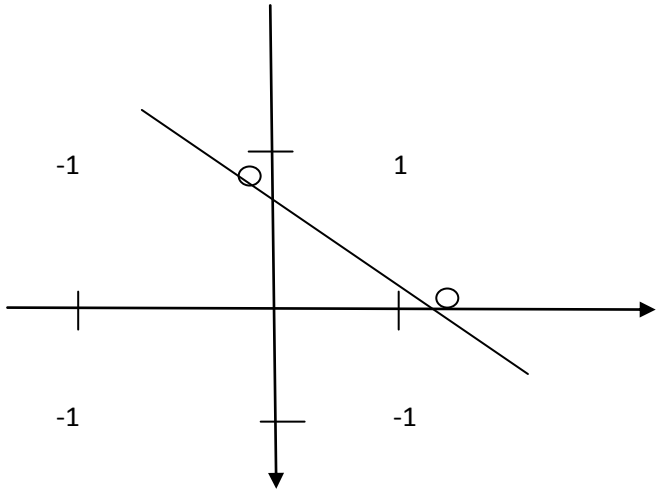


Figura 2.18. Recta determinada por el perceptrón, función lógica AND

Hemos trazado una línea recta, observamos que en un lado de la línea, están los que clasifican con 1 y en el otro lado de la línea los que clasifican con -1.

En el punto (1,1) colocamos su salida que es 1.

En el punto (1,-1) colocamos su salida que es -1.

En el punto (-1,1) colocamos su salida que es -1.

En el punto (-1,-1) colocamos su salida que es -1.

Algoritmo del perceptrón simple

Inicio

Inicializa matriz de entrada

Inicializa vector salidas

Inicializa vector pesos

i=0

```
Mientras (i<longitud de entradas) Hacer
    yi = pesos[0]*entradas[i][0]+pesos[1]*entradas[i][1]+pesos[2]*entradas[i][2]
    Si (yi >0 ) entonces
        Escribir(entrada[i][0],entradas[i][1],salida[i], correcto, yi)
    Sino
        Escribir(entrada[i][0],entradas[i][1],salida[i], incorrecto,yi)
        // corrección de pesos
        Desde k=0 hasta (k<longitud pesos) con incremento 1 Hacer
            pesos[k]=recalculaPesos(pesos[k],salidas[i],entradas[i][k])
        Fin_desde
        i = -1 // para que empiece de nuevo
    Fin_si
    i  =i+1
  Fin_mientras
Fin
Función   recalculaPesos(pesosj,  ti,  xj)
    Real wj
    wj = pesosj + 0.5*ti*xj
    retornar  wj
Fin_funcion
```

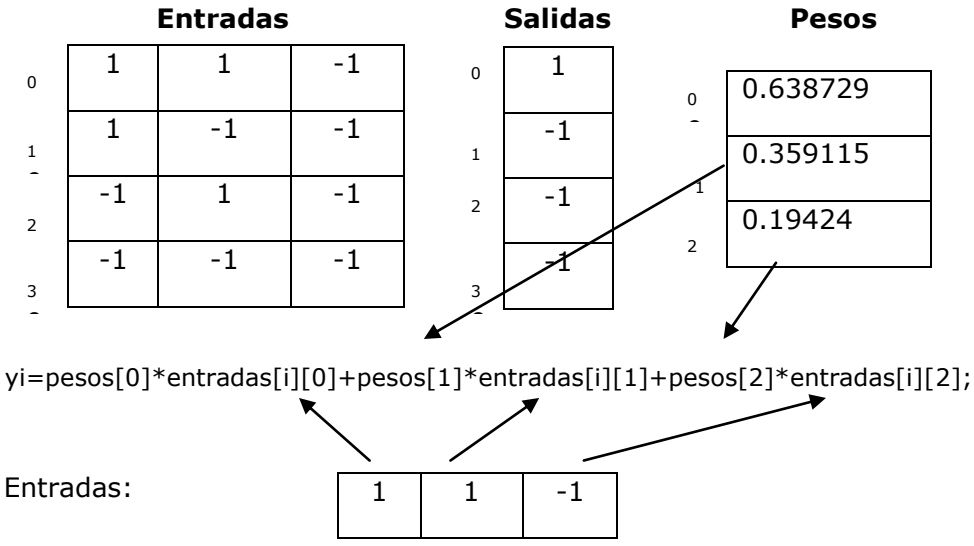


Figura 2.19. Representación gráfica del algoritmo

2.3.9. PERCEPTRON ADALINE

En 1960, Bernard Widrow y Hoff, diseñaron el modelo de la red Adaline (ADAptive LInear NEurón), donde proponen un sistema de aprendizaje que tuviera en cuenta el error producido. La red Adaline presenta la misma limitación del Perceptrón, ambas redes pueden solo resolver problemas linealmente separables, sin embargo el algoritmo ADALINE es más potente que la regla de aprendizaje del Perceptrón ya que minimiza el error medio cuadrático.

$$E = \sum_{i=1}^{m} E^P = \frac{1}{2} \sum_{i=1}^{m} (d^p - y^p)^2$$ (d=salida esperada, y=salida de la red)

Se intentara minimizar este error, mediante la regla del descenso del gradiente. Se realiza un cambio de peso proporcional a la derivada del error, medida en el patrón actual, respecto al peso:

$$\Delta_p w_j = - \gamma \frac{\partial E^P}{w_j}$$

Para el cálculo de la derivada anterior se utiliza la regla de la cadena:

$$\frac{\partial A}{\partial x} = \frac{\partial A}{\partial y} \frac{\partial y}{\partial x}$$

Aplicando la regla de la cadena a la expresión anterior:

$$\frac{\partial E^P}{\partial w_j} = \frac{\partial E^P}{\partial y^P} \frac{\partial y^P}{\partial w_j}$$

Al ser unidades lineales, sin función de activación en la capa de salida, se cumple:

$$\frac{\partial y^P}{\partial w_j} = xj \qquad \frac{\partial E^P}{\partial y^P} = - (d^p - y^p)$$

Sustituyendo queda como sigue:

$\Delta p\ wj = - \gamma\ (dp - yp)\ xj$ [Isasi 2]

Ejemplo decodificador de decimal a binario

El decodificador recibe una entrada en binario y produce como salida su valor en decimal. Como se trata de pasar de binario a decimal se ha elegido se ha elegido una de dimensión 3 como entrada, entonces el número máximo de entradas que se puede generar es de 23=8. En la siguiente tabla se muestra el conjunto de datos de entrenamiento:

Tabla 2.6 Decodificador de decimal a binario

entrada			salida
X1	X2	X3	
0	0	1	1
0	1	0	2
0	1	1	3
1	0	0	4
1	0	1	5
1	1	0	6
1	1	1	7

Consideremos:

Los valores aleatorios serán asignados a los pesos:

w1= 0.840, w2=0.394, w3=0.783 ɣ=0.3

yi = x1w1+ x2w2+ x3w3

Para la entrada (0, 0,1) salida 1

yi = 0*0.840 + 0*0.394 + 1*0.783 = 0.783

Como la salida no es idéntica a la deseada los pesos serán modificados. Se calcula el error: e = d - y= 1 - 0.783 = 0.217

Se recalcula los pesos: wj = wj + ɣ*(d - y)* xj

w1= 0.840 + 0.3* 0.217*0= 0.840

w2= 0.394 + 0.3* 0.217*0= 0.394

w3= 0.783 + 0.3* 0.217*1= 0.848

El proceso se repite para todos los patrones de entrada.

Para la entrada (0, 1,0) salida 2

yi = 0*0.840 + 1*0.394 + 0*0.848 = 0.394

Como la salida no es idéntica a la deseada los pesos serán modificados. Se calcula el error: e = d - y= 2 - 0.394 = 1.61

Se recalcula los pesos: wj = wj + γ*(d - y)* xj

w1= 0.840 + 0.3* 1.61*0= 0.840

w2= 0.394 + 0.3* 1.61*1= 0.876

w3= 0.848 + 0.3* 1.61*0= 0.848

Y así sucesivamente, para la entrada (1, 1, 1) salida 7

yi = 1*3.090 + 1*1.966 + 1*1.828 = 6.881

Como la salida no es idéntica a la deseada los pesos serán modificados. Se calcula el error: e = d - y= 7 - 6.881 = 0.12

Se recalcula los pesos: wj = wj + γ*(d - y)* xj

w1= 3.090 + 0.3* 0.12*1= 3.126

w2= 1.966 + 0.3* 0.12*1= 2.002

w3= 1.828 + 0.3* 0.12*1= 1.861

Se puede apreciar que después de una iteración la red produce una salida muy adecuada para este último patrón, será necesario introducir todos los patrones de nuevo y repetir el proceso un número indefinido de veces si se desea reducir el error.

Tabla 2.7 Decodificador a binario para 10 iteraciones

Iteración	Pesos	Error de patrón
1	[3.126, 2.003, 1,86]	[0.21, 1.65, 1.27, 3.16, 1.98, 2.35, 0.12]
2	[3.61, 1.98, 1,42]	[-0.86, -0.03, -0.60, 0.87, 0.19, 0.73, -0.18]
3	[3.82, 1.98, 1,20]	[0.42, 0.01, -0.28, 0.39, 0.06, 0.35, -0.01]
4	[3.92, 1.98, 1,10]	[-0.20, 0.01, -0.13, 0.17, 0.02, 0.17, -0.04]
5	[3.96, 1.99, 1,05]	[-0.09, 0.01, -0.06, 0.08, 0.00, 0.08, -0.02]
6	[3.98, 1.99, 1,02]	[-0.05, 0.01, -0.03, 0.03, 0.00, 0.04, -0.01]
7	[3.99, 2.00, 1,01]	[-0.02, 0.00, -0.01, 0.02, 0.00, 0.02, 0.00]
8	[4.00, 2.00, 1,00]	[-0.01, 0.00, 0.00, 0.01, 0.00, 0.01, 0.00]
9	[4.00, 2.00, 1,00]	[-0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00]
10	[4.00, 2.00, 1,00]	[-0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00]

Se observa cómo los errores cometidos van reduciéndose de iteración a iteración, hasta alcanzar un valor cero.

```
Algoritmo adaline
Inicio
   Inicializa matriz de entrada
   Inicializa vector salidas
   Inicializa vector pesos
   i=0
   Mientras (i<longitud de entradas) Hacer
      yi = pesos[0]*entradas[i][0]+pesos[1]*entradas[i][1]+pesos[2]*entradas[i][2]
      Si (yi =salida[i]) entonces
               // no hace nada
      Sino
               err=calculaError(,salida[i], yi)
              // corrección de pesos
               Desde k=0 hasta (k<longitud pesos) con incremento 1 Hacer
                  pesos[k]=recalculaPesos(pesos[k], err, entradas[i][k])
               Fin_desde
      Fin_si
      i  =i+1
   Fin_mientras
Fin
Función  calculaError(double ti,double yi){
      double err
      err = (ti-yi)
      retornar err
Fin_funcion
Función recalcularPesos(double pesosj,double err,double xj){
      double wj;
      wj = pesosj + 0.3*err*xj
      retornar wj
Fin-funcion
```

2.3.10. PERCEPTRON MULTICAPA

El perceptrón Frank Rosenblat está limitado a una clasificación de patrones separables linealmente. Un patrón es linealmente separable si existe un plano capaz de separar todos los puntos

que clasifica como 1 y los que clasifica como -1. El algoritmo de Rosenblat está basado en una red neuronal de una sola capa con pesos ajustables que limita la potencia de cálculo del algoritmo, para salvar estas limitaciones se usa el "Perceptrón multicapa". [Haykin 4]

Los siguientes tres puntos resaltan las características básicas del perceptrón multicapa:

- El modelo de cada neurón en la red incluye una función de activación no lineal que es diferenciable.
- La red contiene una o dos capas que están ocultas de los nodos de entrada y salida.
- La red exhibe un alto grado de conectividad, la extensión de la cual es determinado por los pesos sinápticos de la red.

  El método popular para el entrenamiento del perceptrón multicapa es el algoritmo de propagación hacia atrás. Que resuelve el problema del xor, nand, or exclusivo.

El entrenamiento procede en dos fases:

- **En la fase hacia adelante**, los pesos sinápticos de la red son fijos y la señal de entrada es propagada a través de toda la red, capa por capa hasta que alcance la salida.
- **En la fase hacia atrás**, una señal de error es producido al comparar la salida de la red con una respuesta deseada. La señal error resultante es propagada a través de toda la red, capa por capa, pero esta vez la propagación es ejecutada en la dirección hacia atrás.

El back-propagation proporciona un método computacionalmente eficiente para el entrenamiento de un PMC.

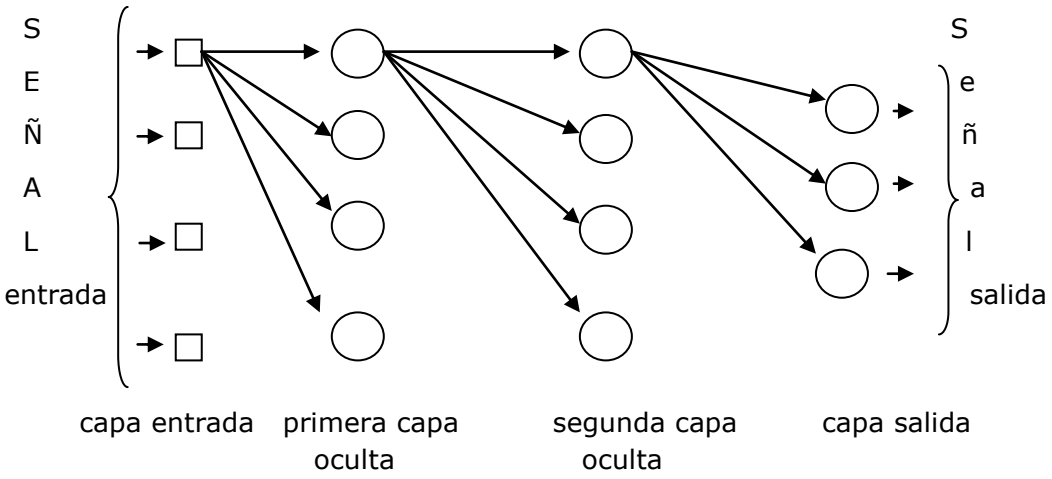


Figura 2.20. Grafico de un PMC con dos capas ocultas y una capa de salida (Fuente propia)

La señal que fluye a través de toda la red avanza en una dirección hacia adelante, de izquierda a derecha y sobre la base de capa por capa.

Los neurones de salida constituyen la capa de salida de la red. Los neurones restantes constituyen las capas ocultas de la red. Así, las unidades ocultas no son parte de la salida o de la entrada de la red, de allí su designación como “oculta”.

La primera capa oculta es alimentada por la capa de entrada, la salida resultante de la primera capa oculta es aplicada a la próxima capa oculta y así sucesivamente para el resto de la red.

Función de las neuronas ocultas.-

Las neuronas ocultas actúan como detectores de características, como tal ellos juegan un rol crítico en la operación de un PMC.

Cuando un proceso de aprendizaje progresa a través del PMC, las neuronas ocultas comienzan gradualmente “descubrir” las características salientes que caracterizan los datos de entrenamiento.

Ellos hacen así ejecutar una transformación no lineal sobre los datos de entrada en un nuevo espacio llamado “el espacio de características”, en este nuevo espacio, las clases de interés en una tarea de clasificación de patrones pueden ser más fácilmente separadas de cada uno del otro, que puede ser el caso en el espacio de datos de entrada original.

En realidad, esto es la formación de este espacio de características a través del aprendizaje supervisado que se distingue el PMC del perceptrón de Rosenblatt.

Función sigmoidea.-

En un principio se pensó que las neuronas usaban una función de umbral, es decir, que permanecían inactivas y se activaban sólo si la estimulación total superaba cierto valor límite. Después se comprobó que las neuronas emitían impulsos de actividad eléctrica con una frecuencia variable, dependiendo de la intensidad de la estimulación recibida, y que tenían cierta actividad hasta en reposo, con estimulación nula. Estos descubrimientos llevaron al uso de funciones no lineales con esas características, como la función sigmoidal, con un perfil parecido al escalón de una función de umbral, pero continúa

Ecuación de la función sigmoidal

$$\frac{1}{1 + e^{-a}}$$

Cuando la entrada es cero la función se aproxima a 0.5, cuando la entrada es negativo la función se aproxima a cero, cuando la entrada es mayor a 0.5 la función se aproxima a 1.

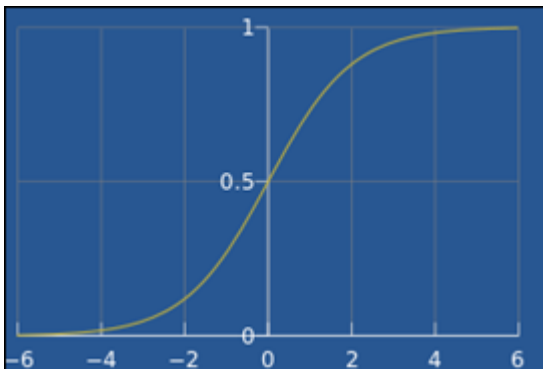


Figura 2.21. La función sigmoidea

La ecuación sigmoidea se utiliza como una función de transferencia entre las neuronas. La ecuación sigmoidea es continua y diferenciable.

$$\sigma(x) = \frac{1}{1+e^{-x}} \qquad \ldots\ldots\ldots\ldots (1)$$

Derivando la ecuación sigmoidea:

$$\frac{d\,\sigma(x)}{dx} = \frac{d}{dx}\left[\frac{1}{1+e^{-x}}\right] = \frac{e^{-x}}{(1+e^{-x})^2}$$

$$\frac{d\,\sigma(x)}{dx} = \frac{e^{-x}}{(1+e^{-x})^2} = \frac{(1 + e^{-x}) - 1}{(1+e^{-x})^2} \text{ (sumando y restando 1)}$$

$$\frac{d\,\sigma(x)}{dx} = \frac{(1 + e^{-x})}{(1+e^{-x})^2} - \frac{1}{(1+e^{-x})^2} = \frac{1}{(1+e^{-x})} - \frac{1}{(1+e^{-x})^2}$$

$$\frac{d\,\sigma(x)}{dx} = \sigma(x) - \left[\frac{1}{1+e^{-x}}\right]^2$$

$$\frac{d\,\sigma(x)}{dx} = \sigma(x) - \sigma(x)^2 = \sigma(x)\,(1 - \sigma(x))$$

$$\sigma' = \sigma\,(1 - \sigma)$$

Demostración de las fórmulas:

Llamaremos:

$\vec{x}$: vector de entrada cuyos “n” elementos denominaremos x_i

$\vec{o}$: vector de salida cuyos “m” elementos denominaremos o_i

$\vec{w}$: vector de pesos cuyos “n*m” elementos denominaremos w_{ij}

$\vec{z}$: vector intermedio previo a $\vec{o}$, cuyos “m” elementos denominaremos $zj = \sum xi\ wij = x\ wj$

El objetivo de la iteración es, dado $\vec{x}$ y $\vec{t}$, obtener un valor Δw, que sumado $\vec{w}$ nos permita disponer de un mejor conjunto de pesos que acerque más los valores de $\vec{o}$ a los de $\vec{t}$. Para ellos definiremos una función de error, que será un número escalar no negativo cuyo objetivo será minimizarlo mediante el ajuste de los pesos.

Una buena medida del error viene dada por la distancia euclidiana de ambos vectores, es decir $\vec{o}$ y $\vec{t}$, debido a que no nos interesa la magnitud del error, agreguemos una constante multiplicativa de 1/2 al principio. Obtenemos ası nuestra función del error como

$$E = \frac{1}{2}\left|o - t\right| = \frac{1}{2}\sum (oj - tj)^2$$

Con el error definido, aplicaremos el operador gradiente para obtener su dirección de máximo crecimiento (siempre derivando con respecto a los wi,j que son nuestras variables independientes. La multiplicaremos por −1 para obtener la dirección de máximo decrecimiento y luego la multiplicaremos por un coeficiente que indicará la velocidad en que el algoritmo avanzara.

Defineremos el gradiente de E con respecto a los pesos, es decir: $\nabla E = \partial E / \partial wij$

Por la regla de la cadena:

Calculemos primero el error de zk

$$\nabla Eu = \frac{\partial E}{\partial zk}\frac{\partial zk}{\partial wij}$$ (siendo u la posición de cada z de cada par i,j)

$$\frac{\partial E}{\partial zk} = \frac{1}{2}\frac{\partial \sum_{k=1}^{n} (on - tn)^2}{\partial zk} = \frac{1}{2}\frac{\partial (ok-tk)^2}{\partial zk}$$

Considerando:

ok= σ(zk)
σ'(zk) = σ(zk) (1 – σ(zk) (derivación de la función sigmoidal)

Entonces:

$$\frac{\partial E}{\partial zk} = \frac{1}{2}\frac{\partial (\sigma(zk) - tk)^2}{\partial zk}$$ (reemplazamos ok= σ(zk)

Derivando obtenemos:

$= \frac{1*2}{}\partial (\sigma(zk))*(\sigma(zk) - tk)$ (reemplazamos la derivación sigmoidal)

2

$$\frac{\partial E}{\partial zk} = \sigma(zk)\,(1 - \sigma(zk)\,*(\,\sigma(zk) - tk)\;(\text{ reemplazamos } ok = \sigma(zk)\,)$$

$$\frac{\partial E}{\partial zk} = ok\,(1 - ok)\,*(\,ok - tk)\;(\text{ reemplazamos } \sigma(zk) = ok\,)$$

Lo multiplicamos por -1 para obtener la dirección de maximo creciente

$$\frac{\partial E}{\partial zk} = ok\,(1 - ok)\,*(\,tk - ok)$$

Definiremos :

$$\delta k = \frac{\partial E}{\partial zk} = (\,tk - ok)\,*\,ok\,(1 - ok)$$

Ya que solo depende de k, con lo cual, al iterar este cálculo puede realizarse una sola vez.

Ahora veremos la segunda parte que es: $\frac{\partial zk}{\partial wij}$

$$\frac{\partial zk}{\partial wij} = \frac{\partial \sum xk\,wkj}{\partial wij} = xi$$

Finalmente obtenemos:

$$\frac{\partial E}{\partial wij} = \delta k\,xi = (\,tk - ok)\,*\,ok*(1 - ok)*xi$$

Capa de módulos ocultos

$$\frac{\partial E}{\partial wij} = \frac{\partial}{\partial wij}\,\frac{1}{2}\sum_{k\,\varepsilon\,K}(tk - ok)^2$$

$$\frac{\partial E}{\partial wjk} = \sum_{k\,\varepsilon\,K}(tk - ok)\,\frac{\partial ok}{\partial wij}$$

$$\frac{\partial E}{\partial wjk} = \sum_{k\,\varepsilon\,K}(tk - ok)\,\frac{\partial \sigma(xk)}{\partial wij}$$

$$\frac{\partial E}{\partial wjk} = \sum_{k \varepsilon K} (tk - ok)\, \sigma(xk)(1 - \sigma(xk))\, \frac{\partial xk}{\partial wij}$$

Por regla de la cadena

$$\frac{\partial E}{\partial wjk} = \sum_{k \varepsilon K} (tk - ok)\, ok\, (1 - ok)\, \frac{\partial xk}{\partial oj}\, \frac{\partial oj}{\partial wij}$$

$$\frac{\partial E}{\partial wjk} = \sum_{k \varepsilon K} (tk - ok)\, ok\, (1 - ok)\, wjk\, \frac{\partial oj}{\partial wij}$$

$$\frac{\partial E}{\partial wjk} = \frac{\partial oj}{\partial wij} \sum_{k \varepsilon K} (tk - ok)\, ok\, (1 - ok)\, wjk$$

$$\frac{\partial E}{\partial wjk} = oj(1-oj)\, \frac{\partial oj}{\partial wij} \sum_{k \varepsilon K} (tk - ok)\, ok\, (1 - ok)\, wjk$$

$$\frac{\partial E}{\partial wjk} = oj(1-oj)\, oi \sum_{k \varepsilon K} (tk - ok)\, ok\, (1 - ok)\, wjk$$

Pero recordemos nuestra definición de δk, podemos escribir esto como:

$$\frac{\partial E}{\partial wij} = oi\, oj\, (1 - oj) \sum \delta k\, wjk$$

Igual que antes, definimos todos los términos, además de oi para ser δj, así tendremos:

$$\frac{\partial E}{\partial wij} = oi\, \delta j$$

Como el peso afecta los errores, para una producción de la capa modulo k ε K

$$\frac{\partial E}{\partial wjk} = oj\, \delta k$$

Donde

$$\delta k = ok(1-ok)\, (tk-ok)$$

Para una capa de modulo oculto:

$$\frac{\partial E}{\partial wij} = oi\, \delta j$$

$$\delta j = oj(1-oj) \sum_{k \varepsilon K} \delta k \, wjk$$

Entrenamiento de red neuronal backpropagation

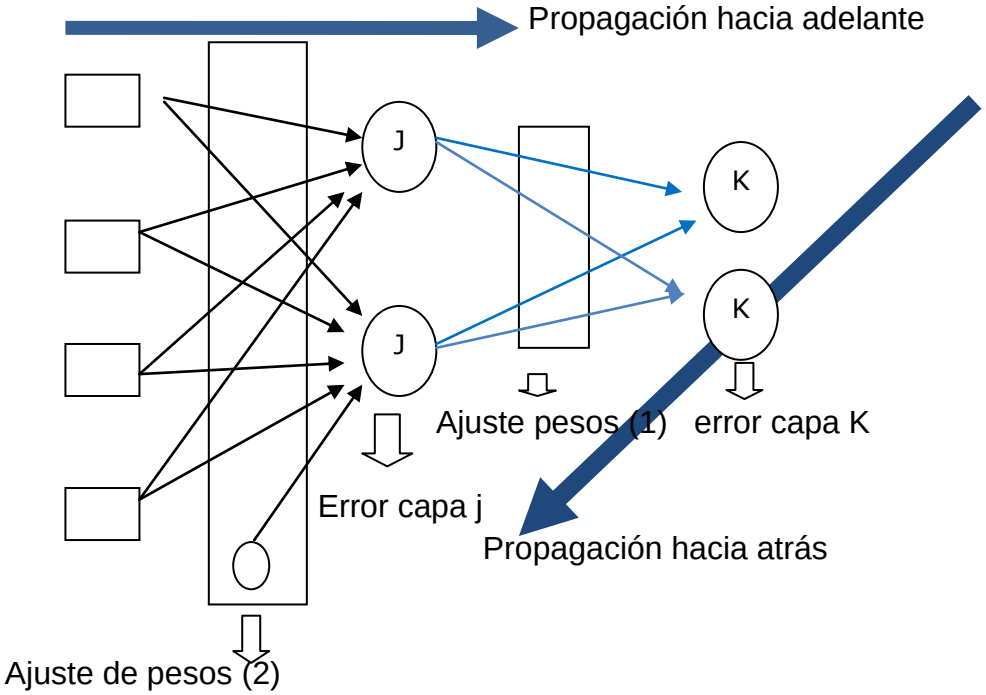


Figura 2.22. Entrenamiento del Backpropagation

Cómo funciona el BackPropagation

1.- Empezar con pesos sinápticos generalmente elegidos al azar.

2.- Introducir los datos de entrada para el entrenamiento

3.- Calcular la salida de la red (propagación hacia adelante)

4.- Comparar la salida generada por la red con la salida deseada

5.- Si no coincide se calcula el error de la salida generada, se usa para ajustar los pesos sinápticos de las neuronas de la capa de salidas.

6.- El error se *propaga hacia atrás* (back-propagation), hacia la capa de neuronas anterior, y se usa para ajustar los pesos sinápticos en esta capa.

7.- Se continua propagando el error hacia atrás y ajustando los pesos hasta que se alcance la capa de entradas.

8.- Este proceso se repetirá con los diferentes datos de entrenamiento, hasta que la red responda correctamente a todos los datos de entrada, se puede considerar una red entrenada y dar por terminado la fase de aprendizaje.

Según la figura 2.22, fórmulas para las diferentes capas:

ERROR en la capa K: Ek = (tk-ok) ok(1-ok)

Donde:
tk: salida correcta
ok: salida actual

Ajustes de pesos (1): wjk=wjk + L*Ek*oj

Donde:
wjk: peso
L: factor de aprendizaje
Oj: entrada al nodo K desde J

Error en la capa oculta (capa j): Ej = oj*(1-oj)*∑Ek*wjk

Ajuste de pesos en la capa de entrada: wij=wij + L*Ej*oi

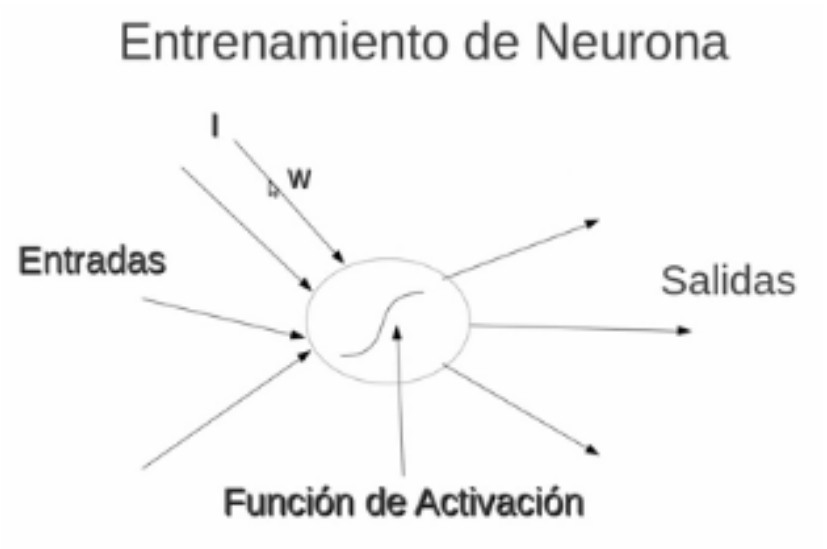


Figura 2.23. función de activación

Aprendizaje de la función xor utilizando backpropagation

Se va incluir la tabla para las distintas entradas

Tabla 2.8. Función lógica XOR

entradas		salidas
x_1	x_2	s
1	1	0
1	0	1
0	1	1
0	0	0

La función xor es verdadera cuando las dos entradas son diferentes, en los demás casos es falso.

Entrenamiento de la red xor

En la primera capa tiene un nodo, con tres entradas incluido el bias, la segunda capa tiene dos nodos, donde cada nodo tiene tres entradas.

Las capas que va a utilizar el XOR

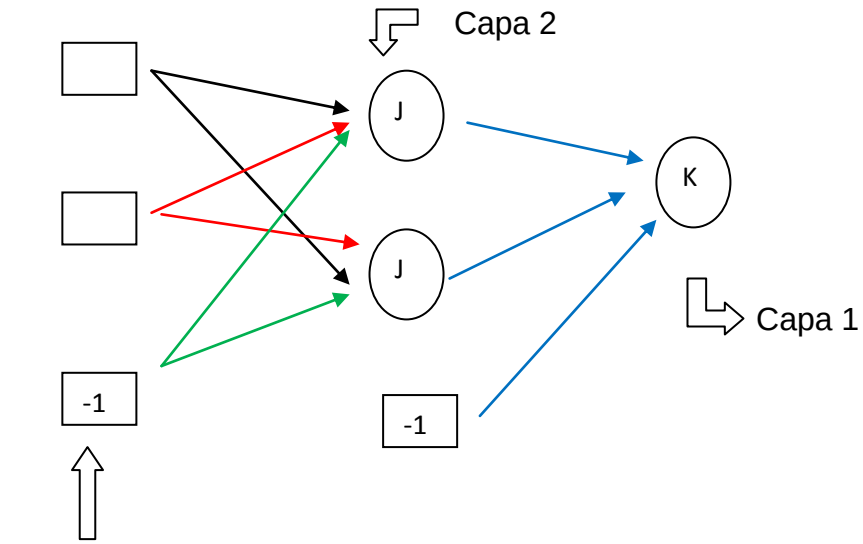


Figura 2.24. Capas que utiliza el xor

Estructura de la red neuronal

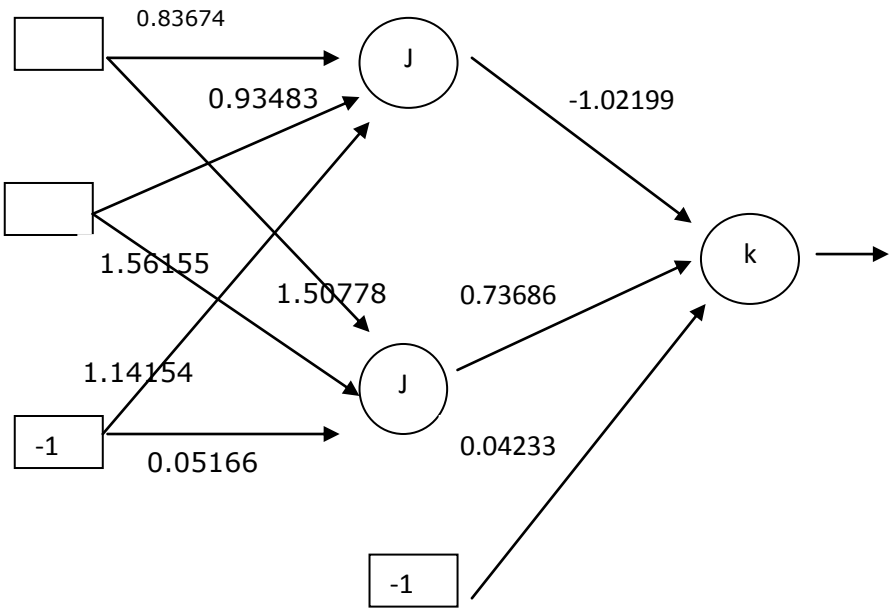


Figura 2.25. Inicialización del BP para el ejemplo del XOR

Pesos: neuronJ[0] y neuronJ[1]

neuronJ[0]	neuronJ[1]
neuronJ[0].pesos[0]=0.83674 neuronJ[0].pesos[1]=0.93483 neuronJ[0].pesos[2]=1.14154	neuronJ[1].pesos[0]=1.50778 neuronJ[1].pesos[1]=1.56155 neuronJ[1].pesos[2]=0.05166

Pesos: neuronK[0]
neuronK[0].pesos[0]=-1.02199
neuronK[0].pesos[1]=0.73686
neuronK[0].pesos[2]=0.04233

Tiene las mismas entradas: neuronJ[0] y neuronJ[1], para 1,1,-1

neuronJ[0]	neuronJ[1]
neuronJ[0].entradas[0]=1 neuronJ[0]. entradas[1]=1 neuronJ[0]. entradas [2]=-1	neuronJ[1]. entradas[0]=1 neuronJ[1]. entradas[1]=1 neuronJ[1]. entradas[2]=-1

La primera entrada (1,1,-1),época 0, i=0, salida =0
Paso hacia adelante: llamaremos a este módulo activacionJ()

∑ entradas[i]*pesos[i]

Para el neuronJ[0] (la entrada es 1,1,-1)

entradas[0]*pesos[0] + entradas[1]*pesos[1] + entradas[2]*pesos[2]
suma = 1*0.83674+ 1*0.93483- 1*1.14154= 0.63003

Aplicando la sigmoide
1/(1+ e-suma) = 1/ (1+e-0.63003) = 0.65249

Para el neuronJ[1] (la entrada es la misma es 1,1,-1)

entradas[0]*pesos[0] + entradas[1]*pesos[1] + entradas[2]*pesos[2]
suma = 1*1.50778+ 1*1.56155- 1*0.05166= 3.01767

Aplicando la sigmoide
1/(1+ e-suma) = 1/ (1+e-3.01767) = 0.95336

Módulo entradasK
toma el valor de activaciónJ y la última toma el valor del bias

neuronK[0].entradas[0]= 0.65249
neuronK[0].entradas[1]= 0.95336
neuronK[0].entradas [2]=-1

Módulo ActivacionK: ∑ entradas[i]*pesos[i]

0.65249*-1.02199+ 0.95336*0.73686 -1*0.04233 = -0.00664

Aplicando la sigmoide
1/(1+ e^{-suma}) = 1/ (1+$e^{+0.00664}$) = **0.49834**

Para hallar la salida de la red, vamos a ver qué valores nos da la función sigmoide.
Como 0.49834 es menor que 0.5 la salida es cero y lo que debe de aprender es 0, cumple la condición 0=0, **no hay retroceso, empezamos nuevamente hacia adelante**

La segunda entrada (1, 0, -1), época 1, i=1, aprender =1
Se repite nuevamente el proceso, pero la entrada es diferente:

Paso hacia adelante: llamaremos a este módulo activacionJ()

∑ entradas[i]*pesos[i]

Para el neuronJ[0] (la entrada es 1,0,-1)

entradas[0]*pesos[0] + entradas[1]*pesos[1] + entradas[2]*pesos[2]
suma = 1*0.83674 + 0*0.93483 - 1*1.14154 = -0.3048

Aplicando la sigmoide
$1/(1+ e^{-suma}) = 1/ (1+e^{+0.3048}) =$ **0.42438**

Para el neuronJ[1] (la entrada es la misma es 1,0,-1)

entradas[0]*pesos[0] + entradas[1]*pesos[1] + entradas[2]*pesos[2]
suma = 1*1.50778+ 0*1.56155 - 1*0.05166= 1.45612

Aplicando la sigmoide
$1/(1+ e^{-suma}) = 1/ (1+e^{-1.45612}) =$ **0.81094**

Módulo entradasK toma el valor de activaciónJ y la última toma el valor del bias

neuronK[0].entradas[0]= 0.42438
neuronK[0].entradas[1]= 0.81094
neuronK[0].entradas [2]=-1

Módulo activacionK: ∑ entradas[i]*pesos[i]

0.42438*(-1.02199) + 0.81094*0.73686 -1*0.04233 = -0.12151

Aplicando la sigmoide

$1/(1+ e^{-suma}) = 1/ (1+e^{-0.12151}) =$ **0.53034**

Para hallar la salida de la red, vamos a ver qué valores nos da la función sigmoide.

Como 0.53034 es mayor que 0.5 la salida es uno y lo que debe de aprender es 1, cumple la condición 1=1, **no hay retroceso, empezamos nuevamente hacia adelante**

La tercera entrada (0, 1, -1), época 2, i=2, la salida es 1

Se repite nuevamente el proceso, pero la entrada es diferente:

Paso hacia adelante: llamaremos a este módulo activacionJ()
∑ entradas[i]*pesos[i]

Para el neuronJ[0] (la entrada es 0, 1,-1)

entradas[0]*pesos[0] + entradas[1]*pesos[1] + entradas[2]*pesos[2]

suma = 0*0.83674 + 1*0.93483 - 1*1.14154 = -0.20671

Aplicando la sigmoide

$1/(1+ e^{-suma}) = 1/ (1+e^{+0.20671}) =$ **0.44850**

Para el neuronJ[1] (la entrada es la misma es 1,0,-1)

entradas[0]*pesos[0] + entradas[1]*pesos[1] + entradas[2]*pesos[2]

suma = 0*1.50778+ 1*1.56155 - 1*0.05166 = 1.50989

Aplicando la sigmoide

$1/(1+ e^{-suma}) = 1/ (1+e^{-1.50989}) =$ **0.81904**

Módulo entradasK

toma el valor de activaciónJ y la última toma el valor del bias

neuronK[0].entradas[0]= **0.44850**

neuronK[0].entradas[1]= **0.81904**

neuronK[0].entradas [2]=-1

Módulo ActivacionK: ∑ entradas[i]*pesos[i]

0.44850*(-1.02199) + 0.81904*0.73686 -1*0.04233 = 0.10282

Aplicando la sigmoide

$1/(1+ e^{-suma}) = 1/ (1+e^{-0.10282)} =$ **0.525682**

Para hallar la salida de la red, vamos a ver qué valores nos da la función sigmoide.

Como 0.525682 es mayor que 0.5 la salida es uno y lo que debe de aprender es 1, cumple la condición 1=1, **no hay retroceso, empezamos nuevamente hacia adelante**

La cuarta entrada (0, 0, -1), época 4, i=3, la salida es 0

Se repite nuevamente el proceso, pero la entrada es diferente:

Paso hacia adelante: llamaremos a este módulo **activacionJ()**

∑ entradas[i]*pesos[i]

Para el neuronJ[0] (la entrada es 0, 0,-1)

entradas[0]*pesos[0] + entradas[1]*pesos[1] + entradas[2]*pesos[2]
suma = 0*0.83674 + 0*0.93483 - 1*1.14154 = -1.14154

Aplicando la sigmoide
$1/(1+ e^{-suma}) = 1/ (1+e^{+1.14154)} =$ **0.24203**

Para el neuronJ[1] (la entrada es la misma es 1,0,-1)

entradas[0]*pesos[0] + entradas[1]*pesos[1] + entradas[2]*pesos[2]
suma = 0*1.50778+ 0*1.56155 - 1*0.05166 = -0.05166

Aplicando la sigmoide
$1/(1+ e^{-suma}) = 1/ (1+e^{+0.05166}) =$ **0.48708**

Módulo entradasK
toma el valor de activaciónJ y la última toma el valor del bias

neuronK[0].entradas[0]= **0.24203**
neuronK[0].entradas[1]= **0.48708**
neuronK[0].entradas [2]=-1

Módulo ActivacionK: ∑ entradas[i]*pesos[i]

0.24203*(-1.02199) + 0.48708*0.73686 -1*0.04233 = 0.06923
Aplicando la sigmoide
$1/(1+ e^{-suma}) = 1/ (1+e^{-(-0.12739))} =$ **0.51730**

Para hallar la salida de la red, vamos a ver qué valores nos da la función sigmoide.
Como 0.51730 es mayor que 0.5 la salida es uno y lo que debe de aprender es 0, no cumple la condición 1=0, **retroceso, empezamos hacia atrás**

Retroceso
Error capaK
Ek = (tk-ok)*ok*(1-ok)

errorK=(0 - 0.51730)*0.51730*(1 - 0.51730) = -**0.12917**

i=-1

Calculo de pesosJK (nuevos pesosJK)

wjk=wjk + L*Ek*oj
observar oj = la entrada del neuronk

neuronk[0].pesos[0] = neuronk[0].pesos[0] +L* EK*0j
neuronk[0].pesos[0] = -1.02199 +0.5*(-0.12917)*0.24203= -**1.03762**

neuronk[0].pesos[1] = neuronk[0].pesos[1] +L*Ek*oj
neuronk[0].pesos[1] = 0.73686 +0.5*(-0.12917)*0.48708 = **0.70541**

neuronk[0].pesos[2] = neuronk[0].pesos[2] +L*EK*oj
neuronk[0].pesos[2] = 0.04233 +0.5*(-0.12917)*(-1) = **0.10691**

Calculo de error capaJ
Ej = oj*(1-oj)*∑Ek*wjk

nodosCapak=2, nodosCapaJ=1
j=0,k=0
Primero hallamos la suma:
suma = suma+Ek*wjk
suma = 0+(-0.12917)*(-1.03762) = 0.13402
Luego hallamos el errorJ
errJ= 0.24203*(1-0.24203)*0.13402=0.02458

j=1,k=0
Primero hallamos la suma:
suma=suma+Ek*wjk
suma= 0+(-0.12917)*0.70541 = -0.09111
Luego hallamos el errorJ
errJ= 0.48708*(1-0.48708)*-0.09111=-0.02276

Calculo de pesosIJ (nuevos pesosIJ)

wij=wij + L*Ej*oi
para el primer error ej=0.02458
nodosCapaJ=3, entradaCapaJ=3 recordar entrada es (0,0,-1)

neuronJ[0].pesos[0]= neuronJ[0].pesos[0]+ 0.5*0.02458*0
neuronJ[0].pesos[0]= 0.83674+ 0 = **0.83674**

neuronJ[0].pesos[1] = neuronJ[0].pesos[1] +0.5*0.02458*0

neuronJ[0].pesos[1] = 0.93483 + 0 = **0.93483**

neuronJ[0].pesos[2] = neuronJ[0].pesos[2] +0.5*0.02458*-1
neuronJ[0].pesos[2] = 1.14154 +0.5*0.02458*(-1) = **1.12925**

para el segundo error ej=-0.02276 recordar entrada es (0,0,-1)

neuronJ[1].pesos[0]= neuronJ[1].pesos[0]+ 0.5*(-0.02276)*0
neuronJ[1].pesos[0]= 0.1.50778 + 0 = **1.50778**

neuronJ[1].pesos[1] = neuronJ[1].pesos[1] +0.5*(-0.02276)*0
neuronJ[1].pesos[1] = 1.56155 +0 = **1.56155**

neuronJ[1].pesos[2] = neuronJ[1].pesos[2] +0.5*(-0.02276)*(-1)
neuronJ[1].pesos[2] = 0.05166 + 0.01138 = **0.06304**

Actualizando la red con los nuevos pesos

Estructura de la red neuronal

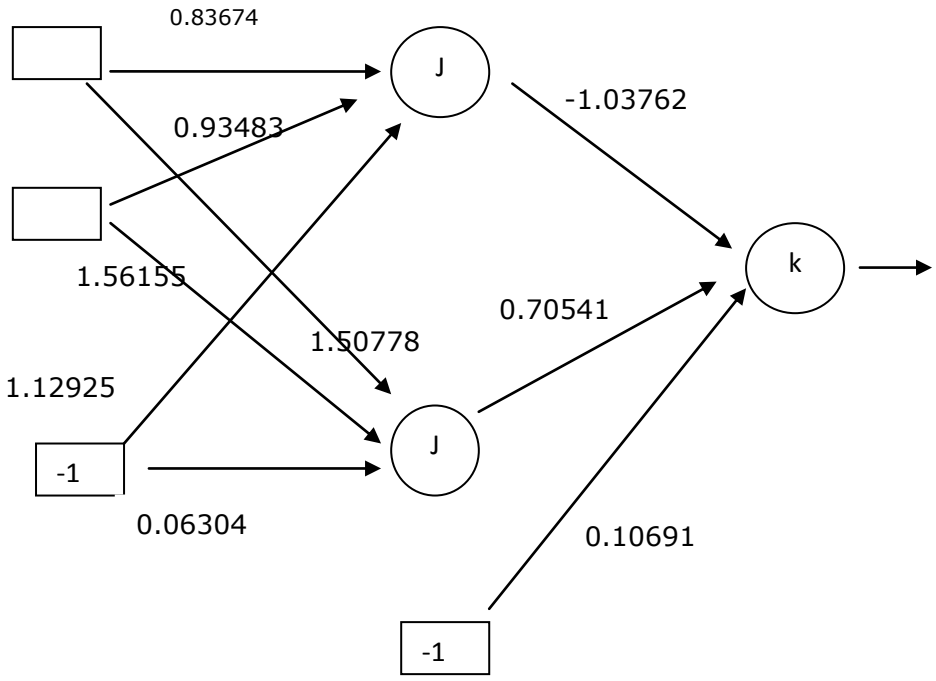


Figura 2.26. BP con nuevos pesos

Empezamos nuevamente
La primer entrada (1,1,-1), época 0, i=0, la salida es 1
Paso hacia adelante: llamaremos a este módulo **activacionJ()**

∑ entradas[i]*pesos[i]

Para el neuronJ[0] (la entrada es 1,1,-1)

entradas[0]*pesos[0] + entradas[1]*pesos[1] + entradas[2]*pesos[2]
suma = 1*0.83674 + 1*0.93483 - 1*1.12925 = 0.64232

Aplicando la sigmoide
$1/(1+ e^{-suma}) = 1/ (1+e^{-0.64232}) =$ **0.65528**

Para el neuronJ[1] (la entrada es la misma es 1,1,-1)

entradas[0]*pesos[0] + entradas[1]*pesos[1] + entradas[2]*pesos[2]
suma = 1*1.50778+ 1*1.56155 - 1*0.06304 = 3.00629

Aplicando la sigmoide
$1/(1+ e^{-suma}) = 1/ (1+e^{-0.94685}) =$ **0.95286**

Módulo entradasK
toma el valor de activaciónJ y la última toma el valor del bias

neuronK[0].entradas[0]= **0.65528**
neuronK[0].entradas[1]= **0.95286**
neuronK[0].entradas [2]=-1

Módulo ActivacionK: ∑ entradas[i]*pesos[i]

0.65528*(-1.03762) + 0.95286*0.70541 -1*0.10691 = -0.11469
Aplicando la sigmoide
$1/(1+ e^{-suma}) = 1/ (1+e^{+0.11469}) =$ **0.47135**

Para hallar la salida de la red, vamos a ver qué valores nos da la función sigmoide.
Como 0.47135 es menor que 0.5 la salida es cero y lo que debe de aprender es 0, cumple la condición 0=0, **no hay retroceso, empezamos nuevamente hacia adelante**

Analizaremos la segunda entrada (1, 0, -1), época 1, i=1, la salida es 1
Se repite nuevamente el proceso, pero la entrada es diferente:

Paso hacia adelante: llamaremos a este módulo **activacionJ()**

∑ entradas[i]*pesos[i]

Para el neuronJ[0] (la entrada es 1,0,-1)

entradas[0]*pesos[0] + entradas[1]*pesos[1] + entradas[2]*pesos[2]
suma = 1*0.83674 + 0*0.93483 - 1*1.12925 = -0.29251

Aplicando la sigmoide
$1/(1+ e^{-suma}) = 1/ (1+e^{+0.29251}) =$ **0.42738**

Para el neuronJ[1] (la entrada es la misma es 1,0,-1)

entradas[0]*pesos[0] + entradas[1]*pesos[1] + entradas[2]*pesos[2]
suma = 1*1.50778+ 0*1.56155 - 1*0.06304 = 1.44474

Aplicando la sigmoide
$1/(1+ e^{-suma}) = 1/ (1+e^{-1.44474}) =$ **0.80918**

Módulo entradasK
toma el valor de activaciónJ y la última toma el valor del bias

neuronK[0].entradas[0]= **0.42738**
neuronK[0].entradas[1]= **0.80918**
neuronK[0].entradas [2]=-1

Módulo ActivacionK: ∑ entradas[i]*pesos[i]

0.42738*(-1.03762) + 0.80918*0.70541 -1*0.10691 = 0.05610
Aplicando la sigmoide
$1/(1+ e^{-suma}) = 1/ (1+e^{-0.05610}) =$ **0.50510**

Para hallar la salida de la red, vamos a ver qué valores nos da la función sigmoide.
Como 0.50510 es mayor que 0.5 la salida es uno y lo que debe de aprender es 1, cumple la condición 1=1, **no hay retroceso, empezamos nuevamente hacia adelante**

La tercera entrada (0, 1, -1), época 2, i=2, la salida es 1
Se repite nuevamente el proceso, pero la entrada es diferente:

Paso hacia adelante: llamaremos a este módulo **activacionJ()**
∑ entradas[i]*pesos[i]

Para el neuronJ[0] (la entrada es 0, 1,-1)
entradas[0]*pesos[0] + entradas[1]*pesos[1] + entradas[2]*pesos[2]
suma = 0*0.83674 + 1*0.93483 - 1*1.12925 = -0.19442

Aplicando la sigmoide
$1/(1+ e^{-suma}) = 1/ (1+e^{+0.19442}) =$ **0.45154**

Para el neuronJ[1] (la entrada es la misma es 1,0,-1)
entradas[0]*pesos[0] + entradas[1]*pesos[1] + entradas[2]*pesos[2]
suma = 0*1.50778+ 1*1.56155 - 1*0.06304 = 1.49851

Aplicando la sigmoide
$1/(1+ e^{-suma}) = 1/ (1+e^{-1.49851}) =$ **0.81735**

Módulo entradasK
toma el valor de activaciónJ y la última toma el valor del bias

neuronK[0].entradas[0]= **0.45154**
neuronK[0].entradas[1]= **0.81735**
neuronK[0].entradas [2]=-1

Módulo ActivacionK: ∑ entradas[i]*pesos[i]

0.45154*(-1.03762)+ 0.81735*0.70541 -1*0.10691 = 0.00113
Aplicando la sigmoide
$1/(1+ e^{-suma}) = 1/ (1+e^{-0.00113)} =$ **0.50028**

Para hallar la salida de la red, vamos a ver qué valores nos da la función sigmoide.
Como 0.50028 es mayor que 0.5 la salida es uno y lo que debe de aprender es 1, cumple la condición 1=1, **no hay retroceso, empezamos nuevamente hacia adelante**

La cuarta entrada (0, 0, -1), época 4, i=3, la salida es 0
Se repite nuevamente el proceso, pero la entrada es diferente:

Paso hacia adelante: llamaremos a este módulo **activacionJ()**

∑ entradas[i]*pesos[i]

Para el neuronJ[0] (la entrada es 0, 0,-1)

entradas[0]*pesos[0] + entradas[1]*pesos[1] + entradas[2]*pesos[2]
suma = 0*0.83674 + 0*0.93483 - 1*1.14154 = -1.14154

Aplicando la sigmoide
$1/(1+ e^{-suma}) = 1/ (1+e^{+1.14154)} =$ **0.24203**

Para el neuronJ[1] (la entrada es la misma es 1,0,-1)
entradas[0]*pesos[0] + entradas[1]*pesos[1] + entradas[2]*pesos[2]
suma = 0*1.50778+ 0*1.56155 - 1*0.06304 = -0.06304

Aplicando la sigmoide
$1/(1+ e^{-suma}) = 1/ (1+e^{+0.06304}) =$ **0.48425**

Módulo entradasK
toma el valor de activaciónJ y la última toma el valor del bias

neuronK[0].entradas[0]= **0.24203**
neuronK[0].entradas[1]= **0.48425**
neuronK[0].entradas [2]=-1

Módulo ActivacionK: ∑ entradas[i]*pesos[i]

0.24203*(-1.03762) + 0.48425*0.70541 -1*0.10691 =-0.01645
Aplicando la sigmoide
$1/(1+ e^{-suma}) = 1/ (1+e^{-(-0.01645))} =$ **0.4958**

Para hallar la salida de la red, vamos a ver qué valores nos da la función sigmoide.

Como 0.4958 es menor que 0.5 la salida es cero y lo que debe de aprender es 0, cumple la condición 0=0, no hay retroceso, la red ha aprendido correctamente

CAPÍTULO III

METODOLOGÍA DE LA INVESTIGACIÓN

3.1. MÉTODO, TIPO DE INVESTIGACIÓN

3.1.1. MÉTODO DE LA INVESTIGACIÓN

Para llevar a cabo la investigación se ha utilizado el enfoque mixto, debido a que el enfoque cualitativo fue utilizado al momento de descubrir y precisar las preguntas de investigación, mientras que el enfoque cuantitativo lo utilizaremos para la recolección y el análisis de datos para contestar las preguntas de investigación y probar las hipótesis planteadas.

3.1.2. TIPO DE INVESTIGACIÓN

El tipo de investigación es cuasi-experimental, prospectivo, longitudinal, analítico.

3.2. DISEÑO Y ESQUEMA DE INVESTIGACIÓN

Se aplicará un Diseño no experimental – transeccional – exploratorio y correcional.

Se selecciona la clasificación transeccional debido a que recolectaremos datos en un solo momento, en un tiempo único.

Asimismo, será un diseño transeccional correlacional-causal porque describiremos las relaciones entre dos o más categorías, conceptos o variables en un momento determinado

Esquema de la investigación:

1. Recolectar 20 fotos en formato bmp y grabarlas en una base de datos.
2. Diseño de la red neuronal artificial utilizando el algoritmo Adaline.
3. Diseño de algoritmo de aprendizaje de la red neuronal artificial para el reconocimiento facial de rostros humanos.
4. Desarrollo y pruebas del software del algoritmo de aprendizaje de la red neuronal artificial para el reconocimiento facial de rostros humanos
5. Prueba de la eficacia del sistema de reconocimiento facial de rostros humanos

Estrategia de pruebas de hipótesis: Para realizar la prueba de la hipótesis se desarrollará el software con el algoritmo Adaline. Utilizando este prototipo se realizará una serie de experiencias que permiten demostrar la hipótesis planteada.

3.3. COBERTURA DE ESTUDIO

3.3.1. POBLACIÓN Y MUESTRA

Se dispondrá de un banco de datos de 38 fotografías de 38 sujetos.

La muestra será de 20 fotografías que ingresarán al proceso de aprendizaje, para ubicarlas en el banco de datos; y las 18 restantes no estarán consideradas en este grupo de aprendizaje.

3.4. MÉTODOS, TÉCNICAS E INSTRUMENTOS DE RECOLECCIÓN DE DATOS

En esta investigación el análisis de datos es cuantitativo, debido a que se dispondrá de información numérica. Para ello, se aplicará estadística descriptiva para las variables tomadas individualmente:

- Desviación estándar
- Varianza

3.5. DISEÑO DEL PROTOTIPO

En la figura 3.1 se muestra el Diagrama de Reconocimiento facial de rostros humanos propuesto en la presente tesis, en el cual se grafica los siguientes procesos:

- APRENDIZAJE
- FUNCIONAMIENTO

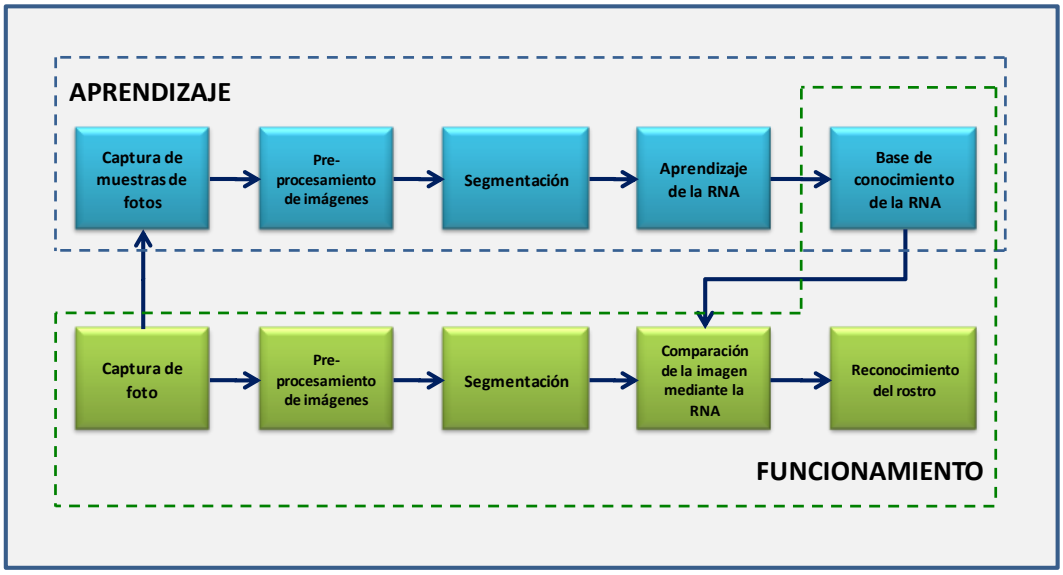


Figura 3.1: Diagrama de reconocimiento facial de rostros humanos

El proceso de **APRENDIZAJE** está conformado por las siguientes fases:

- Captura de muestras de fotos

  En esta fase se captura la imagen con una cámara web. El resultado de la captura es obtener una imagen, luego esta imagen es almacenada en blanco y negro en archivos de imágenes con formato .jpg en 24 bits. Estas imágenes se utilizarán para realizar el paso de aprendizaje de la RNA, para que posteriormente sea procesada por el sistema.

- Pre-procesamiento de imágenes

  En esta fase se convierten las imágenes obtenidas en colores de 24 bits por pixel a una imagen en blanco y negro mediante la binarización. El objetivo es incrementar las muestras de fotos que aporten en la fase de aprendizaje.

- Segmentación

  En esta fase se extrae de las imágenes la porción del rostro al cual se entrenará a la RNA.

- Aprendizaje de la RNA

  Es una fase mediante la cual se ajustan los pesos de la red utilizando las diversas muestras para que pueda reconocer los rostros.

- Base de conocimiento de la RNA

  En esta fase se almacena el conocimiento que ha obtenido la RNA durante la fase de Aprendizaje.

El proceso de **FUNCIONAMIENTO** está conformado por los siguientes pasos:

- Captura de fotos

  Se trata del proceso de captura de las imágenes con una cámara web. El resultado de la captura es obtener una imagen, luego esta imagen es almacenada en blanco y negro en archivos de imágenes con formato .jpg en 24 bits, para que posteriormente sea procesada por el sistema.

- Pre-procesamiento de imágenes

  En esta fase se convierten las imágenes obtenidas en colores de 24 bits por pixel a una imagen en blanco y negro mediante la binarización. El objetivo es incrementar las muestras de fotos que aporten en la fase de aprendizaje.

- Segmentación

  En esta fase se extrae de las imágenes la porción del rostro el cual se comparará con la base de conocimientos.

- Comparación de la imagen mediante la RNA

  En esta fase se compara la imagen mediante la RNA utilizando el conocimiento adquirido durante el proceso de aprendizaje el cual se encuentra en la base de conocimiento.

- Reconocimiento del rostro

  Se coloca en la pantalla el resultado del grado de coincidencias de la imagen con la muestra de imágenes. Si el resultado es mayor o igual a 6,000 significa que la imagen tomada ha sido identificada en la muestra de fotografías almacenadas.

A continuación en la figura 3.2 se muestra cada una de las etapas que se definieron para diseñar el prototipo de la solución planteada:

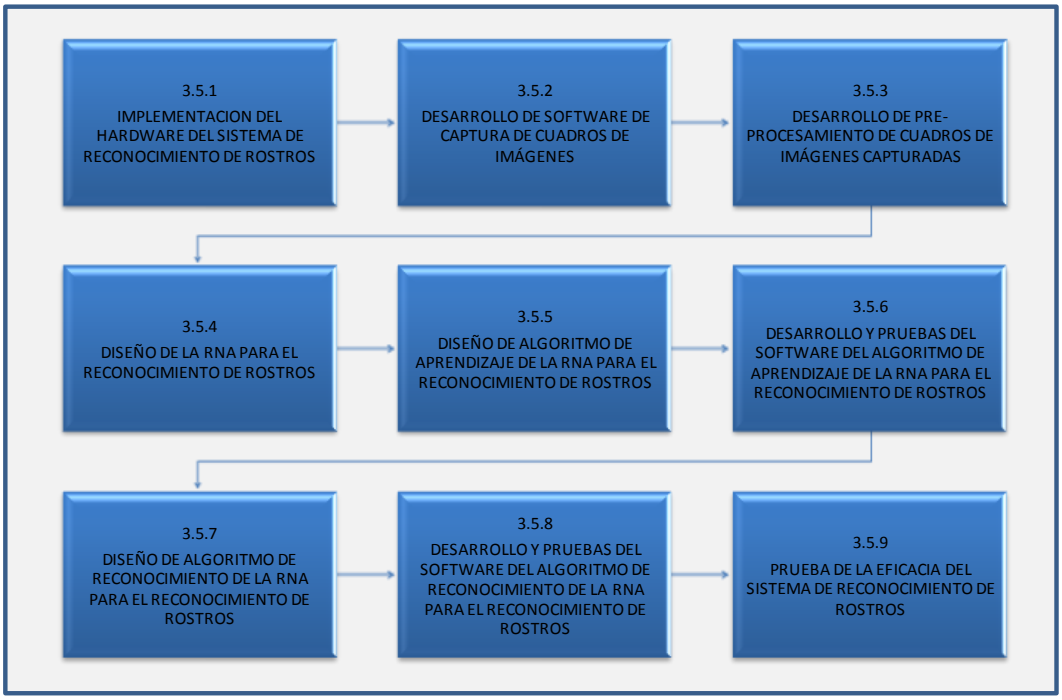


Figura 3.2: Etapas para el diseño del prototipo de la solución planteada

3.5.1. INSTALACIÓN DEL HARDWARE DEL SISTEMA DE RECONOCIMIENTO DE ROSTROS

En esta etapa se llevó a cabo la instalación de una cámara web en la laptop con un soporte para mantenerla fija.

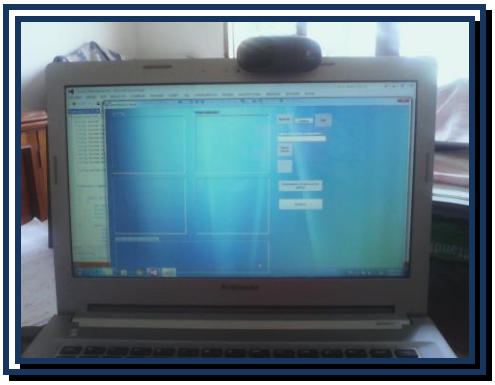


Figura 3.3: Hardware del Sistema de Reconocimiento de Rostros

3.5.2. CONSTRUCCIÓN DE SOFTWARE DE CAPTURA DE CUADROS DE IMÁGENES

Se desarrolló un software en Visual C++ para la captura de las imágenes con la cámara y guardarlas en archivos formato .jpg.

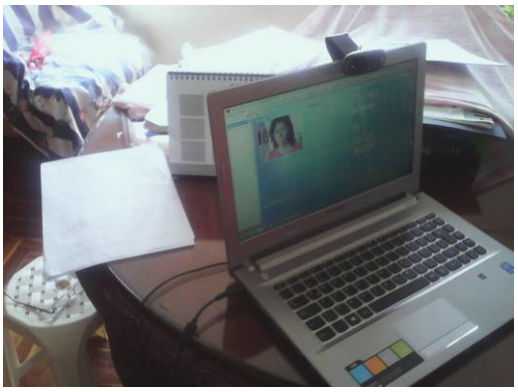


Figura 3.4: Interfaz de usuario para la captura de imágenes

Programa para capturar la foto.

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using System.IO;
using System.Drawing.Imaging;
using System.Runtime.InteropServices;

using System.Text.RegularExpressions;

namespace Camara_Web
{

    public partial class FWEB_CAM : Form
    {
        int contador;
        Bitmap imagei;
        Bitmap imaged;
        Bitmap grafico;

        public struct neurona{
                    public int m;
                    public double[] peso;
            public double n;
            public double a;
            public double u;
            public double k;
            public int capa;
        };

        int[] sred = new int[100];
        neurona[] rna = new neurona[100];
        int[] entrada = new int[150000];
```

```
        System.Windows.Forms.Timer MyTimer = new
System.Windows.Forms.Timer();


        public FWEB_CAM()
        {
            InitializeComponent();
        }

        private void Form1_Load(object sender, EventArgs e)
        {

            this.webCamCapture2.CaptureHeight =
this.pictureBox2.Height;
            this.webCamCapture2.CaptureWidth =
this.pictureBox2.Width;
            validacionCamara2.Text = "";

        }


        private void button5_Click(object sender, EventArgs
e)
        {


this.webCamCapture2.TimeToCapture_milliseconds = 30;
                this.webCamCapture2.Start(0);

        }


if (sNombreCam2Jpeg.EndsWith("Jpeg")){
                pictureBox2.Image.Save(sNombreCam2Jpeg,
ImageFormat.Jpeg);
}
```

3.5.3. CONSTRUCCIÓN DE PRE-PROCESAMIENTO DE CUADROS DE IMÁGENES CAPTURADAS

La imagen capturada mediante la cámara, y almacenada en los archivos formato .jpg se abre para obtener los pixeles de la imagen y se guardan en la memoria operativa de la computadora, para realizar los procesos de binarización, eliminación de ruido y segmentación del objeto.

La imagen capturada por la cámara web y almacenada en formato .jpg se pre-procesa y se obtiene la imagen pre-procesada tal como se puede apreciar en la figura 3.5

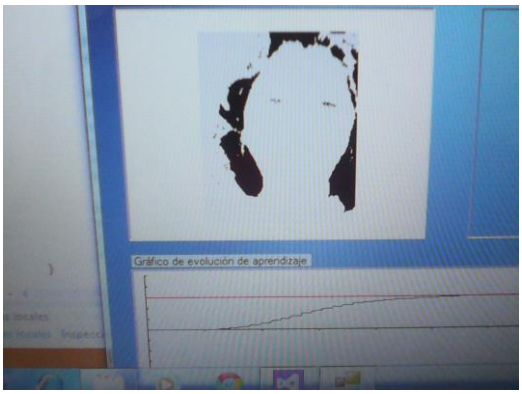


Figura 3.5: Imagen capturada pre-procesada

Programa para binarización la foto capturada.

```
try
        {
           Bitmap image1;
           Bitmap image2;
           Bitmap image3;
           //string pattern;
           Color newColor;

           uint colorAux;
           byte a;
           byte b;
           byte c;
           byte d;
           int x, y;
           uint com;
           int[,] dato = new int[300, 300];
           uint[,] datocolor = new uint[300, 300];

           image2 = new Bitmap(@"" + sNombreCam1Jpeg, true);
           image3 = new Bitmap(@"" + sNombreCam1Jpeg, true);



           image1 = new Bitmap(@"" + sNombreCam2Jpeg, true);
           cad = "";
           Replace = "";
           result = "";
           pattern = "";

           for (x = 0; x < image1.Width; x++)
           {
              for (y = 0; y < image1.Height; y++)
              {
                 Color pixelColor = image1.GetPixel(x, y);

                 colorAux = (uint)((pixelColor.A << 24) | (pixelColor.R << 16) |
(pixelColor.G << 8) | (pixelColor.B << 0));
```

```
        a = (byte)(colorAux >> 24);
        b = (byte)(colorAux >> 16);
        c = (byte)(colorAux >> 8);
        d = (byte)(colorAux >> 0);


        Replace = string.Format(".{0}", Environment.NewLine);

        if (b > 230 && c > 230 && d > 230)
        {
           dato[x, y] = 1;
           newColor = Color.FromArgb(0, 0, 0);
           cad = cad + "1";

        }
        else
        {
           dato[x, y] = 0;
           newColor = Color.FromArgb(230, 230, 230);
           cad = cad + "0";
        }

        datocolor[x, y] = colorAux;
        newColor = Color.FromArgb(b,c,d);
        image2.SetPixel(x, y, newColor);
     }

     cad = cad + ".";

  }

  com = 0;
  for (x = (int)image1.Width / 2 - 2; x < (int)image1.Width / 2 + 2; x++)
  {
     for (y = (int)image1.Height / 4 - 2; y < (int)image1.Height / 4 + 2; y++)
     {

        colorAux = datocolor[x, y];

        a = (byte)(colorAux >> 24);
        b = (byte)(colorAux >> 16);
        c = (byte)(colorAux >> 8);
        d = (byte)(colorAux >> 0);

        Replace = string.Format(".{0}", Environment.NewLine);

        newColor = Color.FromArgb(230, 230, 230);
        cad = cad + "P(" + x + "," + y + ")= " + colorAux + "\n";
        cad = cad + "P(" + x + "," + y + ")= " + b + "," + c + "," + d + "\n";
        cad = cad + "P(" + x + "," + y + ")= " + (b + c + d) / 3 + "\n";
        com = com + (uint)((b + c + d) / 3);
```

```
            image2.SetPixel(x, y, newColor);
        }
    }
    com = (uint)(com / 16);

    ie = 0;
    for (x = 72; x < image1.Width-72; x++)
    {
        for (y = 20; y < image1.Height-20; y++)
        {

            colorAux = datocolor[x,y];

            a = (byte)(colorAux >> 24);
            b = (byte)(colorAux >> 16);
            c = (byte)(colorAux >> 8);
            d = (byte)(colorAux >> 0);


            Replace = string.Format(".{0}", Environment.NewLine);

            if ((((b + c + d) / 3) > (com - 30)) && (((b + c + d) / 3) < (com + 30)))
            {
                dato[x, y] = 1;
                entrada[ie] = 1;
                ie++;
                newColor = Color.FromArgb(0, 0, 0);
                cad = cad + "1";

            }
            else
            {
                dato[x, y] = 0;
                entrada[ie] = 0;
                ie++;
                newColor = Color.FromArgb(230, 230, 230);
                cad = cad + "0";
            }
                image3.SetPixel(x, y, newColor);
        }

        cad = cad + ".";

    }

    pattern = @"([./]{1})";
    result = Regex.Replace(cad, pattern, Replace);

    using (FileStream fs = new FileStream("C:\\CAPTURAS\\txtCamara2.txt",
FileMode.OpenOrCreate, FileAccess.Write))
    {
        byte[] buffer = System.Text.Encoding.Default.GetBytes(result);
```

```
    fs.Write(buffer, 0, buffer.Length);

}
result = "";
cad = "";
pictureBox1.Image = image2;
pictureBox3.Image = image3;
```

3.5.4. DISEÑO DE LA RNA PARA EL RECONOCIMIENTO DE ROSTROS

El diseño de la estructura de la RNA para el reconocimiento de rostro facial, corresponde a una red neuronal artificial de pre alimentación con neuronas de activación lineal y que consta de cuatro capas ver figura 3.6, donde en la primera capa tenemos veinte (20) neuronas, en la segunda capa tiene cuarenta (40) neuronas, en la tercera capa veinte (20) neuronas y en la cuarta y última capa una (1) neurona.

La red tiene 64.965 entradas ya que cada pixel de la imagen capturada que se convierten en una entrada.

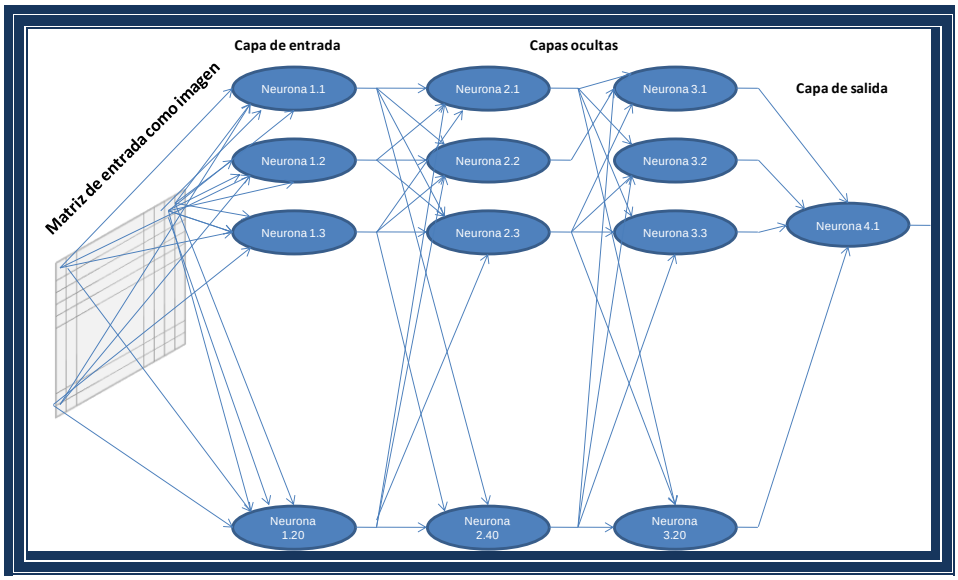


Figura 3.6: Diseño de la estructura de la RNA

3.5.5. DISEÑO DE ALGORITMO DE APRENDIZAJE DE LA RNA PARA EL RECONOCIMIENTO DE ROSTROS

El algoritmo de aprendizaje para la estructura de la red neuronal está representado a través del diagrama de flujo de la figura 3.7

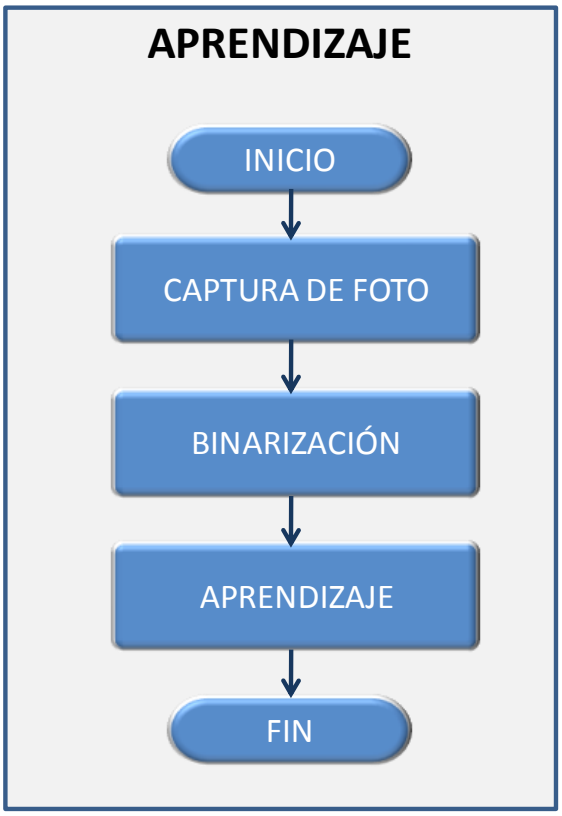


Figura 3.7: Diagrama de flujo del algoritmo de aprendizaje

Para la actualización se usa el algoritmo de Backpropagation modificado, en el cual en lugar de usar el error medio cuadrático se usa el error instantáneo de la RNA. En base a esto para actualizar los pesos se usa la fórmula:

$$\mathbf{peso_{j,i} = peso_{j,i-1} + error * ka * entrada_j}$$

Donde:

$peso_{j,i}$ – Valor del peso j actual.

$peso_{j,i}$ – Valor del peso j anterior.

error = Valor deseado de salida de la RNA - salida real de la RNA

ka – Velocidad de aprendizaje.

Entrada$_j$ – Entrada de la cual se actualiza el peso.

3.5.6. DESARROLLO Y PRUEBAS DEL SOFTWARE DEL ALGORITMO DE APRENDIZAJE DE LA RNA PARA EL RECONOCIMIENTO DE ROSTROS

Se ha realizado un programa en Visual C++ para el aprendizaje de la red neuronal artificial, y se han realizado las pruebas correspondientes para determinar la convergencia del algoritmo.

Programa aprendizaje de la red neuronal artificial.

```
public struct neurona{
            public int m;
            public double[] peso;
          public double n;
          public double a;
          public double u;
          public double k;
          public int capa;
       };

       int[] sred = new int[100];
       neurona[] rna = new neurona[100];
       int[] entrada = new int[150000];


      string sGraficoJpeg = "C:\\CAPTURAS\\grafico.jpg";

      grafico = new Bitmap(@"" + sGraficoJpeg, true);
      GraficoAprendizaje.Image = grafico;
      int Xmax = 600;
      int Ymax = 130;
      int xc1, yc1, xc2, yc2;
      int dx, dy;
      float pmax = 10000;
      float fx = 30000000;
      xc1 = 10;
      yc1 = Ymax / 2;
      dx = Xmax - 20;
      dy = Ymax / 2 - 5;
      xc2 = 10;
      yc2 = Ymax / 2 - 20;

      coordenadas(grafico, xc1, yc1, dx, dy, pmax, fx);
```

```
                /* Definición de la estructura de la red*/

                sred[0] = 4;
                sred[1] = 20;
                sred[2] = 40;
                 sred[3] = 20;
                sred[4] = 1;

                Bitmap imagee;
                string sNombreCam3 = "C:\\CAPTURAS\\blanco.jpg";

                imagee = new Bitmap(@"" + sNombreCam3, true);
                int ancho = imagee.Width-144;
                int alto = imagee.Height-40;
                int nte = ancho * alto;
                double factor = 0.001;

                  Random random = new Random();

                  string Replace;
                  string cad;
                 string result;
                 string pattern;
                 int ie = 0;
                double lim = 6000;

                cad = "";
                Replace = "";
                result = "";
                pattern = "";

int ir = 0;
for (int j = 1; j <= sred[0]; j++)
{
   if (j == 1)
   {
      for (int k = 0; k < sred[j]; k++)
      {
         rna[ir].m = (int)ancho * alto;
         rna[ir].capa = j;
         rna[ir].peso = new double[150000];
         for (int p = 0; p < rna[ir].m; p++)
         {
            rna[ir].peso[p] = random.NextDouble() * factor;
         }
         rna[ir].k = 1;
         rna[ir].u = 6000;
         ir++;
      }
   }
   else
   {
```

```
        for (int k = 0; k < sred[j]; k++)
        {
            rna[ir].m = sred[j - 1];
            rna[ir].capa = j;
            rna[ir].peso = new double[150000];
            for (int p = 0; p < rna[ir].m; p++)
            {
                rna[ir].peso[p] = random.NextDouble() * factor;
            }
            rna[ir].k = 1;
            rna[ir].u = 6000;
            ir++;
        }
    }
}

int nred = 0;
for (int j = 1; j <= sred[0]; j++)
{
    nred = nred + sred[j];
}

cad = cad + "Pesos\n";


int ne = 0;
double AP = 0.005;
double ka = 0.000001;
double na = 0;
double error = 0;
double res1 = 0;
double res = 0;
int ni = 0;
int nia = 0;
double exe = dx / fx;
double eye = dy / pmax;



Color co;
co = Color.FromArgb(255, 0, 0);



   ie = 0;
    for (x = 72; x < image1.Width-72; x++)
    {
        for (y = 20; y < image1.Height-20; y++)
        {


            colorAux = datocolor[x,y];
```

```
                a = (byte)(colorAux >> 24);
                b = (byte)(colorAux >> 16);
                c = (byte)(colorAux >> 8);
                d = (byte)(colorAux >> 0);


                Replace = string.Format(".{0}", Environment.NewLine);

                if ((((b + c + d) / 3) > (com - 30)) && (((b + c + d) / 3) < (com + 30)))
                {
                   entrada[ie] = 1;
                   ie++;
                   newColor = Color.FromArgb(0, 0, 0);
                   cad = cad + "1";

                }
                else
                {
                   entrada[ie] = 0;
                   ie++;
                   newColor = Color.FromArgb(230, 230, 230);
                   cad = cad + "0";
                }
                   image3.SetPixel(x, y, newColor);
             }

             cad = cad + ".";

          }

res = CalculoRed(nred);
cad = cad + "\nResultado inicial de la red = " + res + "\n";


          co = Color.FromArgb(0, 0, 255);

          do
          {
             for (int j = 0; j < nred; j++)
             {
                res = CalculoRed(nred);
                error = lim - res;
                if ((res > (lim + 100)) | (res < (lim - 100)))
                {
                   if (rna[j].capa == 1)
                   {
                      for (ne = 0; ne < rna[j].m; ne++)
                      {
                         if (entrada[ne] > 0)
                         {
                            rna[j].peso[ne] = rna[j].peso[ne] + error * ka * entrada[ne];
```

```
                    }
                    ni++;
                }
            }
            else
            {
                for (ne = 0; ne < rna[j].m; ne++)
                {
                    rna[j].peso[ne] = rna[j].peso[ne] + error * ka;
                    ni++;
                }
            }
        }
        line(grafico, xc1 + (int)(exe * nia), yc1 - (int)(eye * na), xc1 + (int)(exe * ni), yc1 - (int)(eye * res), co);
        nia = ni;
        na = res;

    }
} while ((res > (lim + 100)) | (res < (lim - 100)));
co = Color.FromArgb(100, 100, 100);
line(grafico, xc1 + (int)(exe * ni), yc1, xc1 + (int)(exe * ni), yc1 - dy, co);
```

3.5.7. DISEÑO DE ALGORITMO DE FUNCIONAMIENTO DE LA RNA PARA EL RECONOCIMIENTO DE ROSTROS

El algoritmo de funcionamiento para la estructura de la red neuronal está representado a través del diagrama de flujo de la figura 3.8:

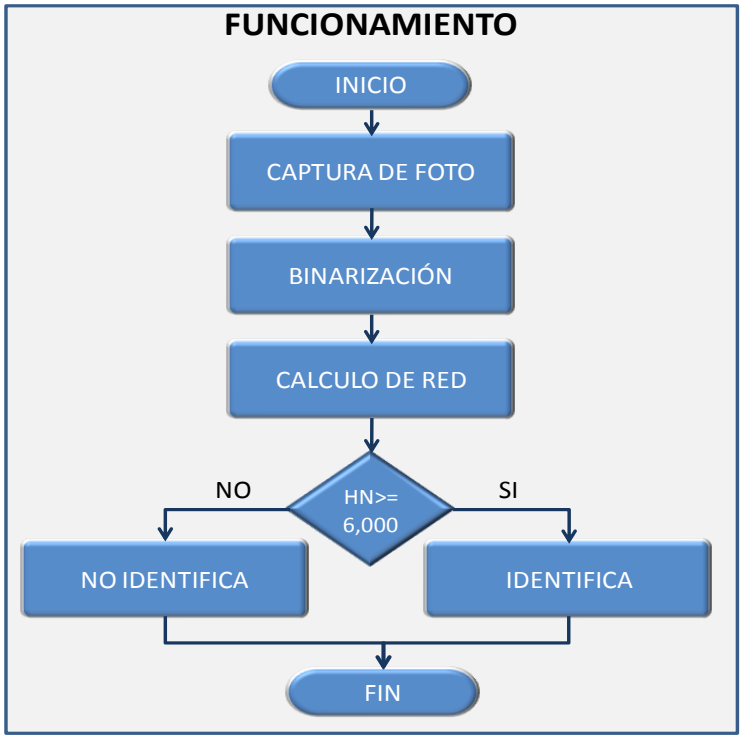


Figura 3.8: Diagrama de flujo del algoritmo de funcionamiento

3.5.8. CONSTRUCCIÓN Y PRUEBAS DEL SOFTWARE DEL ALGORITMO DE FUNCIONAMIENTO DE LA RNA PARA EL RECONOCIMIENTO DE ROSTROS

Se programó en Visual C++ el algoritmo de funcionamiento de la red neuronal artificial tal como se muestra en el código y se realizaron las pruebas correspondientes.

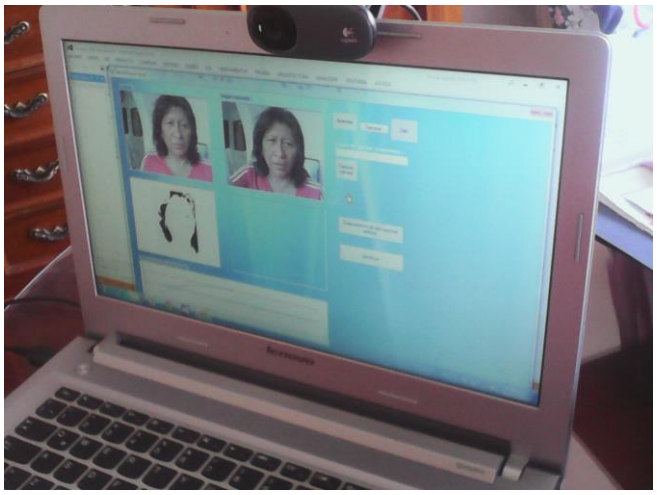


Figura 3.9: Interfaz de funcionamiento para el reconocimiento de rostros

Programa de funcionamiento de la red neuronal artificial.

```
private void button7_Click(object sender, EventArgs e)
      {
         double res;
         int nred = 0;
         int ie = 0;
         string sNombreCam2Jpeg = "C:\\CAPTURAS\\cam2(" + contador + ").Jpeg";
         string sNombreCam1Jpeg = "C:\\CAPTURAS\\blanco.jpg";
         if (sNombreCam2Jpeg.EndsWith("Jpeg"))
         {
            pictureBox2.Image.Save(sNombreCam2Jpeg, ImageFormat.Jpeg);
         }

         string Replace;
         string cad;
         string result;

         try
         {
            Bitmap image1;
            Bitmap image2;
            Bitmap image3;
            string pattern;
            Color newColor;

            uint colorAux;
            byte a;
            byte b;
            byte c;
            byte d;
            uint com;
            int x, y;
            int[,] dato = new int[300, 300];
            uint[,] datocolor = new uint[300, 300];

            image2 = new Bitmap(@"" + sNombreCam1Jpeg, true);
            image3 = new Bitmap(@"" + sNombreCam1Jpeg, true);

            image1 = new Bitmap(@"" + sNombreCam2Jpeg, true);
            cad = "";
            Replace = "";
            result = "";
            pattern = "";

            for (x = 0; x < image1.Width; x++)
            {
               for (y = 0; y < image1.Height; y++)
               {
                  Color pixelColor = image1.GetPixel(x, y);

                  colorAux = (uint)((pixelColor.A << 24) | (pixelColor.R << 16) |
(pixelColor.G << 8) | (pixelColor.B << 0));
```

```
            a = (byte)(colorAux >> 24);
            b = (byte)(colorAux >> 16);
            c = (byte)(colorAux >> 8);
            d = (byte)(colorAux >> 0);


            Replace = string.Format(".{0}", Environment.NewLine);

            if (b > 230 && c > 230 && d > 230)
            {
               dato[x, y] = 1;
               newColor = Color.FromArgb(0, 0, 0);
               cad = cad + "1";

            }
            else
            {
               dato[x, y] = 0;
               newColor = Color.FromArgb(230, 230, 230);
               cad = cad + "0";
            }

            datocolor[x, y] = colorAux;
            newColor = Color.FromArgb(b, c, d);
            image2.SetPixel(x, y, newColor);
         }

         cad = cad + ".";

      }
      com = 0;
      for (x = (int)image1.Width / 2 - 2; x < (int)image1.Width / 2 + 2; x++)
      {
         for (y = (int)image1.Height / 4 - 2; y < (int)image1.Height / 4 + 2; y++)
         {

            colorAux = datocolor[x, y];

            a = (byte)(colorAux >> 24);
            b = (byte)(colorAux >> 16);
            c = (byte)(colorAux >> 8);
            d = (byte)(colorAux >> 0);

            Replace = string.Format(".{0}", Environment.NewLine);

            newColor = Color.FromArgb(230, 230, 230);
            cad = cad + "P(" + x + "," + y + ")= " + colorAux + "\n";
            cad = cad + "P(" + x + "," + y + ")= " + b + "," + c + "," + d + "\n";
            cad = cad + "P(" + x + "," + y + ")= " + (b + c + d) / 3 + "\n";
            com = com + (uint)((b + c + d) / 3);
            image2.SetPixel(x, y, newColor);
         }
```

```
}
com = (uint)(com / 16);

cad = cad + "com = " + com + "\n";

ie = 0;
for (x = 72; x < image1.Width - 72; x++)
{
   for (y = 20; y < image1.Height - 20; y++)
   {

      colorAux = datocolor[x, y];

      a = (byte)(colorAux >> 24);
      b = (byte)(colorAux >> 16);
      c = (byte)(colorAux >> 8);
      d = (byte)(colorAux >> 0);


      Replace = string.Format(".{0}", Environment.NewLine);

      if ((((b + c + d) / 3) > (com-30)) && (((b + c + d) / 3) < (com+30)))
      {
         dato[x, y] = 1;
         entrada[ie] = 1;
         ie++;
         newColor = Color.FromArgb(0, 0, 0);
         cad = cad + "1";

      }
      else
      {
         dato[x, y] = 0;
         entrada[ie] = 0;
         ie++;
         newColor = Color.FromArgb(230, 230, 230);
         cad = cad + "0";
      }
      image3.SetPixel(x, y, newColor);
   }

   cad = cad + "\n";

}


result = Regex.Replace(cad, pattern, Replace);


pictureBox1.Image = image2;
pictureBox3.Image = image3;
cad = cad + "Numero de entradas = " + ie + "\n";
```

```
            for (int j = 1; j <= sred[0]; j++)
            {
               nred = nred + sred[j];
            }
            res = CalculoRed(nred);

            MessageBox.Show("Resultado = " + res);

            cad = cad + "\nResultado inicial de la red = " + res + "\n";


            pattern = @"([\n/]{1})";
            result = Regex.Replace(cad, pattern, Replace);

            using (FileStream fss = new FileStream("C:\\CAPTURAS\\identifica.txt",
FileMode.OpenOrCreate, FileAccess.Write))
            {
               byte[] buffer = System.Text.Encoding.Default.GetBytes(result);
               fss.Write(buffer, 0, buffer.Length);

            }

         }
         catch (ArgumentException)
         {
            MessageBox.Show("Hay un error." + "Revise la ruta del archivo de la
imagen.");
         }
      }
```

3.5.9. PRUEBA DEL SISTEMA

3.5.9.1. APRENDIZAJE DE LA RED NEURONAL ARTIFICIAL

El aprendizaje de la red neuronal artificial se realizó con las fotografías de 20 personas como se muestra en la Figura 3.10.

Figura 3.10 Fotos de pruebas

En las figuras 3.11 y 3.12 se presenta el aprendizaje con las fotografías 1 y 10 de la muestra, y se puede apreciar la curva de evolución de aprendizaje.

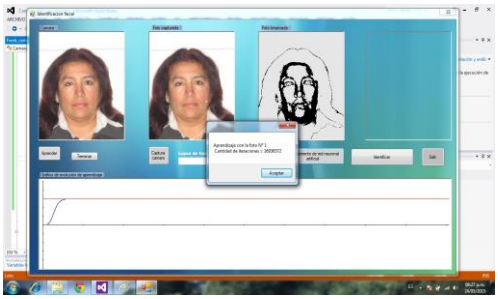


Figura 3.11: Evolución de aprendizaje con la foto 1

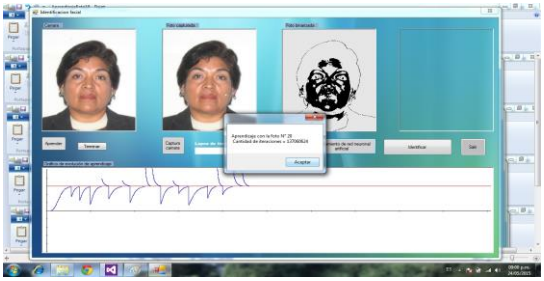


Figura 3.12: Evolución de aprendizaje con la foto 20.

En la Figura 3.13 y 3.14 se presentan la evolución del aprendizaje de la red neuronal artificial de repeticiones consecutivas con todo el conjunto de muestras de fotos y se puede apreciar que en cada repetición el error es menor hasta que finalmente la red converge hacia el umbral con todas las fotografías.

Figura 3.13: Evolución de aprendizaje en la repetición 1 con todo el conjunto de muestras.

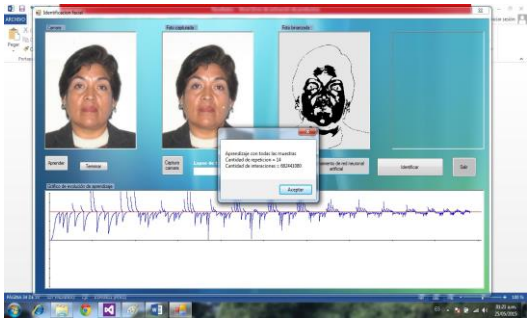


Figura 3.14: Evolución de aprendizaje en la repetición 14 con todo el conjunto de muestras.

3.5.9.2. PRUEBAS DEL FUNCIONAMIENTO DE LA RNA

En las Figuras 3.15 y 3.16 se presentan los resultados de identificación con las 20 fotografías que se realizó el aprendizaje repitiendo 20 veces con todo el conjunto de muestras.

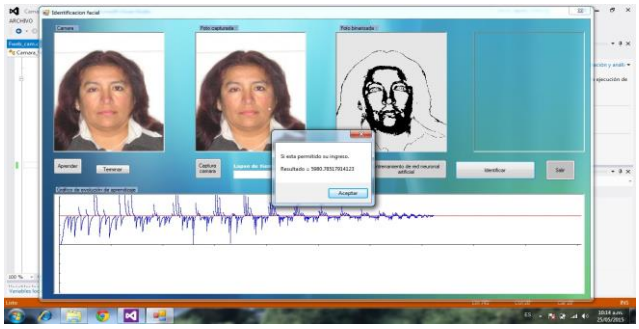


Figura 3.15: Resultados de identificación.

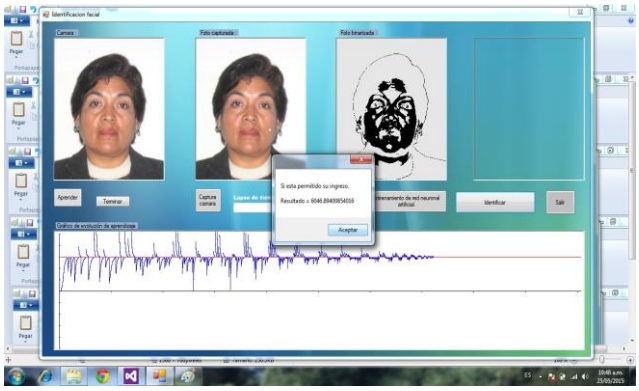


Figura 3.16: Resultados de identificación.

En las Figuras 3.17 y 3.18 se presenta las pruebas con 18 fotos que no pertenecen al conjunto de muestras.

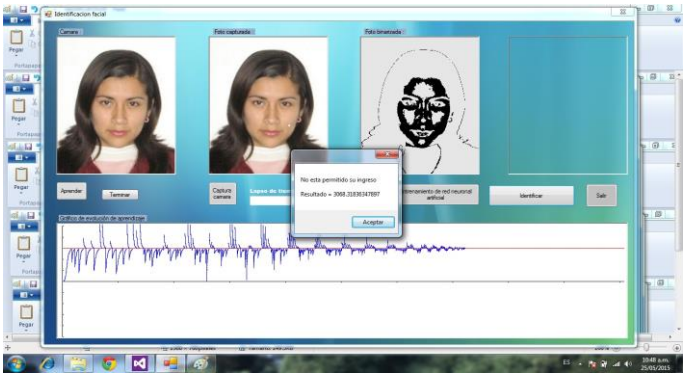


Figura 3.17: Pruebas que no pertenecen al conjunto de muestras.

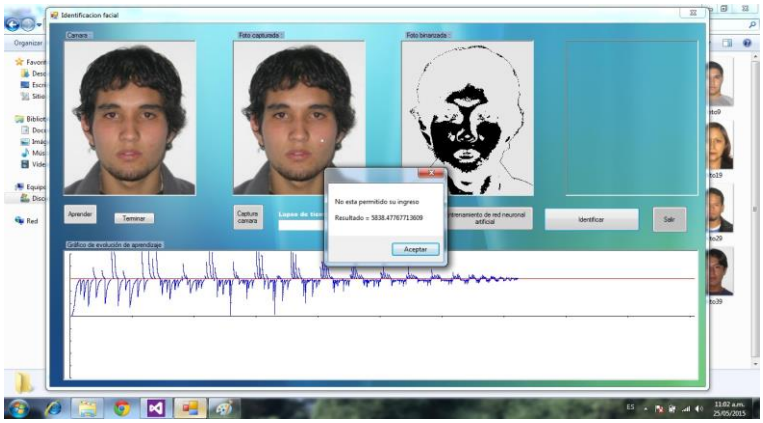


Figura 3.18: Pruebas que no pertenecen al conjunto de muestras.

A continuación se muestra el gráfico de evolución de aprendizaje con 30 repeticiones con todo el conjunto.

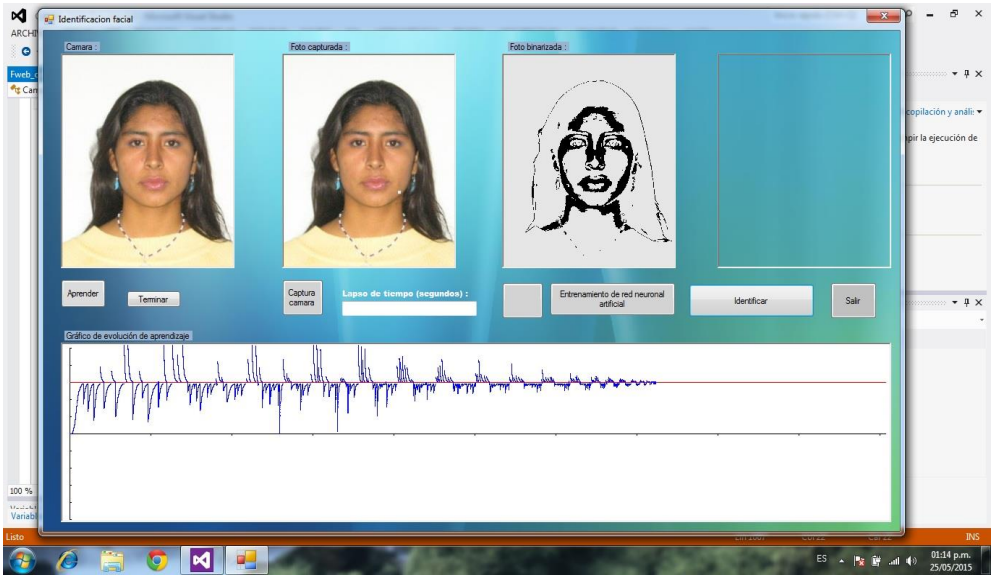


Figura 3.19: Pruebas que no pertenecen al conjunto de muestras

CAPÍTULO IV

RESULTADOS

4.1. PRESENTACIÓN DE RESULTADOS

4.1.1. RESULTADOS DE EVOLUCIÓN DEL APRENDIZAJE

A continuación se muestra la tabla 4.1 con los resultados de la evolución del aprendizaje del total de muestras de fotos (20), en el cual se detalla por c/u de ellas las iteraciones obtenidas.

Tabla 4.1: Evolución del aprendizaje

Registro	Foto	Iteraciones
1	1	16936572
2	2	22549272
3	3	29715712
4	4	37334764
5	5	42111604
6	6	51189424
7	7	56443984
8	8	66476024
9	9	70441112
10	10	77344304
11	11	85466224
12	12	89407504
13	13	96096164
14	14	108661292
15	15	108994604
16	16	114727504
17	17	118906864
18	18	125810056
19	19	132283084
20	20	137060624

4.1.2. EVOLUCIÓN DE APRENDIZAJE EN LA REPETICIÓN CON TODO EL CONJUNTO DE MUESTRAS

En la Tabla 4.2 se muestra la evolución del aprendizaje de la foto 20 con 20 repeticiones y con 30 repeticiones.

Tabla 4.2: Evolución de aprendizaje en la repetición con todo el conjunto de muestras (caso Foto 20)

Registro	20 repeticiones	30 repeticiones
1	137060644	261608498
2	222379840	346975585
3	298255916	453003870
4	353762108	478310086
5	401407148	525955002
6	449144540	573692394
7	497308476	621856330
8	536834116	661381970
9	575614568	700162422
10	609071576	733619430
11	634834228	759382082
12	668976128	793523982
13	682441080	806988934
14	694902574	819450428
15	706901578	831449432
16	718101354	842649208
17	728909639	853457493
18	739656583	864204437
19	750184726	874732580
20	760627848	885175702
21		897521380
22		909867058
23		922212736
24		934558414
25		946904092
26		959249770
27		971595448
28		983941126
29		996286804
30		1008632482

En la Tabla 4.3 se muestra los resultados de identificaciòn de la muestra de las fotos que se consideraron en el Proceso de aprendizaje. Se puede observar que en la ejecuciòn de la prueba con 20 repeticiones la foto 4 no logrò alcanzar el valor mìnimo de 6000 para se considerada vàlida, sin embargo, en la ejecuciòn de la prueba con 30 repeticiones las 20 fotos fueron reconocidas por el sistema.

Tabla 4.3: Resultados de identificación con el grupo de aprendizaje

Registro	Foto	Con 20 repeticiones	Con 30 repeticiones
1	1	6180,785179	7278,785179
2	2	6662,874673	7362,874673
3	3	6392,382255	7009,018478
4	4	5633,954625	6223,392222
5	5	6441,744264	7171,302822
6	6	7338,884929	7897,120443
7	7	8070,339598	8694,659444
8	8	7921,655243	8254,105533
9	9	6095,382541	6786,034475
10	10	7978,246872	8178,242072
11	11	5647,48815	6511,48817
12	12	6073,695347	6541,578412
13	13	6090,530126	6784,254785
14	14	6096,716351	6874,254782
15	15	6093,382891	6748,254125
16	16	6076,348955	7024,348955
17	17	5645,582718	6495,582718
18	18	6094,41404	6854,357843
19	19	6099,290162	6784,254875
20	20	6046,894009	7154,365478

En la Tabla 4.4 se muestra los resultados de identificaciòn de la muestra de las fotos que no se consideraron en el Proceso de aprendizaje. Se puede observar que en la ejecuciòn de la prueba con 20 repeticiones la foto 25 genera un resultado positivo, sin embargo, al ejecutar nuevamente todo el grupo de fotos que no ingresaron al aprendizaje, considerando 30 repeticiones el sistema logra rechazar todo el grupo de fotos, inclusive la foto 25 que en la ejecuciòn anterior diò el resultado como positivo.

Tabla 4.4: Prueba con fotos que no pertenecen al conjunto de muestras

Registro	Caso	Foto	20 Repeticiones	30 Repeticiones
1	1	21	3068,318363	2977,318363
2	2	22	5418,061848	4874,061848
3	3	23	5666,242555	5201,242555
4	4	24	4296,762035	4128,762035
5	5	25	6070,339598	5240,339598
6	6	26	5725,592523	4578,592523
7	7	27	3133,954625	2974,954625
8	8	28	5820,529973	2120,529973
9	9	29	5946,332885	4946,332885
10	10	30	2182,943004	1574,943004
11	11	31	5640,297437	3574,297437
12	12	32	5880,237467	4780,237467
13	13	33	2607,179187	2147,179187
14	14	34	2441,744264	2374,744264
15	15	35	5838,477677	3574,477677
16	16	36	5338,884929	3478,884929
17	17	37	5838,477677	4784,477677
18	18	38	2481,900978	2174,900978

4.1.3. TABLAS DE RESULTADOS DE IDENTIFICACIÓN

En la Tabla 4.5 se muestra la tabulación de los datos resultantes de las pruebas realizadas, con la muestra de fotos que ingresaron al Proceso de Aprendizaje, con la finalidad de analizar el nivel de eficiencia del algoritmo elaborado.

Para ello se ha subdividido por cantidad de repeticiones. Como resultado de la ejecución considerando 20 repeticiones, el sistema identificó como válidas 17 fotos de las 20 fotos de la muestra, 3 de ellas no las identificó (el resultado fue menor que 6000). En este caso, el nivel de eficiencia para la identificación de rostros fue de 85%.

Sin embargo, el resultado de la ejecución considerando 30 repeticiones, el sistema identificó como válidas el total de las 20 fotos de la muestra. En este caso, el nivel de eficiencia para la identificación de rostros fue del 100%.

Tabla 4.5: Resultado de pruebas de identificación para el grupo de aprendizaje.

N°	Cantidad de muestras	Cantidad de repeticiones	Identificados	No identificados	Eficiencia
1	20	20	17	3	85%
2	20	30	20	0	100%

En la Tabla 4.6 se muestra la tabulación de los datos resultantes de las pruebas realizadas, con las 18 fotos que no pertenecen al grupo de Aprendizaje, con la finalidad de analizar el nivel de eficiencia del algoritmo elaborado.

Para ello, se ha subdividido por cantidad de repeticiones. Como resultado de la ejecución considerando 20 repeticiones, el sistema identificó como válida una (01) foto de las 18 que no pertenecen al grupo de Aprendizaje. En este caso, el nivel de eficiencia para la identificación de rostros fue de 94%.

Sin embargo, el resultado de la ejecución considerando 30 repeticiones, el sistema no identificó como válidas el total de las 18 fotos que no pertenecen al grupo de Aprendizaje. En este caso, el nivel de eficiencia para la identificación de rostros fue del 100% debido a que logró rechazar las 18 fotos en su totalidad.

Tabla 4.6: Resultado de pruebas de identificación con un grupo que no forman parte de grupo de aprendizaje

N°	Cantidad de muestras	Cantidad de repeticiones	Identificados	No identificados	Eficiencia
1	18	20	1	17	94%
2	18	30	0	18	100%

CAPÍTULO V

DISCUSION DE RESULTADOS

5.1. ANÁLISIS DE LOS RESULTADOS

En la Figura 4.1 se evidencia el crecimiento de la curva de Evolución del aprendizaje

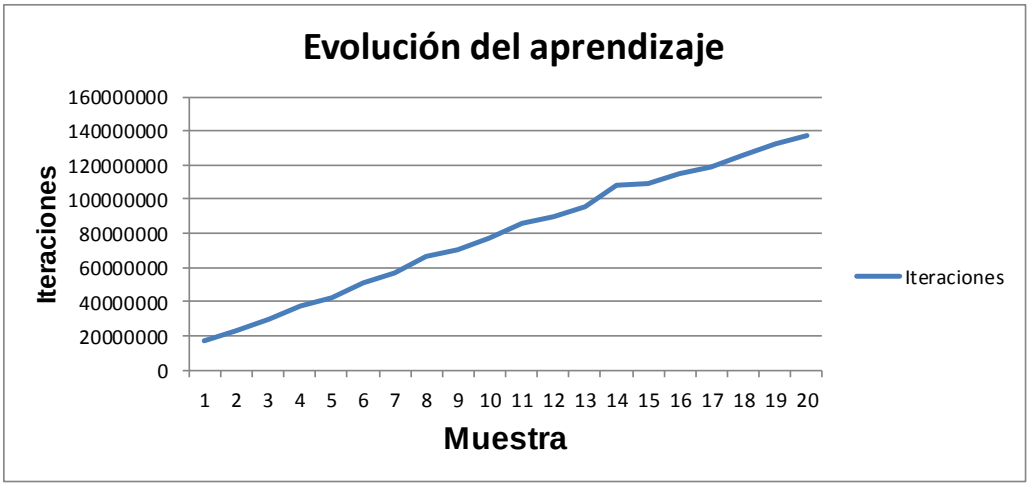


Figura 4.1: Evolución del aprendizaje

En la Figura 4.2 se observa que la curva de evolución de aprendizaje crece con respecto al mayor número de repeticiones que se realicen.

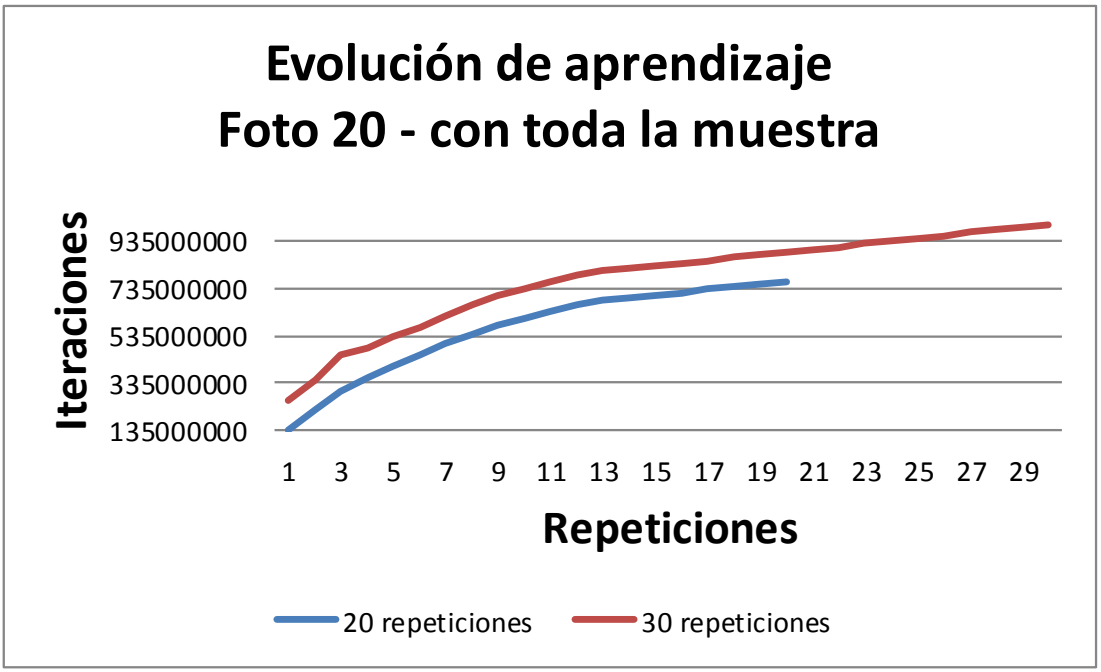


Figura 4.2: Evolución de aprendizaje en la repetición con todo el conjunto de muestras (caso Foto 20)

En la Figura 4.3 se muestra gráficamente los resultados de las ejecuciones de c/u de las fotos de la muestra con respecto al número de repeticiones realizadas. Se puede observar que los valores se incrementan cuando el número de repeticiones es mayor.

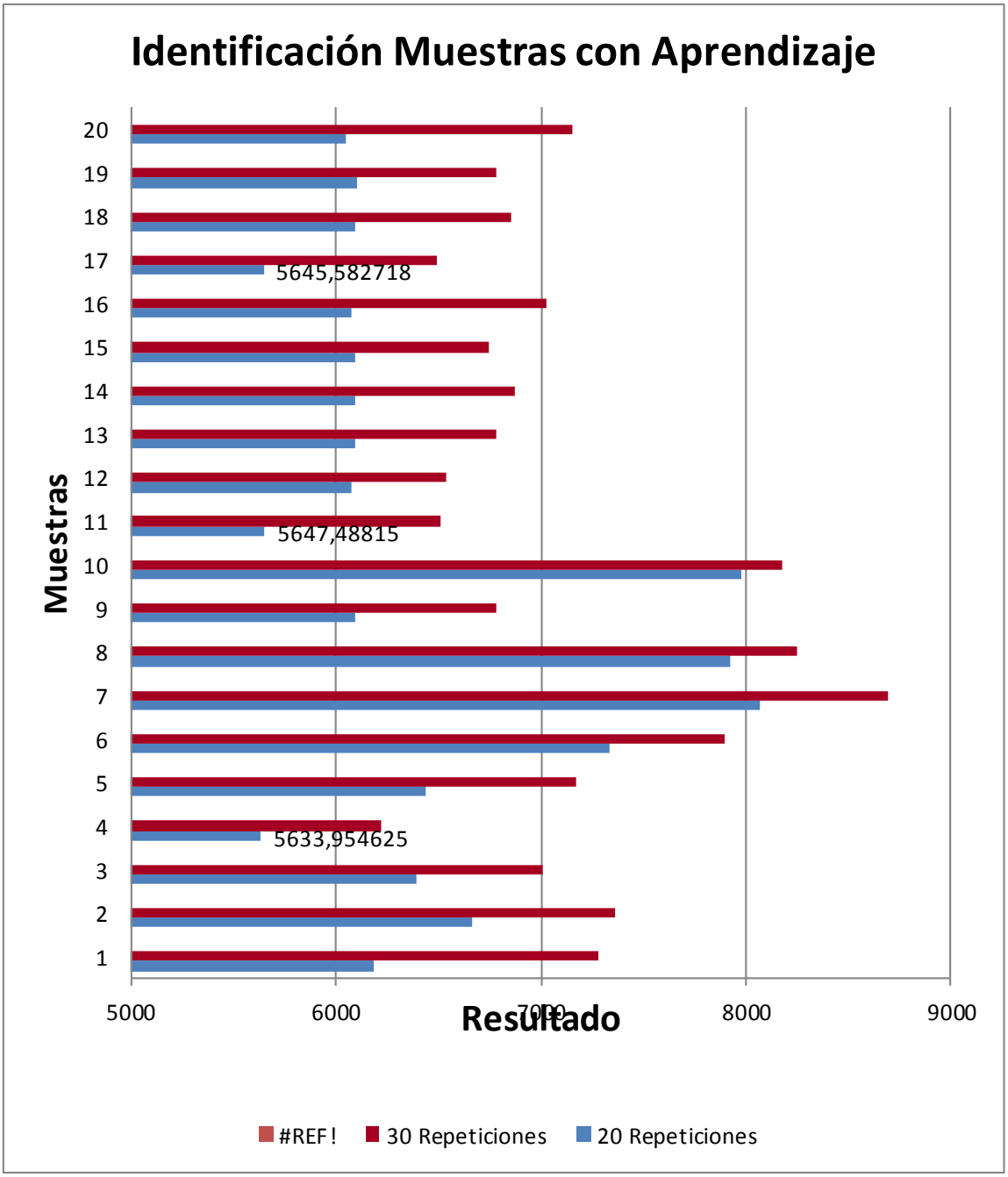


Figura 4.3:: Resultados de identificación con el grupo de aprendizaje

En la Figura 4.4 se muestra gráficamente los resultados de las pruebas realizadas, con la muestra de fotos que ingresaron al Proceso de Aprendizaje.

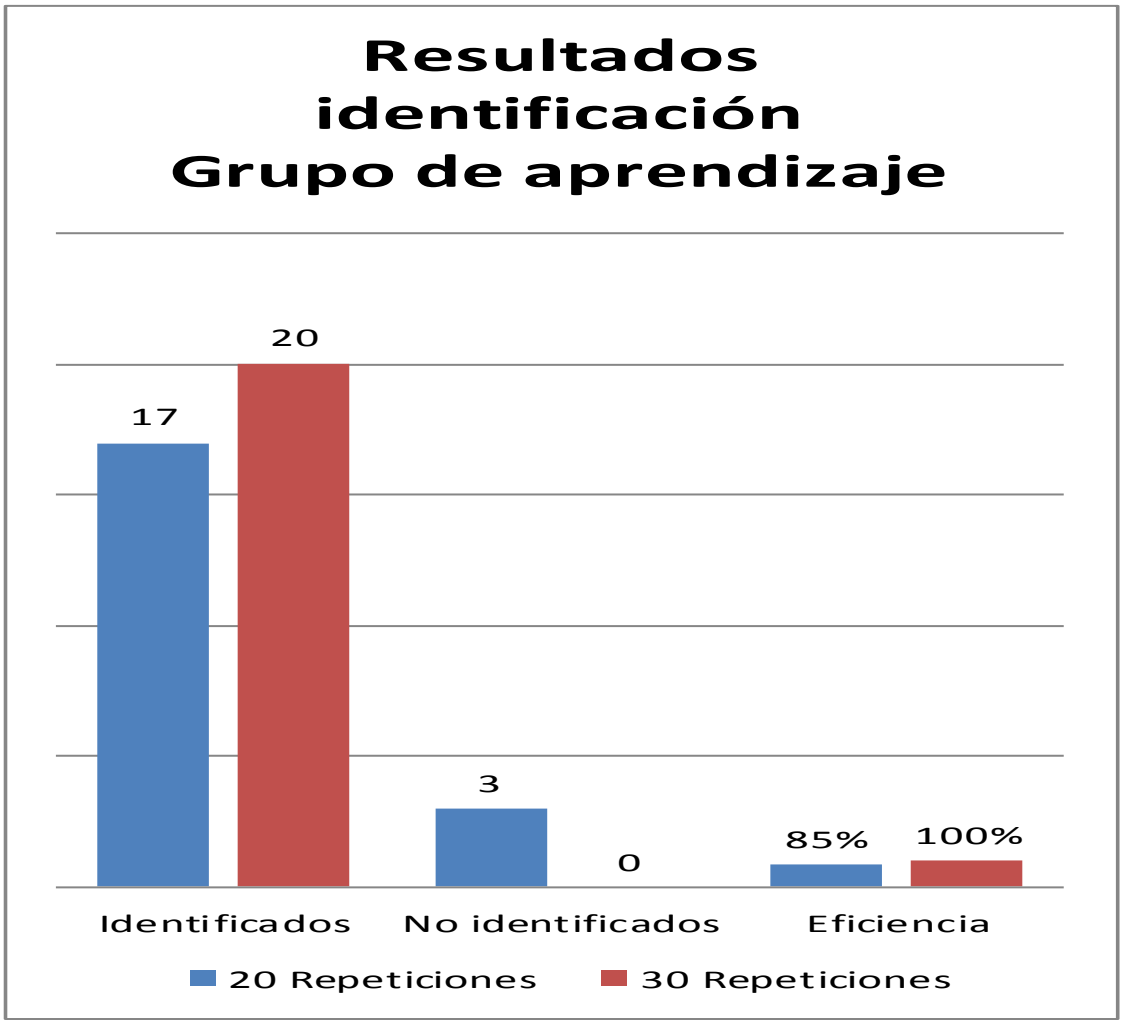


Figura 4.4: Resultado de pruebas de identificación para el grupo de aprendizaje

En la Figura 4.5 se muestra gráficamente los resultados de las pruebas realizadas, con la muestra de fotos que no ingresaron al Proceso de Aprendizaje.

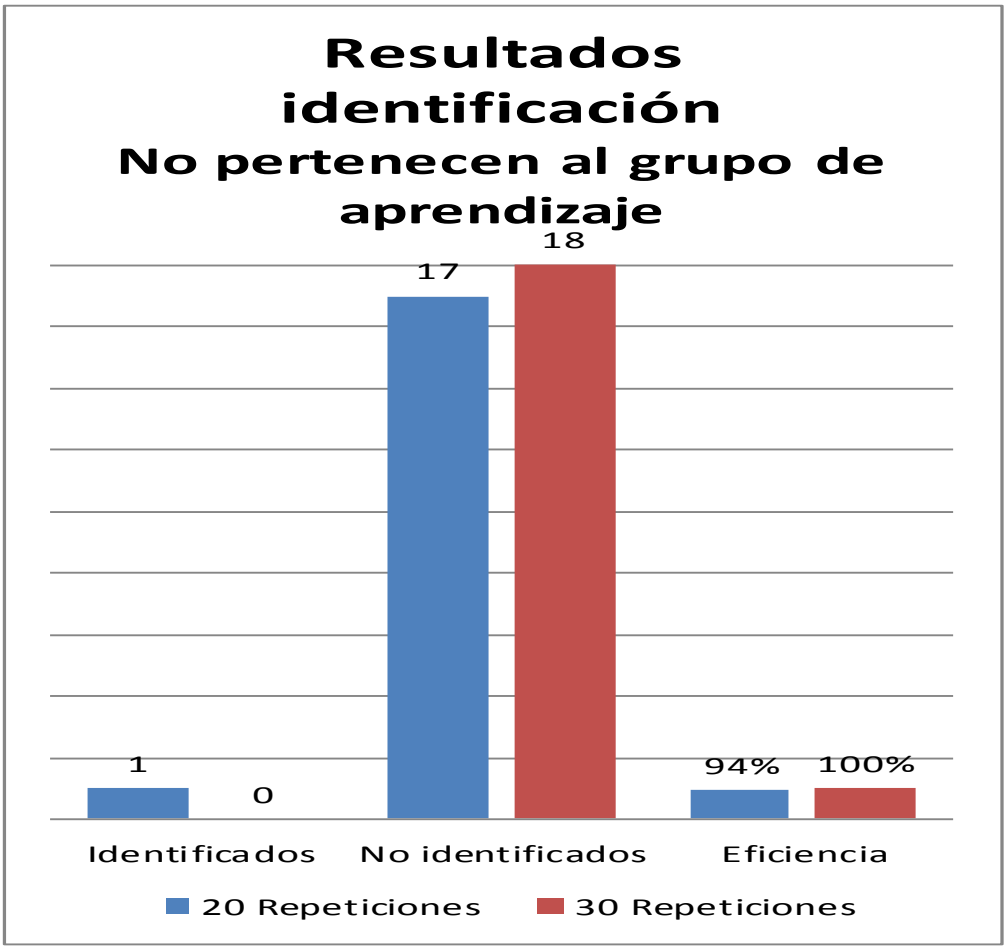


Figura 4.5: Resultado de identificación de fotografías que no pertenecen al grupo de aprendizaje

5.2. VERIFICACIÓN O CONTRASTACIÓN DE HIPÓTESIS

- Es posible diseñar la captura de imágenes utilizando hardware y software para la carga de datos de individuos, por medio de una webcam, y software base para grabarlo en formato jpg.
- Con el resultado del presente trabajo se confirma que es posible diseñar un prototipo para el reconocimiento de rostros utilizando redes neuronales artificiales.

- Por medio del proceso de aprendizaje es posible diseñar un prototipo que identifique personas utilizando redes neuronales artificiales para el aprendizaje de las imágenes capturadas.
- El prototipo fue desarrollado haciendo uso de algoritmos de redes neuronales artificiales los cuales permiten la identificación de personas mediante el reconocimiento de rostros.
- La eficiencia de identificación para el grupo de fotos de personas con las cuales se realizó el aprendizaje de la red neuronal artificial en 20 repeticiones es de 85% de identificación y con 30 repeticiones con todo el grupo es del 100%.
- La eficiencia de no identificación para un grupo de fotos de personas con las que no se realizó el aprendizaje de la red neuronal artificial en 20 repeticiones es de 94% y para 30 repeticiones es de 100%.

CONCLUSIONES

La segmentación y extracción del rostro es un paso muy importante ya que esto elimina mucha información innecesaria al momento del entrenamiento mejorando la eficiencia de la red.

El algoritmo de aprendizaje ADALINE es mucho más eficiente que la red perceptrón de una capa y converge más rápido.

Se comprobó que no es necesario usar la correlación cruzada para centrar la imagen debido a la plasticidad de la red, esto es muy importante ya que permite una codificación más eficiente evitando líneas de programación innecesarias, y tiempos de uso de procesador.

Para la actualización se usa el algoritmo de Backpropagation modificado, en el cual en lugar de usar el error medio cuadrático se usa el error instantáneo de la RNA.

Se ha realizado un programa en Visual C++ para el aprendizaje de la red neuronal artificial, y se han realizado las pruebas correspondientes para determinar la convergencia del algoritmo.

Fácil inserción dentro de la tecnología existente. Debido a que una red puede ser rápidamente entrenada, comprobada, verificada y trasladada a una instalación de hardware de bajo costo, es fácil insertar RNA para aplicaciones específicas dentro de sistemas existentes (chips, por ejemplo). De esta manera, las redes neuronales se pueden utilizar para mejorar sistemas de forma incremental, y cada paso puede ser evaluado antes de acometer un desarrollo más amplio.

La capacidad de las redes neuronales radica en su habilidad de procesar información en paralelo (es decir, procesar múltiples pedazos de datos simultáneamente). Desafortunadamente, las máquinas hoy en día son serie – sólo ejecutan una instrucción a la vez. Por ello, modelar procesos paralelos en máquinas serie puede ser un proceso que consuma mucho tiempo. Como todo en este día y época, el tiempo es esencial, lo que a menudo deja las redes neuronales fuera de las soluciones viables a un problema.

Otros problemas con las reglas neuronales son la falta de reglas definitorias que ayuden a construir una red para un problema dado. Existen muchos factores a tomar en cuenta: el algoritmo de aprendizaje, la arquitectura, el número de neuronas por capa, el número de capas, la representación de los datos.

Con los resultados obtenidos se puede concluir que la eficiencia del sistema depende de la cantidad de repeticiones que se realiza en el aprendizaje.

El tiempo de aprendizaje de una fotografía toma menos de dos minutos. El tiempo de funcionamiento de una fotografía toma menos de un minuto.

RECOMENDACIONES

Propiciar en los centros de estudios el crecimiento del desarrollo de estos tipos de algoritmos los cuales pueden ser utilizados para temas de seguridad.

El algoritmo Adaline también puede ser desarrollado en otros lenguajes de programación diferentes a Visual C++.

El prototipo desarrollado puede también ser diseñado en dispositivos móviles con las adecuaciones de los lenguajes de programación adhoc para este tipo de hardware.

A pesar de que las redes neuronales artificiales no constituyen un área nueva de conocimiento, consideramos que todavía en el país no se ha estado prestando la atención que debería, teniendo en cuenta las ventajas que presenta sobre otras técnicas para la solución de cierto tipo de problemas descritos en el presente trabajo.

BIBLIOGRAFÍA

- [Anderson 7] James A. Anderson.(2007), redes neuronales. 1ra Ed. México.Alfaomega

- [Barro 9] S. Barro y J. Mira (editores). Computación Neuronal. Servicio de Publicaciones de la Universidad de Santiago de Compostela (1995).

- [Bonifacio 1] Bonifacio Martin Del Brío- Alfredo Sanz Molina (2006), Redes Neuronales y Sistemas Borrosos. 3ra Ed. España.Ra-ma

- [Deboeck 10] Deboeck, Guido J. Pattern recognition and prediction with self-organizing maps and ork /AIabtext.html supporting software review: visualization through viscovery [en línea]. <http://www.gordiand-knot.com >>. Gordian Institute Electronic NewsLetter. [Consulta: 20 de febrero 2000.]

- [Freman 5] Freman- Skapura (1997), redes neuronales, algoritmos, aplicaciones y técnicas de programación. 1ra ED. Mexico. Addison-Wesley

- [Haykin 4] Haykin, S. (1994). Neural networks: A comprehensive foundation. New York. Prince_Hall.

- [Hilera 8] José R.Hilera - Víctor J. Martínez (1995). Redes neuronales artificiales: fundamentos, modelos y aplicaciones. 1ra Ed. España. RA-MA.

- [Isasi 2] Pedro Isasi Viñuela, Ines M. Galván León (2004), Redes Neuronales Artificiales un enfoque práctico. 1ra Ed. Madrid. Pearson Prentice Hall.

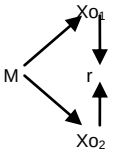
- [López 3] Raquel Flórez López, José Miguel Fernández Fernández (2008) redes neuronales artificiales, 1Ra Ed. Netbiblo.

- [StatSoft 6] StatSoft, Inc. Neural Networks. Statistica for Windows [Computer program manual]. Tulsa, Oklahoma: StatSoft, Inc. <<http://www..statsoft.com>>. [Consulta: diciembre del 2000].

ANEXOS

ANEXO 1
MATRIZ DE CONSISTENCIA DEL PROYECTO DE INVESTIGACIÓN

TÍTULO: Uso de redes neuronales artificiales para la identificación de personas mediante el reconocimiento de rostros de la Municipalidad de La Victoria-Lima 2015

PROBLEMA	OBJETIVOS	HIPÓTESIS	VARIABLES	DIMENSIONES	INDICADOR	INSTRUMENTOS	METODOLOGÍA
GENERAL: ¿En qué medida las redes neuronales artificiales permiten la identificación de personas mediante el reconocimiento de rostros del Padrón de beneficiarios en los Programas Sociales de la Municipalidad de La Victoria, Lima - 2015?	**GENERAL:** Diseñar un prototipo con el uso de redes neuronales artificiales que permitan la identificación de personas mediante el reconocimiento de rostros de la Municipalidad de La Victoria-Lima 2015	**GENERAL:** Las redes neuronales artificiales permiten la identificación de personas mediante el reconocimiento de rostros.	**INDEPENDIENTE:** Redes neuronales artificiales	Diseño del módulo de Captura de imágenes de rostros	Especificación técnica del módulo captura de imágenes	Software Visual C++	*** Población:** Fotografías: 38 Número de personas:38 *** Muestra:** Fotografías:20 Número de personas:20 *** Esquema del proyecto** *** Tipo de investigación:** Cuasi-experimental, prospectivo, longitudinal, analítico *** Diseño** No experimental, transeccional y correlacional Cuyo Diagrama es:  **TÉCNICAS A UTILIZAR** 1. Para acopio de datos: Fotografías 2. Instrumento de recolección de datos: Base de datos-sql 3. Para el procesamiento de datos Codificación en java 4.- Técnicas para el análisis e interpretación de datos: Estadística descriptiva e inferencial para cada variable. 5. Para la presentación de datos: Cuadros, tablas estadísticas y gráficos. 6. Para el informe final: Esquema propuesto por la Universidad
				Diseño del prototipo de redes neuronales artificiales	Especificación técnica del diseño del prototipo de redes neuronales		
				Diseñarr prototipo	Reportes de resultados de las ejecuciones		
ESPECÍFICOS: • ¿De qué manera se capturan imágenes de rostros? • ¿De qué manera se reconocen rostros utilizando redes neuronales artificiales? • ¿En qué medida se identifican personas utilizando redes neuronales artificiales?	**ESPECÍFICOS :** • Instalar el hardware y software para capturar imágenes de rostros • Diseñar un prototipo para el reconocimiento de rostros utilizando redes neuronales artificiales • Diseñar un prototipo que identifique personas utilizando redes neuronales artificiales	**ESPECÍFICAS :** • Es posible diseñar la captura de imágenes utilizando hardware y software • Es posible diseñar un prototipo para el reconocimiento de rostros utilizando redes neuronales artificiales • Es posible diseñar un prototipo que identifique personas utilizando redes neuronales artificiales	**DEPENDIENTE:** Identificación de personas mediante el reconocimiento de rostros	Número de personas identificadas	Eficacia en el reconocimiento de rostros	Observacional	
				Duración del proceso de identificación	Tiempo de ejecución del prototipo		

ANEXO 2
OPERACIONALIZACIÓN DE VARIABLES

VARIABLES	DEFINICIÓN CONCEPTUAL	DEFINICIÓN OPERACIONAL	DIMENSIONES	INDICADOR	ITEM	INSTRUMENTOS VALOR FINAL	TIPO DE VARIABLE	ESCALA
VI = V1 Redes neuronales artificiales	*Las redes neuronales artificiales (RNA) son modelos matemáticos inspirados en el funcionamiento de las redes neuronales biológicos, especialmente en la forma de funcionamiento del cerebro humano.*	Las RNA contienen un gran número de elementos simples de procesamiento llamados nodos o neuronas que están organizados en capas. Cada neurona está conectada con otra neurona mediante enlaces de comunicación, cada una de las cuales tiene asociado un peso. En los pesos se encuentra el conocimiento que tiene la RNA acerca de un determinado problema.	Diseño del módulo de Captura de imágenes de rostros	Especificación técnica del módulo captura de imágenes	¿Cómo se captura las fotografías de los individuos? ¿Cómo se almacena las fotografías de los individuos?	Software Visual C++	cualitativa	Nominal
			Diseño del prototipo de redes neuronales artificiales	Especificación técnica del diseño del prototipo de redes neuronales	¿Cómo funciona el método Adaline? ¿Cómo se diseña la capa de entrada? ¿Cómo se diseña la capa oculta? ¿Cómo se diseña la capa de salida?			
			Diseñar prototipo	Reportes de resultados de las ejecuciones	¿Cuáles reportes genera para mostrar el porcentaje de error en la identificación? ¿Cuáles reportes genera para mostrar la relación de individuos identificados? ¿Cuáles reportes genera para mostrar la relación de individuos no identificados?			
VD = V2 Identificación de personas mediante el reconocimiento de rostros	Es una acción para autenticar a cada individuo.	Es el proceso para autenticar si la persona que dice ser es la que se encuentra en el registro de beneficiarios de los Programas Sociales organizados por la Municipalidad de La Victoria – Lima.	Número de personas identificadas	Eficacia en el reconocimiento de rostros	¿Cuántas personas identifican de un total de 38 fotografías? ¿Cuántas personas no se identifican de un total de 38 fotografías?	Observacional	cuantitativa	Nominal
			Duración del proceso de identificación	Tiempo de ejecución del prototipo	¿Cuánto tiempo demora la ejecución del prototipo para identificar un individuo?			

ANEXO 3: Programa de cálculo de la red neuronal artificial.

```
private double CalculoRed(int nred)
    {
      for (int n = 0; n < nred; n++)
      {
        if (rna[n].capa == 1)
        {
          rna[n].n = 0;
          for (int ne = 0; ne < rna[n].m; ne++)
          {
            rna[n].n = rna[n].n + entrada[ne] * rna[n].peso[ne];
          }
          rna[n].a = rna[n].n * rna[n].k;
        }
        else
        {
          int ni = 0;
          for (int cc = 1; cc < rna[n].capa - 1; cc++)
          {
            ni = ni + sred[cc];
          }
          rna[n].n = 0;
          for (int ne = 0; ne < rna[n].m; ne++)
          {
            rna[n].n = rna[n].n + rna[n].peso[ne] * rna[ni].a;
            ni++;
          }
          rna[n].a = rna[n].n * rna[n].k;
        }
      }
      return rna[nred - 1].a;
    }
```

ANEXO 4: Programa de construcción de sistema de coordenadas

```
private void coordenadas(Bitmap imagen, int Xc, int Yc, int lx, int ly, float my, float
mx)
      {
         int j;
         int i,dx,dy;
         float ix,iy;
         Color color;
         int ndx, ndy;
         ndx = 10;
         ndy = 5;
         ix=mx/ndx;
         iy=my/ndy;
         dx=lx/ndx;
         dy=ly/ndy;
         //pDC->TextOut(Xc+lx/4-30,Yc-ly,titulo);
         //setusercharsize(1,1,1,1);
         //pDC->TextOut(Xc+lx-20,Yc-15,"Muestras");
         //setusercharsize(2,3,2,3);
         //myPen.CreatePen(PS_DOT,1,RGB(0,255,0)); // crear el pincel líneas
punteadas
         //pOldPen=pDC->SelectObject(&myPen);
         color=Color.FromArgb(0, 0, 0);
         line(imagen, Xc, Yc, Xc + lx, Yc, color);
         line(imagen, Xc, Yc - ly, Xc, Yc + ly, color);

            /*pDC->MoveTo(Xc,Yc+dy*i);
            pDC->LineTo(Xc+2,Yc+dy*i);
                              sprintf(tex,"%6.2f",iy*i);
                              pDC->TextOut(Xc+3,Yc+dy*i-5,tex);*/

         for (i = 1; i <= ndy; i++)
         {
            line(imagen, Xc, Yc - dy * i, Xc + 2, Yc - dy * i, color);
         }
         for (i = 1; i <= ndy; i++)
         {
            line(imagen, Xc, Yc + dy * i, Xc + 2, Yc + dy * i, color);
         }
         for (i = 1; i <= ndx; i++)
         {
            line(imagen, Xc + dx * i, Yc, Xc + dx * i, Yc + 2, color);
         }
          /*pDC->SelectObject(pOldPen);
          myPen.DeleteObject();
          pDC->SelectObject(pOldFont);
          myFont.DeleteObject();
                              myPen.CreatePen(PS_SOLID,1,RGB(0,0,255));
                              pOldPen=pDC->SelectObject(&myPen);
                              pDC->MoveTo(Xc,Yc);
```

```
        pDC->MoveTo(Xc,Yc-ly);
        pDC->LineTo(Xc,Yc);

        pDC->MoveTo(Xc,Yc);
        pDC->LineTo(Xc+lx,Yc);
        pDC->MoveTo(Xc,Yc-ly);
        pDC->LineTo(Xc-2,Yc-ly+6);
        pDC->MoveTo(Xc,Yc-ly);
        pDC->LineTo(Xc+2,Yc-ly+6);

        pDC->MoveTo(Xc+lx,Yc);
        pDC->LineTo(Xc+lx-6,Yc+2);
        pDC->MoveTo(Xc+lx,Yc);
        pDC->LineTo(Xc+lx-6,Yc-2);*/
}
```

ANEXO 5: Estructura del Programa General

Elementos verificables	Enunciado
• **Pregunta clave**	¿En qué medida
• **Variable "X"**	**Las Redes neuronales artificiales**
• **Enlace o relacionante**	permiten la
• **Variable "Y"**	**Identificación de personas mediante el reconocimiento de rostros**
• **Muestra/Población**	Del padrón de beneficiarios
• **Ámbito social (accesible)**	en los Programas Sociales
• **Ámbito geográfico (objetivo)**	de la Municipalidad de La Victoria, Lima
• **Tiempo**	2015