COMPUTACIÓN INTELIGENTE Y ESTADOS EMOCIONALES

Franyelit Suárez
Luis Rosales
Ángel Lezama

Editorial: AutanaBooks

Computación Inteligente y Estados Emocionales

Computación Inteligente y Estados Emocionales

Diseño de portada: Alan Druet
Diagramación: Franyelit Suárez
frangelits@gmail.com

Primera edición
Quito-Ecuador 2019

INDICE

Capítulo 1:

Computación Inteligente en el contexto actual

Cuando Alan Turing llevó a cabo su conferencia ante los miembros de la National Physical Laboratory de Londres en el año 1947 , la ya para entonces vieja pregunta ¿Pueden pensar las máquinas? fue abordada desde un punto de vista tan distinto que dio un impulso decisivo para la germinación de un nuevo campo en la aún reciente disciplina de la informática: La Inteligencia Artificial.

Turing vislumbró mejor que cualquiera de su tiempo que la relación entre máquina y pensamiento no implicaba una dicotomía en sí misma; pero también comprendió que dicha relación debía establecerse de una manera precisa, unívoca, que no admitiera elucubraciones. El concepto de máquina en primera instancia y para el uso que quería acotar no le causaba mayores complicaciones al inglés. Ya en 1936 y para responder al problema del Entscheidungsproblems (problema de decisión enunciado por Leibniz y formalizado por el matemático David Hilbert) había desarrollado el concepto de Máquina Universal de Turing , que detalla la idea de un dispositivo computacional lecto-escritor que, mediante el uso de un lenguaje formal sobre una cinta finita de casillas tabuladas, puede realizar cualquier proceso que sea computable. Este concepto, que describe al primer computador moderno [6] es la base fundacional de la informática como se conoce actualmente, además es muy avanzado debido a que refleja aspectos que le acercan a la manera en que el ser humano procesa ciertos aspectos naturales de decisión, ¿No es el lenguaje un conjunto de símbolos que se utilizan según un patrón establecido? ¿No es el uso del lenguaje una de las formas que empleamos y que denotan inteligencia y que además nos diferencia de los otros animales?; pero pensar es un acto reflexivo que va más allá. Definir pensar no era sencillo en su época ni ahora, y sabía que debía tratarse con cuidado.

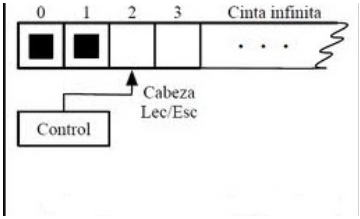


Fig 1. Máquina Universal de Turing

Para lograr un uso sólido de la abstracción de pensar, Turing ofreció una alternativa para ello: El juego de Imitación. Dicho juego es la base de lo que se conoce en nuestros días como el Test de Turing y cuyo objetivo es lograr mediante un chat a distancia entre tres jugadores decidir si se está tratando con un ente inteligente no humano. El juego se estructura de la siguiente manera: los jugadores, por colocar un escenario, se ubican dentro de una cámara con dos estancias que se comunican a través de una pared o tabique donde se practica una ventanilla que es por donde se intercambiaran los mensajes mecanografiados. En la primera instancia está el interrogador y en la segunda dos jugadores (supongamos un hombre y una mujer). Si mediante preguntas y respuestas mecanografiadas a través de la ventanilla, el interrogador aislado no puede distinguir quién de los interlocutores es una computadora, se dice que la máquina ha mostrado síntomas de inteligencia. Nos encontramos ante el primer intento formal estructurado de conocer cuán inteligente puede ser una máquina que es capaz de computar problemas de decisión.

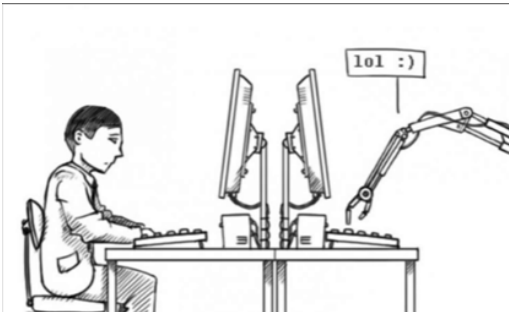


Fig. 2. Test de Turing simplificado.

Turing intuía que el potencial de llegar a un punto tal de desarrollo en la

capacidad de almacenamiento de las máquinas, que podrían guardar instrucciones con los matices que diferencian la inteligencia de la acción puramente mecánica o programada, parafraseando a Arthur C. Clark y asumiendo un punto referencial positivista, diríamos que si una máquina realiza acciones inteligentes y toma decisiones inteligentes es porque es inteligente, y dicha inteligencia en acción no se distingue de la inteligencia humana en acción.

Metas remotas

Estos conceptos entusiasmaron a muchos y la percepción general era que en muy poco tiempo desde aquellos deslumbrantes avances de los años 50, la inteligencia artificial coexistiría con el ser humano en un tiempo muy corto.

Desafortunadamente, en la conferencia de Dartmouth de 1956 se crearon visiones triunfalistas para un intervalo de diez años que no se cumplieron, esto produjo el abandono masivo de las investigaciones durante los quince años siguientes. La realidad resultó ser mucho más compleja de lo que las inocentes expectativas esperaban. A pesar de los increíbles avances en la capacidad de almacenamiento de las computadoras , el desarrollo del software y el hardware resultaban insuficientes para el manejo ingente de datos necesarios para equiparar o simular el comportamiento inteligente. La inteligencia artificial entonces pasó a ser una rama de estudio con un consumo ingente de recursos cuyos resultados estaban muy lejos de ser alentadores y cuyo abordaje no respondía a metodologías estructuradas que permitieran marcar un rumbo concreto en las investigaciones . En resumen, se trazaron tales expectativas en tan corto tiempo que los resultados escasos desalentaron la continuidad de las investigaciones y por un lapso de tiempo considerable pasó al olvido, hundiéndose parcialmente en la oscuridad de los años.

Y dijo Frank… que exista el perceptrón

Frank Rosenblatt fue un investigador proveniente del mundo de la filosofía y las letras; sin embargo parte de su carrera como investigador la realizó en el Laboratorio Cornell Aeronáutico de Búfalo, New York. Allí, en 1958, inspirado por los trabajos de Mcculloch y Pitts de 1947 mientras era el encargado de la sección de sistemas cognitivos, realizó las primeras investigaciones de lo que posteriormente se conocería como el perceptrón, la detonación de salida para las denominadas redes neuronales artificiales. La red neuronal básica es la mostrada abajo en la Fig. 3.

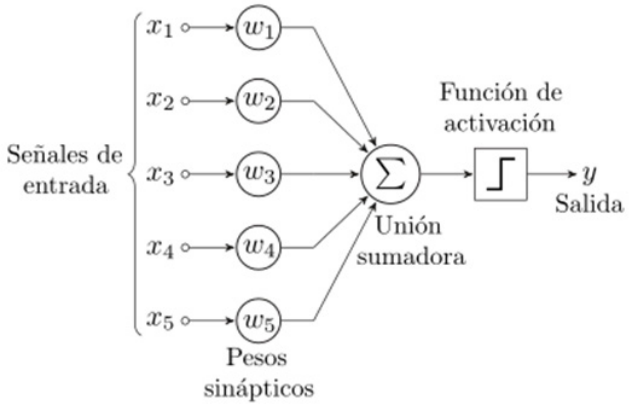


Fig. 3. Modelo de Perceptrón básico.

Para entenderlo mejor debemos conocer un poco la anatomía de una neurona y su fisiología. Las neuronas son las células más especializadas que existen, han perdido la capacidad de realizar cualquier otra función distinta a la que ejecutan, incluso no pueden dividirse.

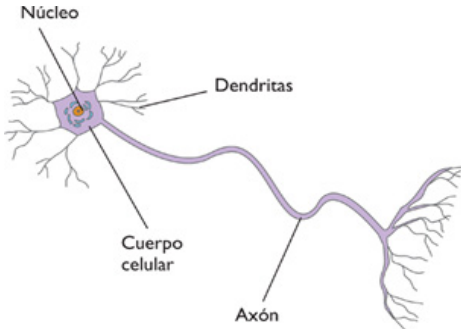


Fig. 4. Esquema básico de una neurona.

Como se puede apreciar en la Fig. 4. Las neuronas está constituida, a grandes rasgos por varios elementos. Las Dendritas que son las terminales nerviosas de la neurona, ellas permiten principalmente enlazar una neurona con otra, transmitiendo el impulso nervioso enviado desde el axón de otra neurona hasta su soma. El Soma o pericarion es el cuerpo de la neurona y es donde se ubica el núcleo de la misma. El axón es una prolongación de la neurona que sirve para la conducción del impulso nervioso y también para el transporte de sustancias.

Las neuronas reciben y procesan la información recibida, esto ocurren en las dendritas y en el axón de la neurona. Los impulsos nerviosos que recibe una neurona pueden ser de dos tipos: excitatorias o inhibitorias. por lo general las neuronas poseen ramificaciones de dendritas que las conectan con otras neuronas; en ellas se reciben miles de impulsos nerviosos; si la sumatoria de impulsos de tipo excitatorio logra activar la respuesta de la neurona, la misma libera un impulso nervioso o potencial de acción que viaja a través del axón hasta las dendritas terminales que se conectan con otras neuronas objetivo o blanco mediante la conexión con las dendritas de otra neurona o directamente en glándulas o músculos del cuerpo.

Haciendo el seguimiento de lo dicho anteriormente, una neurona:

1. Recibe señales nerviosas (información).

2. Realiza una sumatoria integral de las señales que recibe (para conocer si se debe o no liberar un impulso nervioso o potencial de acción).

3. Comunicar impulsos nerviosos a otras neuronas o directamente a glándulas y músculos.

4. Tomando la información descrita es posible comprender el funcionamiento de un perceptrón.

El perceptrón se inspira en una red neuronal para buscar un sistema que posea el comportamiento de una neurona. Como se puede observar en la Fig. 3. el perceptrón está formado por un conjunto de señales de entrada que asemejan a los impulsos nerviosos que reciben las neuronas mediante sus dendritas. A cada valor de entrada se le asocia un peso que equivale a la fuerza inhibitoria o excitatoria del impulso que recibe y se realiza la suma de impulso para conocer mediante la función umbral si el resultado de todos los valores de entrada permite que la función se active o no.

De Nuevo en las Tinieblas

En el año 1969, en su libro Percetrons, Marvin Lee Minsky y Seymour Papert demostraron matemáticamente que los perceptrones eran incapaces de realizar correctamente la función lógica XOR, lo que llevó a la comunidad científica, dado el tono impreso en la conclusiones del libro, que el modelo de la redes neuronales estaba agotado, a abandonar masivamente las investigaciones que se estaban desarrollando en esta área. A pesar de ello, algunos científicos entre los que se pueden mencionar a Stephen Grossberg y a Gail Carpentier siguieron experimentando con redes neuronales, ellos presentaron el modelo neuronal ART (Adaptive Resonance Theory) mostrado en la Fig. 5.

Otros científicos que desarrollaron teoría basadas en redes neuronales en la década de los setenta fueron Jim Anderson y Geoff Hinto, quienes organizaron el primer evento neo conexionista en 1979 y al cual asistieron David Rumelhart, James Lloyd Mc Clelland, Geoff Hinton, Jim Anderson, Jerry Feldman y Terry Sejnowski.

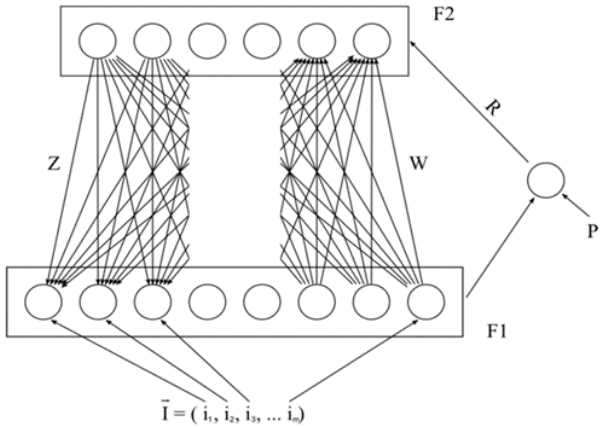


Fig.5. Red Neuronal ART.

Ya para el año 1986, la aparición del libro de James McClelland y David Rumelhart publicado en dos volúmenes Parallel Distributed Processing: Explorations in the Microstruture of Cognition, impulsó nuevamente a la palestra científica mundial el estudio de la redes neuronales artificiales. Esto se debió a que mostraron las ventajas y desventajas de ésta técnica frente a otros sistemas de procesamiento de información e introdujeron el concepto de RNA multicapa.

Perceptrón Multicapa

Como hemos podido apreciar, las limitaciones del perceptrón simple empujaron la búsqueda de una generalización que permitiera la posibilidad de resolver mediante la técnica de RNA problemas más complejos de los que podía solucionar un perceptrón simple, como lo eran aquellos que involucran la separación no lineal de datos. El Perceptrón multicapa viene siendo

una generalización del perceptrón simple. a pesar de que Minsky y Papert mostraron en 1969 que combinando varios perceptrones simples se podía alcanzar soluciones a problemas no lineales; sin embargo solo especularon con la posibilidad -incluso introdujeron el concepto de capa oculta- de que dicha estructura permitiera abordar problemas más complejos sin indicar cómo se procedería con los pesos y su transferencia desde la capa de entrada a las capas ocultas, pero no podría ser como en el perceptrón simple ya que la regla de aprendizaje del perceptrón simple no funciona en esta nueva estructura.

Pero estas ideas inspiraron los estudios de Rummelhart, Hiton y Wilians, que en 1986 como se mencionó ya en un párrafo anterior, crearon una algoritmo de retropropagación de errores medidos en la salida de la red hacia las neuronas ocultas que es el origen de la denominada regla delta generalizada.

Múltiples autores han demostrado de manera independiente que el perceptrón multicapa en un aproximador universal, esto se entiende como la capacidad de hacer que cualquier función continua en una espacio real cualquiera (Rn) pueda modelarse o aproximarse mediante un perceptrón multicapa con al menos una capa oculta de neuronas. Esto le permite al perceptrón multicapa ser muy útil al momento de aproximar relaciones no lineales entre datos de entrada y salida. esta cualidad le ha permitido ser la arquitectura más utilizada en la resolución de problemas donde su naturaleza de aproximador universal se combina con su fácil implementación, esto sin de mejorar potencia de cálculo y calidad de resultados en sus áreas de aplicación. Todo esto no quiere decir que el perceptrón multicapa no posea limitación importante. Una de ellas es el tiempo computacional extenso que necesita para aprender a resolver problemas que involucren un gran número de variables. La difícil tarea de poder realizar un análisis teórico de la red debido la presencia de componentes no lineales y también motivados por el alto grado de conexiones que se realizan en sus diversas capas.

Estructura de una RNA multicapa

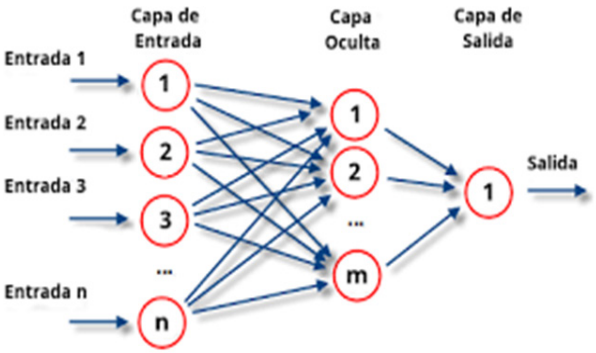


Fig. 6. Modelo de Perceptrón Multicapa, en este caso particular con tres capas; entrada, intermedia y salida.

La arquitectura de un perceptrón multicapa se caracteriza por tener neurona agrupadas en al menos tres niveles diferenciados. Estos niveles se describen a continuación:

Capa de entrada: Las neuronas de esta capa no accionan como neuronas propiamente dichas, sino que operan como receptoras de señales o patrones que provienen del exterior y los retransmiten a todas las neuronas de la siguiente capa.

Capa de salida: La última capa opera como salida de la red, donde se entrega los resultados de las acciones llevadas a cabo por la red sobre los datos o patrones de entrada.

Capas Ocultas: realizan el procesamiento no lineal de los datos o patrones recibidos en la entrada de la red.

En la Fig. 6 se observa que las conexiones del perceptrón multicapa se dirigen hacia adelante, esto recibe el nombre de redes alimentadas hacia delante o feedforward. Las conexiones entre las neuronas llevan asociadas un umbral que es tratada como otra conexión más a la neurona, cuyo valor es constante e igual a 1. Por lo general existe una conexión completa entre las neuronas de la capa siguiente y la anterior; se dice entonces que existe una red completamente conectada.

Entre las ventajas de la Redes Neuronales Artificiales tenemos:

Las Redes Neuronales Artificiales (RNA) son muy potentes a la hora de resolver problemas no lineales.

Los procesos de aprendizaje a los que se somete a las RNA les permiten la síntesis de algoritmos.

Permite a las personas que no poseen un conocimiento profundo de las matemáticas que describen a la tecnología, utilizarla. lo que se necesita es familiarizarse con los datos asociados al trabajo a desarrollar.

La adaptabilidad es una de las más destacadas características de las RNA. Puede suceder que algunos elementos en el procesamiento de la información fallen; sin embargo la red se mantendrá operando. esto la diferencia de los programas informáticos tradicionales.

Las desventajas de las redes neurales

Se necesita entrenar a las RNA por cada problema al cual le enfrentan. Se deben realizar una cantidad importante de pruebas para determinar la estructura que debe poseer la red. El proceso de entrenamiento es complejo y por tanto consume una cantidad importante de tiempo el hacerlo (tiempo computacional).

Se requieren cantidades ingentes de datos debido a la naturaleza de la red neuronal. Las RNA se entrenan y por ello requieren de un conjunto de datos elevado de entrada.

Desde el punto de vista de un observador externo que desee realizar cambios en la red, las RNA resultan un problema complejo. Esto se debe a que para entrenar a la RNA e incorporar un nuevo aprendizaje es necesario cambiar muchas iteraciones en diferentes unidades de la misma. Por tanto se hace necesario que la red sea una red distribuida con un esquema de aprendizaje determinado.

El amanecer de la inteligencia

Actualmente es posible afirmar que las máquinas pueden aprender, los avances alcanzados a través de la disciplina de Machine Learning así lo atestigua; pero ¿Realmente una máquina puede ser inteligente? Esta cuestión no tiene una respuesta incontestable. Es cierto que se ha logrado que las máquinas aprendan a realizar actividades específicas y que puedan explorar campos que antes estaban en completo dominio de los seres humanos como podría ser las disciplinas creativas o generativas, tales como las asociadas a las artes. La capacidad de abordar estos campos nuevamente colocan en la palestra hasta dónde puede llegar el aprendizaje de estas nuevas gene-

raciones de computadores; sin embargo seguimos tratando con lo que se denomina aprendizaje específico. Las máquinas se están volviendo expertas en determinados campos… pero no están desarrollando inteligencia. Sin duda aprenden, pero este aprendizaje se sigue circunscribiendo en estrechos márgenes del saber, aún la tarea de lograr un aprendizaje como la desarrollaría una persona sigue siendo tarea pendiente. En parte esta limitante está estrechamente relacionada con lo poco que se sabe de cómo el ser humano aprende. Ya la capacidad del software y del hardware ha llegado a tal punto del arte que es posible llegar a la capacidad teórica computacional de un cerebro humano, el problema ahora se dirige a la forma en que los seres humanos adquieren la inteligencia.

Esto se explica tomando en cuenta el poco desarrollo que se tiene en la actualidad en neurociencia. Los modelos desarrollados en inteligencia artificial siguen los caminos de lo poco que se conoce sobre este órgano, por tanto, las limitaciones de la inteligencia actual solo son el reflejo de las limitaciones del ser humano de conocer la manera en que funciona su cerebro.

Superando las limitaciones

A pesar del conocimiento apenas incipiente sobre la fisiología del cerebro, las aplicaciones de la inteligencia artificial (IA) está tomando cada día mayor participación en áreas estratégicas de cara al futuro en muchos frentes, entre ellos podemos mencionar:

1. Lingüística computacional
2. Medicina
3. Educación
4. Robótica
5. Finanzas
6. Marketing
7. Biología
8. Otras áreas aplicables

Lingüística computacional

Aunque el término no suene familiar ni su uso pueda considerarse extendido dentro de la sociedad; su estudio y desarrollo, aplicando técnicas y herramientas informáticas, permitirá desentrañar el funcionamiento de al-

gunos procesos humanos complejos y beneficiarnos de ese conocimiento para la instrumentación de soluciones computacionales en diversas áreas.

Para clarificar, La lingüística estudia todas las facetas que caracterizan a los lenguajes utilizados por el ser humano. Esto comprende la estructura gramatical de una lengua en específico, su evolución en el tiempo y la manera en que estas estructuras transmiten significados. Colocando un ejemplo muy específico, una palabra o frase puede tener un significado muy diferente dentro de una oración, si la misma está condicionada por elementos culturales o contextuales que no se hallan explicitados dentro de la misma.

La lingüística computacional, en contraste con la tradicional que se focaliza en el modelado teórico de las estructuras gramaticales y sintácticas de los lenguajes orales y escritos, busca trasladar las fórmulas del lenguaje natural humano y aplicarlas a sistemas informatizados. El propósito es lograr cada vez mejores:

1. Traductores.

2. Asistentes virtuales

3. Sistemas de consulta de Bases de Datos.

4. Buscadores de información (herramientas de Procesamiento de Lenguaje Natural (NPL) por sus siglas en inglés).

5. Sistemas Biométricos de Seguridad.

6. Procesamiento de textos e información (minería de datos).

También se puede hacer mención de los aportes que la lingüística computacional puede brindar a la lingüística convencional, entre ellas pueden destacar:

La automatización de tareas de búsqueda puede ayudar a tener una base de análisis más amplia y con mayores matices de relaciones que permitan realizar labores más efectivas que las que pudiesen obtenerse de manera manual; tampoco pudiesen ser tan detalladas ni con el nivel de profundidad del método computacional.

A través de un NLP se puede realizar el barrido de todas las entradas que traten un tema específico tanto en una publicación científica indexada como en una red social.

Medicina

En medicina se extendido el uso de redes neuronales artificiales (RNA) entrenadas para apoyar el diagnóstico médico a través de la interpretación de imágenes. La capacidad computacional ayuda a realizar diagnósticos más precisos permitiéndole al médico un informe detallado de lo obtenido a través de los escáneres. Esto mejora sustancialmente la toma de decisiones por parte del personal de salud.

Un ejemplo directo puede tomarse de una de las áreas más demandadas a nivel mundial por su incidencia en la salud mundial, y es el uso de RNA para el diagnóstico de estenosis en pacientes con riesgo de infarto. La estenosis son lesiones que ocurren cuando en las arterias disminuye el diámetro de los conductos arteriales debido a la acumulación de grasa en sus paredes, esto impide el libre flujo de sangre a lo largo del conducto, generando un elevado riesgo de sufrir un ataque cardíaco.

El entrenamiento de la red neuronal se realiza según el formato de variaciones en la entrada del sistema, tomando información de un numeroso conjunto imágenes debidamente registradas de pacientes con problemas de estenosis y de pacientes que no padecen la enfermedad a fin de poder hacer comparaciones y entrenar a la red para que pueda discriminar unas de otras.

Para este caso particular, algunas de las características que podrían emplearse de los patrones de entrada son las siguientes:

-Promedio en los rangos del diámetro arterial.

-La desviación estándar en el diámetro arterial.

-La sumatoria de los diámetros arteriales.

Para que estas tres características puedan obtenerse de manera automática se necesita una etapa de entrenamiento, que es una etapa previa de reconocimiento de patrones de entrada, donde es fundamental el cálculo historiográfico de los diámetros arteriales para con ellos obtener y discriminar los valores de los demás factores.

Cualquier sistema informatizado que sea destinado al soporte de decisiones médicas parte del procesamiento de considerables cantidades de información para obtener un conjunto importantes de características que modelan el comportamiento de una dolencia particular. De otra manera sería impracticable un diagnóstico preciso, lo cual es fundamental para ofrecer beneficios reales a la salud de los pacientes de forma directa.

La Inteligencia computacional también se está posicionando en otras

áreas de la medicina aparte del diagnóstico médico. Una de estas se ubica en el monitoreo de dispositivos de cuidado intensivo, aprendiendo de los estados biológicos de los pacientes para tomar medidas en caso de que se manifiesten cuadros específicos que indiquen que se han rebasado ciertos límites o líneas para hacer sonar las alarmas cuando suceda.

Otra área que ha ido evolucionando constantemente es la de equipos de detección temprana de posibles enfermedades que un paciente pudiese desarrollar a corto o mediano plazo, que dependen de un conjunto de variables genéticas, culturales y laborales entre otras, y así poder planificar un tratamiento preventivo adecuado.

El uso de robots equipados con IA para realizar cirugías supervisadas ya ha superado el primer escalón técnico y ya es una realidad en desarrollo. En Fort Lauderdale ya se han practicado cirugías asistidas con el apoyo de un robot. Destaca el uso de sensores que supervisan constantemente el estado del paciente para advertir cualquier circunstancia de peligro. Lo más destacable de estas cirugías es que se hacen con un mínimo de incisión en el cuerpo ya que un delgado instrumento colocado en la extremidad del robot es suficiente para los objetivos de la cirugía. La extremidad del cual posee la información necesaria para ir de forma precisa al órgano afectado y proceder con la cirugía. En dicha operación no interviene mano humana, solo el mínimo necesario para dirigir al robot en los puntos de mayor complejidad del recorrido.

Educación

La IA en educación se ha decantado por la tutoría del aprendizaje asistido. La forma tradicional de enseñanza a venido siendo fuertemente cuestionado por los resultados generales que presenta. Tener un maestro para 30 o más alumnos dificultad conocer los ritmos de aprendizaje de los estudiantes de manera individual y los refuerzos que deben ser aplicados para lograr que los conocimientos impartidos sean asimilados adecuadamente. Los investigadores y desarrolladores se han enfocado en brindar a los estudiantes programas informáticos que le asisten en el aprendizaje de temas concretos, donde el sistema aprende del alumno y de éste averigua cuál es el estado de sus conocimientos para guiarlo y plantearle el orden de los conocimientos a reforzar y lo que debería estudiar.

Existe una gran demanda y por tanto una evolución importante en las áreas destinadas al entrenamiento por simulación de escenarios laborales

técnicos a través de videojuegos; entre los cuales se destaca el entrenamiento de pilotos de aviones comerciales, operadores de maquinaria pesada y práctica de intervenciones quirúrgicas.

Robótica e Industria

En robótica, el uso de IA ha tenido un empleo extensivo apoyado principalmente por los grandes centros de investigación tecnológica gubernamentales y multiregionales. Los ejemplos más destacados son los de las agencias espaciales. Estos organismos debido a la necesidad de dar soluciones a problemas complejos derivados de sus actividades investigativas han necesitado desarrollar dispositivos que puedan realizar un conjunto de actividades autónomas o en conjunto con humanos en ambientes poco explorados o conocidos, o donde se requiere una exposición a factores a los que no puede exponerse el ser humano. Un ejemplo a destacar es el de CIMON (Crew Interactive Mobile Companion por sus siglas en inglés) un robot flotante con rostro desarrollado de forma colaborativa entre Airbus e IBM para la agencia espacial alemana. Ha dicho dispositivo se le han efectuado pruebas a sus sistemas de orientación, navegación y conducción y ahora estará en la Estación Internacional involucrado en el desarrollo de ciertos experimentos cuando esté ya flotando allí y que incluyen un cubo de Rubik, cristales y un experimento médico en el que el robot hará las veces de cámara flotante. En cuanto al manejo de CIMOS, está pensado para que no requiera las manos de los astronautas pues se activará mediante con control por voz, y se dirigirá al astronauta que le hable, pudiendo ejecutar expresiones faciales, moviéndose de manera autónoma. El aprendizaje de CIMON será mediante el entrenamiento en campo a través de las órdenes de los astronautas que interactúan con él ya que no posee capacidad de aprendizaje autónomo.

La empresa privada está también interesada en desarrollar ambientes robotizados que puedan aprender directamente de la experiencia exterior y adaptarse al entorno de forma instantánea, este es el principio que subyace en los desarrollos de la conducción automática. La creación de mapas lo suficientemente sofisticados y precisos le permitirán a los sistemas de conducción concentrarse en temas puntuales como el tráfico de peatonal y las estadísticas telemétricas de obstáculos viales.

Varias universidades del mundo en equipo con grandes conglomerados como podría ser Alphabet (casa matriz de Google, Android, Maps, etc.) están promocionando tanto laboratorios, centros de investigación robótica y galardones para aquellos logros en estas áreas otorgan reconocimientos

a aquellos que logran ganar concursos y competencias internacionales. Por ejemplo, estudiantes de Tecnológico Nacional de México ganaron el Robotchallenge de 2016 realizado en Viena Austria, uno de los de mayor prestigio mundial, imponiéndose por encima de dos mil robots pertenecientes a 56 naciones.

Finanzas y gestión de la información

Las finanzas es una de las áreas que más está apostando por las implementaciones de inteligencia artificial aparte del marketing y la automoción. Uno de los ejemplos que podemos utilizar es la presencia los robo-advisors que son asesores virtuales con la capacidad de proponer opciones de inversión para pequeñas empresas y personas interesadas en el mercado de valores. Esta opción hace muy poco estaba restringida a grandes corporaciones, hoy día esta tecnología permite la masificación al conocimiento financiero, permitiendo mayor economía y agilidad en la toma de decisiones.

Son muchos los ejemplos que evidencian que esta tecnología puede suponer un antes y un después en el mundo de las finanzas. Por ejemplo, los robo-advisors, que son ya un fenómeno de asesoramiento financiero en Wall Street, democratizan el acceso a este tipo de servicios. Pequeños inversores que antes no podían optar por este servicio debido a su alto coste, ahora recurren a estos "robots" de forma fácil, económica y rápida.

La banca desde siempre se ha interesado en los temas asociados a la seguridad de los usuarios y de las transacciones mercantiles. La prevención y detección de fraudes y control en el blanqueo de capitales se ha beneficiado del aprendizaje profundo (deep learning) pues las máquinas logran detectar irregularidades en diferentes escenarios es cuestión de pocos segundos.

Otro aspecto importante es la gestión de riesgos financieros al momento de la concesión de un crédito. Las evaluaciones crediticias a través del Robotic Process Automatization permiten conocer rápidamente la calidad crediticia del prestatario. Además permite identificar nuevos escenarios crediticios y oportunidades de negocio en los mercados que podrían aprovecharse.

La inteligencia artificial ha venido simplificando la experiencia del usuario al acceder a los servicios financieros electrónicos, haciendo de esta plataforma indispensable en la cotidianidad de los clientes. La cada vez mayor dependencia de las plataformas virtuales también representa un desafío y un cambio de paradigma para el usuario. Se debe alertar los riesgos de este tipo de tecnología y prever los desafíos que estas conllevan. A pesar de

las ventajas que nos pueden ofrecer, y del error de creer que los algoritmos a estas alturas están más humanizados, aún las máquinas con inteligencia artificial no son empáticas y suponen un apoyo más que una sustitución del trabajo humano.

Marketing

La transformación de marketing se viene dando a un ritmo acelerado. El uso de herramientas manuales paulatinamente se va descartando por herramientas automatizadas. Los tests A/B, la gestión de contactos, las campañas vía email, ranquin de seguimiento de consumo o lead scoring son algunas de las herramientas utilizadas en marketing que has sido impactadas por la inteligencia artificial de forma notable.

Y el impacto positivo en la rentabilidad y alcance indicado en la automatización de campañas y procesos publicitarios no fuese posible sin los algoritmos desarrollados para la industria del mercadeo de marcas y productos]. Pongamos algunos ejemplos:

NETFLIX [34] ha desarrollado un algoritmo que le sugiere al suscriptor posibles contenidos que se amoldan a sus preferencias dentro de la oferta programática de la plataforma. A partir de las preferencias de los abonados NETFLIX se nutre para crear contenido original que satisfaga dichas preferencias. Ha sido tan exitoso el programa, que el 75% del material consumido viene de alguna recomendación hecha en la plataforma.

La compañía norteamericana UPS se ahorra millones de dólares utilizando un programa que le indica cuales son las rutas óptimas para la entrega de sus paquetes. El programa se llama ORION y es el responsable de indicar las rutas de envío de mayor rendimiento.

Y por último, FACEBOOK ha desarrollado un algoritmo que elige cuales son las publicaciones más destacadas para el usuario y las coloca en su página de inicio.

Cada día más actores se suman al mercadeo digital. Se estima que el 70% de las marcas aún no están compitiendo en esta plataforma de negocios, lo que muestra el nivel de complejidad al cual se enfrentan los publicistas en un futuro muy próximo. Se estima que el futuro de las ventas esté dirigidas a satisfacer el mercado individual; cada individuo representará un mercado al que será necesario atender y por ello, el esfuerzo de las campañas publicitarias irá al microcosmo de las verdaderas necesidades individuales. Esto representa un gigantesco reto a niveles aún poco explorados motivado a la

inmensa microsegmentación a la que debe hacer frente; sin embargo la potencia de los algoritmos deep learning indican que ya pueden manejar datos de bases externas como internas y así procesar la información necesaria de cada individuo para su uso publicitario y tenerlos disponibles en muy poco tiempo.

Innovaciones como SIRI y CORTANA abren muchas posibilidades de opciones para los consumidores y agencias que buscan hacer contacto con aquellas personas que requieren de sus productos y servicios, motivado al grado de facilidad e interactividad que se puede obtener a través de estos asistentes virtuales. Sin duda el futuro de esta industria pasará por cuan efectivos puedan ser los algoritmos en un mercado cada día más competitivo.

Biología

Con el propósito de evitar la desaparición de la diversidad genética de animales y plantas sobre el planeta se están desarrollando aplicaciones que puedan pronosticar el comportamiento de grupos de especies animales, plantas y de factores medioambientales con el propósito de definir las líneas evolutivas que se podrían presentar uniendo todos los elementos que configuran los ecosistemas que previsiblemente formen parte de los escenarios futuros donde estas especies harán su trayecto vital, dependiendo de las condiciones actuales. Las técnicas de modelado matemático de Big Data y de reconocimiento de patrones (RNA) anudado al Machine Learning están haciendo posible estas investigaciones. Se aprovecha los datos obtenidos de imágenes satelitales y estadísticas de diferentes estaciones temporales para generar mapas de desplazamientos de animales y plantas y con ellos conocer las amenazas a las cuales se enfrentan y planificar medidas para disminuir los impactos negativos. Con estas técnicas también se avanza en el seguimiento de patrones conductuales y evolutivos de la especies a niveles embrionarios.

Otras áreas aplicables

En la resolución de crímenes, el apoyo de la inteligencia artificial al campo jurídico ha tenido un destacado aporte en la identificación de personas a partir de unas cuantas muestras de huellas, ADN, biotipo otros datos de importancia que acoten las opciones y den una mayor exactitud.

En diversas ciudades del mundo, el tráfico urbano ha derivado en un problema importante a resolver. El desarrollo de gestores inteligentes del tráfico está colaborando con los arquitectos y urbanistas al diseño de me-

didas que permitan un mejor desplazamiento vehicular, prediciendo el comportamiento de los mismos y a partir de ello proponer alternativas y nuevos espacios asociados a la optimización del tránsito.

Bibliografía

1. A. M. Turing, "¿Puede pensar una máquina?", Eds. KRK, p3. 2012.

2. A. M. Turing, "Computing Machinery and Intelligence", Mind, 59, pp 433-460. 1950.

3. A. M. Turing, "¿Puede pensar una máquina?", Eds. KRK, pp 10-17. 2012.

4. A.M. Turing, "On computable numbers, with an application to the Entscheidungsproblem", Proceedings of the London Mathematical Society, Series 2, 42, pp 230 - 265. 1936.

5. A. M. Turing, "On Computable Numbers, With an Application to the Entscheidungsproblem", Proceedings of the London Mathematical Society, 2 (1937) 43 (6): 544-6. Reimpreso en Martin Davis ed. *The Undecidable*, Raven Press, Hewlett NY pp. 115–154. 1965.

6. M. De León, A. Timón, "¿Pueden pensar las máquinas?". Citado en el portal web: http://esmateria.com/2014/03/25/pueden-pensar-las-maquinas/

7. L. Barón, "El juego de imitación de Turing y el pensamiento humano". Avances en Psicología Latinoamericana, Vol. 26(2), pp 180-194. 2008.

8. A. C. Clark. "Cualquier tecnología suficientemente avanzada es indistinguible de la magia" El bosque mágico, Lev Grossman. Ediciones. 2015.

9. J. McCarthy *et. al.* "A Proposal for The Dartmouth Summer Research Project on Artificial Intelligence", Stanford University, Computer Science Department. EEUU. 1955. Citado en el portal web: URL: http://www-formal.stanford.edu/jmc/history/dartmout h/

10. V. Torres, "Así fue la evolución del almacenamiento", 2017. Citado en el portal web: https://www.elobservador.com.uy/nota/asi-fue-la-evolucion-del-almacenamiento--201735500

11. J. Marcos, "¿Pueden pensar las máquinas? ¿La mente humana fue el laboratorio de quien descifró Enigma?" 2014. Citado en el portal web: http://www.fronterad.com/index.php?q=pueden-pensar-maquinas-men-

te-humana-fue-laboratorio-quien-descifro-enigma/

12. F. Rosenblatt, "Principles of Neurodynamics: Perceptrons and the Theory of Brain Mechanisms". Spartan Books,Nueva York, 1962.

13. W. S. McCulloch, W. Pitts, "A logical calculus of the ideas immanent in nervous activity". Bulletin of Mathematical Biophysics, 5, 115-133, 1943.

14. E. Kandel *et. al* "Neurociencia y Conducta" Pearson educación, pp 89-107. 1996.

15. B. Martín del Brío, & A. Sanz, "Redes Neuronales y Sistemas Borrosos" Eds. Ra-Ma, pp. 4-10. 2001.

16. M. Minsky, S. Papert, "Perceptrons: An Introduction to Computational Geometry". MIT Press, 1969.

17. G. Carpenter, S. Grossberg, "Absolutely stable learning of recognition codes by a self-organizing neural network". *Conference Proceedings 151: Neural Networks for Computing,* pp. 77-85. American Institute of Physic (AIP). 1986.

18. D. Rumelhart, J. McClelland, Grupo de investigación PDP, "Parallel Distributed Processing: Explorations in the Microstructure of Cognition". MIT Press, 1987.

19. D. Rumelhart, G. Hinton,R. Williams, "Learning representations by back-propagating errors". *Nature*, 323 (6088), pp. 533–536. 8 October 1986.

20. K. Funahashi, "On the approximate realization of continuous mappings by neural networks". Neural Networks, 2, pp.183- 192, 1989.

21. B. Martín del Brío, "Redes Neuronales y Sistemas Borrosos" Eds Ra-Ma, pp. 43-47. 2001.

22. P. Flach, "Machine Learning: The Art and Science of Algorithms that Make Sense of Data". Cambridge University Press. 2012.

23. M. Pérez, "¿Podremos crear máquinas verdaderamente inteligentes?"2018. Citado en el portal web:https://elpais.com/tecnologia/2018/03/27/actualidad/1522139782_730950.html.

24. J. Mora, "La Evolución de la Inteligencia Artificial", 2016. Citado en el portal web:https://mundocontact.com/la-evolucion-de-la-inteligencia-artificial/

25. J. Sanz, "¿Qué es la Lingüística Computacional? 2018. Citado en el portal web:http://www.full-on-net.com/blog/inteligencia-artificial/

qu-es-la-lingstica-computacional.

26. I. Cruz, "Avances en las cirugías asistidas". 2016. Citado en el portal web:https://centrosconacyt.mx/objeto/cardio/

27. Conacyt, 2016. "Inteligencia computacional y una aplicación en la cardiología". Citado en el portal web:http://www.cronica.com.mx/notas/2016/983394.html

28. B. Gros, "Tres usos de la Inteligencia Artificial en la educación". 2018. Citado en el portal web:https://www.educaciontrespuntocero.com/opinion/inteligencia-artificial-en-educacion/95072.html.

29. A. Martí. 2018. "Un robot flotante con inteligencia artificial y <cara> será el próximo astronauta en la Estación Espacial Internacional". Citado en el portal web:https://www.xataka.com/espacio/robot-flotante-inteligencia-artificial-cara-sera-proximo-astronauta-estacion-espacial-internacional.

30. AFP, "Inaugura Google laboratorio de IA en París". 2018. Citado en el portal web:https://www.jornada.com.mx/ultimas/2018/09/18/inaugura-google-laboratorio-de-ia-en-paris-8204.html.

31. TecNM/DCD, "Estudiantes del TecNM ganan dos medallas de oro en Mundial de Robótica en Austria" 2016. citado en el portal web: https://www.tecnm.mx/ciencia-y-tecnologia/estudiantes-del-tecnm-ganan-dos-medallas-de-oro-en-mundial-de-robotica-en-austria.

32. Redacción CEU IAM, 2018. "¿Cómo modela la inteligencia artificial el mundo de las finanzas? Citado en el portal web: https://www.ceuiam.com/business-school/como-la-inteligencia-artificial-modela-el-mundo-de-las-finanzas--post.

33. N. Maynez, "9 aplicaciones de la Inteligencia Artificial en el Marketing Digital que revolucionará tu negocio" 2018. Citado en el portal web: https://blog.adext.com/es/aplicaciones-inteligencia-artificial-marketing-digital.

34. A. Moreo, "Por qué dotar de inteligencia artificial las herramientas de marketing" 2016. Citado en el portal web: https://www.cyberclick.es/numerical-blog/por-que-dotar-de-inteligencia-artificial-las-herramientas-de-marketing.

35. J. Garcia-Algarra, *et.al.,* "A Structural Approach to Disentangle the Visualization of Bipartite Biological Networks". *COMPLEXITY* 2018. Número de artículo: 6204947.

36. Universidad Politécnica de Madrid, "Una herramienta para descubrir ecosistemas en riesgos de extinción", 2018. citado en el portal web: https://www.agenciasinc.es/Noticias/Una-herramienta-para-descubrir-ecosistemas-en-riesgo-de-extincion.

CAPÍTULO 2:

Técnicas de Inteligencia Computacional

Los desafíos a los que se enfrenta la población humana actual son cada vez más complejos y la necesidad de abordarlos se encuentran dentro de un intervalo de tiempo que se cuenta en décadas; en este escenario los actores políticos, científicos y sociales han observado la importancia que posee el manejo de información como herramienta estratégica al momento de afrontar los retos que se avecinan y darles una solución adecuada. Esta situación ha llevado al estudio de alternativas para que el procesamiento de información sea mejor y más eficiente, con el objeto de brindar alternativas sustentables de vida, economía y resguardo social.

Sin duda la era de la informática ha condicionado ostensiblemente el desarrollo de las potencialidades de integración e intercambio de toda la humanidad en diferentes niveles: sociales, económicos y científicos. Dentro de este marco la humanidad se enfrenta a desafíos enormes como lo pueden ser la sustentabilidad de la vida, el uso racional de los recursos naturales, la mitigación de las desigualdades económicas, el bienestar social y el acceso a la educación y a las nuevas tecnologías. Para ello se ha buscado alternativas que puedan hacer frente a estos problemas de una forma eficiente. En este punto algunos sectores de las ciencias se han volcado al estudio de la forma en que el cerebro humano realiza el procesamiento de información. A esta línea de investigación pertenece la *Computación Inteligente* (*CI*) .

La CI reúne un cuerpo metodológico matemático con una característica muy importante en el contexto actual: la capacidad de adaptarse a nuevos escenarios de manera flexible. Posee elementos como la capacidad de abstracción, asociación, descubrimiento y generalización que le permite generar medidas que responden con una acción a un escenario donde deba tomarse una decisión, o realizando una labor predictiva conociendo los elementos que integran un problema particular. Por ello la CI maneja algoritmos adaptativos, conceptos asociados a la filosofía de paradigmas y de decisión para generar respuestas en escenarios complejos donde el cambio o transición de

variables es una constante. Esta característica define al denominado comportamiento inteligente.

En 1994, en Orlando, Florida, se realiza el primer congreso sobre CI en el mundo, allí James C. Bezdek propuso la siguiente definición para la Computación Inteligente:

"Un sistema se llama computacionalmente inteligente si trata con datos de bajo nivel como datos numéricos, tiene un componente de reconocimiento de patrones y no usa conocimiento en el sentido IA, adicionalmente cuando comienza a exhibir adaptabilidad computacional, tolerancia a fallas, velocidad de respuesta y tasas de error que se aproximan al rendimiento humano."

Y propone el siguiente mapa, visto en la Fig. 7. de asociaciones que modelan la inteligencia en un sistema computacional.

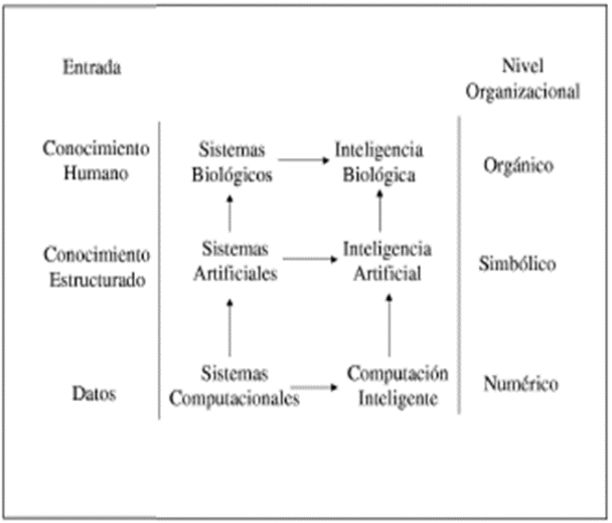


Figura. 2.1. Relación entre los componentes de un sistema inteligente.

En la complejidad de los sistemas inteligentes aumenta de menor a mayor de izquierda a derecha y de abajo hacia arriba. Este esquema responde más a la concepción de *implementación* computacional de la inteligencia (considerando a la inteligencia biológica como la más relevante) que el modelado del paradigma de lo que es inteligente independiente de su

fuente, por ello ha sido fuertemente cuestionado en la comunidad científica.

Desarrollo de la computación inteligente

La computación inteligente se ha abordado desde dos vertientes: las que se basan en el paradigma inmerso en el problema a resolver y los que se inspiran en las diferentes maneras en que el cerebro humano procesa datos e información. En este último enfoque se ha presentado la hibridación de técnicas para compensar las deficiencias inherentes de cada técnica en particular, para así potenciar su utilización. Esto ha sido fundamental para el desarrollo de la CI.

Tomando en cuenta lo antes descrito, la definición de las dos vertientes mediante las cuales se ha desarrollado la computación inteligente se resumen de la siguiente manera:

La que sigue los paradigmas de la Inteligencia Artificial que se inclinan por los aspectos cognitivos y estructurales de los problemas a abordar. Este esquema sigue las investigaciones desarrolladas por Simon y Newell donde se utiliza a la Inteligencia Artificial como medio para el desarrollo de máquinas que puedan desempeñar trabajos de la misma forma o mejor que los seres humanos, para lograrlo se debe dotar a las máquinas de comportamiento inteligente, esto engloba ciertas capacidades como lo pueden ser la percepción, la comprensión de variables en ambientes complejos, la capacidad de aprendizaje, comunicación y toma de decisiones. También la Inteligencia Artificial busca comprender la razón de los comportamientos de animales, máquinas y seres humanos. Se puede apreciar en este enfoque un objetivo inclinado hacia la ingeniería, que es ejecutar a través de máquinas tareas complejas que realiza el ser humano como es la comprensión lingüística, el procesamiento de imágenes y el razonamiento abstracto.

El que se inspira en el procesamiento distributivo y de diferentes líneas de acción (procesamiento paralelo) que presenta el cerebro humano. Esta escuela inspirada en los trabajos adelantados por James McClelland entre otros, indican que el procesamiento de información inteligente está formado por un gran conjunto de elementos de procesamiento sencillos, los cuales operan simultáneamente y comparten información entre sí para lograr el procesamiento de toda la información. A este esquema se le ha denominado *sistema conexionista.*

Técnicas utilizadas en la computación inteligente

La computación inteligente se ha desarrollado a través de esta segunda vertiente, con técnicas que la investigación científica ha ido perfilando a lo largo del tiempo de su proceso de desarrollo. Entre las técnicas utilizadas actualmente se tiene:

1. Redes Neuronales Artificiales.
2. Sistemas difusos.
3. Computación Evolutiva.
4. Vida Artificial.
5. Sistemas Expertos.

Redes Neuronales Artificiales

Introducción

Están inspiradas en el funcionamiento de las células nerviosas del cerebro y la manera en que procesan enormes cantidades de información en una estructura de conexión masiva y paralela con elementos relativamente simples de procesamiento. Al ser un sistema distributivo y adaptativo, una de las cualidades que más se destacan es su capacidad de aprender a partir de ejemplos. Desde el punto de vista del paradigma que desarrolla se puede decir que las redes neuronales buscan emular la estructura de cerebro. Para realizar una analogía que vislumbre mejor esta premisa, las redes neuronales buscan emular el *hardware* del cerebro.

Uno de los inconvenientes que presentan las redes neuronales es la dificultad de realizar una representación estructurada del conocimiento, su dificultad para interactuar con los sistemas de bases de datos actuales y la imposibilidad de explicar cómo el sistema llega a una conclusión determinada.

Diferencias entre las redes neuronales naturales y las computadoras

Como se ha indicado, las redes neuronales imitan la forma en que el cerebro procesa información. pero entre el cerebro y las unidades de cómputo que se aplican en informática existen diferencias que es recomendable conocer.

Los computadores de la actualidad poseen la arquitectura Von Neumann que es una máquina de procesamiento (el hardware) que ejecuta de manera secuencial (una acción tras otra) un conjunto de instrucciones (el software)

que se almacena en lugares específicos denominados como memoria. El computador está constituido por cuatro unidades básicas: a) La unidad de entrada de información. b) La unidad de salida. c) La unidad de procesamiento (CPU) que está compuesta por las subunidades algorítmicas y de control. y d) La unidad de memoria.

Este sistema constituye una máquina universal de Turing, por lo que teóricamente puede realizar cualquier tarea que pueda programarse de forma adecuada. En esto radica su versatilidad, ya que con solo cambiar el programa que la rige puede realizar diversas tareas, desde cálculos balísticos como control se sistemas automáticos entre muchos otros.

Aunque la potencia de unidad de procesamiento ha tenido un desarrollo enorme en la última década, su capacidad para ciertos problemas, como aquellos provenientes del mundo natural (reconocimiento visual, por ejemplo) requerían tiempos de cómputo inviables para una máquina con arquitectura Von Neumann. Esto se debe, en el caso del ejemplo, al manejo de gigantescas cantidades de datos que componen una imagen y la necesidad de una respuesta en los márgenes que maneja el tiempo real. Adicionalmente, en los problemas de la realidad es necesario el manejo de bases de datos enormes. El ordenador fue diseñado para trabajar con datos específicos de manera serial. La información de la vida real el imprecisa, masiva y redundante.

Entonces, para poder hacer frente a estas dificultades, la idea más aproximada para resolver este problema de computo derivaría hacia un conjunto de computadores conectados en paralelo; pero esto traería una cantidad de dificultades asociadas como son el elevado coste de operación, el software requerido para procesar los datos, el espacio físico que ocuparía dicho conjunto de computadores.

El cerebro por otra parte funciona de una manera diferente al marco de las computadoras con arquitectura Von Neumann. En primera instancia no tienen una unidad de procesamiento semejante a un microprocesador de la potencia que poseen las computadoras actuales; ni siquiera está constituido por un puñado de CPUs. el cerebro posee neuronas que son semejantes a procesadores elementales, altamente interconectados que es lo que da forma la red neuronal cerebral.

Las neuronas son pequeños procesadores, más lentos que cualquier microprocesador actual, sencillos y de fiabilidad reducida; sin embargo, coexisten en el cerebro humano al menos cien mil millones de neuronas trabajando en paralelo, de lo que se deriva su poder de cómputo

Las neuronas no necesitan ninguna programación más allá del aprendizaje que obtienen del entorno siguiendo un esquema distinto al que emplean las computadoras tradicionales. Este esquema se denomina como *autoorganización*. mientras las computadoras poseen una unidad central de control, las neuronas se influencian unas a otras mediante señales inhibidoras o activadoras, lo que les permite de forma dinámica ajustar sus estados según las necesidades de la red. De estas interacciones derivan variadas propiedades de procesamiento que definen la capacidad de percepción del ser humano y al final, la manera en que se llevan a cabo las actividades que definen el pensamiento.

El cerebro entonces en un sistema complejo no lineal, de procesamiento en paralelo y altamente adaptativo. Para dejar fijadas la diferencia entre las computadoras tradicionales y el cerebro humano observe la Tabla 1.

Tabla 1. Comparativa entre el cerebro humano y un computador.

	Cerebro	Computador
Velocidad del Proceso	$\approx 10^{-2} seg. (100Hz)$	$\approx 10^{-9} seg. (1000\ MHz)$
Estilo de procesamiento	paralelo	Secuencial
Número de procesadores	$10^{11} - 10^{14}$	pocos
Conexiones	10.000 por procesador	Pocas
Almacenamiento del conocimiento	distribuido	Direcciones fijas
Tolerancia a fallos	amplia	Nula
Tipo de control de procesos	Auto organizado	centralizado

Modelos de Red Neuronal Artificial

Las redes neuronales artificiales intentan emular el comportamiento funcional de las redes biológicas. Con este objetivo se busca el modelamiento de las redes neuronales biológicas. En la Fig. 2.2. se puede apreciar el modelo artificial del funcionamiento de una neurona. Allí se observa un conjunto de entradas provenientes de otras neuronas o de estímulos externos; un conjunto de pesos que indican la fuerza de las conexiones sinápticas con las dendritas de la neurona; el cuerpo de la neurona donde se acumulan los estímulos; la función de activación "T", que representa el umbral o condición que debe cumplirse para activar una señal o en caso contrario inhibir una señal, y por último el axón que transmite las señales, si se ha producido una activación, a otras neuronas .

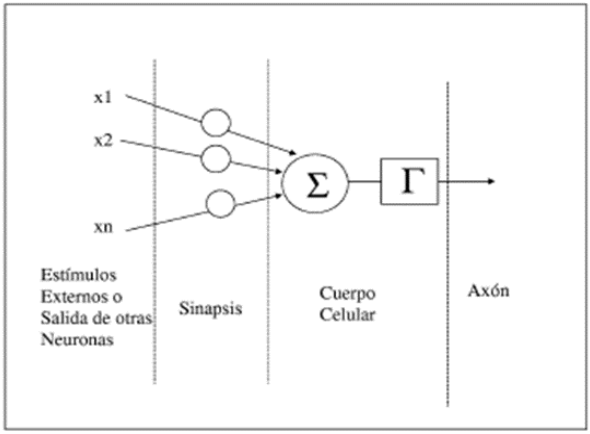


Fig. 2.2. Modelo de una neurona artificial según la referencia biológica.

La función de activación ayuda a definir un estado excitatorio o inhibitorio de la neurona; la misma se define como:

$$Sign(x) = \begin{cases} 1 & Si\ x \geq 0 \\ -1 & Si\ x < 0 \end{cases} \tag{2.1}$$

La función Sign (o signo) es una función no lineal, puesto que su argumento cambia de estado sin una transición indicada.

Las redes neuronales biológicas tienden a mantener el estado inhibitorio. Por ello una aproximación a este fenómeno se podría ilustrar como se observa en la Fig. 2.3. en ella se añade el componente artificial mediante el término . Como la predisposición al estado inhibitorio no es un valor dado de antemano se le trata como un término adicional de ajuste de la red neuronal, el cual se denomina como *sesgo*, *umbral* o *desviación*.

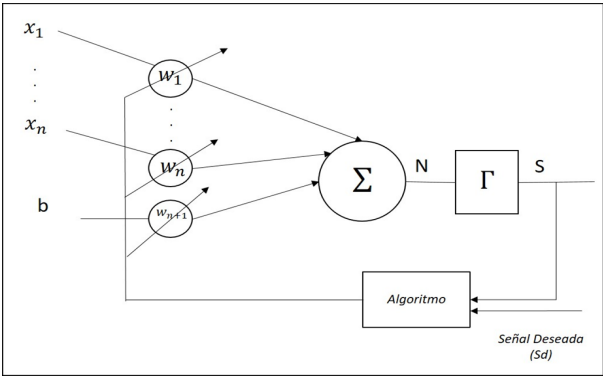


Fig. 2.3 Modelo de neurona artificial añadiendo el estado de predisposición inhibitorio.

De este desarrollo se obtienen las siguientes fórmulas que se extraen de este modelo.

$$N = \sum_{i=1}^{n} w_i x_i + w_{n+1} b \tag{2.2}$$

$$S = \Gamma(N) \tag{2.3}$$

A partir de estos datos se puede entonces, para simplificar las expresiones resultantes, definir los valores de entrada a la neurona como un vector *X* que incluya al elemento (b) que modela el *sesgo* en la red. Se tendría entonces que:

$$X = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \\ b \end{bmatrix} \tag{2.4}$$

Un vector de pesos *W* que contiene la suma de las intensidades de las interconexiones sinápticas ante todas las entradas incluyendo las que poseen el sesgo o desviación:

$$W = \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_n \\ w_{n+1} \end{bmatrix} \tag{2.5}$$

El modelo matemático que se extrae de estas acciones es el siguiente:

$$N = W^T X \tag{2.6}$$

$$S = \Gamma(N) \tag{2.7}$$

Luego del desarrollo realizado, se puede extraer un breve recuento de los elementos que constituyen a una neurona artificial:

El conjunto de entradas x_i con *i:* {1,2,3…, n}.

El conjunto de los pesos de la neurona *w* que representa el grado de intensidad de las interacciones entre las neuronas presinápticas y postsinápticas.

En el cuerpo celular se observa la sumatoria o acumulación de las señales de entrada operando sobre los pesos. A esta relación se le denomina *Regla de Propagación*. Esto proporciona el nivel de potencia postsináptica resultante en función de los pesos y entradas.

Luego se obtiene la *Función de Activación* "Γ". Esta proporciona el estado de activación actual de la neurona, en función de su estado postsináptico actual.

La función de salida S es la señal que resulta del procesamiento del estado de activación de la neurona.

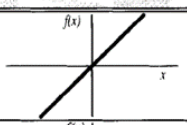
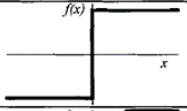
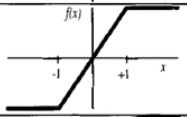
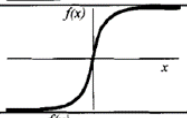
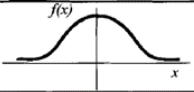
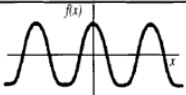
Funciones de activación

Se han utilizado diferentes funciones de activación a lo largo del desarrollo de las diferentes redes neuronales. Como se puede observar en la Tabla 2. entre las primeras funciones se encuentra la función identidad, empleada, por ejemplo, en la red neuronal Adaline. La función escalón que fue empleada en el desarrollo del Perceptrón simple también se empleó en la red Hopfield utilizada como memoria asociativa binaria, y en la neurona clásica de McCulloch-Pitts. Por último, la función lineal por tramos es computacionalmente sencilla y de factibilidad biológica, esto debido a que una neurona mientras mayor excitación recibe mayor probabilidad de activación presenta,

hasta que llega a un nivel de saturación determinado donde la respuesta no varía.

Después de la introducción de las funciones anteriores, las investigaciones fueron necesitando progresivamente que, en algunos casos, las funciones permitieran la redefinición de los valores del estado operacional de las neuronas, por ellos era indispensable que la función de activación cumpliese con la condición de ser derivable.

Tabla 2. Funciones activación utilizadas en redes neuronales.

	Función	Rango	Gráfica
Identidad	$y = x$	$[-\infty, +\infty]$	
Escalón	$y = \text{signo}(x)$ $y = H(x)$	$\{-1, +1\}$ $\{0, +1\}$	
Lineal a tramos	$y = \begin{cases} -1, & si\ x < -l \\ x, & si\ +l \le x \le -l \\ +1, & si\ x > +l \end{cases}$	$[-1, +1]$	
Sigmoidea	$y = \dfrac{1}{1+e^{-x}}$ $y = \text{tgh}(x)$	$[0, +1]$ $[-1, +1]$	
Gaussiana	$y = A.e^{-Bx^2}$	$[0,+1]$	
Sinusoidal	$y = A.\text{sen}(\omega x + \varphi)$	$[-1,+1]$	

Las funciones que cumplen la condición antes mencionada son las denominadas sigmoideos y se utilizan en redes neuronales que poseen propagación hacia atrás como método de aprendizaje supervisado. Este tipo de funciones permiten emplear técnicas extraídas de las matemáticas para corregir el error en el proceso de aprendizaje. Para ilustrar esta cualidad, se supondrá que se tiene un coeficiente numérico positivo λ junto a la variable (x) como se indica a continuación:

$$y(x) = \frac{1}{1 + e^{-\lambda x}} \tag{2.8}$$

Este factor λ permite aproximar la curva según sean su variación como se aprecia en ecuación (2.9). Para valores negativos de la variable (x) la función tenderá a cero, y para valores positivos tenderá a uno, cumpliendo con lo que se espera al momento de emular el comportamiento de los dos estados de una neurona. Esta variación de la función sigma se le denomina *sigmoidea unipolar*.

$$y(x) = \frac{2}{1 + e^{-\lambda x}} - 1 \tag{2.9}$$

A semejanza del comportamiento de la función unipolar, cuando la variable (x) es negativa, la función tiende a -1, debido a que la primera expresión de la función tiende a cero; si por el contrario la variable (x) tiende a un valor positivo, la función tiende a 1. La función se denomina *sigma bipolar*.

Otra función que posee cualidades similares a las funciones sigmoides, es la función Tangente hiperbólica tangh(x).

Otra función utilizada es la denominada *función gaussiana*, que se emplea mayormente cuando existen reglas de propagación que involucran el cálculo de cuadrados de distancia (ejemplo, la distancia geométrica euclidiana). entre los vectores de entrada y los pesos.

Para cerrar, en ocasiones muy específicas se utiliza funciones sinusoidales, como en aquellas que se requiera mostrar específicamente un comportamiento periódico en función del tiempo.

Función de salida

Representa el resultado de las operaciones llevadas por la neurona y de la cual se obtiene un estado de activación o de inhibición que será transmitido a la neurona siguiente. Cuando se afirma lo anterior se declara que la función de activación *es idéntica a la función de salida*. Esto ocurre en modelos neuronales como el Adeline y el perceptrón multicapa. Cuando la función es de tipo escalón quiere decir que para que se produzca la activación, la neurona

debe superar cierto umbral, como el caso del modelo McCulloch-Pitts. En modelos como la *Máquina de Boltzmann* la neurona presentará un comportamiento estocástico en la función de activación, esto indica que su comportamiento será probabilístico.

Modelo Estándar de Neurona Artificial

Los modelos anteriores de neuronas están muy generalizados. En la práctica se utiliza la denominada Neurona Estándar, que es un ejemplo particular de las neuronas con modelo de procesamiento distribuido en paralelo, considerando que la suma de las entradas y los pesos modelan las reglas de propagación y que la función de salida en la identidad de la función de activación.

Teniendo en cuenta lo anterior, se puede resumir que la neurona estándar consiste en:

Un conjunto de entradas y pesos sinápticos .

Una regla de propagación común que se expresa en la ecuación (2.10):

$$h_i(t) = \sigma(w_{ij}, x_j(t)); h_i(t) = \sum w_{ij} x_j \quad (2.10)$$

Una función de activación , que representa a la vez la salida de la neurona y su estado de activación.

El parámetro se añade con frecuencia entre las entradas de la neurona para modelar una cantidad que se resta del potencial postsináptico y que representa la resistencia natural de la neurona para pasar del modo inhibida al modo activo; esto significa que la neurona debe alcanzar un cierto nivel mínimo de potencial postsináptico que permita la activación de la misma. De esta manera, el argumento de la función se simplifica a la siguiente expresión:

$$\sum_j w_{ij} x_i - \theta_i \quad (2.11)$$

Con este argumento desarrollado se puede escribir que el modelo de neurona estándar posee la siguiente forma matemática:

$$S = \Gamma(N) \Rightarrow y_i(t) = f_i(\sum_j (w_{ij}\, x_i - \theta_i) \tag{2.12}$$

Si los índices *i* y *j* y se hace que coincidan con 0, se puede decir que y ; de esta manera el potencial postsináptico sería la sumatoria desde j=0. La expresión quedaría como

$$y_i(t) = f_i(\sum_{j=0}^{n} (w_{ij}\, x_i) \tag{2.13}$$

Ya con esta expresión lo que resta es definir la función de activación de alguna de la mostradas en la Tabla 2. para determinar por completo a la neurona estándar.

Con entradas de tipo digital

Observe el siguiente ejemplo. Supóngase que la función seleccionada de la Tabla 2. es la función escalón. La misma posee un comportamiento digital debido a que

$$H(x) = 1, si \sum_j (w_{ij} x_i) \geq \theta_i \tag{2.14a}$$

y

$$H(x) = 0, si \sum_0 (w_{ij} x_i) \leq \theta_i \tag{2.14b}$$

Esto implica que si el potencial de membrana supera el umbral , entonces la neurona pasa a estado activo; si por el contrario no lo logra la neurona no se activará. Más adelante se verá que éste es el modelo de activación del perceptrón original y que bajo ciertas consideraciones se denominará como de *tipo umbral* , existiendo una lógica establecida a partir de los elementos de esta clase.

Si se considera el caso de que una neurona se inhiba a partir de una señal de entrada inhibitoria, esto quiere decir que con la presencia de tan solo una señal inhibitoria de entrada ya la neurona no se activa, y además se introducen en el modelo retardos en la propagación de señales, se tendrá como resultado el modelo de neurona de McCulloch-Pitts introducida en la década del cuarenta del siglo XX como primer intento de modelado de la operación de una neurona.

Los investigadores demostraron que redes basadas en este modelo podrían realizar la operación de cualquier función lógica. Si por ejemplo se dan dos entradas binarias y , a partir de esta información, una persona con la noción básica de compuertas puede construir denominada tabla de verdad de esta neurona y observar que se obtiene la función lógica tipo NAND. Concretamente, una sola neurona de tipo umbral sólo puede realizar operaciones lógicas de separación lineal, es decir, puede modelar funciones separables linealmente, no así funciones como la XOR o OR exclusivo que no son separables linealmente. Para ello se necesita implementar variaciones como retroalimentación o añadir capas extras. Esto se desarrollará con mayor amplitud más adelante.

Con entradas de tipo analógico

Ya se consideró el caso en que las entradas tienen un comportamiento de tipo digital o discontinuo, en esta sección se trabajara con entradas de tipo continua como es el caso de las sigmoides [23]. Las funciones de tipo sigmoide poseen una gráfica que se asemeja a una "S" aplastada hacia sus extremos y que son funciones continuas. De las sigmoides más habituales debido a que pueden ser diferenciables en su rango se tiene

$$y = f(x) = \frac{1}{1+e^{-x}} \text{ , con } y \in [0,\ 1] \tag{2.15}$$

$$y = f(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}} = tgh(x), \text{ con } y \in [-1,\ 1] \tag{2.16}$$

El perceptrón multicapa utiliza este modelo de entradas debido a las necesidades de aprendizaje de este tipo de neuronas, que requieren funciones diferenciables para imponer reglas de aprendizaje como en las denominadas de *Back Propagation* (BP) [12]. Se profundizará más adelante en este tipo de neuronas.

Se debe recordar también que el modelo de neurona con entrada de tipo sigmoide puede interpretarse también de manera probabilística donde su operación deja de ser determinista. Como en los casos antes mencionados, se verá un ejemplo de este comportamiento más adelante cuando se desarrolle el tema asociado con estas características de funcionamiento neuronal.

Estructura de las Redes Neuronales Artificiales (RNA)

La estructura de una red neuronal es la configuración o topología que posee dicha red; esta parte de la manera en que están interconectados los nodos o neuronas que la conforman. En una RNA los nodos se interconectan a través de sinapsis y dichas conexiones según el patrón que posean son las que definen el comportamiento final de la red. Cabe destacar que dichas topologías están inspiradas en los modelos biológicos que presentan las neuronas en el sistema nervioso cuyas terminales se encuentran cercanas a los receptores de los [16]. Las conexiones sinápticas poseen una direccionalidad que define la manera en que se propaga los impulsos y estos llevan un único sentido que es de una neurona presináptica a una postsináptica como se puede observar en la Fig. 2.8.

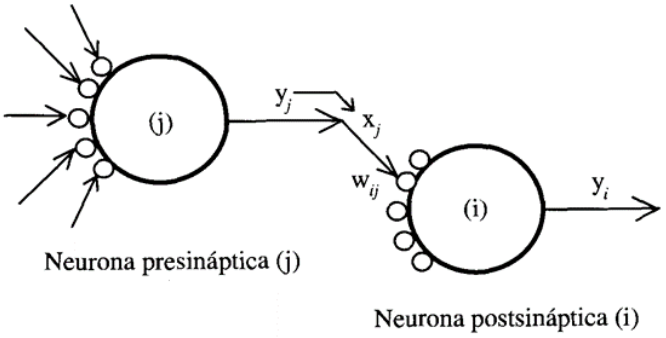


Fig. 2.8. conexión entre una neurona presináptica y una postsináptica

Por lo general las neuronas se agrupan en unidades topológicas y estructurales denominadas capas, estas a su vez en grupos de redes con características no siempre similares denominados *clusters*. Cuando no existen dichos grupos lo más recurrente es que en una capa las neuronas sean del mismo tipo. cuando existe una o varias capas formando un conjunto, existe una red neuronal.

Según la función que desempeñe, las capas pueden ser de tres tipos como se puede ver en la Fig. 2.9.

Capa de entrada: está compuesta por neuronas que captan señales o datos que provienen de su entorno; para especificar más su función se ejemplifica suponiendo que existen sensores de luz, movimiento, calor etc., que transmiten los valores recogidos del ambiente a la capa receptora que es la capa de entrada, ésta dependiendo de su configuración realizará una acción con los estímulos recibidos.

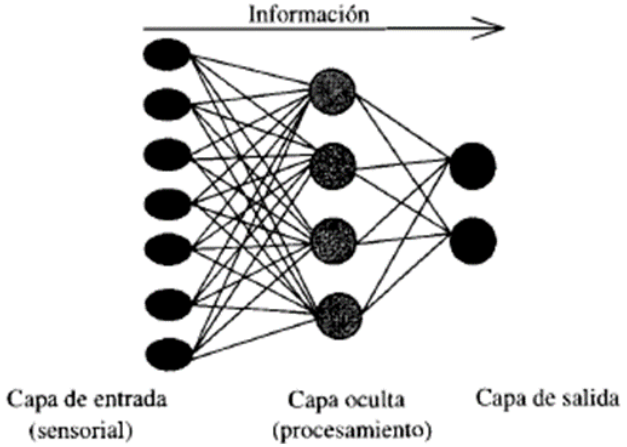


Fig. 2.9. Estructura de red neuronal unidireccional. se aprecian los tres tipos de capas: entrada, oculta y salida.

Capa de salida: es la donde las neuronas generan una respuesta de la red sobre los estímulos recibidos. En esta capa, las neuronas pueden estar conectadas a elementos que procesen esa respuesta como podrían ser los efectores.

Capa oculta: es la capa intermedia que no tiene una conexión directa ni con sensores ni con efectores, lo que quiere decir que no posee una conexión directa con su entorno. Este tipo de capas añade flexibilidad a las redes proporcionándoles libertad para conseguir. Representaciones del entorno que le permitan caracterizarlo con un mayor detalle, lo que representa un mayor nivel computacional.

Las neuronas presentan conexiones que como ya se han indicado pueden ser excitatorias o inhibitorias; si el *peso* sináptico es positivo, entonces se tiene

una conexión excitatoria, si por el contrario le *peso* es negativo, se producirá una conexión inhibitoria. Cabe mencionar que las técnicas de entrenamiento actuales para Redes Neuronales Artificiales no contemplan que las conexiones sean inhibitorias o excitatorias *per se*, sino que se entrenan para escoger la magnitud y signo que deben tener los pesos en la red.

Aparte a lo anterior descrito, las neuronas también pueden presentar conexiones *laterales* o *intra capas*, estas se producen entre las neuronas que se encuentran en el mismo nivel o capa. Existen también las neuronas que poseen conexiones *inter capas* que son las que se producen entre neuronas de capas diferentes; e incluso existen las conexiones *retroalimentadas*, donde a diferencia de las conexiones entrada salida estas conexiones dirigen impulsos a neuronas que les preceden o incluso poseen conexiones consigo mismas [12].

Una vez establecidos los elementos constitutivos más elementales de las Redes Neuronales Artificiales, se procede a exponer las arquitecturas más comunes que se han adoptado hasta los momentos, tomando en cuenta aspectos estructurales y también dependiendo de cómo fluye la información entre sus conexiones.

Atendiendo en primera instancia los aspectos estructurales, la clasificación de las RNA queda de la siguiente manera.

Arquitecturas de RNA definidas por su estructura

Redes Monocapa

Son aquellas RNA cuyas neuronas están organizadas en una sola capa.

Redes Multicapa

Son aquellas RNA cuyas neuronas están organizadas en dos o más capas.

Arquitecturas de RNA definidas por su flujo de información

Redes unidireccionales

Son aquellas RNA cuyo flujo de información se realiza en una sola direc-

ción y es desde las neuronas de entrada hacia las neuronas de salida. Como se puede apreciar en la Fig. 2.10. La ilustración corresponde a una RNA multicapa y unidireccional.

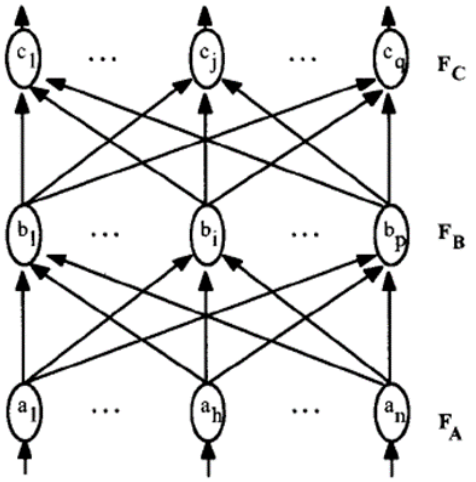


Fig. 2.10. Estructura de red neuronal multicapa unidireccional.

Redes Realimentadas

También denominadas redes *recurrentes*, en las RNA realimentadas la información puede fluir en una dirección cualquiera, incluso en los arreglos salida-entrada como se puede apreciar en la Fig. 2.11. que representa a una red monocapa realimentada.

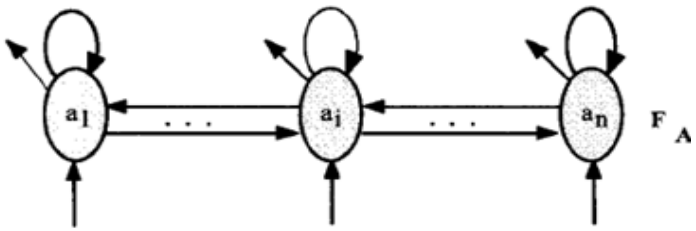


Fig. 2.11. Representación de una RNA monocapa realimentada.

Arquitecturas de RNA definidas por manejo de memoria

Redes Heteroasociativas

Son aquellas redes entrenadas para que teniendo un conjunto de entradas **A** se produzcan como respuestas de salida un conjunto de salidas **B**. Esto corresponde con la idea general que una RNA se interpreta en su conjunto como una *red de memoria asociativa* que responde con un comportamiento determinístico. En el caso de un conjunto de entradas establecido se produce un conjunto de salidas determinado.

Redes Autoasociativas

Son aquellas RNA que se auto referencian para obtener la señal de salida. Esto significa que la RNA en presencia de un conjunto **A** de señales de entrada, el patrón de referencia es la misma RNA para producir las señales de salida. Por ejemplo, como sucede con la RNA desarrollada por Hopfield las señales de entrada sean **A**=**A'** + *ruido*, la RNA como respuesta elimine el ruido y reproduzca la señal **A** original [12, 16, 20].

Hacia una definición formal de una Red Neuronal Artificial

Hasta el momento no se ha aproximado a las RNA de forma intuitiva, siguiendo el camino que referencian a los prototipos biológicos de los cuales se inspiraron inicialmente; sin embargo, es necesaria una definición que contenga una mayor rigurosidad y precisión, y es lógico que la misma tenga una fundamentación matemática. Para ello se utiliza el concepto de *grafo* como herramienta para definir una RNA.

Un grafo es un objeto que está constituido por un conjunto de nodos o vértices que se conectan a través de arcos o enlaces. En el caso que nos atañe, el grafo representa la arquitectura del sistema junto con sus diferentes enlaces, quienes fungen como los canales por los que discurrirán los datos. Los grafos como queda representados gráficamente en la Fig. 2.12 pueden ser *dirigidos* cuando se desplazan en sentido específico o *no dirigidos* cuando hacen uso de los enlaces de forma bidireccional. También se puede indicar que los grafos pueden ser *densos* cuando existe una gran interconectividad entre sus enlaces o *dispersos* cuando la interconectividad es escasa. Los grafos pueden estar constituidos de diversas formas tomando en cuenta las características antes mencionadas [12, 16, 20].

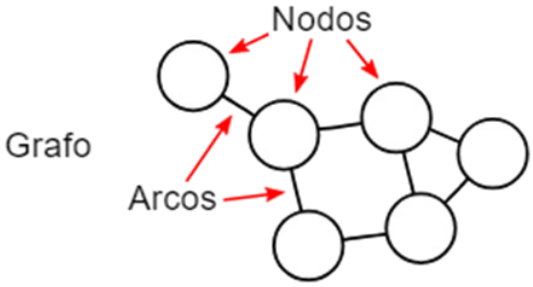
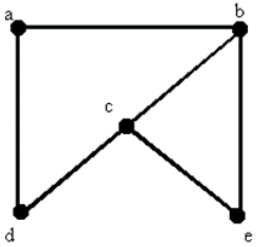


Fig. 2.12. Representación gráfica de un grafo.

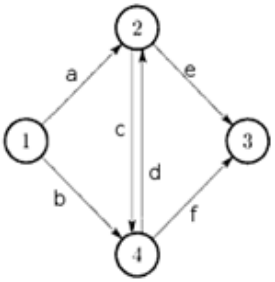
Otra representación muy utilizada de los grafos es a través de una matriz de conexiones. Se observa en la Fig. 2.13 que, para el caso de grafos no dirigidos, la matriz es simétrica.



$$A = \begin{array}{c} \\ a \\ b \\ c \\ d \\ e \end{array} \begin{array}{c} \begin{array}{ccccc} a & b & c & d & e \end{array} \\ \left[\begin{array}{ccccc} 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 \end{array}\right] \end{array}$$

Fig. 2.13. Matriz de un Grafo no dirigido.

En el caso de un grafo direccionado la matriz queda representada como se ilustra a continuación en la Fig. 2.14.



$$\begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \end{array} \begin{array}{c} \begin{array}{cccccc} \mathbf{a} & \mathbf{b} & \mathbf{c} & \mathbf{d} & \mathbf{e} & \mathbf{f} \end{array} \\ \left[\begin{array}{cccccc} 1 & 1 & 0 & 0 & 0 & 0 \\ -1 & 0 & 1 & -1 & 1 & 0 \\ 0 & 0 & 0 & 0 & -1 & -1 \\ 0 & -1 & -1 & 1 & 0 & 1 \end{array}\right] \end{array}$$

Fig. 2.14. Matriz de un Grafo dirigido.

Una tercera de representar un grafo es a través de sus listas de adyacencias o conexiones que indican cómo se encuentran conectados los nodos o vértices a través de sus diferentes enlaces. Una representación de esto se tiene en la Fig. 2.15 a continuación.

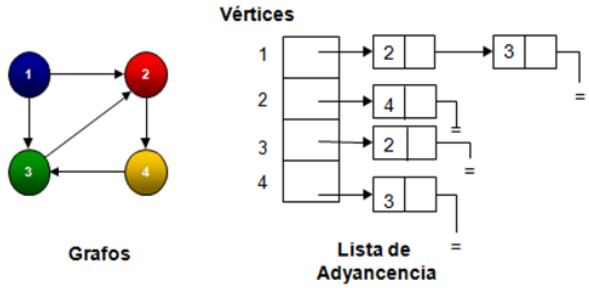


Fig. 2.14. Tabla de adyacencias de un grafo.

Ya con un conocimiento general de grafos, la definición de una RNA puede hacerse de la siguiente manera, adoptando la aproximación de Müller [23]:

Definición de Red Neuronal Artificial

Una RNA es un grafo dirigido con las siguientes características constitutivas:

Todo nodo está asociado con una variable de estado .

A una conexión de los nodos y se le atribuye un peso perteneciente a

Todo nodo posee un umbral .

Para todo nodo existe una función que lo define que se escribe como , y que depende de los pesos de las conexiones del nodo, del valor del umbral y de los estados de los nodos a él conectados. De esta manera la función define el nuevo estado del nodo.

Para una mayor claridad en los conceptos mencionados, en la práctica de las redes neuronales, los nodos simbolizan a las neuronas y las conexiones simbolizan las sinapsis. Se reformula entonces lo anteriormente descrito de la siguiente manera:

Las neuronas de entrada no poseen sinapsis previas.

Las neuronas de salida no poseen sinapsis que sean estimulaciones de entrada de otras neuronas.

Las neuronas que no son ni de entrada ni d salida son denominadas neuronas ocultas.

Una red de neuronas es unidireccional si no posee bucles dirigidos a neuronas en la misma capa o capas anteriores.

Una red es recurrente cuando existen bucles de retroalimentación, que son bucles que provienen de neuronas de la misma capa o de capas anteriores dirigidas hacia adelante.

Arquitecturas de Redes Neuronales Artificiales según el modelo de aprendizaje

Las Redes neuronales artificiales pueden entre las que hacen uso de memoria para ejecutar instrucciones y las que utilizan aprendizaje de patrones. Esta última característica a resultado de gran versatilidad he importancia en las redes neuronales artificiales ya que la propiedad de ser entrenadas a producido un impacto profundo en la sociedad moderna actual [24].

A grandes rasgos, y sin incluir todos los tipos de entrenamiento que pueden recibir las redes neuronales para el aprendizaje de ciertas pautas, se tienen los siguientes tipos:

Aquellas redes neuronales cuyos algoritmos de entrenamiento utilizan supervisión.

Aquellas redes neuronales cuyos algoritmos de entrenamiento no poseen supervisión.

Aquellas redes neuronales cuyos algoritmos de entrenamiento utilizan

reforzamiento [25], estoy indica una combinación de las anteriores.

Aquellas redes neuronales cuyos algoritmos de entrenamiento son híbridos.

Redes Supervisadas

Como puede observarse en la Fig. 2.15. A las redes neuronales con supervisión se le suministra cierta información externa sobre el comportamiento de las funciones de entrada/salida que serán modeladas en la red. Dentro de dicha información se suministra el reconocimiento de patrones de la función de salida con respecto a la función de entrada y la forma en que se organizan las clases; esto persigue el objetivo de, mediante la iteración y ajuste de los pesos de la red, la función de salida del sistema coincida con el objetivo buscado. A pesar que este tipo de entrenamiento presenta una mayor complejidad de computo en el reconocimiento de las reglas que definen a las funciones de la red, los resultados son a cambio mucho más precisos

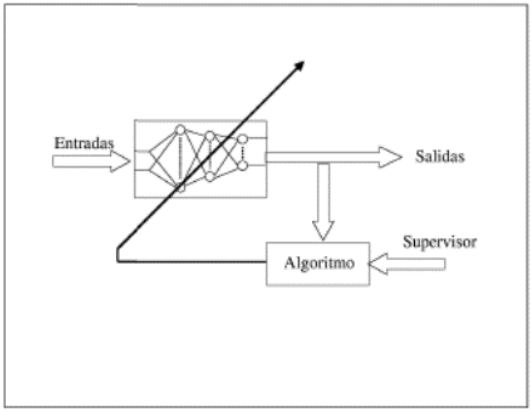


Fig. 2.15. Ejemplo de una red neuronal con supervisor.

Entre las redes supervisadas se encuentra a menudo las denominadas redes neuronales unidireccionales, que son ampliamente utilizadas para el reconocimiento de patrones y evaluación de funciones [26]. Se menciona entre las redes de este tipo a los *perceptrones simples*, *perceptrones multicapa* y las redes tipo *ADALINE* [18]. El algoritmo de *retro propagación* o BP (Back Propagation) es sin lugar a dudas el de mayor empleo en la actualidad para

redes multicapas [27].

Proceso de entrenamiento de una red neuronal artificial (RNA) supervisada

Las RNA reciben datos de entrada que luego mediante una función de activación producen datos de salida. Entre los objetivos ulteriores se encuentras clasificar los datos de salida o ajustar la función de activación. El por qué de esto estriba en que teóricamente si existen suficientes neuronas en la red artificial se podrá modelar una función continua con una precisión adecuada si se seleccionan los valores adecuados para ajustar los parámetros que posea dicha red. Dichos parámetros por lo general son los pesos sinápticos, los cuales son la memoria donde se almacena el conocimiento que la red posee sobre el problema que está resolviendo.

Una RNA logra almacenar el conocimiento sobre un problema mediante el entrenamiento que reciba; dicho proceso se basa en la modificación de los pesos sinápticos de dicha red a fin de conseguir el mejor rendimiento que pueda lograr. El aprendizaje se genera mediante la comparación de los resultados obtenidos con un conjunto de referencia o de entrenamiento. Es importante aclarar en este punto que el objetivo del aprendizaje en una RNA no es memorizar respuestas a estímulos específicos, es decir, reconocer las salidas de datos productos de las entradas presentes, sino que logre reconocer los patrones y modelarlos de forma general, entendiendo el por qué se han generado. Por ello es muy importante que las referencias en cuanto a ejemplos para el entrenamiento sean suficientes y representativos del problema al cual se le busca solución para que, una vez entrenada la RNA, pueda generalizar y manejar datos que no hayan estado en los ejemplos de referencia sin que ello signifique una pérdida de precisión en sus resultados [28].

Cuando el aprendizaje es supervisado, por lo general se le suministra a la RNA cierto conjunto de entradas y salidas deseadas que la red debería ser capaz de producir y entender la relación que las une. Matemáticamente esta relación se describe como sigue

$$\mathbf{X} = \{(\mathbf{x}^n, \mathbf{t}^n)\}_{n=1}^{N} \tag{2.17}$$

Donde X es el vector de entradas, t el correspondiente a las salidas y N es el tamaño del conjunto de entrenamiento. Si se toma un modelo genérico f(.) para representar a la red, se tendría que los parámetros de peso w serían los únicos a definir para producir una aproximación adecuada de t. tomando en cuenta todo esto, Se obtiene la ecuación siguiente:

$$y = f(x \mid w) \tag{2.18}$$

Donde *y* es la salida que provee la red. El algoritmo de aprendizaje buscará optimizar la relación entrada salida hasta hacerla parecer lo mayor posible a los parámetros dados por el conjunto de entrenamiento. Esto significa que se busca un conjunto w*w donde el error de aproximación se mínimo. Esto conlleva a definir una función de Error *E* que controle y verifique en cada etapa del aprendizaje cuánto se está aproximando a la solución; esto indica que, por cada iteración *n* de aprendizaje se compare y^n con t^n a fin de observar la diferencia entre ellas y las estimadas por las señales de control de entrenamiento.

La manera en que se mida el error dependerá de la naturaleza del problema que se esté buscando resolver; en el caso que sea un problema de regresión [29] la salida es una variable continua y se utilizaría el Error Cuadrático Medio (ECM)

$$E = \frac{1}{N}\sum_{n=1}^{N}(y^n - t^n)^2 \tag{2.19}$$

De esta manera se logrará que las salidas de la red modelen la función de distribución media de las salidas que quieren utilizarse durante el entrenamiento.

Si por otra parte el problema es de clasificación, se buscará no solo la señal de salida de la red, sino también a la clase C a la que pertenece; entonces se tiene que la salida que se busca identificar es un vector t = (t_1, t_2, . . ., t_C) cuyos valores son binarios (0 o 1) y donde solo aquel valor que pertenezca a la clase adecuada tomará el valor de 1. Entonces se recomienda el uso de la función de Entropía Cruzada (EC) que permite modelar las probabilidades que tiene una salida particular en pertenecer a una de las clases consideradas en el entrenamiento. La ecuación de la EC es

$$E = -\sum_{n=1}^{N}\sum_{k=1}^{C}(y_k^n)^{t_k^n} \tag{2.20}$$

De la expresión representa a la neurona k en la iteración n. En este caso no se toma en cuenta tanto la diferencia entre los valores buscados de para cada neurona k de salida y lo obtenido ; sino que la clasificación del resultado esté en la clase adecuada, esto quiere decir que el resultado de la clasificación sea el correcto; esto quiere decir que se le atribuya a la neurona de la clase correcta el mayor valor de activación a la salida asociada a ella.

Métodos de aprendizaje supervisado

En la metodología de aprendizaje supervisado se tiene un conjunto N: $\{x^1, x^2, ..., x^N\}$ que sirve como patrón para el tipo de entradas que se requieren modelar y obtener una correspondencia determinada con un conjunto t: $\{t^1, t^2, ..., t^N\}$ de salidas que se esperan obtener, minimizando la función de error *E* entre los valores deseados y las salidas y: $\{y^1, y^2, \ldots, y^N\}$ obtenidas.

Buscando una mejor comprensión de los conceptos a mostrar más adelante, se realizará el análisis de comportamiento de un perceptrón simple, tomando en cuenta solo una única salida; a pesar de no ser de mucha aplicación en la actualidad, ayuda mucho para entender el funcionamiento básico.

Regla de Hebb

Siguiendo el modelo de McCulloch-Pitts [12] de la neurona estándar, para que una neurona se active dependerá del valor de sus entradas junto con el valor de sus pesos de tal manera que dependiendo de las variaciones que puedan generarse en los pesos, estos modifican los valores de salida para los mismos valores en las entradas. Como se indica en la Tabla 3.

Tabla 3. Simulación de compuertas AND y OR para la neurona estándar, con w_0=-1.5 para la compuerta AND y w_0=-0.5 para la compuerta OR.

x_1	x_2	w_1	w_2	$\sum_{i=1}^{2} w_i x_i$	AND	OR
0	0	1	1	0	0	0
0	1	1	1	1	0	1
1	0	1	1	1	0	1
1	1	1	1	2	1	1

La Tabla 3. nos muestra la capacidad del modelo para aprender el comportamiento de las funciones lógicas AND y OR. Como se observa, los valores w_1 y w_2 poseen con el mismo valor y no varían; con solo variar el valor del umbral w_0, se obtiene una función u otra.

Para que la RNA logre el comportamiento que se desee, los pesos deben ser entrenados, lo que supone un proceso de aprendizaje. Entre los primeros métodos de aprendizaje se encuentra el desarrollado por D. Hebb [30] en 1949 y se basa en un hecho comprobado de la biología y es que cuando *dos neuronas se activan de manera simultánea, la conexión entre ambas se refuerza* [31]. Esto indica que cuando una neurona posee una actividad positiva y la misma influye sobre la actividad de otra neurona, la conexión que las une se refuerza, el caso inverso produce de la misma forma un efecto contrario, si una neurona inhibe el funcionamiento de otra, la conexión que las une tiende a debilitarse. Si se supone que en la Fig. 2.7 la entrada x_i es la salida de otra neurona, el valor del peso w_i se incrementa si la entrada x_i y la salida *y* son ambas positivas. Por el contrario, el peso w_i disminuirá si las variables x_i, *y* poseen signos contrarios. Esta idea se puede representar mediante la siguiente formulación matemática:

$$w_i(t+1) = w_i(t) + \eta x_i y \tag{2.21}$$

Indicando que, cuando un peso w_i en un instante determinado (t+1), es igual al peso en el instante del entrenamiento anterior (t) más el añadido de un incremento o decremento, la magnitud de dicho decremento o incremento viene dado por los valores de x_i y la salida *y* multiplicado por el *ratio o tasa de aprendizaje* . Esta ratio η es un factor constante positivo y es quien nos indica en cuanto se modifica el peso sináptico. Esto implica que de cierta

manera controla la velocidad de aprendizaje y posee cualidades no supervisadas ya que no depende de los valores de salidas **t** de referencia.

El Perceptrón

Frank Rosenblatt inspirándose en el modelo de McCulloch-Pitts y añadiendo como base de aprendizaje un método modificado de la regla de Hebb presentó el perceptrón en el año 1957 [19]. El mismo consiste como ya se ha mencionado en el capítulo anterior, en una capa de *j* cantidad de neuronas x_j que se encargan de recibir y modificar un subconjunto de datos de entrada mediante un conjunto de valores de pesos previamente fijados a los cuales se les aplica una función de tipo escalón (véase Tabla 2.) si se toma el sesgo θ_j como un valor de 1, se obtiene la ecuación

$$y_i(t) = f(\sum_{j=1}^{N}(w_{ij} - \theta_i) \tag{2.22}$$

Si se toma en cuenta que las neuronas de entrada no ejecutan algún tipo de cómputo, sólo llevan la información (nótese que el termino información indica datos ya procesados y no datos sin procesar provenientes del exterior. Así era en principio el modelo original) a las neuronas de salida. La activación se realiza a través de la función escalón. De esta forma se llega al funcionamiento de un perceptrón simple, escribiendo la ecuación como sigue:

$$y_i(t) = H(\sum_{j=1}^{N}(w_{ij} - \theta_i) \tag{2.23}$$

Donde la función H o Heaviside es la función escalón.

El perceptrón puede emplearse como clasificador o como representación de funciones booleanas (como pudo observarse en la Tabla 3.). debido a que su modelo neuronal es básicamente de tipo McCulloch-Pitts cuya salida es binaria, su importancia se debe a que es un dispositivo que puede ser entrenado, ya que permite determinar cuáles son los pesos sinápticos adecuados de manera automática teniendo en cuenta el error entre la salida que se obtiene y la que se desearía obtener lo que lo convierte en el primer

método de aprendizaje supervisado.

Se verá cómo un perceptrón puede realizar labores de clasificación. Toda neurona de un perceptrón representa una determinada clase, de tal manera que un vector de entrada, una determinada neurona responde con un **0**, si no pertenece a la clase a la cual representa y con un **1** si pertenece a la clase. Como se puede notar, una RNA de tipo perceptrón solo puede discriminar entre dos clases que se puedan separar mediante separación lineal, esto quiere decir que las clases puedan separarse a través de una única condición que esté sobre una línea recta o hiperplano.

Entonces, se dice que hay una RNA tipo perceptrón con dos entradas x_1 y x_2; Según la ecuación que la define se tiene que:

$$y = H(w_1 x_1 + w_2 x_2 - \theta \tag{2.24}$$

Lo que tomando en cuenta la función escalón se tiene que

$$y = \begin{cases} 1, si\ w_1 x_1 + w_2 x_2 \geq \theta \\ 0, si\ w_1 x_1 + w_2 x_2 < \theta \end{cases} \tag{2.25}$$

Si x_2 se hace depender de los valore de los pesos y de x_1, se puede realizar el siguiente despeje

$$x_2 = -\frac{w_1}{w_2} x_1 + \frac{\theta}{w_2} \tag{2.26}$$

Si se realiza la representación gráfica de esta ecuación lineal (ya que obtiene una recta de pendiente negativa) efectivamente se puede observar la separación lineal en dos regiones, una región donde estarán las salidas asociadas a la clase '0' y las asociadas a la clase '1', como se observa en la Fig. 2.16.

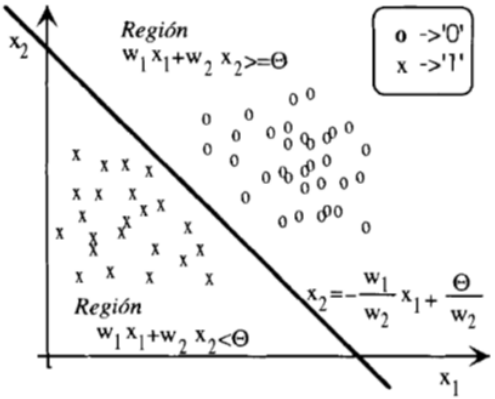


Fig. 2.16. Regiones de decisión divididas por una recta o hiperplano.

Se concluye que el perceptrón es un discriminador de tipo lineal, debido a que separa dos regiones en el espacio que pertenecen a dos clases diferentes de datos, utilizando como filtro una condición lineal.

Lo anterior nos lleva a concluir que el perceptrón puede modelar el funcionamiento de una compuerta AND y sus variantes (NAND, OR, etc.); sin embargo, frente a funciones lógicas de tipo exclusiva como la XOR el rendimiento del perceptrón es bastante lamentable. Al tomar como referencia la Fig. 2.17. Se puede observar que el desempeño de perceptrón simple para modelar las regiones de forma adecuadas según la función a imitar es muy reducido.

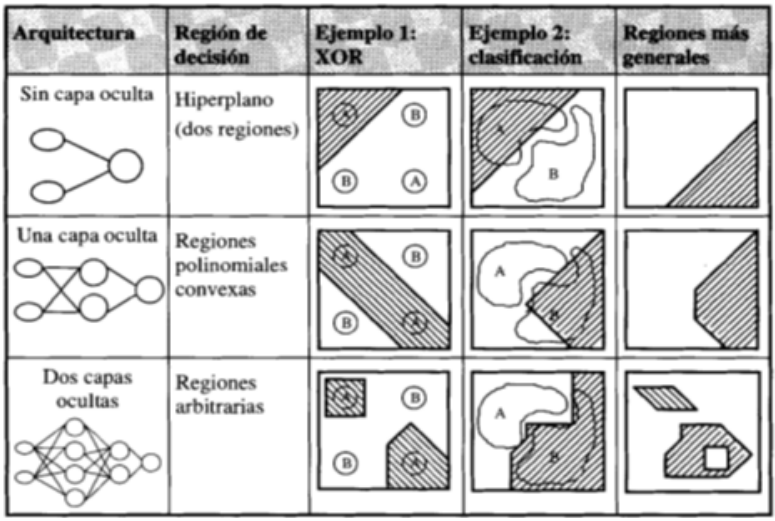


Fig. 2.17. regiones de decisión del perceptrón según su tipo.

En el caso de la función lógica XOR que produce '0' si las entradas son iguales y '1' si son distintas. Como puede observarse, las regiones no pueden discriminarse con una única condición lineal y siendo el perceptrón simple un discriminador lineal, no está capacitado para modelar adecuadamente esta función. Se concluye que para funciones no lineales el perceptrón simple no puede representarlas correctamente. También se puede observar que otras redes neuronales, como las que poseen capas adicionales, modelan de mejor manera la función lógica XOR. Entonces se debe acudir a las RNA multicapas para lograr el modelamiento de estas funciones más complejas.

Denker [32] pasado el primer lustro de los años 80 logró observa junto con otros colegas que cualquier función booleana es susceptible a ser representada por una RNA de tipo unidireccional que posea tan solo una capa oculta, este tipo de capa es aquella que no tiene conexión directa con estímulos del exterior y que tampoco posee conexiones hacia la salida de la red. Esta conclusión ya se venía vislumbrando desde finales de lo 70, que para solucionar las carencias del perceptrón era necesario añadir capas ocultas; sin embargo, y a pesar que se tenía disposición del algoritmo del perceptrón simple, se carecía de un procedimiento que lograse calcular de

forma automática los pesos adecuados en una red multicapa, este problema denominado asignación de crédito nacía del desconocimiento de cómo contribuía al error total en la salida del sistema de RNA, los diferentes nodos ocultos. Este problema fue resuelto en primera instancia por Paul Werbos [33]; sin embargo, se tuvo que esperar a que el grupo PDP (Parallel Distributed Processing Research Group) redescubriera el algoritmo BP (Back Propagation) [34] para resolver los inconvenientes y dar a conocer el potencial de esta tecnología para la resolución de problemas.

Aprendizaje del Perceptrón

El algoritmo aprendizaje del perceptrón se realiza a través de la corrección de errores, del mismo modo que el BP y el ADALINE [35]. Esto quiere decir que el perceptrón mediante iteraciones va corrigiendo el error de salida comparándolo con ciertos valores que fungen como patrones de referencia. Significa que los pesos son ajustados a medida que los resultados no coinciden con los patrones de referencia teóricos hasta que el error sea el mínimo posible.

Entonces

Se tiene un conjunto de referencia N: $\{x^1, x^2, ..., x^N\}$ que sirve como patrón para el tipo de entradas.

Un conjunto t: $\{t^1, t^2, ..., t^N\}$ de salidas que se esperan obtener.

Un conjunto y: $\{y^1, y^2, \ldots, y^N\}$ de salidas obtenidas

La función de error *E* que compara los resultados e indica si es necesaria otra iteración para la búsqueda de mejores valores para los pesos.

Los valores de entradas y salidas se encuentras en el rango de valores comprendidos en intervalo que va desde el 1 al -1, o desde el 1 al 0 dependiendo de la definición de nivel lógico. Los pesos w inician en valores aleatorios, y se busca clasificarlos correctamente a fin de que el resultado presente una separación lineal de los pesos si ello es posible. Si en una iteración los valores de pesos son correctos no se actualizan, en caso contrario, se aplica la regla de Hebb de la siguiente manera

$$\Delta w_{ij}^{N}(t) = \begin{cases} \eta t_i^N x_j^N, & si\ y_i^N \neq t_i^N \\ 0, & si\ y_i^N = t_i^N \end{cases} \tag{2.27}$$

La regla para conseguir los pesos adecuados sería

$$\Delta w_{ij}^{N}(t) = \frac{\eta}{2}(t_i^N - y_i^N)\, x_j^N \tag{2.28}$$

A esta se le denominada la **regla del perceptrón**. Hay que llegar a un acuerdo para definir el valor de variación η para que el aprendizaje no sea lento y poco práctico; o muy rápido y que no refleje adecuadamente el aprendizaje. Cabe mencionar que, siendo las entradas y salidas discretas, el resultado del valor de los pesos será de la misma manera y no tendrá un valor mayor a η.

Adaline (ADAptative LInear NEtwork)

El modelo ADELINE (red lineal adaptativa) es un modelo de RNA similar al perceptrón dado que utiliza también un aprendizaje por corrección de errores, pero posee respuesta lineal para entradas que pudiesen ser continuas, a diferencia del Perceptrón que utiliza funciones discretas; además el proceso de entrenamiento es distinto. Para una RNA ADALINE se utiliza la regla de aprendizaje Widrow-Hoff [35] que es mejor conocida como regla LMS (Least Mean Square, por sus siglas en inglés, lo que se podría traducir como mínimos cuadrados), De ella deriva la Regla Delta.

Regla Delta

Como se mencionó en la última línea del párrafo anterior, la regla delta es un caso derivado de la regla de aprendizaje Widrow-Hoff y que se utiliza para el aprendizaje del sistema de RNA tipo ADALINE. Se observó que la *regla de Hebb* puede almacenar pares de patrones de vectores perpendiculares entre sí, estableciendo una ratio de aprendizaje que le permite corregir y depurar errores; *la regla de mínimos cuadrados* produciría pares de vectores adecuados cuando se cumpla la condición de ser linealmente independientes, lo que llevaría a un conjunto o matriz de pesos optimizados.

Como se verá a continuación, la deriva de la regla LMS o *mínimos cuadrados* de una función optimizada denominada *coste* o *error* permite la obtención de algoritmos que pueden aplicarse en el campo de la computación neuronal.

Aprendizaje derivado de la optimización de una función de error o coste

Al proponer una función de error o coste que monitorice el desempeño de una RNA, se busca una función dependiente de los valores de los pesos sinápticos presentes en la RNA, quienes pueden ser optimizados de forma iterativa con el propósito de obtener un conjunto de valores correspondientes a mínimos de la función. Esta operación generalizará un método de optimización que lleve al conocimiento de una regla para obtener los valores adecuados que deben tener los pesos sinápticos en la red para alcanzar los objetivos de aprendizaje.

La técnica de optimización más extendida es la del *descenso por el gradiente* [3x cita]. El primer paso en esta técnica es definir una función E dependiente de los valores W de los pesos $E(W)$, donde E: R^n R que representa una superficie que posee picos y valles como se observa en la Fig. 2. 18. Un valle en la superficie representa W^* un mínimo local óptimo, donde el objetivo principal del aprendizaje es hallar el conjunto de pesos que coinciden con el mínimo global de la función de error, aunque este cabe mencionar que, en presencia de redes de propósito general los mínimos alcanzados sean locales y no globales pero que sean lo suficientemente buenos para el alcanzar los objetivos de aprendizaje.

Luego de definir la función de error E, y para descubrir el conjunto de pesos ideales que corresponden al mínimo global de la función se inicia el descenso por el gradiente desde un punto W_0 para t=0, observando el sentido de máxima variación, que equivale al máximo gradiente, de la función de error $E(W)$ en W (0) que viene dado por el gradiente en ese punto. Este resultado apuntará hacia los picos de la función, por lo que se invierte el sentido para llegar al mínimo local luego de varias iteraciones [33].

Este proceso se puede representar mediante la siguiente ecuación

$$W(t+1) = W(t) - \frac{\eta}{2}\nabla E(W) \tag{2.29}$$

Donde $\eta=2$□ , puede variar según el peso, indica el tamaño de la muestra en cada iteración, que debe escogerse con cuidado ya que una muestra muy pequeña haría el proceso muy lento; en cambio, una muestra muy grande impediría que la iteración logre converger hacia el punto en el cual se encuentra el mínimo buscado.

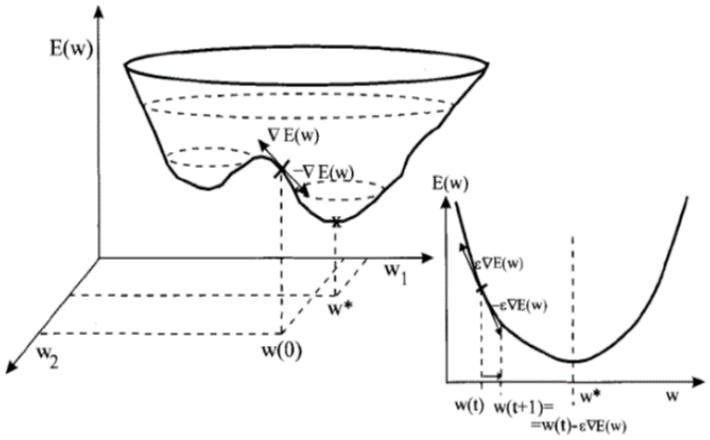


Fig. 2.18. Superficie de la función de error E(W) donde se encuentra el espacio de los pesos.

Si se tiene el conjunto de los pesos W: $\{w_{ij}\}$ y como se sabe, cada iteración tiene el propósito de acercar los pesos al mínimo de la función, al calcular la variación de la función *E(W)* para una iteración *t* se tiene que:

$$\delta\left(E\left(w_{ij}\right)\right) = \sum_{ij} \frac{\partial(E(\sum_{ij} w_{ij}))}{\partial w_{ij}} \delta w_{ij} \tag{2.30}$$

Y conociendo que $-2\eta=-$□ que es la variación de los pesos, entonces multiplicando por el gradiente

$$\delta\left(f\left(w_{ij}\right)\right) = \sum_{ij} \frac{\partial(E(\sum_{ij} w_{ij}))}{\partial w_{ij}} \left(-\varepsilon \frac{\partial(E(\sum_{ij} w_{ij}))}{\partial w_{ij}}\right) \tag{2.31}$$

Lo que nos resulta en

$$\delta\left(f\left(w_{ij}\right)\right) = -\varepsilon \sum_{ij} \left(\frac{\partial (E(\sum_{ij} w_{ij}))}{\partial w_{ij}} \right)^2 \leq 0 \tag{2.32}$$

Indicando que la variación siempre será menor a cero por lo que siempre tenderá a disminuir. Este procedimiento asegura al menos hallar un mínimo local.

Si en vez de una función de activación lineal se utiliza una función sigmoidea se tendrá entonces el algoritmo de la *Regla Delta* [35]. La actualización de los pesos tomando en cuenta los conjuntos de referencias **N** y **t** para el aprendizaje supervisado, conociendo que la variación vista (2.26) es igual a la regla del perceptrón y que indica la ratio de aprendizaje, se obtendrá la ecuación

$$w_{ij}(t+1) = w_{ij}(t) + \Delta w_{ij} = w_{ij}(t) + \frac{''}{2}(t_i^N + y)x_j^N \tag{2.33}$$

Que es a la vez la generalización de la regla LMS o de mínimos cuadrados vista anteriormente [35].

Perceptrón Multicapa

Anteriormente se observó que el perceptrón simple posee limitaciones importantes en su diseño; sin embargo, si se le añade conexiones intermedias con nodos o sinapsis a neuronas entre aquellas neuronas que son de entrada y aquellas que son de salida, se tendrá lo que se denomina *Perceptrón Multicapas* [33]. Las neuronas que son intermedias son las que forman la denominada *capa oculta*.

Un perceptrón Multicapa (PMC) utiliza por lo general para su entrenamiento el algoritmo denominado de retro propagación de errores o *Back Propagation (BP sus siglas en inglés)* y que, a través de sus variantes suele discriminarse la acción del algoritmo de BP y el PMC combinada como la red de retro propagación; aunque en alguno caso solo se identifique por el método.

Según Hecht-Nielsen [37] el inicio del método se aprendizaje inicia con la tesis doctoral de Werbos [34] de 1974 pero el hecho no trascendió en la comunidad científica de la época. Fue una década después el algoritmo BP fue redescubierto por David Parker y por el equipo PDP conformado por los doctores Rumelhart, Hinton, MacClelland [38] quienes lo pusieron en el radar de la investigación en redes neuronales y cuyo uso ahora es generalizado.

Para la época de Werbos, la capacidad de cómputo no era lo suficientemente potente como para realizar prácticas en problemas de interés. Fue ya a mediados de los ochenta cuanto esta dificultad fue resuelta y con ello las limitaciones que impedían un mejor estudio del método. Con el apoyo de grupo PDP el interés sobre esta técnica para la resolución de problemas complejos fue despertado en la comunidad internacional. La estructura del perceptrón multicapa se observa en la Fig. 2.19 junto con una función de activación sigmoidea.

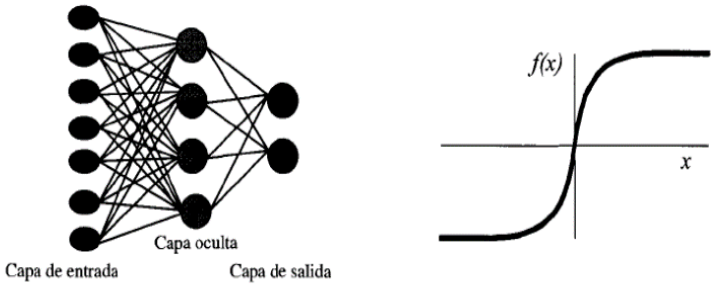


Fig. 2.19. Perceptrón Multicapa y su función de activación.

Se x_i los valores de las entradas de la red, y_j las salidas procedentes de la capa oculta y z_k las salidas de la capa final; t_k representa las salidas que se buscan como objetivo y w_{ij} los pesos de la capa de salida y θ_j los umbrales, w'_{kj} los pesos de la capa de salida, θ'_k sus umbrales obtenidos en la operación y que se debe corregir.

La expresión matemática para un PMC con una capa oculta y neuronas con salida lineal es la siguiente

$$z_k = \sum_i w'_{kj}\, y_i - \theta'_i = \sum_i w'_{kj} f\Big(\sum_i w'_{ji}\; x_i - \theta_j\Big) - \theta'_i \qquad (2.34)$$

Donde la función *f ()* es de tipo sigmoideo como las que se indican en la Tabla 2. Y como aparece en la Fig. 2.9. Ejemplo

$$f(x) = \frac{1}{1+e^{-x}} \qquad (2.35) \qquad \text{y} \qquad f(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}} = \tanh(x) \qquad (2.34b)$$

Teniendo con estas funciones las salidas entre los intervalos [0, +1] para la primera función, y [-1, +1] para la segunda.

En líneas generales, esta es la configuración tradicional del PMC; sin embargo, existes variaciones al modelo que incluyen: funciones no lineales en la capa de salida, como el ejemplo visto en (2.34a) y que es utilizado ampliamente en problemas de clasificación; más capas ocultas; el empleo de funciones de activación alternativas; introducción de retroalimentación en la configuración del sistema o dependencias temporales entre otras variantes que se han ensayado [34].

Perceptrón Multicapa como aproximador de funciones

Durante los años que ha sido objeto de investigación a los cuales se ha sometido al perceptrón a múltiples pruebas de operación y de resolución de problemas, se ha podido coludir que, sobreponiéndose a las limitaciones iniciales de perceptrón monocapa, el perceptrón multicapa ha demostrado poseer una capacidad robusta en la representación de *mapas* y de realizar complejas clasificaciones de una manera eficaz y con el empleo de pocos recursos relativamente.

Sin embargo y a pesar de las ventajas que muestran las RNA tipo PMC no se conocía de forma teórica el por qué, una demostración teórica que mostrase de forma clara esas potencialidades computacionales. Ese camino se inició con las demostraciones de McCulloch y Pitts [21] sobre la capacidad del PMC de modelar funciones booleanas; luego Denker [32] pudo demostrar que una red unidireccional multicapa con una sola capa oculta

puede representar toda función booleana; simultáneamente, Richard Lippmann [39] llegó a la conclusión que con una red tipo PMC, con dos capas ocultas, se puede representar cualquier región de complejidad arbitraria, incluso puede representar funciones boolenas y no booleanas, sin reparar en la arbitrariedad de la misma [40].

Por otra parte, Kolmogorov [41] teorizó y demostró que, en una arquitectura similar a un PMC, con una capa oculta, dicho sistema puede ser un aproximador universal de funciones; de igual manera Hornik [42] y Funahashi [43] llegaron a la misma conclusión una década después.

Aunque existen diversas demostraciones sobre las afirmaciones antes mencionadas, no se desarrollará las mismas en este libro, para quienes estén interesados en profundizar en ellas, se puede acceder a la literatura citada [41, 42, 43].

Conociendo que existen demostraciones teóricas que confirman lo hasta ahora expuesto, se puede indicar que *el PMC con una capa oculta puede modelar de manera satisfactoria una función continua dentro de un intervalo, si se cumplen ciertas condiciones iniciales*. Esto nos lleva a concluir además que las RNA multicapa también son modeladores o aproximadores de funciones. Por tanto, si se tiene que:

$$s'_k = \sum_j w'_{kj}\, y_i - \theta'_i = \sum_j w'_{kj}\, f\Big(\sum_i w_{ji}\, x_i - \theta_j\Big) - \theta'_i \quad (2.35)$$

Donde se observa que s'(x) que representa un PCM es una función s(x) constituida por funciones sigmoideas f(x), lo que tiene muchas características en común con las funciones continuas y periódicas que se desarrollan en las series sinusoides como las de Fourier [44]. Estos soportes teóricos en conjunto con los teoremas a los cuales se ha hecho mención (Hornik, Funahashi, etc.) han incidido de forma definitiva en forma tanto funcional como filosófica de concebir a las redes neuronales artificiales. Esto no quiere decir que no existan dificultades que deben ser resueltas con respecto al modelo adoptado para desarrollar redes neuronales artificiales. En primera instancia existe el problema de sobre aprendizaje debido a que no existe un claro concepto en los teoremas acerca de las capas ocultas que son necesarias para

abordar la solución a determinado problema. Esta excesiva amplitud sobre lo abarcable en el modelado de funciones puede generar el caso donde, al no conocer la cantidad de capas ocultas necesarias para aproximar una función, la misma pueda ser inabordable e indefinible. Estos casos están en la palestra actual y son objeto de un estudio importante [15, 44].

Aprendizaje mediante retro propagación de errores (BP)

De igual manera que el perceptrón simple, el PMC aprende el valor adecuado de sus pesos mediante una regla de ajuste de errores, con lo cual se busca que los valores de salida de la red se correspondan con las salidas que idealmente se desean o en su defecto, que sean lo más próximas posibles.

Como se ha mencionado, el PMC utiliza en su proceso de aprendizaje el método de descenso por el gradiente para ajustar los pesos de la red. Estos ajustes se realizan en la capa salida de la red, según la desviación producida entre el valor entregado y el deseado; este error se propaga hacia las capaz anteriores hasta llegar a la capa de los valores de entrada. De aquí su nombre, *retro propagación*.

El proceso para minimizar el error consta de los siguientes pasos:

Se aplica un conjunto de valores de entrada que siguen un patrón determinado y que serán procesados por la red hacia adelante capa por capa. La red tendrá una respuesta a dicho conjunto de datos, lo que significa que se calcula los valores de la ecuación:

$$s'^{n}_{k} = \sum_{j} w'_{kj}\, y^{n}_{j} - \theta'_{k} = \sum_{j} w'_{kj}\, f\Big(\sum_{i} w_{ji}\, x^{n}_{i} - \theta_{j}\Big) - \theta'_{k} \quad (2.36)$$

Para un conjunto de pesos determinados, por lo general son valores aleatorios pequeños.

El siguiente paso es realizar la propagación del error calculado hacia las capas anteriores según la regla de ajuste del error, que compara la respuesta obtenida de la red con la respuesta que se desea obtener. El detalle de este proceso es el siguiente

Para cada valor *n* del conjunto referencia o patrón de aprendizaje

Se ejecuta el valor n-ésimo del patrón de referencia según la fórmula (2.36).

Calcular el valor de la señal de error mediante la expresión

$$\delta w_{kj} = \frac{\eta}{2}\sum_{n} \Delta'^{n}_{k} y'^{n}_{j}\,, con\ \Delta'^{n}_{k} = [t^{n}_{k} - f(v'^{n}_{k})]\frac{\partial f(v'^{n}_{k})}{\partial v'^{n}_{k}} \quad (2.37)$$

$$\partial w_{kj} = \frac{\eta}{2}\sum_{n} \Delta'^{n}_{i} y'^{n}_{j} \quad con\ \Delta^{n}_{j} = (\sum_{n} \Delta'^{n}_{k} w'_{kj})\frac{\partial f(v^{n}_{j})}{\partial v^{n}_{j}} \quad (2.38)$$

Se calcula el incremento parcial de los pesos y umbrales de cada elemento *n* del conjunto de referencia (2.37) (2.38).

Se calcula el incremento total para todos los elementos del conjunto de referencia *n* de los pesos y umbrales y (2.37) (2.38).

Se actualiza pesos y umbrales.

Calcular el actual que se rige por el error cuadrático medio según (2.38), luego en *t=t+1*, volver al paso 1 si la iteración no ha cumplido con los patrones de referencia.

$$E(w_{ji}, \theta_j, w'_{kj}, \theta'_k) = \frac{1}{2}\sum_{n}\sum_{k}\left[t^{n}_{k} - f(\sum_{j} w'_{kj} y^{n}_{j} - \theta'_k)\right]^2 \quad (2.40)$$

Se puede resumir de la siguiente manera el procedimiento de retro propagación. Los *pesos iniciales* deben ser valores pequeños y aleatorios entre negativos y positivos, nunca nulos. En el procedimiento presentado se nota que la utilización de la técnica de descenso por el gradiente surge de forma necesaria, donde se da la ejecución inicial para cada uno de los valores del patrón de entrenamiento o de referencia, luego se calcula los valores de desviación de los pesos debido a cada patrón, se suman y es entonces cuando se realiza la actualización de los pesos. A este procedimiento se le denomina *aprendizaje por lotes* o *Batch Learning* y es el procedimiento natural del apren-

dizaje por BP. Existen variaciones de este método, debido a que a medida que el procesamiento de datos se hace más extensivo, el método por lotes presenta muchas dificultades debidas al tiempo empleado para la computación de valores y la muestra de resultados. Por ello existen la variante *on-line* que es la actualización de los valores de los pesos por cada iteración de la regla. La ventaja de esta variante en su capacidad de trabajar con grandes bases de información, donde mucha de ella es redundante. Si se realizara un proceso de aprendizaje de 50000 datos de entrada como referencia, con el sistema por lotes tardaría 50 veces en procesar toda la información para el entrenamiento.

Redes no supervisadas

Las redes no supervisadas carecen de patrones de referencia como sucedía en el caso de las RNA supervisadas. Estas poseían de lo que se denominaba conjunto referencia o patrón. En el caso de las RNA que carecen de supervisión existe los que se denomina *proceso de autoorganización*. En este modelo las relaciones de similitud, es decir, en qué se parecen las diversas entradas al sistema, es en lo que se fundamentarán los resultados para conocer los valores en la salida que son los adecuado para el mismo. Las aplicaciones más extendidas de este tipo de arreglos neuronales son la *minería de datos*, *el agrupamiento de patrones* y *visualización*. Luego de introducidos los modelos indicados haciendo una panorámica de sus rasgos más importantes, se introducirá la arquitectura, operación y reglas de aprendizaje de los *Mapas de Autoorganizativos de Kohonen* [45], uno de los modelos autoorganizados más estudiados y más utilizados a nivel práctico.

En una red no supervisada como la mostrada en la Figura. 2.16. A la red no se le suministra información externa sobre las funciones de entrada/salida, sino que la misma red se autoorganiza para reconocer los patrones o pautas de dichas funciones.

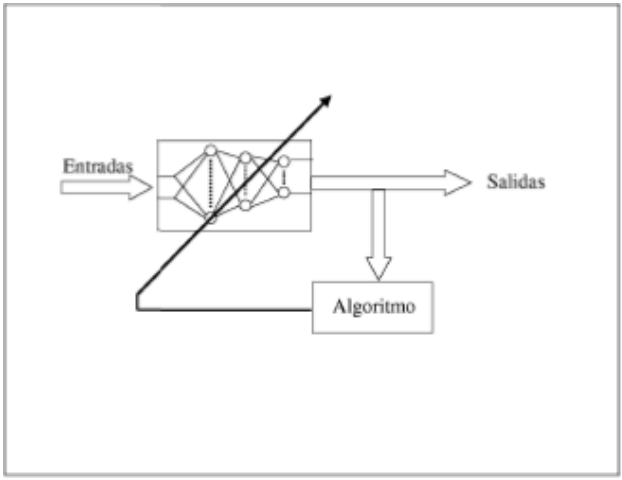


Figura. 2.16. Red neuronal no supervisada.

A lo denominado autoorganización en este sistema, solo es el resultado probabilístico que produce la red a partir de los datos suministrados en la entrada de la red, la cual que genera un tipo de salida de la misma. A medida que más información en la entrada se va procesando, del mismo modo la red va descubriendo patrones, extrayendo y agrupando características similares, no necesita de un *guía* ajeno a la red para obtener resultados adecuados, ya que el objetivo no es tender a uno, sino descubrir patrones, encontrar similitudes y de ellas marcar sus propias referencias. Para resumir lo hasta ahora dicho, se tiene que, dependiendo de su arquitectura, las redes no supervisadas pueden:

Clasificar valores de salida cuya naturaleza es discreta [0, 1], para luego agruparlos según ciertos criterios de categorización que son los que indican a cuál grupo pertenecen. Dichos criterios son descubiertos por la propia red durante los ciclos de aprendizaje, luego estos se organizan en función de las distancias entre los patrones obtenidos.

Hacer generalizaciones del caso anterior, al que se le denomina *memoria asociativa*, la cuál es la representación generalizada de redes por agrupamiento

donde no solo se obtienen 1 o 0 en la salida, sino que se obtiene un vector que es representativo de la categoría o grupo.

Hacer *etiquetas* en lugar del vector representativo de la clase, esto permite reducir la cantidad de bits necesarios para salvaguardar la información relevante.

Analizar similitudes entre patrones en una neurona con salida continua, comparando los grados de similitud actual con los registrados en procesos anteriores; esto se registra en los pesos sinápticos de la neurona y le indica que es un patrón típico dentro de un conjunto de salida. Si se amplía a varias neuronas cuyas salidas son continuas, entonces se opera sobre un espacio en donde se encuentras los valores de entrada que producen los vectores que rigen a los pesos sinápticos de todas las neuronas, resaltando las características más importantes del espacio sensorial. A esto se le llama *análisis de componentes principales.*

Mapeo del espacio sensorial de entrada, realizando una proyección donde las neuronas se organizan de forma geométrica (como en el caso de una matriz) y cuyo objetivo es conservar la topología de la mejor manera en unas dimensiones más pequeñas.

Métodos de aprendizaje no supervisados

Los sistemas neuronales no supervisados se encuentran representados en tres grupos según el método que emplean para actualizar las características internas de la red. Se mencionan entonces:

Los que se basan en la regla de Hebb. Dichas redes tienen como característica distintiva la activación simultánea de un número elevado de neuronas de salida. Por ejemplo, las redes tipo *Principal Component Analysis* (PCA) quienes realizan análisis de componentes principales [46] pertenecen a esta clasificación.

Las que se basan en modelos competitivos. En este modelo la competencia entre neuronas produce que sólo una neurona, o un conjunto específico de ellas, se active. Este comportamiento se basa en las teorías de Von

der Malburg [47]. Cuando la neurona o el conjunto de vecinas ganadoras se activan, se refuerzan sus conexiones sinápticas; este comportamiento competitivo es elemental en muchos modelos de redes neuronales artificiales; Se puede observar en las redes *adaptive resonace teory* (ART) o redes que funcionan bajo la teoría de resonancia adaptativa [48], las de tipo Neocogitrón de Fukushima [49] y los mapas autoorganizados de Kohonen; todas ellas realizan agrupamiento por patrones, que convergen en los denominados *clusters*, que son caracterizadores de patrones.

Los basados en las teorías de la información. Linsker [50] en su trabajo de 1988 adelantó un modelo para maximizar la cantidad de información que se preserva dentro del sistema al momento de realizar el procesamiento de los datos, tratando de minimizar la *entropía* en la red; esto permite realizar el análisis de componentes principales como las redes con método competitivo.

Mapas Autoorganizativos de Kohonen

Kohonen propuso un sistema que ayuda a trabajar con conjuntos de datos de entrada de complejidad dimensional importante. Las complejas relaciones estadísticas entre los valores de entrada se transforman en relaciones geométricas más manejables dentro de un espacio dimensional menor. Como se aprecia en la Figura 2.17, se tiene una malla bidimensional de *N* neuronas con un vector w_j (j= 1, 2, …, m) que tomará valores en el mismo espacio de los valores de entrada.

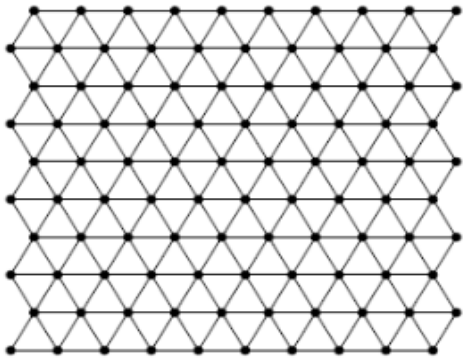


Figura 2.17. Malla bidimensional donde se ubican las neuronas antes del entrenamiento.

Luego de realizadas varias iteraciones con los valores de entrada, la malla va organizándose según los patrones que ha descubierto de los valores introducidos al sistema. En la Figura 2.19 se puede por ejemplo observar la malla una vez realizado varios ciclos de entrenamiento según la información procesada.

Figura 2.18. Malla ya ejecutados varios ciclos de iteraciones de entrenamiento.

Una de las diferencias más notables de los mapas autoorganizativos en comparación con otros sistemas de aprendizaje por competencia es que se actualizan los pesos de manera general y no solamente de la neurona ganadora. Si se tiene un vector de entradas x^n= {1,2, 3, …, n), la neurona ganadora será aquella que se aproxime más al valor de entrada, lo que significa que gana la neurona cuya salida sea de tal manera que su desviación sea la menos alejada del valor de entrada, lo que equivale a la menor distancia euclidiana. Esto puede escribirse de la siguiente manera:

$$d(\boldsymbol{w}_g, \boldsymbol{x}) = \boldsymbol{min}_{\boldsymbol{ij}}\{d(\boldsymbol{w}_{\boldsymbol{ij}}, \boldsymbol{x}) \tag{2.41}$$

Una de las diferencias principales del mapa autoorganizativo frente a otros sistemas de aprendizaje competitivos, es que en cada paso del aprendizaje se actualizan no sólo los pesos de la neurona ganadora **g**, sino también los pesos del resto de neuronas de la malla. Esto se puede escribir de la siguiente manera:

$$w_{ijk}(t+1) = w_{ijk}(t) + \eta(t).h(|\boldsymbol{i} - \boldsymbol{g}|, t).\left(x_k(t) - w_{ijk}(t)\right) \quad (2.42)$$

Donde $\eta(t)$ es el denominado índice de aprendizaje, $h(t)$ la función de vecindad que establece cuáles son las neuronas vecinas de la neurona ganadora; todo lo indicado representa un procesamiento matricial *mxm* de nodos elementales **i**(i, j), de tipo bidimensional como se observa en la Figura 2.19 donde x(t) □ R^n son las entradas dentro del espacio sensorial y los pesos vienen dados por $w_{ij}(t)$, donde $w_{ij}(t)$ □ R^n que funge de referencia primaria para las posteriores iteraciones. La diferencia entre **i** y **g** representa la distancia entre la neurona de referencia y la neurona ganadora. Cuando la neurona ganadora no pertenece al entorno de vecindad de la neurona referencial el resultado es nulo, y por ende los pesos no se actualizan si ocurre el caso contrario su valor es positivo, y se ejecuta la actualización de los pesos. Este proceso se ejecuta hasta que se logra actualizar la neurona ganadora, lo que implica que la vecindad se va reduciendo a medida que se ejecutan las iteraciones necesarias para lograr ubicar una única neurona ganadora definitiva.

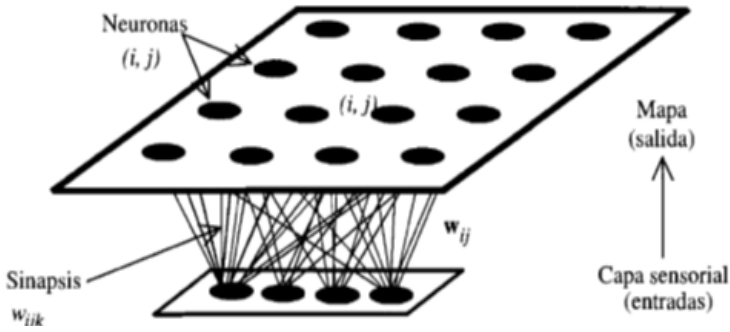


Figura 2.19. Configuración de los mapas autoorganizativos.

Proceso de ejecución de algoritmo autoorganizado

Aunque no existe una única manera para ejecutar el proceso de aprendizaje para un algoritmo autoorganizado, los resultados tienden a ser bastante independientes de los medios utilizados; sin embargo, se intentará resumir dicha operación de la siguiente manera [12, 45]:

Se inicializan los pesos sinápticos w_{ijk}. Esto se puede llevar a cabo de diversas maneras; sin embargo, los más habitual y de práctica extendida es inicializar los pesos con un valor positivo pequeño en t=0.

Tomando en cuenta que se toman las muestras iniciales **x**(t) de un espacio definido como *espacio sensorial* p(x) de entrada, se pueden presentar a la red cualquier valor aleatorio de este espacio por cada iteración. En el caso poco frecuente de tener varios espacios sensoriales, debe alternarse los valores aleatorios de todos los espacios presentes como entradas.

Las neuronas **i**≡ (i, j) calculan sus pesos sinápticos $\mathbf{w}_{ij}$ y el resultado de la operación del vector **x** de entrada. Para medir la similitud de los resultados se emplea por lo general la denominada distancia euclídea según la siguiente formula:

$$d^2(\boldsymbol{w}_{ij}, \boldsymbol{x}) = \sum_{k=1}^{n} (w_{ijk} - x_k)^2 \tag{2.43}$$

Una vez hechas las comparaciones pertinentes, se ubica la distancia menor y con ello la neurona ganadora **g**.

Se procede a la actualización de los valores de los pesos sinápticos de la neurona ganadora según el criterio de la ecuación 2.43.

Se repite el proceso desde el paso 2 si no se ha llegado al número de iteraciones posibles; de lo contrario finaliza el proceso de aprendizaje.

El número de iteraciones necesarias para entrenar una red autoorganizada dependerá en primera instancia de la cantidad de neuronas a entrenar; por lo general para las redes de tipo autoorganizadas como los mapas de Kohonen, se necesitan en las simulaciones computarizadas al menos 100000 iteraciones. Es importante no olvidar que deben ser compatibles en su métrica tanto el criterio de similitud como la regla de aprendizaje. Esto puede notarse en la ecuación 2.43 como en la 2.42. Si en las fases de recuerdo y de actualización de pesos los criterios son distintos, el desarrollo del mapa como lo dice Demartines [51] será verá comprometido.

Una medida que podría ser equivalente a la distancia euclídea es la denominada *producto escalar*

$$C_{ij} = \sum_{k=1}^{n} w_{ijk}\, x_k \tag{2.44}$$

Que puede incluirse en el proceso de ejecución junto con (2.42). sin embargo, se debe tener mucho cuidado de normalizar los vectores según la norma 1 donde la distancia euclídea y el producto escalar coinciden. Si se utiliza la distancia euclídea (2.43) y la regla de actualización de pesos (2.42) *no es necesario normalizar*. Se normaliza si un caso particular lo requiere como el mantenimiento dinámico de un conjunto de valores de entrada.

Modelos de mapas autoorganizados

Aparte de la regla euclídea o la de producto escalar para la actualización de los pesos del sistema, existen otros modelos que solo se mencionarán en la Tabla 4 como información de interés.

Tabla 4. Modelos de mapas autoorganizados

MODELO DE NEURONA	FUNCIÓN DISTANCIA	REGLA DE APRENDIZAJE
1) Manhattan	$d(\mathbf{w}_{ij},\mathbf{x}) = \sum_{k=1}^{n} \lvert w_{ijk} - x_k \rvert$	$\Delta w_{ijk}(t) = \alpha(t).sign\left(x_k(t) - w_{ijk}(t)\right)$
2) Produc. escalar		
2a) $\lVert x \rVert$=cte=1	$d(\mathbf{w}_{ij},\mathbf{x}) = \sum_{k=1}^{n} w_{ijk} x_k$	$\Delta w_{ijk}(t) = \alpha(t).\left(x_k(t) - w_{ijk}(t)\right)$
2b) $\lVert x \rVert$≠cte.	$d(\mathbf{w}_{ij},\mathbf{x}) = \sum_{k=1}^{n} w_{ijk} x_k$	$w_{ijk}(t+1) = \frac{w_{ijk}(t) + \alpha(t).x_k(t)}{\lVert \mathbf{w}_{ij}(t) + \alpha(t).\mathbf{x}(t) \rVert}$
2c) $\lVert x \rVert$≠ cte.	$d(\mathbf{w}_{ij},\mathbf{x}) = \sum_{k=1}^{n} w_{ijk} x_k \equiv y_{ij}$	$\Delta w_{ijk}(t) = \alpha(t).\left(x_k - y_{ij}(t).w_{ijk}(t)\right)$
2d) $\lVert x \rVert$≠ cte.	$d(\mathbf{w}_{ij},\mathbf{x}) = \frac{\sum_{k=1}^{n} w_{ijk} x_k}{\lVert \mathbf{w}_{ij} \rVert . \lVert \mathbf{x} \rVert}$	$\Delta w_{ijk}(t) = \alpha(t).\left(x_k(t) - w_{ijk}(t)\right)$
3) Euclídea	$d^2(\mathbf{w}_{ij},\mathbf{x}) = \sum_{k=1}^{n} \left(w_{ijk} - x_k\right)^2$	$\Delta w_{ijk}(t) = \alpha(t).\left(x_k(t) - w_{ijk}(t)\right)$

Otros tipos de redes neuronales

Redes neuronales retroalimentadas

Las redes neuronales retroalimentadas, tienen una diferencia fundamental que las separa de las redes unidireccionales observadas hasta el momento. En las redes unidireccionales la información se propaga desde el conjunto de entrada hasta el conjunto de salida. En las redes de neuronas retroalimentadas, la situación se vuelve más compleja, debido a que las neuronas reciben información de la salida para luego inyectarla posteriormente en el conjunto de entrada, obteniendo una propagación de información tanto hacia adelante como hacia atrás de la red. Esto introduce un grado de complejidad que las redes neuronales unidireccionales no posee y por tanto para las redes retroalimentadas en crítico garantizar la estabilidad de la respuesta que emita.

Se han desarrollado teoremas que definen las condiciones que debe reunir una red retroalimentada para que sea estable [52][53] y la demostración se apoya en la *función V de Lyapunov*. La función parte de la condición que, para un sistema dinámico de variables de entrada con un sistema de ecuaciones que cumple que para x_i: $\{x_1, x_2, \ldots, x_n\}$ descrito por un sistema de ecuaciones parciales de la forma

$$\dot{x}_i \equiv \frac{dx_i}{dt} = F(t, x_1, x_2, \ldots, x_n) \tag{2.45}$$

Donde sólo en el origen el sistema se encuentra en reposo y las derivadas de las funciones que la determinan existen en todo el dominio y las variables se encuentren definidas y acotadas, entonces puede buscarse una función V: de Lyapunov de la forma

$$V = \sum_{i=1}^{n} \frac{\partial V}{\partial x_i} \leq 0, \forall x_i \tag{2.46}$$

Que sea convergente para todas las entradas posibles (x_1, x_2, …, x_n) y que además sea estable de forma global.

La función de Lyapunov es una generalización del concepto físico de la energía, que establece que si se encuentra una función de energía que dis-

minuya al ser operada, esta es estable. Este hecho ayuda a estudiar los problemas de estabilidad de una manera más sencilla. De esta propiedad se apoyó Hopfield [20] para proponer una variante de red neuronal retroalimentada.

Red Neuronal Tipo Hopfield

La red neuronal de tipo Hopfield es una red de una sola capa que se interconecta entre si como se aprecia en la figura 2.20. Las señales de salida que se obtienen en un tiempo t se redirigen a la entrada de cada neurona convirtiéndolas en valores de entrada dentro de un tiempo t+1.

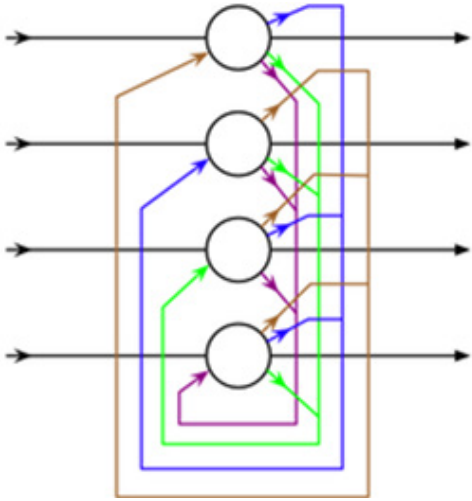


Figura 2.20. Red neuronal tipo Hopfield.

Como se observa, el núcleo de la red es un conjunto de neuronas en una sola capa como lo puede ser el perceptrón, incluso la red está siendo definida por una función de entrada tipo binaria, con salidas del mismo tipo . Se puede entonces indicar que los valores a la entrada pueden ser {0, 1} o {-1, 1}. Cada neurona calcula el potencial postsináptico sumando las entradas y los pesos relacionados, menos un valor de umbral de activación. Esto se expresa de la siguiente manera:

$$h_i(t) = \sum_j w_{ij} x_j(t) - \theta_i \tag{2.46}$$

Por último, al valor obtenido de la neurona se le aplica la función escalón para obtener la salida de tipo digital.

$$y_i(t) = f(h_i(t)) = f\left(\sum_j w_{ij}\, x_j(t) - \theta_i\right) \tag{2.47}$$

Si lo valores de entrada a las neuronas son {-1, 1}, el valor positivo correspondería a un estado de activación para la neurona, y el caso negativo sería una inhibición equivalente y opuesto. En la primera iteración para t=0, las neuronas recibirán el primer estímulo del exterior, el proceso dará como resultado unas salidas que son reenvidas a las entradas como valores . La operatividad de la red tipo Hopfield y su dinámica se puede entonces describir como

$$x_i(t+1) = f\left(\sum_j w_{ij}\, x_j(t) - \theta_i\right) \tag{2.48}$$

A partir del momento en que es retroalimentada las salidas como nuevas entradas, ya la red no recibe información del exterior hasta tanto la red llegue a un punto de equilibrio donde la entrada en igual a la salida del sistema. Se dirá en general que el estado **x** de la red neuronal de Hopfield en un tiempo t, está condicionado por el estado de todas las neuronas que la constituyen. Esto se expresa como

$$\boldsymbol{x}(t) = (x_1(t) \ldots\ x_i(t) \ldots\ x_n(t))^T \tag{2.49}$$

La neurona, como elemento básico de la red neuronal de Hopfield puede representarse como en la Figura 2.21, semejante a otras representaciones antes mencionadas, pero teniendo en cuenta la retroalimentación

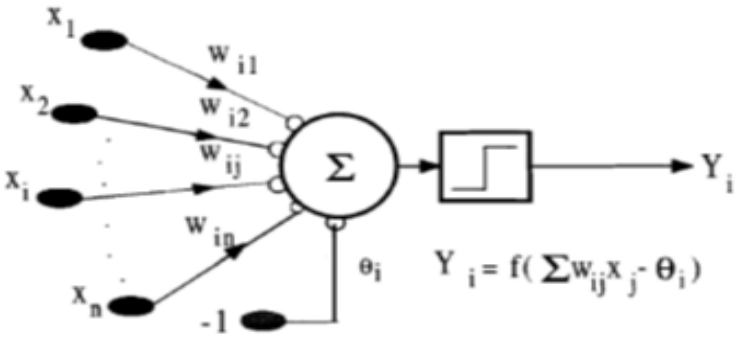


Figura 2.21. Neurona de una red de Hopfield con valor umbral y función de activación tipo escalón.

La forma de actualización del estado de la red hará que la misma exhiba un comportamiento u otro, lo que quiere decir que el estado final de la red depende directamente de su dinámica de actualización; entonces dependerá de la implementación la dinámica de la misma y esta puede ser de dos maneras: **síncrona o asíncrona**. En el primer caso las neuronas se van activando una a la vez de forma aleatoria o siguiendo una directiva de activación. En el caso síncrono, un conjunto o todas las neuronas se activan a la vez, el caso más extendido es cuando se activan todas al mismo tiempo.

Cuando la red de Hopfield llega a su estado estable, es decir que se cumple que

$$\boldsymbol{x}(t+1) = \boldsymbol{x}(t) \equiv \boldsymbol{x}^* \tag{2.50}$$

La salida de la red ya no cambia, lo que indica que la red a alcanzado su punto de equilibrio, estimando con esto que ya la red neuronal terminó de procesar la información que se recibió del exterior, es decir, en , siendo la salida final del sistema. A este estado se le relaciona con la *memoria asociativa* de la red y posee ventajas prácticas importantes.

Al realizar el entrenamiento de esta red se puede hacer que *recuerde* ciertos

patrones de entradas como estados estables de la red, de tal manera que para determinados valores **x** de entrada, el resultado de salida sea alguno de los patrones $\mathbf{x}^N$ memorizados. Esta asociación sirve para que cuando a la red se le presente una entrada $\mathbf{x}^N$ + *ruido* pueda acercarse al estado estable más próximo memorizado que sería $\mathbf{x}^N$. Esta característica de la red nos podría ayudar a recuperar datos que por alguna razón se han visto dañados (con ruido) y a través de la red recuperar el original ($\mathbf{x}^N$ sin ruido).

CAPÍTULO 3:

Aplicaciones de la Inteligencia Computacional

Los intereses de las sociedades humanas son muy diversos y siguen derroteros muy distintos. Estos intereses engloban todas las actividades que son posibles imaginarse y cada una de ellas demandan cada día una mayor especialización tanto en ejecución como en gestión. Para aquellas actividades que son muy demandadas y que sin distingo de origen o estatus son necesidades básicas de todo ser humano como la salud, la educación y la seguridad, las exigencias cada día son mayores y el uso de herramientas tecnológicas cada día se vuelve más común e indispensables.

La informatización de la salud, la educación y la seguridad entre muchas otras no es sino una consecuencia natural que responde a la complejidad de la sociedad cambiante de los actuales momentos. De la misma manera otras disciplinas como la economía, que necesitan manejar datos en cuanto a desempeño, oportunidad y dirección de recursos, requieren de sistemas computacionales que les brinden el soporte y la confianza necesaria para poder tomar las mejores decisiones posibles.

Ejemplos sobre el uso de sistemas computacionales inteligentes para mejorar el diagnóstico de enfermedades, para predecir el comportamiento de ciertos ecosistemas que se ven afectados por la actividad humana o para determinar cuáles métodos son más eficientes al momento de impartir conocimientos en educación existen muchos; sin embargo el objetivo de este capítulo no es realizar un compendio de trabajos que muestren las aplicaciones de la inteligencia computacional en muchas áreas del quehacer humano, sino reflejar cómo el empleo de las técnicas de Computación Inteligente han ido integrándose de forma acelerada en casi todos los aspectos de dichas actividades, cambiando notablemente la manera en que se hacían las cosas.

Elecciones

En aquellos países donde se ha adoptado el sistema democrático como método de ordenamiento del Estado, las elecciones son el reflejo y termómetro de la salud del mismo. En ellas confluyen todos los intereses de la nación, pues mediante su ejecución la mayoría de las instituciones se renuevan cada cierto tiempo, sustituyendo o manteniendo en funciones a aquellas personas que deben encargarse de todos los aspectos de su competencia y con ello permitir el adecuado funcionamiento de la nación.

Por su naturaleza masiva, el desarrollo de las campañas electorales ha representado para los candidatos y sus auspiciantes un esfuerzo importante de despliegue a través de los medios tradicionales de difusión como son la radio, la televisión y la prensa escrita [1]; el manejo de encuestas de opinión pública cara a cara y a través de contactos telefónicos, permitían conocer con un nivel de incertidumbre controlado, las preferencias e inclinaciones políticas del electorado en cada estadio de desarrollo de las campañas y con ello coordinar las estrategias más convenientes para corregir el rumbo o para apalancar las ventajas.

Sin embargo, hechos recientes han evidenciado un cambio de paradigma en la concepción de los métodos más adecuados para medir el desempeño de los candidatos y predecir a través de ellos sus posibilidades de victoria.

Los resultados electorales más notorios a nivel mundial ocurridos en 2016, como lo fueron las elecciones presidenciales en Estados Unidos, el referéndum consultivo sobre la salida del Reino Unido de la comunidad europea, el plebiscito sobre los Acuerdos de Paz en Colombia, las primarias francesas entre otros procesos acontecidos en ese año reflejaron una incapacidad notable de las herramientas tradicionales para predecir los resultados que a la postre fueron obtenidos. Las victorias de Trump y del No en el Reino Unido, significando la salida del país de la comunidad europea, por nombrar las más representativas, resultaron sorpresivas para los analistas políticos y para los grandes conglomerados mediáticos que daban como lógicas las victorias de Clinton y la permanencia del Reino Unido en la comunidad europea. Las encuestas fallaron estruendosamente en predecir lo que se estaba produciendo en la sociedad británica y estadounidense.

Ante tan abrumadora evidencia, el cambio de las herramientas que mejor modelaran el comportamiento de las masas antes de los comicios políticos era una necesidad. Por ello se comenzó la búsqueda de las señales que arrojaran alguna luz sobre qué había pasado, cuándo comenzaron a cambiar los hábitos de comunicación del electorado, por qué las encuestas de opinión

y los medios de comunicación no supieron entender las dinámicas internas de las sociedades que antes dominaban al momento de saber hacia dónde se inclinaba la balanza de las preferencias en estos acontecimientos de tanto impacto.

La búsqueda de los antecedentes de los hechos arrojó varias líneas de procesos sociales que explicarían las dinámicas electorales que se observan desde la primera década del siglo XXI.

En primera instancia se encuentra la irrupción de las redes sociales (RR. SS.) como medios de acercamiento directo a los actores políticos y sociales en los diferentes círculos de competencia política y social, sean estas municipales, regionales o nacionales. Este tipo de dinámicas sin intermediarios que van evolucionando y refinándose con el tiempo, avizoran un panorama probable donde a pesar de las capacidades tanto económicas como técnicas los medios tradicionales, su papel y relevancia estarán por detrás de las prácticas más horizontales que conectan a los actores políticos con el electorado a través del uso de redes sociales.

Las redes sociales también confirman de alguna manera la tesis desarrollada por Elisabeth Noelle-Neumann acerca de un comportamiento de masas que denominó la espiral del silencio [2]. Este comportamiento refleja la disconformidad de ciertos grupos que no comulgan con las tesis aprobadas por los grandes conglomerados de la información y de las sociedades. Esta situación se reflejó con mucho énfasis en el referéndum británico y en las elecciones presidenciales estadounidenses.

La espiral del silencio añadió un nivel de incertidumbre notorio en las encuestas públicas de cara a los comicios antes nombrado, ya que las mismas arrojaban las victorias de las opciones que abogaban por la permanencia del Reino Unido en la comunidad europea y la opción de Clinton como futura presidenta estadounidense. Esto arrojó que una masa importante de personas con poder electivo no expresó sus verdaderas afinidades o simpatías electorales, se las reservaron o, incluso, las mezclaron con las de los electores que no estaban alineados con sus preferencias, ocultando así sus verdaderas opiniones y gustos.

Este comportamiento también posee otra arista que debe ser tomada en cuenta, y es aquella que evita sentirse aislado o señalado por sus preferencias puesto que estas no entran dentro de las opiniones que son "políticamente" correctas o aceptadas por los grandes conglomerados tanto sociales como de medios. Y este comportamiento sucede, aparentemente, cuando existen marcadas preferencias entre candidatos: entre aquellos que se alinean con los

pensamientos, de nuevo, "políticamente correctos" y que reciben un apoyo de difusión y de acceso a los medios masivos mucho mayor, que aquellos que representan posturas opuestas e incómodas que no son muy populares.

Ante posiciones muy contrastadas entre candidatos, donde unos son apoyados por las élites intelectuales y por los medios y otros mucho menos aceptados por estas, el voto oculto crece, generando que las verdaderas correlaciones de fuerza e intención de voto no se vea reflejada en los estudios demoscópicos usuales.

Es en este escenario donde las RR.SS. pueden echar algo de luz a la real balanza de intenciones de voto y hacia dónde esta se inclina. En los comicios estadounidenses, cuatro días antes de los mismos, Donald Trump poseía en Twitter 12,9 millones de seguidores y 11,9 millones de me gusta en Facebook, en contraparte con los 10,1 millones de seguidores en Twitter que poseía Hillary Clinton y los 7.8 millones de me gusta reportaba su Facebook. Esto reflejaba un 35% menos me gusta y un 27% menos de seguidores en Twitter que su contrincante [1].

En contraste con las muestras demoscópicas llevadas a cabo hasta entonces por los grandes medios de comunicación, las RR.SS. modelaron un escenario más cercano a los resultados que posteriormente se concretaron.

Barack Obama aparece entre los pioneros en el uso de estrategias 2.0. en su campaña presidencial de 2008. El uso de grandes bases de datos unidas a las RR.SS. como medio para la captación de simpatizantes estaba en sus primeros estadios de desarrollo y a partir de su aplicación y de la posterior victoria de Obama, el estudio sobre cómo interactúan los votantes con los candidatos políticos en RR.SS. comenzó a profundizarse entre grandes centros de investigación e importantes universidades [3]. Para 2012, ya existían bases de datos con millones de posibles votantes indecisos que podías ser clasificados y agrupados para luego ser abordados comunicacionalmente mediante estrategias acordes con sus perfiles [4].

Con la campaña presidencial de Donald Trump en 2016, ya se habían desarrollado algoritmos que le permitieron a su equipo de asesores crear mensajes personalizados a cientos de miles de perfiles de usuarios de RR.SS. a partir de la intervención de Trump en el tercer debate nacional televisado que sostuvo con Clinton ese año [5].

A partir de entonces el uso de RR.SS. se ha vuelto parte estratégica de toda campaña electoral en el mundo y un indicador que está siendo profundamente analizado a día de hoy por la dinámica que genera entorno a

su interesante capacidad de plasmar las tendencias que ponen el foco sobre aquellos candidatos con mayores probabilidades de victoria. A pesar del hecho de ser una disciplina aún en fase experimental, el incremento sustancial del poder de procesamiento de la computación actual está permitiendo canalizar las enormes cantidades de datos que generan las RR.SS. y poder utilizarlas para clasificar perfiles de usuarios en internet [1].

Entre todas las redes sociales, la que más destaca como herramienta de pulsión política es Twitter, y sobre ella se estudia la dinámica del usuario de la red y su adhesión a aquellas corrientes de información, proyectos y figuras a las cuales apoya como un actor libre, independiente y en algunos casos como difusor de estas ideas. El estudio de la generación de datos e información de millones de usuarios en Twitter permite modelar la dinámica ciudadana y los nexos políticos con los cuales se vincula.

Se plantea que, bajo el enfoque de red, lo importante no es solo el uso de Inteligencia Artificial como generador de contenidos políticos estratégicos, sino como herramienta exploratoria que reúne opiniones, tendencias, declaraciones, comentarios que fungen como elementos complementarios que forman posiblemente una correlación directa con las preferencias electorales, prediciendo con efectividad los resultados de las mismas.

Con este enfoque se realizó una investigación de campo en las elecciones primarias presidenciales chilenas del 2017 donde un equipo multidisciplinar, puso a prueba la premisa de la capacidad predictiva de la red social twitter para identificar los ganadores de procesos comiciales. En el trabajo se buscó seleccionar el algoritmo de inteligencia computacional que mejor extraiga de las diversas interacciones de los usuarios en dicha red, el patrón que modele y prediga con mayor asertividad los resultados electorales. Para ello se construyó a partir de las diversas métricas que se pueden aplicar a la RR.SS. un modelo de inteligencia computacional predictivo que incluyera como factores la realidad social del entorno y las opiniones compartidas entre usuarios [1].

El procesamiento computacional de lenguaje natural permitió que las métricas antes mencionadas como el nombre del usuario que emitió un mensaje, la fecha de publicación del mensaje, la cantidad de seguidores del remitente, la cantidad de personas a las que el remitente sigue y la cantidad de mensajes emitidos por el remitente antes de la fecha del último mensaje, entre otros parámetros, pudiesen ser procesados -en principio manualmente por colaboradores humanos y después a través algoritmos computacionales entrenados- almacenados y luego clasificados por el sesgo del cri-

terio emitido, bien sea a favor, en contra o neutro hacia una de las opciones electorales disponibles y contrastados con la premisa de correspondencia tomada como referencia de veracidad, la cual es como se ha mencionado en este trabajo, la relación directa entre niveles de simpatía por una opción que se refleja en un flujo de mensajes positivos hacia esa tendencia[1].

Los investigadores tomaron en cuenta la posibilidad de robots generadores de comentarios que introdujeran distorsión en el análisis y posterior entrenamiento de los algoritmos, por lo que fue necesario tomar información de otras investigaciones que hablaran sobre mecanismos de detección de robots según criterios de valoración de usuarios.

Una vez logrado todos esto se entrenaron dos algoritmos modeladores que registraron información directamente de Twitter en un período de tiempo de 11 días antes de las elecciones primarias chilenas. Los resultados predijeron, con un nivel de probabilidad de error muy ajustados, los resultados que al final ocurrieron [1].

Investigaciones experimentales como la antes descrita arrojan resultados promisorios para seguir ahondando en las capacidades de la inteligencia computacional de conocer tendencias, no solo electorales, sino de múltiples actividades donde las opiniones de los usuarios modelan las preferencias e inclinaciones de los mismos. Quizá haya que esperar y seguir desarrollando investigaciones acerca de la potencia de los algoritmos para entender las corrientes dominantes basadas en RR.SS. pero es innegable que ahora la fuerza de la computación en un mundo más fragmentado en opiniones con canales más directos de opinión estará informando con mayor precisión las preocupaciones y tendencias colectivas que los antiguos medios masivos.

Redes eléctricas inteligentes

Desde principios de siglo XX, con la corriente alterna ganando la guerra de los modelos eléctricos, poco es lo que había cambiado en esta área. Las mejoras que se sucedieron con el tiempo estaban más enfocadas en la forma en que sé administraba la red que en el modelo de funcionamiento; esto ha venido cambiando de forma paulatina, sin embargo con la irrupción de la inteligencia artificial y la inteligencia computacional se ha acelerado el modelo de cambio, convirtiendo en impostergable la conmutación de redes unidireccionales a redes bidireccionales: las smart grids o redes eléctricas inteligentes serán en muy poco tiempo el nuevo estándar de la industria.

El concepto de red eléctrica inteligente o smart grids no es novedoso,

pero su necesidad en Latinoamérica se perfila cada día con mayor nitidez. La importancia de implementaciones de este tipo viene de la mano de la evidente obsolescencia de los sistemas eléctricos de esta región y de la necesidad de optimizar los recursos disponibles para la paulatina sustitución de las anteriores redes unidireccionales por redes más eficientes y que permitan un mejor uso de las inversiones hechas para mantener los niveles de servicio y, además, sostener las proyecciones de expansión que son necesarias debido al aumento de la población y de llegar a áreas que aún no cuentan con electricidad [6].

Para introducir en la redes eléctricas inteligentes y ejemplos concretos es necesario conocer la manera tradicional del sistema eléctrico tradicional.

El modelo eléctrico tradicional inicia desde las fuentes de producción de electricidad; dichas fuentes poseen diversos orígenes entre las que destacan el gas natural, el carbón, la energía nuclear y las fuentes hídricas. Otras fuentes se han añadido a la creación de electricidad como la proveniente de la energía eólica y de la energía solar.

Una vez creada la electricidad es necesario transmitirla para que pueda ser utilizada por los conglomerados humanos. La transmisión de electricidad se lleva a cabo a través de líneas eléctricas de alto voltaje, esto es así por la necesidad de evitar la mayor pérdida de energía posible; a mayor voltaje menor es la corriente y por ende menor pérdida de energía, sobre todo por calentamiento inducido en los conductores. Los conductores son cables que están sujetos a grandes torres que están dispuestas de tal manera que puedan llevar la electricidad a los centros poblados.

Una vez la electricidad llega a los centros poblados se inicia el proceso de distribución. La distribución consiste en recibir la energía transmitida para después, mediante el uso de transformadores, llevar el voltaje a los niveles de uso industrial y residencial. Los transformadores conviertes altas tensión en bajas tensiones, ideales para el consumo de empresas y hogares.

Las industrias, los hogares y todos aquellos elementos que utilizan energía eléctrica y que están conectados a la red eléctrica son las denominadas cargas. La carga total es la demanda de electricidad en un momento determinado es el aspecto más importante y delicado de la gestión de la red de energía eléctrica pues es fundamental que en todo momento la red se encuentre en equilibrio; el flujo de electricidad debe ser constante y predecible a fin de proporcionar la cantidad de electricidad adecuada en un momento dado, por tanto el manejo de información a medida que las redes de energía eléctrica se hicieron más grande, vino a ser un aspecto esencial para el cor-

recto funcionamiento de la misma.

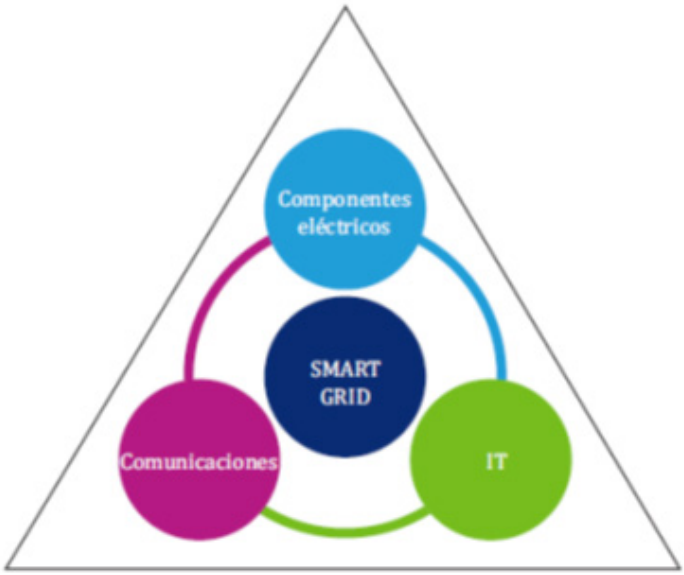
Cuando las redes eléctricas mundiales alcanzaron altos niveles de complejidad entre las décadas de los 70 y 90 del siglo XX, las dificultades en la administración se multiplicaron. Esta situación redundaba en el alza de los costes por facturación debido a las medidas de respaldo y redundancia de la red para evitar fallos y cortes del suministro. Esta situación era producto en parte de la escasez de datos para la toma de decisiones puntuales de optimización y control [6].

Los primeros pasos para el desarrollo de prototipos de sensores que pudiesen analizar en intervalos de tiempo muy cortos si existían problemas en el suministro de electricidad sobre grandes áreas iniciaron a principios de 1990 con las investigaciones lideradas por la Boneville Power Administration, oficina federal estadounidense, hasta que en 2000 se puso en operatividad el Wide Area Management System (WAMS), la administración de operaciones de medición primaria en áreas extensas, que es el origen de las Smart Grids [7].

A partir de esta experiencia, varias compañías y estados se interesaron en poner en funcionamiento en sus sistemas de control dicho modelo, replicándose en Europa, China entre otros, comenzando el proceso de actualización y cambio de la forma operativa de los sistemas eléctricos.

Ya con la información del modelo tradicional de redes eléctricas y conociendo los orígenes de las redes eléctricas inteligentes, se podría definir, en síntesis -a pesar de las diferencias encontradas en la bibliografía- a las redes inteligentes como un modelo sustentable de actualización de la red eléctrica en la cual convergen áreas de las ciencias de la información y comunicación e ingenierías conexas para la gestión y operación de manera inteligente de los procesos de generación, transmisión, distribución y consumo de la energía eléctrica, incluyendo el ecosistema de mercado de dicha energía [6].

Las redes eléctricas inteligentes funcionan como un todo interconectado como se puede apreciar en la Figura 3.1.



ıtes

La gráfica muestra una característica de las redes eléctricas inteligentes que es la bidireccionalidad de flujo tanto de información como de energía entre cliente y compañía; esto consiste en un esquema que pasa de un flujo de energía unidireccional desde las fuentes de energía hasta el consumidor, a otro que soporta el flujo bidireccional de recursos e información y que aprovecha las capacidades de generación distribuida -concepto analizado más adelante- y almacenamiento producto de la integración de redes con fuentes tradicionales y aquellas cuyas fuentes son predominantemente renovables [6].

La concepción de las redes eléctricas inteligentes permite anticiparse a posibles eventos como averías, perturbaciones o fallas del sistema eléctrico que pudiesen tener un impacto importante en el servicio. Esta antelación ayuda también en lo referente a planes de generación y almacenamiento, integrando a los clientes en las actividades del sector, integrando de forma posible, fuentes renovables de energía que brindarían soporte y recursos a la red [8].

Pero todo sistema tiene sus ventajas y desventajas. Las ventajas de las redes inteligentes son:

a)Estabilidad y seguridad energética: la integración en un mismo sistema de fuentes renovables y no renovables permiten un mejor flujo de energía, estabilizando los precios y la capacidad de entrega.

b)Amigable al medio ambiente: se reducen emisiones contaminantes al integrar más fuentes renovables al sistema.

c)Mayor eficiencia energética: al lograr un suministro con una menor incidencia de fallas, se maximiza el uso de la infraestructura y los recursos.

d)Sectorización de generación de energía o energía distribuida: se pueden integrar pequeños generadores de clientes ubicados en lugares residenciales a la red general para brindar energía a otros usuarios. Un ejemplo serían los paneles solares residenciales.

e)Integración geográfica de recursos: Los estados con mayor producción de energía pueden complementar de forma rentable y sostenible redes eléctricas de estados que no cubren su demanda energética.

Entre las desventajas se pueden nombrar

a)Costos: aún la concepción general de los estados soberanos sobre las energías de fuentes renovables es que son costosas, y que su implementación no está por encima de otras necesidades nacionales.

b)Políticas de calidad y servicio desactualizadas: todavía no existen en general entes regulatorios que promuevan la implementación de redes inteligentes como norma.

c)Estándares globales: aún se está trabajando en estándares de tal manera que las redes eléctricas inteligentes poseen especificaciones técnicas abiertas y de uso general a nivel mundial.

Cabe resaltar que la inteligencia computacional está presente en la implementación de redes inteligentes, como puede apreciarse en algunos ejemplos concretos.

En Portoviejo, capital de Manabí [8], provincia del Ecuador, la Universidad Técnica de Manabí diseñó un plan de red eléctrica de características inteligentes para un edificio de la sede utilizando herramientas de inteligencia computacional para modelado de red y monitorización de estados de operatividad.

Se tomo en cuenta, para respaldo de red el almacenamiento de un mini-generador fotovoltaico con capacidad de 7kWh que podía complementar los requerimientos de energía del edificio por tramos de operación, principalmente nocturna, desconectando del edificio al transformador principal y con ello ahorrando consumo y pérdidas asociadas al transformador.

Para ello entrenaron una red neuronal artificial de tipo función base radial RBF con norma euclídea y con los siguientes valores de entrada a la red neuronal:

a)Tensión eléctrica

b)Corriente

c)Factor de Potencia

d)Potencia

e)Frecuencia

f)Energía efectiva entregada

Estos datos fueron recopilados en un intervalo de 10 minutos por 20 días para tener un modelo de comportamiento de la carga tanto en días de actividad como los fines de semana y la entrega de energía en ese período. Al analizar los datos en este período se logró realizar un modelado de la red eléctrica en MatLab para conmutar de forma automática y arrojando todos los parámetros correctos de funcionamiento, manteniendo un monitoreo útil mediante el desarrollo de una interfaz dónde aparecen los parámetros de entrada y los arrojados por el sistema de control automatizado [8].

En Argentina se está desarrollando la primera red eléctrica inteligente en la ciudad de Armstrong, departamento de Belgrano, en la Provincia de Santa Fe [9].

El sistema cuenta con una planta fotovoltaica con una potencia de 200Kw, tres molinos de viento y una red paneles solares en edificios públicos y viviendas que se encargan del suministro eléctrico en la ciudad. Los hogares poseen receptores inalámbricos que le proporciona información sobre el estado de la red y características del consumo en la vivienda entre otras estadísticas.

El proyecto consta de una primera etapa que la de control, donde se instalaron aproximadamente mil medidores inteligentes de un universo de cinco mil medidores, con sistemas de maniobra y control de la red que están

conformados por seccionadores y reconectores que monitorean la red. La segunda etapa es la implementación la red distribuida con energías renovables de tal manera que la generación este cerca de los usuarios, evitando las pérdidas de la transmisión desde largas distancias.

Las tecnologías son fundamentales y están equipadas con herramientas de inteligencia computacional para adaptarse a la realidad que arrojen los estudios de factibilidad y operatividad.

Se estima en general que para el 2030 ya se avance en general hacia los marcos regulatorios que permitan la puesta en marcha de redes inteligentes en Latinoamérica [6].

Visión artificial

La visión artificial es un campo de estudio que engloba un conjunto de técnicas para el procesamiento de imágenes como la extracción de características, normalización espacial, segmentación, reconocimiento de patrones entre otros [10]. En medicina, la segmentación es una de las tareas más importantes al momento de analizar una imagen, siendo la segmentación la partición de una imagen según los componentes estructurales que resalten en un área homogénea según criterios definidos como textura o nivel de intensidad.

Las imágenes resonancias magnéticas son de particular interés para esta área en desarrollo debido a la necesidad de una clasificación correcta de tejidos, órganos, lesiones y anomalías en regiones específicas que identifiquen algún tipo de tumor o daños en el tejido. En este sentido se han desarrollado programas, basados en sistemas automáticos, que buscan interpretar grandes volúmenes de información de manera correcta procesando, analizando e interpretando imágenes que de otra manera no hubiese sido posible abordar.

Una de las técnicas más actuales es la captación de imágenes médicas es a través del sistema de Recuperación de Imágenes Basadas en Contenidos CBIR (Content-Base Image Retrieval) que combina la localización de imágenes con el método convencional de texto con el uso de color, textura, forma y colocación espacial para definir las Regiones de Interés ROI (Region of Interest) a estudiar. El objetivo de esto es comparar las conclusiones a las que llega un experto humano mediante la visualización de imágenes convencionales de resonancia magnética y las generadas por sistemas automáticos basadas en las técnicas antes descritas. La importancia de estos desarrolla estriba en la necesidad de dilucidar con mayor precisión aquellas

cuestiones que quedan a interpretación de los expertos como, por ejemplo, el estado de los tejidos alrededor o cerca de una tumoración. La necesidad de precisar mejor estos elementos es lo que hace a la computación inteligente indispensable para lograr mayor rigor médico y un mejor diagnóstico de cara al futuro [10].

Dentro de las técnicas de computación inteligente, destaca la lógica difusa con principal herramienta para la implementación de soluciones de esta índole; sin embargo, la combinación de varias técnicas permite a los sistemas un mejor desempeño.

Fuentes renovables de energía y computación inteligente

Las fuentes renovables de energía son la alternativa para combatir los efectos del calentamiento global. Actualmente la producción de energía a nivel planetario está conformada por un 80% de fuentes de origen fósil como el petróleo y el gas y en un 20% de fuentes renovable como la energía eólica, la energía térmica o la energía solar[11].

Los combustibles fósiles tienen la ventaja que producir energía mediante su explotación tiene una complicación técnica mucho más baja que la necesaria para producir la misma cantidad de energía mediante fuentes como el aire; sin embargo, posee múltiples desventajas que no tienen aquellas. La principal es el impacto sobre el medio ambiente debido a la generación extensiva de gases de efecto invernadero que se acumulan en la atmósfera, coadyuvando al descongelamiento de los polos. La otra desventaja es que su ciclo de reposición es mucho menor a la demanda energética mundial, presentándose como recursos finitos en el tiempo.

A partir de este cuadro es que puede entenderse la importancia de impulsar la consolidación de energías renovables, que no producen emisiones de efecto invernadero y que por su abundancia pueden ser consideradas fuentes de recursos infinitas; sin embargo, poseen un aspecto problemático, que es las dificultades de implementación debido a las exigencias técnicas que conllevan.

Las fuentes renovables de energía presentan una importante variabilidad, como es el caso del viento. La energía eólica se alimenta del aire en una región determinada, y la misma no presenta un comportamiento continuo sino aleatorio, ya que las condiciones atmosféricas no son definibles en el tiempo. El control sobre sistemas no lineales es un reto que a través de técnicas de computación inteligente se han podido ir abordando, como es el

caso de los generadores basados en viento.

Los generadores basados en viento o generadores eólicos son sistemas no lineales, de gran complejidad y dependientes del lugar de funcionamiento y por mucho tiempo han estado supervisados por sistemas de control basados en modelos teóricos de sistemas lineales adaptados a ciertas circunstancias operativas. Este criterio no evalúa correctamente los niveles de operación en las zonas transitivas, produciendo cargas por saturación que merman la vida útil del generador[11].

Los nuevos sistemas de control basados en lógica difusa permiten transiciones limpias entre puntos de operación, mejorando el desempeño del generador y prolonga su tiempo operativo. Esto también ayuda a la adopción de esta tecnología limpia, ya que se abaratan los costos de supervisión y mantenimiento.

Bibliografía

[1] P. Santander, C. Elórtegui, C. González, H. Allende-Cid, & W. Palma. "Redes sociales, inteligencia computacional y predicción electoral: el caso de las primarias presidenciales de Chile 2017". Cuadernos.info, (41), 41-56. 2017.

[2]E. Noëlle-Neumann, E. "La espiral del silencio. Opinión pública: nuestra piel social". Cap 20. 1995.

[3]L. Deltell, F. Claes & J. Osteso. "Predicción de tendencia política por Twitter: Elecciones Andaluzas 2012". Ámbitos. Revista Internacional de Comunicación. Consultado directamente desde la página web el 20 de junio de 2019 6:45 pm. http://institucional.us.es/ambitos/?p=148

[4] F. Provost & T. Fawcett, T. "Data Science and its Relationship to Big Data and Data-Driven Decision Making". Big Data, pp 52-53. 2013.

[5] M. Hilbert (2017). "The More You Know, the More You Can Grow: An Information Theoretic Approach to Growth in the Information Age". Entropy, Vol 19(2), pp 82. 2017.

[6] J. Gers. "América Latina y el Caribe: Estado del arte de las redes eléctricas inteligentes". EnerLac. Revista de Energía de Latinoamérica y el Caribe. Vol 1(1), Pp 25-41. 2017.

[7] Colaboradores de Wikipedia. Red eléctrica inteligente [en línea]. Wikipedia, La enciclopedia libre. [fecha de consulta: 19 de julio del 2019]. Dispo-

nible en <https://es.wikipedia.org/w/index.php?title=Red_el%C3%A9ctrica_inteligente&oldid=116180890>. 2019.

[8] N. Balderramo, Y. Llosas, L. Neves & L. Cuenca. "Diseño de Redes Eléctricas Inteligentes para una Gestión Energética". Memorias de la Novena Conferencia Iberoamericana de Complejidad, Informática y Cibernética (CICIC). 2019.

[9] Infobae, Portal de noticias. ¿Cómo funciona la primera red eléctrica inteligente de Argentina? [fecha de consulta: 17 de julio del 2019]. Disponible en https://www.infobae.com/def/desarrollo/2018/07/14/como-funciona-la-primera-red-electrica-inteligente-de-argentina/. 2019

[10] I. Rodríguez. "Desarrollo de módulo de aplicación para sistemas Open source (PACS): sistema de recuperación de imágenes de contenido (CBIR), para diagnosis y tratamiento en el área de Atención Primaria usando imágenes de resonancia magnética cerebral". Universidad de Granada. 2019.

[11] M. Santos, E. Miranda. "Aplicación de la lógica difusa en el ámbito de las

energías renovables ". Revista Elementos,Vol 2(1). 2012.

CAPÍTULO 4:

Validación y optimización

Este capítulo es producto de la estrategia a seguir para dictar un curso a profesionales con experiencia en usar estrategias computacionales en investigación, y en muchos casos, con experiencia en simulaciones. Está basado en experiencias propias y de otros autores en simulaciones en Ciencias e Ingeniería. Uno de los objetivos del capítulo es identificarse con el problema físico, el problema matemático, los posibles esquemas de solución y las posibles fuentes de errores. Además de familiarizarse con algún lenguaje de programación o con técnicas de computación inteligente. Lo planteado aquí para ecuaciones diferenciales en derivadas parciales sirva como elemento a considerar en cualquier simulación, bién sea, por redes neuronales o algoritmos evolutivos u otras técnicas de inteligencia artificial. También se intenta resaltar, que la persona que lleva acabo una simulación se convierte en un "experimentalista" por todo lo que involucra y el tiempo que se requiere para llevarla a cabo.

Los recientes avances en los métodos de simulación y la gran disponibilidad de software que actualmente existen en el mercado, han hecho que las simulaciones se realicen en todas casi todas las áreas científico--tecnológicas: sistemas empresariales, fenómenos climáticos y meteorológicos, fenómenos físicos, químicos, biológicos y en problemas de Ciencias e ingeniería. Una simulación presenta las siguientes ventajas:

• Permite estudiar el efecto de cambios internos y externos del sistema, al hacer alteraciones en el modelo del sistema y observando los efectos de esas alteraciones en el comportamiento del mismo.

• Puede conducir a un mejor entendimiento del sistema y por consi-

guiente a sugerir estrategias que mejoren la operación y eficiencia del mismo.

• Puede ayudar a entender mejor la operación del sistema, a detectar las variables más importantes que interactúan en el sistema y a entender mejor las interrelaciones entre estas variables.

• Permite experimentar con nuevas situaciones, sobre las cuales tiene poca o ninguna información. A través de esta experimentación se puede anticipar mejor a posibles resultados no previstos.

Las simulaciones son necesarias en los siguientes casos:

1.- Los métodos analíticos para resolver el modelo matemático no se han desarrollado aún.

2.- Los métodos analíticos están disponibles, pero los procedimientos matemáticos son tan complejos y difíciles, que la simulación proporciona un método más simple de solución.

3.- La simulación puede ser la única posibilidad, debido a la dificultad para realizar experimentos y observar fenómenos en su entorno real, por ejemplo, estudios de vehículos espaciales en sus vuelos interplanetarios.

4.- Se requiere respuestas rápidas para sistemas o procesos que requieren de largo tiempo para realizarse.

Las áreas de aplicación de la simulación son muy amplias, numerosas y diversas, basta mencionar sólo algunas de ellas: Descargas atmosféricas y su impacto en líneas de trasmisión, sub-estaciones eléctricas, las ecuación del calor, las ecuaciones de Maxwell análisis del impacto ambiental causado por diversas fuentes, análisis y diseño de sistemas de manufactura, análisis y diseño de sistemas de comunicaciones, análisis de sistemas de transporte terrestre, marítimo o por aire, adiestramiento de operadores (centrales carboeléctricas, termoeléctricas, nucleoeléctricas, aviones), análisis financiero de sistemas económicos y evaluación de sistemas tácticos o de defensa

militar, entre otros.

Cuando se estudia el modelo matemático de un sistema físico, usando simulación, se le llama modelo de simulación. Se evalúa el comportamiento a valores específicos de las variables de entrada, ejecutando el modelo de simulación por un período finito de tiempo. Una simulación se puede definir como una prueba o una serie de pruebas en las que se realizan cambios significativos a las variables de entrada de un modelo y así observar e identificar las razones de los cambios en las variables de salida. Cuando el número de variables de entrada es grande y el modelo de simulación es complejo, la simulación puede llegar a ser computacionalmente prohibitiva. Además del alto costo computacional, se incurre en mayor costo cuando la variable de entrada depende de un rango de valores, en el proceso de encontrar la mejor entrada.

La experimentación numérica es parte de la simulación, como se infiere en el caso de las ecuaciones diferenciales en derivadas parciales. La solución numérica de ecuaciones en derivadas parciales es uno de los campos de más actividad en la matemática aplicada actual ya que se refiere a como discretizar una variedad de problemas que aparecen en Ciencias e Ingeniería.

Cómo lograr soluciones numéricas confiables, como comprobar que las soluciones encontradas no están erradas y sobre como verificar que las soluciones encontradas es la más cercana a la solución real se trata más adelante. Si las soluciones son válidas en todo el espacio y para todo tiempo o si son válidas en un espacio y tiempo limitados. La experiencia nos dice que, en general, las soluciones son válidas en un rango limitado de puntos. Un código puede ser estable y convergente solo con una malla mediana o gruesa de puntos sin ir a un refinamiento con mallas muy finas, aunque en teoría, mientras más refinada la malla la solución numérica se aproxima más a la solución real.

Otro punto a resaltar es que, no necesariamente, los códigos son válidos para todas las mallas lo que permite hablar de un dominio de validez. Por eso, los apasionados de las simulaciones son considerados experimentalistas numéricos. Es por esto, que las simulaciones deben blindarse experimentando con un refinamiento de malla, distintas condiciones de contorno y en algunos casos variando los métodos de solución (cuando esto ocurre, las

pruebas de convergencia y estabilidad deben realizarse nuevamente). Esto permite ir a la optimización de los códigos, lo que se traduce en la confiabilidad, convergencia y estabilidad de las soluciones.

Para tratar la complejidad que representa la optimización de problemas donde involucre inteligencia artificial u otras técnicas de simulación, al final del capítulo se realiza una propuesta metodológica, basada en experiencias propias y de varios autores, la cual permite integrar aspectos no comunes en la literatura sobre optimización, y al mismo tiempo combina el modelado con la simulación, análisis y optimización, permitiendo transformar un modelo conceptual en un modelo de simulación, y un problema de decisión en un problema de búsqueda mediante la experimentación numérica.

El objetivo de la optimización de la simulación es minimizar los errores mientras maximiza la Información obtenida en un experimento de simulación. La optimización basada en simulación integra técnicas de optimización en el análisis de simulación.

Una vez que se modela matemáticamente un sistema, las simulaciones basadas en computadora proporcionan información sobre su comportamiento. La entrada de cada variable varía con otros parámetros que permanecen constantes y se observa el efecto sobre el objetivo de diseño. Este es un método que consume tiempo y mejora parcialmente el rendimiento. Para obtener la solución óptima con un mínimo de cómputo y tiempo, el problema se resuelve de forma iterativa donde, en cada iteración, la solución se acerca a la solución óptima. Tales métodos se conocen como "optimización numérica" o "optimización basada en simulación".

En el experimento de simulación, el objetivo es evaluar el efecto de diferentes valores de las variables de entrada en un sistema. Sin embargo, el interés a veces es encontrar el valor óptimo para las variables de entrada en términos de los resultados del sistema. Una forma podría ser ejecutar experimentos de simulación para todas las posibles variables de entrada. Sin embargo, este enfoque no siempre es práctico debido a varias situaciones posibles y solo hace que sea difícil de manejar los experimentos para cada escenario. Por ejemplo, puede haber demasiados valores posibles para las variables de entrada, o el modelo de simulación puede ser demasiado complicado y costoso de ejecutar para valores de variables de entrada subóptimos.

En estos casos, el objetivo es encontrar valores óptimos para las variables de entrada en lugar de probar todos los valores posibles.

En la Optimización paramétrica (estática el objetivo es encontrar los valores de los parámetros, que son "estáticos" para todos los estados, con el objetivo de maximizar o minimizar una función. En este caso, se puede utilizar la programación matemática, como la programación lineal. En este escenario, la simulación ayuda cuando los parámetros contienen ruido o la evaluación del problema exigiría un tiempo de computadora excesivo, debido a su complejidad.

La técnica de control de optimización se utiliza principalmente en informática e ingeniería eléctrica. El control óptimo es por estado y los resultados cambian en cada uno de ellos. Uno puede usar la programación matemática, así como la programación dinámica. En este escenario, la simulación puede generar muestras aleatorias y resolver problemas complejos y de gran escala.

Para optimizar es necesario:

-Correr con mallas muy finas para lograr precisión numérica de los resultados

-Cambiar los métodos matemáticos para mejorar la solución y mejor precisión en los resultados.

-Obtener resultados en menos tiempo.

Formulación del problema y soluciones

Problema numérico: Descripción precisa de la relación funcional entre un conjunto finito de datos de entrada y un conjunto finito de datos de salida. Mientras que Algoritmo es la secuencia ordenada y finita de pasos, excento de ambigüedades que seguido en su lógica nos conduce a la solución de un problema específico.

Por otra parte, el Método numérico es el procedimiento para transformar un problema matemático en numérico para su simulación.

Antes de realizar una simulación, por sencilla que sea, se deben tener claro los objetivos y alcances de la misma:

a)La simulación como un todo resuelve el problema,

b)El problema se divide en partes, donde cada parte involucra una simulación

c)La simulación es solo una parte de la solución del problema.

Los problemas frecuentes a simular presentan las siguientes características: Obtener resultados ya reportados por otros autores usando otro método de solución; simular un problema original (no reportado) o simular problemas ya estudiados haciendo variaciones tales como imponer otras condiciones iniciales y de contorno o variar el modelo físico. Finalmente, simular un problema como continuación de otro ya estudiado. Una vez seleccionado el problema el usuario escoge entre simular con un software comercial o programar en el lenguaje de programación que desee.

El modelo físico y el modelo matemático

El modelo físico proporciona una descripción lo más precisa posible del sistema a simular, y a su vez muy sencilla para permitir construir el modelo matemático y las condiciones iniciales y de borde. Debemos tener claro que las ecuaciones que se usan para describir el comportamiento del sistema mecánico, eléctrico, electrónico son modelos ideales o aproximados y la solución no necesariamente da un comportamiento real y preciso del sistema.

En muchos problemas de Ciencias e Ingeniería el modelo matemático involucra ecuaciones diferenciales en derivadas parciales. Las ecuaciones diferenciales en derivadas parciales se pueden resolver numéricamente por distintos métodos. Entre ellos: diferencias finitas, métodos espectrales, aproxi-

maciones sucesivas, elementos finitos. Aquí nos enfocaremos solamente en el método de diferencias finitas, uno de los más comunes y cuyo tratamiento es parecido al de otros métodos, en cuanto a los aspectos a considerar en la simulación. Las técnicas numéricas para resolver las ecuaciones diferenciales en derivadas parciales, depende si la ecuación es parabólica, elíptica o hiperbólica y en el caso de las diferencias finitas, se han usado y verificado su aplicabilidad en diversos problemas en Ciencias e Ingeniería.

Existe otra variedad de problemas, donde el sistema se describe mediante un sistema de ecuaciones lineales. Cada caso requiere un tratamiento adecuado al contexto.

Las ecuaciones diferenciales en derivadas parciales se clasifican en elípticas, hiperbólicas y parabólicas. Por ejemplo, la ecuación de Poisson se ha resuelto para encontrar la solución numérica del potencial eléctrico en una placa cuadrada en dos dimensiones.

$$\frac{\partial^2 \phi}{\partial x^2} + \frac{\partial^2 \phi}{\partial y^2} = \rho \tag{1.1}$$

Un ejemplo de ecuación parabólica es la ecuación de Difusión: ecuación de transferencia de calor en 1D (una dimensión espacial y una temporal)

$$\frac{\partial T}{\partial t} = \alpha \frac{\partial^2 T}{\partial x^2} \tag{1.2}$$

El ejemplo más común de ecuaciones hiperbólicas es el de las ecuaciones tipo onda cuya forma es la siguiente:

$$\frac{\partial^2 f}{\partial x^2} - \frac{1}{v^2} \frac{\partial^2 f}{\partial t^2} = 0$$

Procedimiento para construir un algoritmo

Descripción acertada del sistema físico. Esto es, lo más cercano posible al sistema real. El sistema de ecuaciones diferenciales que describen el sistema físico. Esto incluye una elección apropiada de las variables dependientes e independientes y condiciones iniciales y de contorno.

Las derivadas en las ecuaciones diferenciales que describen el sistema físico se aproximan por operadores discretos. Por ejemplo, el operador derivada parcial ∂ se aproxima a un operador D_Δ.

Una discretización consistente implica que lim $\lim_{\Delta \to 0} D_\Delta f \Delta = \partial f$ para cualquier función f suficientemente suave.

Describir el problema, estrategias de discretización, estrategias de solución, posibles soluciones, y de ser posible resolverlo en forma algebraica y discreta para un valor finito de Δ.

En el límite de discretización extremadamente fino, Δ→0, se debe obtener una solución numérica aproximada a la solución analítica.

una vez seleccionada una aproximación de diferencia finita consistente, la estabilidad es una condición necesaria y suficiente para que la solución numérica converja a la solución analítica.

Diferencias finitas

Como mencionamos antes, el método que usaremos para resolver las ecuaciones diferenciales es diferencias finitas. La deducción de los esquemas de diferencias para las primeras y segundas derivadas, se hace haciendo un desarrollo en serie de Taylor. Este es uno de los métodos más usado para obtener modelos numéricos y en el análisis de errores en las simulaciones. Las soluciones numéricas son aproximaciones a las soluciones exactas. El desarrollo de Taylor (es una serie infinita de términos) de f_{i+1} alrededor de x_i con $f_{i+1} = f(x_i)\, f_{i+1} = f(x_{i+1})$ es

El desarrollo de Taylor

$$f_{i+1} = f_i + h\frac{df}{dx} + h^2 \tag{1.3}$$

$$\frac{df}{dx} = \frac{f_{i+1} - f_i}{h} + h\frac{d^2f}{2dx^2} + \cdots \tag{1.4}$$

De (1.3) y (1.4) se tiene las aproximaciones en diferencias finitas para la primera y segunda derivada a segundo orden.

$$\frac{df}{dx} = \frac{f_{i+1} - fi}{h} + O(\Delta^2) + \cdots \tag{1.5}$$

$$\frac{d^2f}{dx^2} = \frac{f_{i+1} - 2f_i + f_{i-1}}{2\Delta^2} + O(\Delta^2) \tag{1.6}$$

Aproximaciones numéricas para una primera derivada temporal usando los esquemas hacia adelante, centrada y hacia atrás respectivamente.

$$\frac{df}{dt} = \frac{f^{n+1} - f^n}{\Delta t} + O(\Delta^2) \tag{1.7}$$

$$\frac{df}{dt} = \frac{f^{n+1} - f^{n-1}}{2\Delta t} + O(\Delta^2) \tag{1.8}$$

$$\frac{df}{dt} = \frac{f^n - f^{n-1}}{\Delta t} + O(\Delta^2) \tag{1.9}$$

Aproximaciones numéricas para una primera derivada espacial usando los esquemas hacia adelante, centrada y hacia atrás respectivamente.

$$\frac{df}{dx} = \frac{f(i+1) - f(i)}{\Delta x} + O(\Delta^2)$$

$$\frac{df}{dx} = \frac{f(i+1) - f(i-1)}{2\Delta x} + O(\Delta^2)$$

$$\frac{df}{dx} = \frac{f(i) - f(i-1)}{\Delta x} + O(\Delta^2)$$

En este caso, $\Delta = h$ es el espaciado de la malla. El término $O(\Delta^2)$ representa el error por el truncamiento de la serie.

Dependiendo del esquema que se quiera usar, las expresiones discretizadas se sustituyen en la ecuación diferencial y se tiene la versión discretizada de la ecuación, lista para ser implementada en el lenguaje de programación que desee el usuario.

Condiciones iniciales y de frontera

Los problemas de evolución involucran típicamente una variable espacial (o tres en el caso más general) sobre la que se dan condiciones de contorno y una variable temporal sobre la que se dan condiciones iniciales.

Unas condiciones iniciales y de frontera adecuadas al problema a resolver completan el modelo matemático. Existen criterios en los problemas reales donde las condiciones iniciales y de contorno se escogen de acuerdo al sistema bajo estudio. Esto hace que la simulación sea más confiable. Por otra parte, existen problemas en donde las condiciones de contorno y las condiciones iniciales son puestas a mano por la dificultad de adecuarlas al problema. Existen dos tipos de condiciones de contorno:

Dirichlet: Solo se requieren los valores de la función a determinar al inicio y en los bordes. Por ejemplo, la ecuación de onda en una dimensión espacial y la temporal, sujeta a las condiciones de frontera f(0,t)=1y f(L,t)=4.

Newmann: Se requiere especificar las derivadas de la función al inicio y en los bordes.

$$\frac{df}{dx}(0) = \frac{df}{dx}(2) = 10$$

En las simulaciones se pueden usar las condiciones de contorno de Dirichlet o las de Newmann dependiendo del problema a resolver. Es común encontrar simulaciones donde se usan ambas condiciones de contorno. Los capítulos 2 y 3, son dos artículos publicados en la revista Universidad, Ciencia y Tecnología el año 99. En ello, se ilustran todo el formalismo matemático para la simulación y luego la simulación en sí. Se observa que en la solución de la ecuación de transferencia de calor se usan ambas condiciones de contorno. Desde el punto de vista del modelo matemático y del modelo físico, es un problema muy bien planteado.

Algunas consideraciones hasta este punto:

¿Qué condiciones se le debe imponer a un esquema numérico para obtener una aproximación aceptable del problema diferencial (de la simulación)?

¿Porqué dos esquemas simples (formulación explícita o implícita, o esquemas hacia atrás o hacia adelante) pueden conducir a comportamientos completamente diferentes en una simulación?

¿Cómo se puede tener información cuantitativa sobre la precisión de la aproximación numérica?

Para tener confianza en los resultados de una simulación, necesariamente tienen que realizarse pruebas de convergencia y estabilidad.

Convergencia y error

Las pruebas de convergencia es la relación o comparación entre la solución numérica y la solución analítica (o exacta). Es claro que en muchos problemas la solución analítica no se conoce. Mientras la estabilidad es la relación entre el espaciado espacial Δx y el espaciado temporal Δt, en el caso unidimensional. Este concepto es extendible para los casos en dos y tres

dimensiones espaciales y una temporal. Una forma de estudiar la estabilidad de un código es evolucionar las cantidades que dependen del tiempo. Si la gráfica no presenta oscilaciones se concluye que la solución es estable.

En las simulaciones aparte de los errores de truncamiento y redondeo, existen los errores inherentes al modelo físico y las limitaciones de modelo matemático (condiciones de contorno). Una forma de comprobar la validez de las soluciones numéricas y detectar posibles errores en la programación o elaboración del código o programa (o que el esquema de discretización escogido no funciona) es mediante las cantidades L1 y L2, conocidas en análisis numérico como ``normas"

$$L_1 = \sum |fj_i - Fij|$$

$$L_2 = \sum \sqrt{(f_{ij} - F_{ij})^2}$$

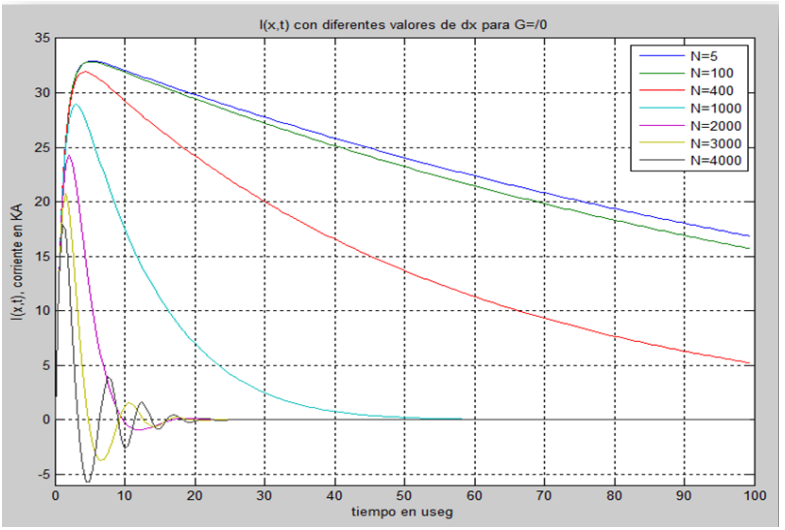


Figura 1: Solución numérica para diferentes valores del número de puntos de la malla N y y el parámetro de conductancia, G=0 en el estudio de descargas atmosféricas sobre una línea de transmisión. El rango de validez

del código es entre 400 y 1000 puntos. Fuera de ese rango el código es inestable.

donde f_{ij} es la solución numérica y f_{ij} es la solución analítica. La idea es ir variando el número de puntos de la malla computacional hasta que se logre la convergencia y demostrar que la solución es aceptable.

Por ejemplo, si se resuelve numéricamente la ecuación de Laplace para una placa cuadrada en dos dimensiones, la idea básica es calcular el potencial eléctrico variando el número de puntos de la malla, lo que implica variar el Δx o Δy. Si se usa en la discretización diferencias finitas a segundo orden, se espera que el error en la simulación sea de segundo orden, esto es,

Variar el número de puntos, Nx=100...100000 (implica tener un Δ para cada Nx o Ny) lo que implica obtener distintos valores del potencial para cada punto. Al graficar

$$\log \varepsilon = 2 \log \Delta$$

Se debe obtener una recta con pendiente ≈2, en total acuerdo con el error esperado a segundo orden de aproximación.

$$\varepsilon \, \alpha \, \Delta^2$$

Esto permite hallar el rango de estabilidad y convergencia del código o programa. Un punto importante es que los códigos no son estables y convergentes para cualquier número de puntos. La prueba de convergencia también se usa para estudiar el rango de validez del código.

Estabilidad

La condición de Counrant--Friedrichs--Levy, sobre el paso en el tiempo t, es una condición necesaria para la convergencia de un algoritmo de evolución numérica. Consiste en evolucionar el dato inicial para un espaciado de la coordenada espacial constante y variando el espaciado Δt. La condición CFL establece que el intervalo temporal para la evolución numérica está

restringido por

$$\Delta t \leq k\Delta x$$

donde k es un número del orden de la unidad y Δx es el espaciado de la malla espacial. De tal forma que para cada valor de Δx se selecciona un valor de Δt que cumpla con la condición CFL y se encuentra experimentalmente cuando la evolución numérica es estable.

Malla computacional

En general, no existe problemas cuando se calculan las derivadas numéricas en los puntos internos de la malla computacional. Sino se manejan adecuadamente las condiciones de contorno pudiesen aparecer problemas en los puntos del borde, lo que necesariamente implica fijar estrategias para los puntos ubicados en el borde de la malla.

Para minimizar los errores y buscar que la simulación sea lo más cercana posible a la realidad, un paso necesario es hacer el refinamiento de malla. A medida que se aumenta el número de puntos, el Δ se hace más pequeño y la solución numérica se acerca más a la solución verdadera o real. El refinamiento de malla tiene como limitación la máquina que se está usando, debido que para mallas muy finas, el tiempo de cálculo es mayor.

Malla gruesa, N=100 . . . . . .

Malla mediana, N=10000 . . . . . .

Malla fina, N=100000 ……

En cada caso Δ=1/N. Si N es muy grande Δ es pequeño y la solución numérica se aproxima más a la solución análitica.

Criterios de convergencia

Es claro que lo planteado para el cálculo de la convergencia con las normas, no es factible aplicar cuando las soluciones analíticas no son conocidas o no se conoce la solución exacta del problema. En este caso se usan los criterios de convergencia, esto es, buscar elementos inherentes al problema

que se está resolviendo que permitan concluir al usuario que la solución es confiable.

Discretizar la ecuación diferencial. Como vimos antes, existen varios esquemas de discretización en diferencias finitas que han sido probados y verificados que son estables dependiendo si la ecuación diferenciales elíptica, parabólica o hiperbólica: Usando diferencias finitas hacia adelante y sustituyendo en se tiene la versión Pseudocódigo. A este nivel, el pseudocódigo sustituye al diagrama de flujo. Ahora más importante es el ``que hacer" que el "cómo hacer". El pseudocódigo contiene, a grandes rasgos, las estrategias de programación, preferiblemente en el lenguaje donde se codificará el programa.

Medidas experimentales

Resultados obtenidos al aplicar una metodología de optimización en simulación, a un proceso de una planta maderera. Se utilizó un modelo de simulación en el software ARENA, integrado con una heurística de algoritmos genéticos. Al aplicar la metodología demuestran que utilizando una configuración diferente de recursos en la empresa, es posible reducir el tiempo promedio de permanencia de los productos dentro del sistema, en aproximadamente un 18%. La configuración necesaria para lograr este resultado se obtuvo evaluando solamente un 1.6% de todas las combinaciones posibles.

En algunos casos un modelo de simulación y una heurística de inteligencia artificial pueden ser usados con el fin de optimizar un sistema simulado. La optimización de modelos de simulación puede ser definida como el proceso de ligar un método de optimización con un modelo de simulación para determinar los valores apropiados de ciertos parámetros de entrada de manera tal que se maximice el desempeño del sistema simulado. Específicamente, un modelo de simulación representando el sistema en estudio fue integrado con una rutina de algoritmos genéticos con el fin de encontrar una solución "óptima" o cercana al óptimo. Los resultados muestran que al usar algoritmos genéticos como estrategia de optimización, es posible encontrar una "buena" solución al problema explorando solamente un 4.44% del total de espacio total de posibles combinaciones o escenarios.

Los modelos de simulación como herramienta de optimización facilitan la elección de la mejor alternativa de entre todas las posibles mediante

comparación de su desempeño. Sin embargo, este procedimiento es inviable cuando el número de posibles soluciones es grande, incluso infinito, siendo imposible en la práctica su enumeración y evaluación por parte del usuario del modelo de simulación. Esta limitación del uso directo del modelo de simulación se solventa mediante la combinación de un optimizador, generalmente basado en algoritmos metaheurísticos y en métodos de la inteligencia artificial, con el modelo de simulación: el optimizador determina una solución prometedora y la envía al simulador, el cual la evalúa y devuelve medidas de desempeño de esta solución, con ellas, y habiendo aprendido del historial de evaluaciones de soluciones ya realizado, el optimizador genera una nueva solución prometedora y se realiza un nuevo ciclo optimizador-simulador.

El procedimiento de búsqueda de la mejor solución se puede automatizar, permitiendo la evaluación, en general, de miles de soluciones, en la búsqueda de la mejor, de la óptima. Existen, además del procedimiento descrito anteriormente, otras posibilidades de combinación de técnicas de optimización con el modelo de simulación, como la estimación de los llamados modelos subrogados, haciendo uso de técnicas de tipo estadístico, de redes neuronales, entre otras.

Como la optimización de la simulación implica el uso de Algoritmos que surgieron de campos muy diferentes, tiene relaciones con muchos diversos. disciplinas, y se ha aplicado a diferentes campos de investigación. Para la optimización basada en la simulación, optimización estocástica, optimización paramétrica, caja negra optimización, y optimización a través de simulación, donde las versiones continuas y discretas.

Las entradas a la simulación pueden ser diversas. referido como ajustes de parámetros, entrada Ajustes, variables, controles, soluciones, diseños, experimentos (o diseños experimentales), factores, o configuraciones.

El rendimiento de una simulación también puede ser referido como un experimento, una evaluación de la función objetiva, o simplemente una evaluación de funciones. Se usa el término 'iteración' para referirse a un número fijo de evaluaciones de funciones generalmente uno) realizado para una optimización de simulación. Una simulación optimizada, debe estar sometida a pruebas tales como: sensibilidad de las respuestas ante cambios en la entrada, valores objetivos para respuestas esperadas, réplicas de simulación.

Optimización de la simulación

Una nota de precaución al usar métodos y esquemas numéricos es que los esquemas no son válidos en todas las ecuaciones ni en todos los dominios. En la practica la experimentación numérica permite determinar el rango en el cual son válidas las soluciones (2) escalamiento y reducción de rango de las variables de decisión, (3) proporcionar buenas conjeturas iniciales para el algoritmo; y (4) información recopilada de un problema conocido estructura, tales como estimaciones derivadas.

Las técnicas que suelen ser. más adecuadas en la práctica para diferentes escenarios en el universo de los problemas de optimización. Cierto clases de algoritmos, tales como búsqueda aleatoria Métodos, pueden ser aplicables a todos estos tipos de problemas, pero a menudo son más adecuados cuando tratar con problemas con discontinuidades, no suavidad) y son a menudo se usan porque son relativamente fáciles de implementar. Las posibilidades de combinar simulación y los procedimientos de optimización son muy amplios: simulación con iteraciones basadas en la optimización; optimización basadas en simulación; simulación secuencial y optimización; y alternar la simulación y diferentes técnicas pueden ser aplicables o más adecuadas dependiendo de cuánto se sabe sobre la simulación subyacente.

Validación

La validación de la red neuronal analizando su comportamiento con datos que no han sido utilizados en el proceso de entrenamiento. La validación de una red neuronal se realiza hasta 2000 patrones en experiencias recientes. El error cometido en el cálculo de las salidas y el ajuste de las salidas con los valores de los pesos y umbrales óptimos conseguidos en el entrenamiento de la red neuronal. Se deduce que tan solo unos pocos patrones, 13, son confundidos por la red neuronal. Esto significa que la red clasifica correctamente en más de un 99 % de los casos.

En otros estudios realizados con las redes neuronales entrenadas con distribución uniforme de la velocidad de retorno del rayo (problemas de descargas atmosféricas ampliamente estudiadas en literatura), el error resultante entre la aplicación del algoritmo de cálculo y la aplicación de la red neuronal es muy alto dependiente del parámetro W; sin embargo, la razón principal está ahora en los valores extremos que pueden tomar las velocidades de retorno, con W = 50 se pueden alcanzar valores de hasta 270000km/s, mientras que con W = 500 se pueden tener valores inferiores a 30000 km/s; si se tieneen cuenta que la red neuronal se ha entrenado con

valores comprendidos entre 30000 y150000 km/s, es lógico esperar errores más grandes que cuando se asumió una distribución uniforme de velocidades; por otra parte, los mejores resultados se han obtenido para líneas apantalladas el estudio realizado con las redes neuronales entrenadas con velocidad de retorno del rayo en función de la intensidad de pico de la descarga, ha dado resultados similares a los obtenidos al utilizar las redes neuronales entrenadas con distribución de velocidad uniforme; tan solo los resultados obtenidos con el algoritmo de Chowdhuri para líneas sin apantallar han mejorado considerablemente.

Si se comparan los dos estudios realizados, al utilizar las redes neuronales entrenadas con la velocidad del rayo en función de la intensidad pico de la descarga solo se consigue disminuir el error total con el algoritmo de Chowdhuri en líneas sin apantallar. La falta de datos utilizados en este entrenamiento ha sido la causa principal de que no se haya obtenido una mejora general con ambos algoritmos y ambos tipos de línea.

Análisis de sensibilidad

Los estudios de sensibilidad permiten determinar que tanto y cómo influyenen las soluciones la variación de los parámetros de entrada y condiciones de contorno. Por ejemplo, en el estudio de descargas atmosféricas en líneas de transmisión, varios estudios de sensibilidad se han realizado con respecto a la Tensión inducida en la línea por descargas a tierra en Intensidad de la descarga de retorno del rayo. Con esta opción se obtiene gráficamente la tensión inducida en la línea por descargas que caen a tierra en sus cercanías, y la tensión máxima inducida en función de una serie de parámetros importantes que caracterizan a la descarga o a la relación descarga-línea. Se utilizaron dos métodos diferentes para calcular la tensión inducida: el método de Chowdhuri, y el método de Rusck. El programa también permitió comparar los distintos métodos entre sí.

Algoritmo de optimización

La idea es conseguir la solución óptima o la mejor solución cuando se realiza una simulación. No importa el método usado en la simulación y si es producto de cualquier técnica de computación inteligente, bien sea, redes neuronales, algoritmo genético, lógica difusa, entre otras o de la solución de ecuaciones en derivadas parciales. Una persona que realiza una simulación es equivalente a una persona que investiga en un laboratorio, tanto en tiempo

de dedicación como en detalles tales como puesta a punto y calibración y manejo de equipos, minimización de errores.

Un reciente diálogo entre un estudiante que usa redes neuronales para determinación de estrés, se muestra a continuación:

1.¿has probado tus resultados con varios modelos de redes neuronales?

Realmente probé con backpropagation y no resultó muy bien. En el caso de este tipo de red se requieren muchos datos de entrada y no disponemos de ellos. De lo contario arroja demasiado error. No fue fácil ponerla a punto. También hay que tener claro que no es una sola red para todo el sistema. Se utilizan diferentes redes en diferentes partes. Por ejemplo, usé una perceptron simple, porque solo eran dos tipos de información y dos categorías. Pero para otras partes de la investigación use red bayesiana que es probabilística. En todos los casos se utilizan según lo que sea más sencillo de trabajar, que no dé mayores complicaciones.

2.¿variaste el número de iteraciones?

Si. El número de capas, la función de activación también porque a veces da la impresión que con una función "bien", por la teoría, pero al probar no es así, resulta mejor la otra. (LE FALTA)

3.Cambiaste pesos y tendencias?

No. Los pesos no cambiaron, esos los arroja el sistema automáticamente. Se cambian las iteraciones, el número de capas, tratando que sean el mínimo. También se agrega la cantidad de neuronas en la capa intermedia, eso si es complicado, se experimenta hasta que se logran los resultados, si te excedes se desconfigura todo, si estas por debajo no alcanzas el resultado. Entonces hay que buscar los valores poco a poco hasta obtener el resultado deseado.

4.Las redes neuronales fueron entrenadas y verificadas con otros parámetros distintos a los del entrenamiento?

Si claro. Se entrenan luego se comparan con valores teóricos que salen de las bases de datos estándares, por ejemplo, dice la teoría que si la frecuencia de la voz está entre 17 Hz ≥ que una función f(b) ≥15 Hz la persona presenta alto nivel de estrés, entonces se compara con valores teóricos todo.

5.¿En fin le hiciste maldad a tus códigos, de tal forma de deducir si estos son estables y convergentes y si los resultados obtenidos son óptimos?.

Hasta donde he visto, si. El Médico de la Universidad Católica lo probó y le gustó pero, todo es perfectible. Debo revisar nuevamente, porque he tenido problemas en cuanto a sistema y computadoras.

A continuación se muestra un algoritmo de optimización que aporta los elementos necesarios para determinar cuál es la mejor solución o solución óptima de una simulación:

 # Problema
 # Modelo físico, modelo matemático.
#Suposiciones
#entrenamiento y pruebas de la RN
 # Solución analítica
#Esquemas de discretización
 # Pseudocódigo
 # Pruebas de convergencia
 # Refinamiento de malla. Por lo menos considerar malla gruesa, mediana y fina. Al variar el número de puntos N implica variar Δ. Esto permitirá hallar el rango de convergencia y estabilidad del código.
#Usar distintos esquemas de discretización (hacia adelante, hacia atrás
o centrada)
#cambiar condiciones iniciales y de contorno
 # Simular con parámetros distintos a los usados en el entrenamiento.
 # pruebas de convergencia y estabilidad con la norma L1.
 # Cambiar la malla. Variar el número de puntos
 # Superponer la solución numérica con la analítica
 # Análisis de sensibilidad
 # Comparar varios métodos de solución.
 # Medidas experimentales
 # Simular con distintos software
 # Variar el número de iteraciones
 # Comparar con soluciones obtenidas previamente.
 # Usar los criterios de convergencia

Para validar la utilidad y validez de la metodología propuesta en una simulación, se propone este algoritmo de optimización que trata de englobar

los aspectos necesarios para lograr la solución óptima. Todos los aspectos considerados en el algoritmo han sido implementados como parte de investigaciones ya desarrolladas. El algoritmo, puede ser utilizado tanto para la obtención de soluciones factibles como para obtener soluciones óptimas en simulaciones, siempre y cuando los requisitos computacionales de tiempo y memoria necesarios para obtener tales soluciones no superen los recursos disponibles.

Azadivar, F.; Shu, J.; Ahmad, M. 1996. Simulation Optimization in Strategic Location of Semi-finished Products in a Pull-Type Production System. Proceedings of the 1996 Winter Simulation Conference. pp. 1123-1128. [Links]

Carson, Y.; Maria, A. 1997. Simulation Optimization: Methods and Applications. Proceedings of the 1997 Winter Simulation Conference. pp. 118-126.

Gen, M.; Cheng, R. 1997. Genetic Algorithms and Engineering Design. Wiley. [Goldberg, D. 1989. Genetic Algorithms in Search, Optimization and Machine Learning, Addison-Wesley, Reading, MA.

Haupt, R.L.; Haupt, S.E. 1998. Practical Genetic Algorithms. Wiley.

Pierreval, H. 1997. Using Evolutionary Algorithms and Simulation for the Optimization of Manufacturing Systems. IIE Transactions. Vol. 29, 3, pp. 181-190. Rosenblatt, M.J.; Roll, Y.; Zyser, V. 1993. A combined Optimization and Simulation Approach for Designing Automated Storage/ Retrieval Systems. IIE Transactions. Vol. 25, pp. 25-50.

Sammons, S. M.; Cochran, J.K. 1996. The Use of Simulation in the Optimization of a Cellular Manufacturing System. Proceedings of the 1996 Winter Simulation Conference. pp. 1129-1134.

CAPÍTULO 5:

Aplicación de Redes Neuronales en el calculo de Sobretensiones y tasa de contornamientos

6.1. INTRODUCCIÓN

Este capítulo presenta los resultados obtenidos al analizar el comportamiento de líneas aéreas de distribución frente al rayo empleando redes neuronales. El cálculo de la tasa de contorneamientos se realizará de manera estadística aplicando el método de Monte Carlo, siguiendo el procedimiento que se ha comentado en el capítulo anterior.

La aplicación del método de Monte Carlo en el análisis estadístico de sobretensiones atmosféricas en líneas aéreas de distribución puede ser una tarea muy laboriosa debido al elevado número de parámetros que intervienen en una sobretensión. Además, la duración de esta tarea puede ser muy larga si los algoritmos de cálculos de sobretensiones son muy sofisticados. Debido a esto, en este capítulo se ha desarrollado una aproximación basada en el empleo de redes neuronales.

Una red neuronal es un procesador paralelo que puede ser entrenado para reproducir un determinado comportamiento natural o artificial. La principal ventaja del empleo de una red neuronal frente a un algoritmo tradicional podría estar por tanto en la velocidad de cálculo si el algoritmo es muy sofisticado. Esto sería todavía más evidente si el número de casos a calcular fuese muy elevado, lo que puede ocurrir con fenómenos como el rayo. Sin embargo, es necesario tener en cuenta que al tiempo de cálculo mediante red neuronal es necesario añadir el tiempo de entrenamiento, y además que actualmente no existe un consenso total sobre el algoritmo de

cálculo más adecuado para obtener sobretensiones inducidas en una línea de distribución por descargas que caen a tierra en sus cercanías. Solo cuando la red neuronal fuese entrenada con datos de campo las ventajas frente a cualquier algoritmo tradicional serían evidentes.

Este capítulo se ha dividido en tres partes.

·La sección 2 presenta una introducción a las redes neuronales y muy especialmente al modelo "Backpropagation" empleado en este estudio.

·En la sección 3 se detalla el proceso de entrenamiento de redes neuronales que se ha seguido y los resultados obtenidos con los distintos estudios realizados, concretamente la aplicación de una red neuronal en la clasificación de descargas atmosféricas y en el cálculo de sobretensiones originadas por estas descargas. El cálculo de las sobretensiones inducidas por descargas a tierra se ha realizado mediante la fórmula de Rusck y mediante el método de Chowdhuri.

·Finalmente, la última sección presenta el estudio de validación que se basará fundamentalmente en el cálculo de la tasa de contorneamientos mediante las redes neuronales desarrolladas anteriormente y mediante los algoritmos de cálculo empleados en el entrenamiento.

6.2. INTRODUCCIÓN A LAS REDES NEURONALES

6.2.1. Conceptos generales

6.2.1.1. Definición

Existen numerosas formas de definir lo que es una red neuronal, como por ejemplo

Red neuronal artificial es una nueva forma de computación, inspirada en modelos biológicos.

Red neuronal artificial es un modelo matemático compuesto por un gran número de elementos procesales organizados en niveles.

Red neuronal artificial es un sistema de computación hecho por un gran número de elementos simples, elementos de proceso muy interconectados,

los cuales procesan información por medio de su estado dinámico como respuesta a entradas externas.

Redes neuronales artificiales son redes interconectadas masivamente, en paralelo, de elementos simples (usualmente adaptativos) y con organización jerárquica las cuales intentan interactuar con los objetos del mundo real del mismo modo que lo hace el sistema nervioso biológico.

En las redes neuronales biológicas, las células (neuronas) corresponden a los elementos de proceso anteriores. Las interconexiones se realizan por medio de las ramas de salida (axones) que producen un número variable de conexiones (sinapsis) con otras neuronas (o quizás con otras partes como músculos y glándulas). Las redes neuronales son sistemas con elementos de proceso muy interconectados.

La compleja operación de las redes neuronales es el resultado de abundantes lazos de realimentación junto con no linealidades de los elementos de proceso y cambios adaptativos de sus parámetros, que pueden definir incluso fenómenos dinámicos muy complicados.

La programación de una red se puede realizar de dos formas

·alterando las estructuras de interconexión entre las células

·cambiando las fuerzas de estas interconexiónes.

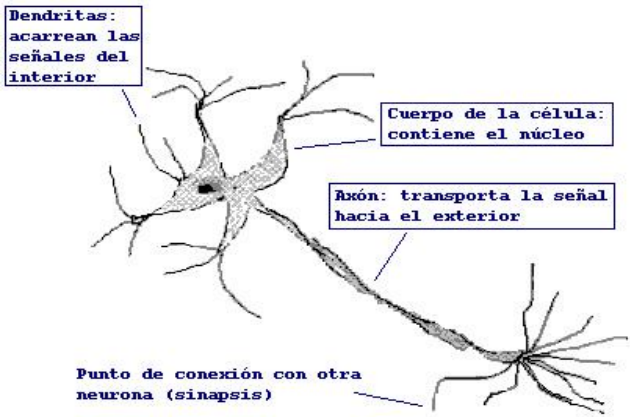


Figura 6.1. Forma general de una neurona.

Parece que existen bastantes estrategias claras sobre como cambiar las fuerzas en la dirección correcta, mientras que los cambios en las interconexiones son mas difíciles de definir, porque suelen tener efectos radicales en el comportamiento de la red, especialmente en lo concerniente a la operación secuencial y las funciones jerárquicas.

6.2.1.2. Aplicaciones

Las redes neuronales forman una teoría computacional emergente que puede utilizarse en gran número y variedad de aplicaciones. Se pueden desarrollar redes neuronales en un periodo de tiempo razonable y pueden realizarse tareas concretas mejor que con otras tecnologías convencionales, incluyendo los sistemas expertos. Cuando se implementan mediante hardware (redes neuronales con chips VLSI), presentan una alta tolerancia a fallos del sistema y proporcionan un grado de paralelismo en el proceso de datos muy grande. Esto hace posible insertar redes neuronales de bajo coste en sistemas existentes y recientemente desarrollados.

Hay muchos tipos diferentes de redes neuronales, cada uno de los cuales tiene una aplicación particular más apropiada. A continuación se muestran algunos ejemplos

·Obtención de modelos de retina (Biología)

·Identificación de candidatos para posiciones específicas (Empresa)

·Previsión del tiempo (Medio ambiente)

·Valoración del riesgo de los créditos (Finanzas)

·Identificación de falsificaciones (Finanzas)

·Inspección de calidad (Manufacturación)

·Control de producción en líneas de proceso (Manufacturación)

·Lectores de rayos X (Medicina)

·Clasificación de las señales de radar (Militar)

·Creación de armas inteligentes (Militar).

Este trabajo está enfocado hacia el campo de reconocimiento de patrones, es decir, se introducen una serie de variables de entrada (intensidad, velocidad,...) y se obtienen varias variables de salida como por ejemplo la tensión inducida en una línea. Serán estas variables de salida las que tendrán que ser estimadas en la fase de aprendizaje de la red neuronal. Para este estudio se empleará el modelo de red neuronal Backpropagation, la cual será descrita en el apartado 6.2.2.

6.2.1.3. Tipos de redes neuronales en el reconocimiento de patrones

Dentro del campo de reconocimiento de patrones, existen actualmente varios modelos de redes neuronales. Entre los más importantes se pueden mencionar los siguientes

1)Self-Organizing-Map (SOM)/Topology-Preserving-Map (TPM)

·aparecen entre los años 1980-1984

·realizan mapas de características comunes de los datos aprendidos

·una limitación importante es la necesidad de un largo proceso de entrenamiento

·creadas por Teuvo Kohonen.

2)Backpropagation

·Aparece en el año 1985 (aunque se conocía desde 1974)

·Es la red más popular

·Se han llevado a cabo numerosas aplicaciones con éxito

·Utiliza un aprendizaje sencillo

·Es potente

·Tiene como limitaciones un largo tiempo de aprendizaje y la necesidad de una cantidad considerable de ejemplos

·Creada por Paul Werbos, David Parker y David Rumelhart.

3)Máquinas de Boltzman y Cauchy

·Aparecen entre los años 1985-1986

·Son un tipo de redes bastante simples

·Representación de patrones de manera óptima

·Necesitan un tiempo muy largo de aprendizaje

·Creada por Jeffrey Hinton, Terry Sejnowski, Harold Szu.

4)Teoría de la resonancia adaptativa (ART)

·Aparece en el año 1986

·Es sofisticada pero poco utilizada

·Como limitaciones tiene la sensibilidad a la translación, distorsión y escala

·Creada por Gail Carpenter y Stephen Grossberg.

6.2.1.4. Elementos de una red neuronal

Una red neuronal es un procesador paralelo que intenta reproducir el comportamiento de una neurona biológica. Cualquier modelo de red neuronal consta de dispositivos elementales de proceso, las neuronas. Cada neurona está caracterizada en cualquier instante por un valor numérico denominado valor o estado de activación ai(t); asociado a cada unidad, existe

una función de salida o de transferencia fi, que transforma el estado actual de activación en una señal de salida yi. Dicha señal es enviada a través de los canales de comunicación unidireccionales a otras unidades de la red; en estos canales la señal se modifica de acuerdo con la sinapsis (el peso wji) asociada a cada uno de ellos según una determinada regla. Las señales moduladas que han llegado a la señal j-ésima se combinan entre ellas, generando así la entrada neta o total.

$$Net_j = \sum_i y_i w_{ji} \qquad (6.1)$$

Una función de activación F, determina el nuevo estado de activación aj(t+1) de la neurona, teniendo en cuenta la entrada total calculada y el anterior estado de activación aj(t).

La dinámica que rige la actualización de los estados de las unidades (evolución de la red neuronal) puede ser de dos tipos: modo asíncrono y modo síncrono. En el primer caso, las neuronas evalúan su estado continuamente, según les va llegando información, y lo hacen de forma independiente. En el caso síncrono, la información también llega de forma continua, pero los cambios se realizan simultáneamente, como si existiera un reloj interno que decidiera cuando deben cambiar su estado. Los sistemas biológicos se encuentran probablemente entre ambas posibilidades.

Es importante resaltar el concepto general de aprendizaje. Se puede considerar que el conocimiento en una red neuronal artificial se encuentra representado en los pesos de las conexiones entre neuronas. Todo proceso de aprendizaje implica cierto número de cambios en estas conexiones. En realidad puede decirse que se aprende modificando los valores de los pesos de la red. Al igual que el funcionamiento de una red depende del número de neuronas de las que disponga y de como estén conectadas entre sí, cada modelo dispone de sus propias técnicas de aprendizaje.

6.2.1.5. Estructura de una red neuronal artificial

Una vez definidos los componentes más importantes de una red neuronal: Unidades de procesamiento (la neurona artificial), estado de activación

de cada neurona, patrón de conexión entre neuronas, regla de propagación, función de transferencia, regla de activación, y regla de aprendizaje; será importante destacar que una red se puede organizar también en función de: Número de niveles o capas, número de neuronas por nivel, patrones de conexión, y flujo de información.

6.2.1.6. Características de las redes neuronales artificiales

Existen cuatro aspectos que caracterizan una red neuronal: la arquitectura, el mecanismo de aprendizaje, el tipo de asociación realizada entre la información de entrada y de salida, y por último, la forma de representación de estas informaciones.

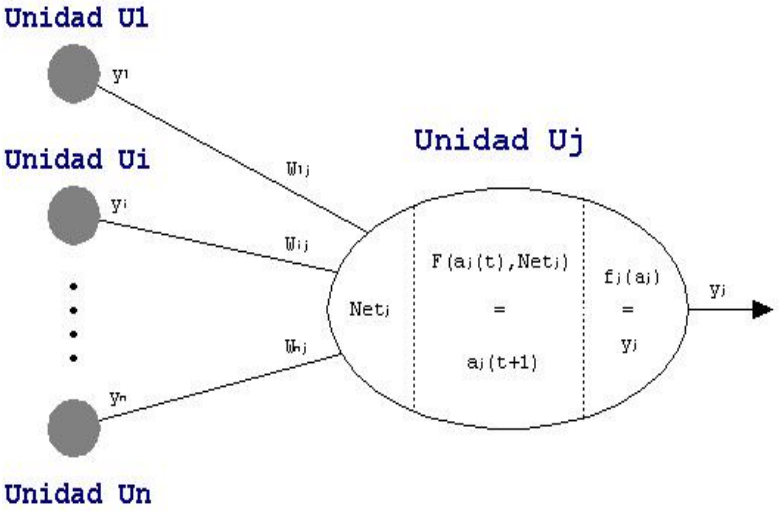


Figura 6.2. Entradas y salidas de una neurona Uj.

a)Arquitectura

La arquitectura o topología de una red neuronal, consiste en la organización y disposición de las neuronas formando capas o agrupaciones de neuronas más o menos alejadas de la entrada y salida de la red. En este sentido los parámetros fundamentales de la red son: el número de capas, el número de neuronas por capa, el grado de conectividad y el tipo de conexiones entre

neuronas.

Cuando se realiza una clasificación de la red en términos topológicos, se suele distinguir entre las redes con una sola capa o nivel de neuronas y las redes con múltiples capas (2, 3, etc.).

b)Mecanismo de aprendizaje

El aprendizaje es el proceso por el cual una red neuronal modifica sus pesos en respuesta a una información de entrada. Los cambios que se producen durante el proceso de aprendizaje se reducen a la destrucción, modificación y creación de conexiones entre las neuronas. En los sistemas biológicos existe una continua creación y destrucción de conexiones. En los modelos de redes neuronales artificiales, la creación de una nueva conexión implica que el peso de la misma pasa a tener un valor distinto de cero.

Durante el proceso de aprendizaje los pesos de las conexiones de la red sufren modificaciones, por tanto se puede afirmar que este proceso ha terminado (la red ha aprendido) cuando los valores de los pesos permanecen estables.

$$\frac{dw_{ij}}{dt} = 0 \qquad (6.2)$$

Un aspecto importante en el aprendizaje de las redes neuronales es conocer como se modifican los valores de los pesos, es decir, cuales son los criterios que se siguen para cambiar el valor asignado a las conexiones cuando se pretende que la red aprenda una nueva información.

Estos criterios determinan lo que se conoce como regla de aprendizaje de la red. De forma general se suelen considerar dos tipos de reglas: las que responden a lo que habitualmente se conoce aprendizaje supervisado, y las correspondientes a un aprendizaje no supervisado. La diferencia fundamental entre ambos tipos estriba en la existencia o no de un agente externo (supervisor) que controle el proceso de aprendizaje de la red.

Otro criterio que se puede utilizar para diferenciar las reglas de aprendizaje se basa en considerar si la red puede aprender durante su funcionamiento habitual o si el aprendizaje supone la desconexión de la red, es decir, su inhabilitación hasta que el proceso termine. En el primer caso, se trataría de un aprendizaje ON LINE, mientras que el segundo es lo que se conoce como aprendizaje OFF LINE.

Cuando el aprendizaje es OFF LINE, se distingue entre una fase de aprendizaje o entrenamiento y una fase de operación o funcionamiento, existiendo un conjunto de datos de entrenamiento y un conjunto de datos de test o prueba que serán utilizados en la correspondiente fase. En las redes con aprendizaje OFF LINE, los pesos de las conexiones permanecen fijos después que termina la etapa de entrenamiento de la red. Debido precisamente a su carácter estático, estos sistemas no presentan problemas de estabilidad en su funcionamiento.

En las redes con aprendizaje ON LINE no se distingue entre fase de entrenamiento y de operación, de tal forma que los pesos varían dinámicamente siempre que se presente una nueva información del sistema. En este tipo de redes, debido al carácter dinámico de las mismas, el estudio de la estabilidad suele ser un aspecto fundamental del estudio.

c) Tipo de asociación entre las informaciones de entrada y salida

Existen dos formas primarias de realizar la asociación entre entrada/salida que se corresponden con la naturaleza de la información almacenada en la red. Una primera sería la denominada heteroasociación, que se refiere al caso en que la red aprende parejas de datos

$$\left[(A_1, B_1), (A_2 , B_2),...,(A_n , B_n)\right] \tag{6.3}$$

de tal forma que cuando se le presenta una información de entrada Ai, deberá responder generando la correspondiente salida asociada Bi. La segunda se conoce como autoasociación, donde la red aprende ciertas informaciones A1,A2,...,An, de tal forma que cuando se le presenta una información de entrada realizará una autocorrelación, respondiendo con uno de los datos almacenados, el más parecido al de la entrada.

Estos dos mecanismos de asociación dan lugar a dos tipos de redes neuronales: las redes heteroasociativas y las autoasociativas. Una red heteroasociativa podría considerarse aquella que computa cierta función, que en la mayoría de los casos no podrá expresarse analíticamente, entre un conjunto de entradas y un conjunto de salidas, correspondiendo a cada posible entrada una determinada salida. Por otra parte, una red asociativa es una red cuya principal misión es reconstruir una determinada información de entrada que se presenta incompleta o distorsionada (le asocia el dato almacenado más parecido).

En realidad estos dos tipos de modelos de redes no son diferentes en principio, porque una red heteroasociativa puede ser siempre reducida a una autoasociativa mediante la concatenación de una información de entrada y su salida (respuesta) asociada, para obtener la información de entrada de la red autoasociativa equivalente.

d) Representación de la información de entrada y salida

Las redes neuronales pueden también clasificarse en función de la forma en que se representan las informaciones de entrada y las respuestas o datos de salida. Así, en un gran número de redes, tanto los datos de entrada como los de salida son de naturaleza analógica, es decir, son valores reales continuos, normalmente normalizados, y su valor absoluto será menor que la unidad. Cuando esto ocurre, las funciones de activación de las neuronas serán también continuas, de tipo lineal o sigmoidal.

Otras redes, por el contrario, sólo admiten valores discretos o binarios {0,1} en su entrada, generando también unas respuestas a la salida de tipo binario. En este caso, las funciones de activación de las neuronas serán del tipo escalón.

La Tabla 6.1 muestra los modelos de red neuronal más conocidos dentro del campo de aplicación del reconocimiento de patrones y las características más relevantes de cada uno según la clasificación que se ha desarrollado.

Tabla 6.1. Características de los modelos de redes neuronales.

MODELO DE RED	TOPOLOGÍA	APRENDIZAJE			ASOCIACIÓN	INFORMACIÓN DE ENTRADA Y SALIDA
		ON LINE/ OFF LINE	SUPERVISADO/ NO SUPERVISADO	REGLA		
TPM	2 capas Feedforward Conexiones laterales	Off	No supervisado	Competitivo	Heteroasociación	Analógica
Backpropagation	n capas Feedforward	Off	Supervisado	Corrección error	Heteroasociación	Analógica
Boltzman Machine	1 capa Conexiones laterales 3 capas Feedforward	Off	Supervisado	Estocástico, Hebbiano, Corrección error	Heteroasociación	Binaria
Cauchy Machine	1 capa Conexiones laterales 3 capas Feedforward	Off	No supervisado	Estocástico	Heteroasociación	E: analógica S: binaria
ART	2 capas Feedforward /Feedback Conexiones laterales	On	No supervisado	Competitivo	Heteroasociación	Analógica

6.2.2. Modelo de red neuronal Backpropagation

6.2.2.1. Introducción

El algoritmo de propagación hacia atrás, o retropropagación (Backpropagation), es una regla de aprendizaje que se puede aplicar en modelos de redes con más de dos capas. De forma simplificada, el funcionamiento de una red Backpropagation consiste en el aprendizaje de un conjunto predefinido de pares de entradas-salidas dados como ejemplo, empleando un ciclo propagación-adaptación de dos fases: primero se aplica un patrón de entrada como estímulo para la primera capa de las neuronas de la red, este estímulo se va propagando a través de todas las capas superiores hasta generar una salida, el resultado obtenido en las neuronas de salida se compara con la salida que se desea obtener y se calcula un valor del error para cada neurona de salida. A continuación, estos errores se transmiten hacia atrás, partiendo de la capa de salida, hacia todas las neuronas de la capa intermedia que contribuyan directamente a la salida, recibiendo el porcentaje de error aproximado a la participación de la neurona intermedia de la salida original. Este proceso se repite capa por capa, hasta que todas las neuronas de la red hayan recibido un error que describa su aportación relativa al error total.

Basándose en el valor del error recibido, se reajustan los pesos de conexión de cada neurona, de manera que la siguiente vez que se presente el mismo patrón, la salida esté más cercana a la deseada.

En la etapa de entrenamiento, este tipo de red neuronal es capaz de adaptar los pesos de las neuronas de las capas intermedias con el objetivo de minimizar el error entre un conjunto de patrones dados como ejemplo y sus salidas estimadas. En la etapa de validación, se aplica esa misma relación a nuevos patrones de entrada no utilizados durante el proceso de entrenamiento. Esta es una de las características más importantes de las redes neuronales. Se entiende como capacidad de generalización a la facilidad que tiene una red neuronal de dar salidas satisfactorias a entradas que el sistema no ha visto nunca en su fase de entrenamiento. En general, la red debe encontrar una representación interna que le permita generar las salidas deseadas cuando se le dan las entradas de entrenamiento, y que pueda aplicar, además, a entradas no presentadas durante la etapa de aprendizaje para clasificarlas según las características que compartan con los ejemplos de entrenamiento.

6.2.2.2. Proceso de aprendizaje de la red neuronal

6.2.2.2.1. Algoritmo de entrenamiento

En una red Backpropagation existe una capa de entrada con n neuronas, una capa de salida con m neuronas y al menos una capa oculta de neuronas internas. Cada neurona de una capa (excepto las de entrada) recibe entrada de todas las neuronas de la capa anterior y envía su salida a todas las neuronas de la capa posterior (excepto las de salida). No hay conexiones hacia atrás (feedback) ni conexiones laterales entre neuronas de la misma capa. La figura 6.3 muestra la estructura general.

A continuación se presentan los pasos y fórmulas a utilizar para aplicar el algoritmo de entrenamiento.

1)Se inicializan los pesos de la red con valores pequeños aleatorios.

2)Se presenta un patrón de entrada, X_P (x_{P1},x_{P2},...,x_{PN}), y se especifica la salida deseada que debe generar la red (d_1,d_2,...,d_m). Si por ejemplo la red es utilizada como un clasificador todas las salidas serán cero, salvo una, que será la de la clase a la que pertenece el patrón de entrada.

3)Se calcula la salida actual de la red. Para ello se presentan las entradas a la red y se va calculando la salida que presenta cada capa hasta llegar a la capa de salida (y1,y2,...,ym). Los pasos serían los siguientes

·Se calculan las entradas netas para las neuronas ocultas procedentes de las neuronas de entrada; para una neurona j oculta

$$net^{(h)}_{pj} = \sum_i w^{(h)}_{ji} x_{pi} + \theta^{(h)}_j \qquad (6.4)$$

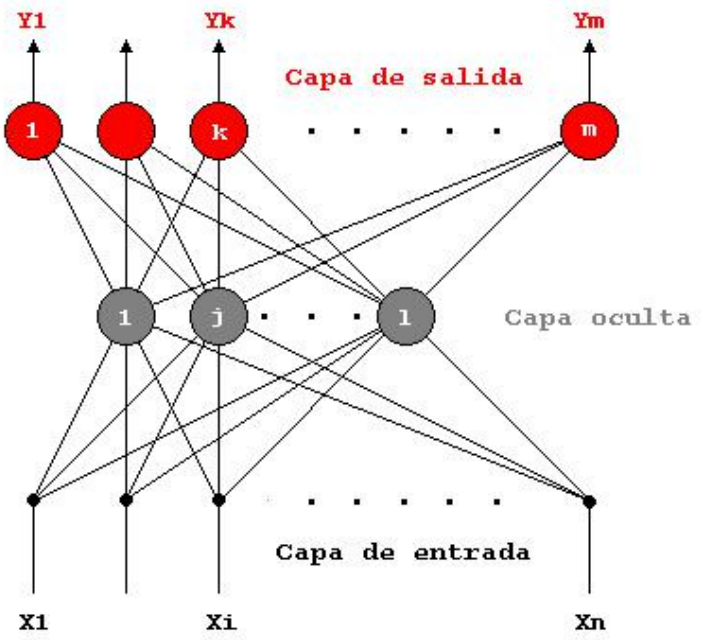


Figura 6.3. Modelo de arquitectura de una red Backpropagation.

en donde el índice h se refiere a magnitudes de la capa oculta (hidden), el subíndice p, al p-ésimo vector de entrenamiento, j a la j-ésima neurona oculta, y θ es el umbral de activación de la neurona y puede ser opcional; el índice i se varía entre 1 y el número de neuronas de la capa de entrada.

·Se calculan las salidas de las neuronas ocultas

$$y_{pj} = f_j^{(h)} (net^{(h)}_{pj}) \tag{6.5}$$

·Se realizan los mismos cálculos para obtener las salidas de las neuronas de salida (output)

$$net^{(o)}_{pk} = \sum w_{kj}^{(o)} y_{pj} + \theta_k^{(o)} \tag{6.6}$$

$$y_{pk} = f_k^{(o)} (net^{(o)}_{pk}) \tag{6.7}$$

donde el índice j varía entre 1 y el número de neuronas de la capa oculta.

4)Se calculan los términos de error de las salidas de todas las neuronas. Si la neurona k está en la capa de salida el error es

$$\delta^{(o)}_{pk} = (d_{pk} - y_{pk}) f_k^{(o)'} (net^{(o)}_{pk}) \tag{6.8}$$

siendo $fk^{(o)'}(net^{(o)}pk)$ la tasa de cambio o pendiente de la función de transferencia. Si la neurona j no es de salida el error es

$$\delta_{pj}^{(h)} = f_j^{(h)'} (net^{(h)}_{pj}) \sum \delta^{(o)}_{pk} w_{kj}^{(o)} \tag{6.9}$$

valor que depende de todos los términos de error de la capa de salida.

Los umbrales internos de las neuronas se adaptan de forma similar, considerando que están conectados con pesos desde entradas auxiliares de valor constante.

5)Se actualizan los pesos. Para ello, se utiliza el algoritmo recursivo, comenzando por las neuronas de salida y trabajando hacia atrás hasta llegar a la capa de entrada, ajustando los pesos de la siguiente forma

·Para los pesos de las neuronas de la capa de salida

$$w^{(o)}_{kj}(t+1) = w^{(o)}_{kj}(t) + \Delta w^{(o)}_{kj}(t+1) \tag{6.10}$$

$$\Delta w^{(o)}_{kj}(t+1) = \alpha \delta^{(o)}_{pk} y_{pj} \tag{6.11}$$

·Para los pesos de las neuronas de una capa oculta

$$w^{(h)}_{ji}(t+1) = w^{(h)}_{ji}(t) + \Delta w^{(h)}_{ji}(t+1) \tag{6.12}$$

$$\Delta w^{(h)}_{ji}(t+1) = \alpha \delta^{(h)}_{pj} x_{pi} \tag{6.13}$$

siendo α la tasa de aprendizaje (amplitud de paso).

A mayor tasa de aprendizaje, mayor es la modificación de los pesos en cada iteración, con lo que el aprendizaje será en general más rápido. Aunque, por otro lado, puede dar lugar a oscilaciones. Para filtrar esas oscilaciones se añade en las expresiones anteriores el término constante β (momento) que determina el efecto en t+1 del cambio de los pesos en el instante t

$$... + \beta \left[w_{ji}(t) - w_{ji}(t-1) \right] \tag{6.14}$$

6) El proceso se repite hasta que el error

$$E_p = \frac{1}{2} \sum_{k=1}^{m} \left[(d_{pk} - y_{pk}) \right]^2 \tag{6.15}$$

Resulta aceptablemente pequeño para cada uno de los patrones aprendidos.

6.2.2.2.2. Variables significativas del entrenamiento

A continuación se describen las variables más importantes que actúan en el algoritmo de aprendizaje. La variación de dichas variables puede alterar de manera considerable los resultados del entrenamiento, y por tanto el funcionamiento de la red neuronal.

·Número de neuronas de la capa oculta: Generalmente un aumento de neuronas ocultas aumenta la eficacia del aprendizaje. Sin embargo, hay que tener en cuenta que existe un margen óptimo en el número de neuronas ocultas, es decir, fuera de ese intervalo los resultados pueden empeorar. Con un error elevado en la salida, un aumento del número de neuronas ocultas puede hacer que disminuya dicho error. De todas formas la tendencia es tener el menor número posible de ellas ya que el proceso de aprendizaje es más rápido.

·Número de capas ocultas: Un aumento del número de capas de neuronas ocultas se traduce en un cambio en la estructura de la red, pudiéndose obtener resultados diferentes. Generalmente con una capa oculta es suficiente. En una red neuronal sin capas ocultas normalmente solo se pueden tener relaciones lineales entre la entrada y la salida.

·Condiciones iniciales: Una mejora en las condiciones iniciales de pesos y umbrales da la posibilidad de que se alcance un mínimo global y no uno local. Diferentes condiciones iniciales pueden hacer que se alcancen diferentes mínimos, y por tanto diferentes resultados.

·Número de iteraciones: Un número elevado de ellas significa un entrenamiento lento. Mediante control de los otros parámetros se puede conseguir que la red entrene con un número más reducido de iteraciones. Recuérdese que el objetivo del entrenamiento de una red neuronal es adaptar los pesos y los umbrales de las neuronas para así minimizar el error entre un conjunto de patrones dados como ejemplo y sus salidas estimadas. La figura 6.4 muestra un ejemplo de superficie de error típico con una función de transferencia sigmoidal logística.

·Patrones de entrada y salida: Un aumento del número de patrones de entrada y salida hace que la red tenga un mejor aprendizaje, y que por tanto cuando se muestren a la red datos desconocidos (diferentes de los utilizados en el entrenamiento) el error en el cálculo de las salidas sea más pequeño. Sin

embargo, cuanto mayor sea el número de datos más largo es el proceso de aprendizaje de la red neuronal. Es muy importante hacer un estudio previo de los datos para evitar confusiones a la red neuronal. Cuanto más definidos estén los datos más rápido aprenderá la red.

·Tasa de aprendizaje: La tasa de aprendizaje es la encargada de acelerar el proceso de aprendizaje. Se suelen escoger valores pequeños, empezando con una tasa constante. Lógicamente es deseable que, hasta alcanzar el mínimo, el entrenamiento sea rápido mientras que en el entorno del mínimo sea lento para poder alcanzarlo plenamente. Por este motivo se utiliza una tasa de aprendizaje variable, adaptativa a lo largo del proceso de entrenamiento.

·Momento: Contrarresta las posibles inestabilidades que se crean en la variación de los pesos, y es importante porque reduce la posibilidad de caer en un mínimo local, además puede acelerar enormemente el proceso.

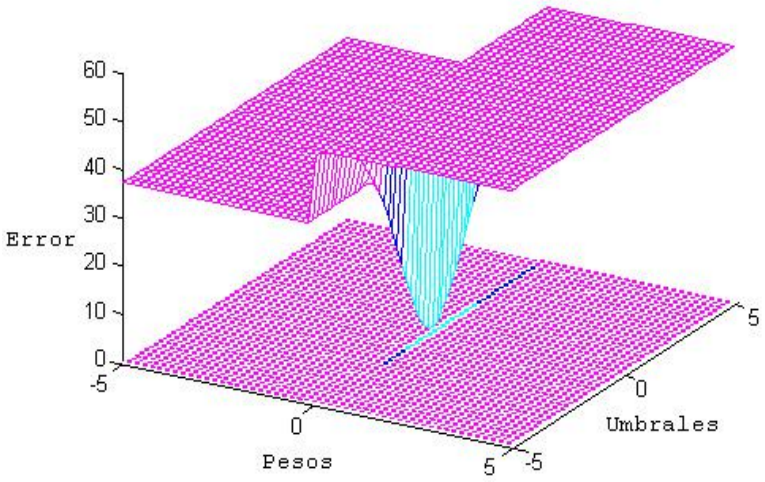


Figura 6.4. Superficie del error.

·Función de transferencia: La función de transferencia o función de activación es la encargada de representar las salidas en función de las entradas a cada neurona. Para el modelo de red Backpropagation dicha función debe ser derivable. Las figuras 6.5, 6.6 y 6.7 muestran algunas de las funciones de transferencia más utilizadas con este modelo. Las dos sigmoidales han de ser adaptadas a las salidas, es decir, los datos de salida han de ser norma-

lizados. También es muy importante normalizar las entradas netas, y una vez normalizadas elegir su mejor intervalo. A continuación se explican más detalladamente estas técnicas.

6.2.2.2.3. Técnicas para la mejora del entrenamiento

A continuación se comentan una serie de técnicas interesantes que pueden mejorar en algunos casos la eficacia del entrenamiento de una red neuronal.

a)Variación de la pendiente de la función de transferencia La función de transferencia lineal responde a la expresión

$$f(x) = a \times x + b \tag{6.16}$$

Donde a es la pendiente, x es la entrada neta de la neurona y b es el umbral de la neurona. Variando la pendiente se puede conseguir una mejor adaptación de las salidas de la red neuronal. En la figura 6.8 se puede observar como varía la función de transferencia al variar la pendiente.

Con la función sigmoidal, para la mayoría de los valores del estímulo de entrada (variable independiente), el valor dado por la función es cercano a uno de los valores asintóticos. Esto hace que en la mayoría de los casos, el valor de salida esté comprendido en la zona alta o baja del sigmoide. De hecho, cuando la pendiente es elevada, esta función tiende a la función escalón. Sin embargo, la importancia de la función sigmoidal es que su derivada es siempre positiva y cercana a cero para los valores grandes positivos o negativos; además, toma su valor máximo cuando x es 0. Esto hace que se puedan utilizar las reglas de aprendizaje definidas para la funcion escalón, con la ventaja, respecto a esta función, de que la derivada está definida en todo el intervalo. La función sigmoidal logística responde a la expresión

$$f(x) = \frac{c}{1 + \exp\{-a(x + b)\}} + d \tag{6.17}$$

Con d=0 y c=1 se tiene una función sigmoidal que toma los valores asintóticos 0 y 1.

Ajustando estos parámetros se pueden obtener salidas deseadas por el usuario.

La función de transferencia sigmoidal tangencial hiperbólica es similar a la anterior, aunque en este caso la salida también puede ser negativa. Esta función responde a la expresión.

$$f(x) = c \times \left(\frac{\exp\{2a(x+b)\} - 1}{\exp\{2a(x+b)\} + 1} \right) + d \qquad (6.18)$$

Con d=0 y c=1 se tiene una función sigmoidal que toma los valores asintóticos -1 y 1.

Las figuras 6.9 y 6.10 muestran la importancia de adaptar la pendiente de estas funciones. Al disminuir la pendiente hay más probabilidad de que las salidas no sean las asintóticas.

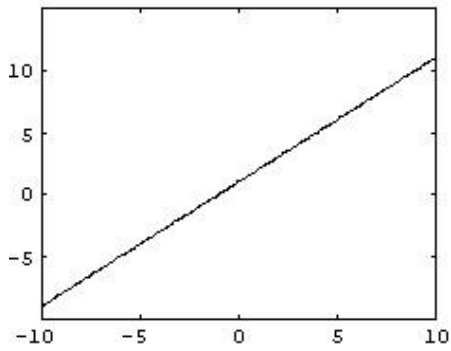


Figura 6.6. Función de transferencia sigmoidal logística.

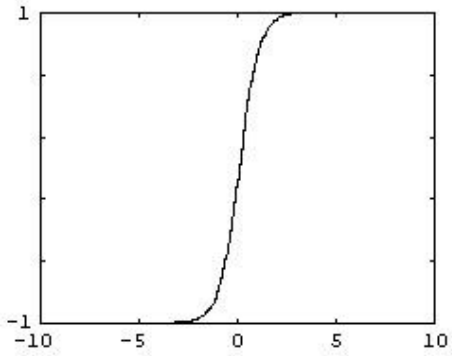


Figura 6.7. Función de transferencia sigmoidal tangencial hiperbólica.

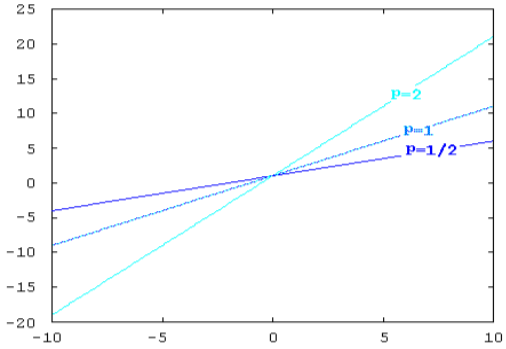


Figura 6.8. Función de transferencia lineal. Variación de la pendiente.

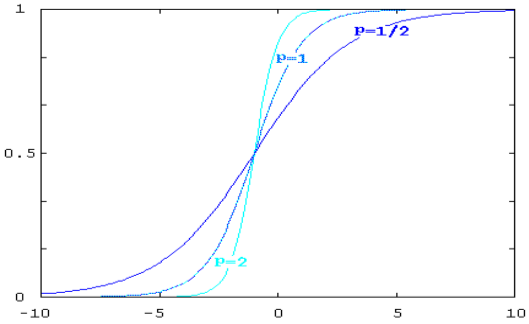


Figura 6.9. Función de transferencia log-sigmoidal. Variación de la

pendiente.

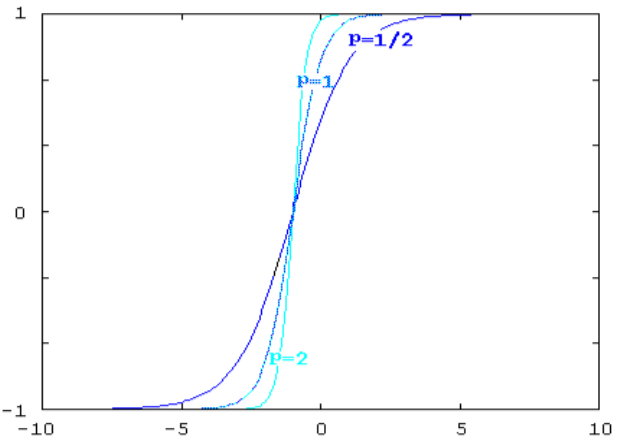


Figura 6.10. Función de transferencia tan-sigmoidal. Variación de la pendiente.

b)Intervalo de las entradas netas

El mínimo error que alcanza la red durante el entrenamiento en algunos casos depende considerablemente del intervalo escogido para las entradas netas de las neuronas. En ambas funciones de transferencia, sigmoidal logística y sigmoidal tangencial, las salidas están acotadas entre 0 y 1 o entre -1 y 1 respectivamente, por tanto habrá un intervalo máximo a partir del cual las salidas o bien no variarán o bien empeorarán. En el caso de utilizar una función lineal, dado que la salida no está acotada, el intervalo de las entradas netas no tiene límite de apertura.

Veamos algunos ejemplos. La figura 6.11 muestra la función de transferencia sigmoidal logística con un umbral (variable b) igual a 1. Para conseguir que la función proporcione valores entre 0 y 1, las entradas han de tener un valor cercano al 0. Se puede observar que los valores de salida comienzan a ser asintóticos a partir de unos valores de entrada entre -5 y 5. No obstante este intervalo no tiene porque ser el mejor, puesto que esto depende de los datos de entrada y del umbral de activación de la neurona. Las figuras 6.12 y 6.13 muestran los diferentes intervalos de salida según se escojan los intervalos {0,1} o {-1,1} para las entradas. Dará mejores resultados un intervalo

{-1,1}, porque es mayor. También hay que tener en cuenta que adaptando el umbral se puede obtener cualquier salida dentro del intervalo {0,1}. La figura 6.14 muestra que un cambio en el umbral de la neurona (por ejemplo, b=3), desplaza la gráfica de la función de transferencia a lo largo del eje de abcisas.

Las directrices a seguir para el correcto ajuste del intervalo de las entradas netas pueden ser

1)Normalizar las entradas netas simétricamente, es decir, tomando como extremos 1 y -1

2)Aumentar el intervalo de dichas entradas hasta obtener un error mínimo.

c)Normalización de los datos

A continuación se comentan algunos de los métodos más utilizados para normalizar datos. Método 1: MÁXIMO

Supóngase que los datos que se quieren normalizar se encuentran dentro del vector DATOS(i), con i=1,...,n. El procedimiento a seguir es el siguiente:

1)Se busca el máximo del vector DATOS(i)

2)Se normalizan los datos según la relación

$$DATOS\_norm(i) = DATOS(i)/Max \tag{6.19}$$

Los datos normalizados caen dentro del intervalo {0,1} en el caso de que los datos originales sean positivos, si también son negativos caerán dentro del intervalo {-1,1}.

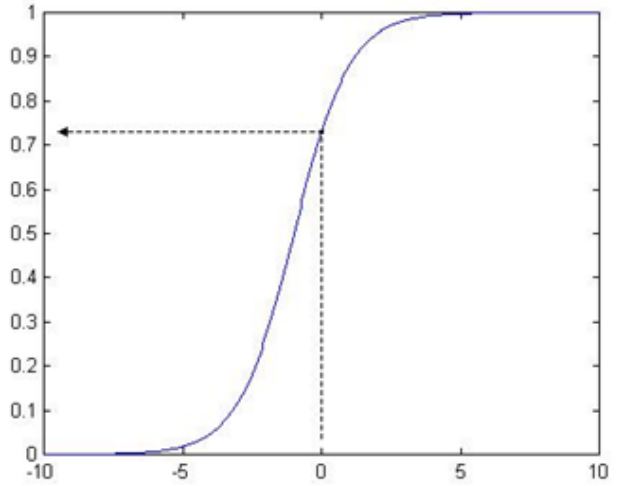


Figura 6.11. Función de transferencia log-sigmoidal (completa/b=1).

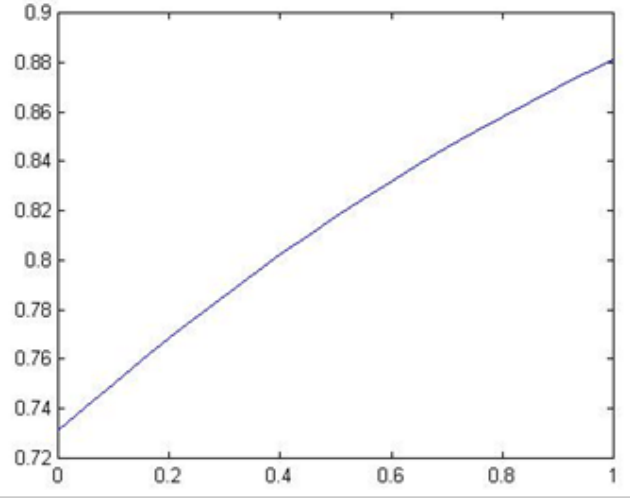


Figura 6.12. Función de transferencia log-sigmoidal (intervalo {0,1}/ b=1).

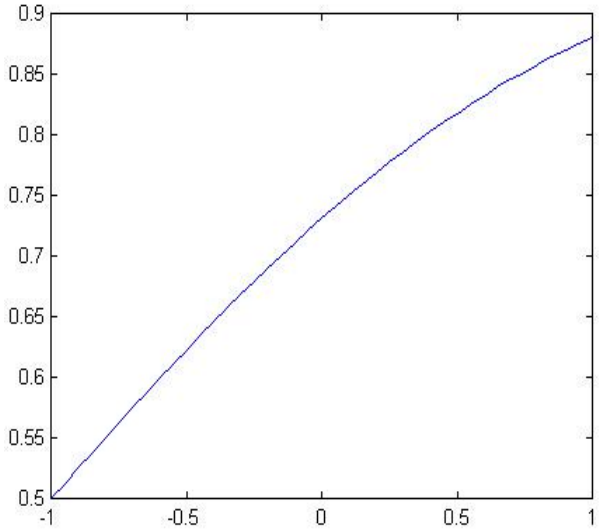


Figura 6.13. Función de transferencia log-sigmoidal (intervalo {-1,1}/ b=1).

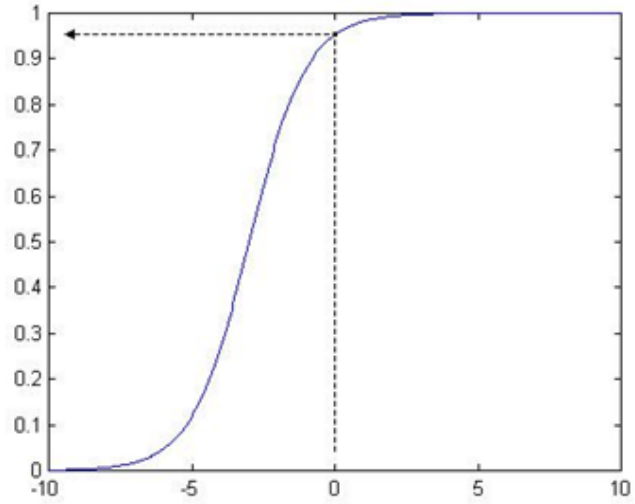


Figura 6.14. Función de transferencia log-sigmoidal (completa/b=3).

Método 2: MÍNIMO-MÁXIMO

El procedimiento a seguir es bastante parecido al anterior:

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

2)Se normalizan los datos según la relación

$$DATOS\_norm(i) = \frac{DATOS(i) - Min}{Max - Min} \tag{6.20}$$

Estos datos normalizados caen dentro del intervalo {0,1} sean cuales sean los datos originales. No obstante es importante destacar que en los datos normalizados siempre habrá un valor que será 0 y otro que será 1.

Método 3: MEDIA-DESVIACIÓN TIPO

Con este método se pueden obtener datos normalizados dentro del intervalo {-1,1} o dentro del intervalo {0,1}. Inicialmente se calcula.

$$DATOS\_norm(i) = \frac{DATOS(i) - \mu}{\sigma} \tag{6.21}$$

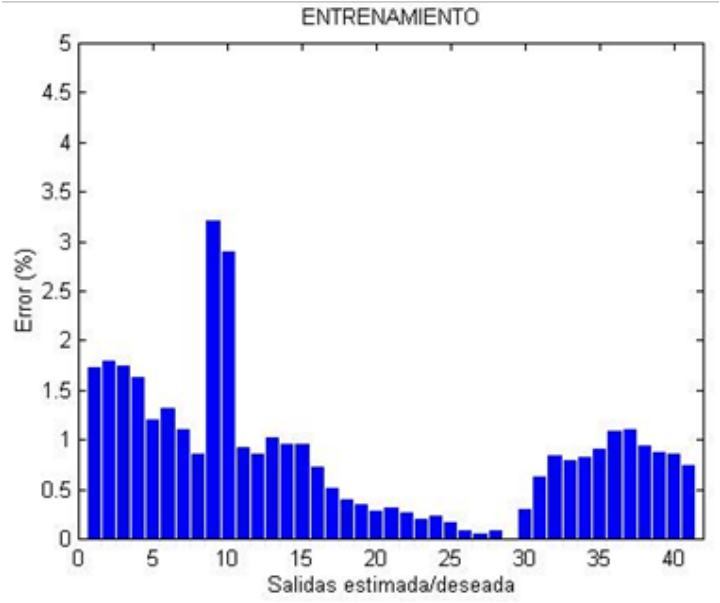
Los datos normalizados caen dentro del intervalo {-k1,k2}. Nótese que k1 y k2 son números reales positivos, y que μ y σ son la media y la desviación tipo de los datos de entrada, respectivamente. Para que los datos normalizados caigan dentro del intervalo {-1,1}, se dividen por el mayor de k1 o k2. Para que los datos normalizados caigan dentro del intervalo {0,1} se hace lo siguiente:

1)A los datos normalizados se les suma el mayor de k1 y k2. Ahora estos datos caen dentro del intervalo {0,k3}, siendo k3=k1+k2.

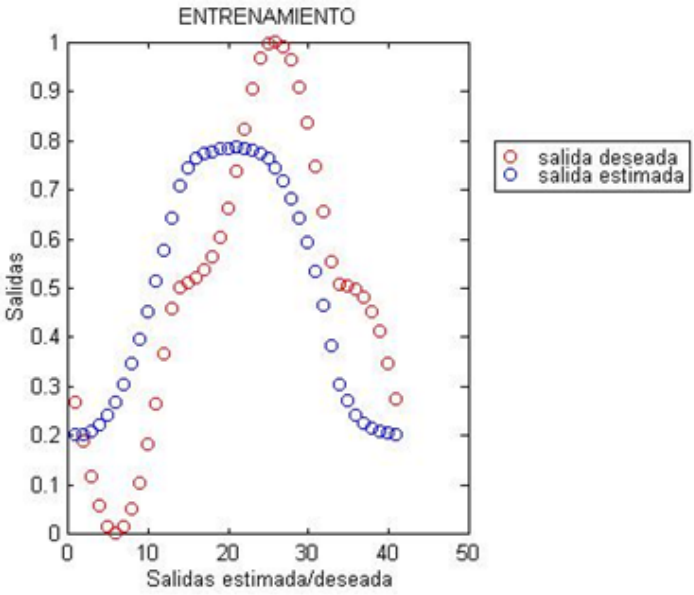
2)Los datos normalizados se dividen por k3, obteniéndose así el intervalo de {0,1}.

6.2.2.3. Proceso de validación de la red neuronal

Una vez que la red neuronal ha sido entrenada, se procede a su validación, es decir, se presentan datos diferentes de los utilizados en el proceso de aprendizaje. El objetivo es estudiar la respuesta de la red neuronal frente a datos que no han sido utilizados durante el entrenamiento. Para visualizar la validación de la red se dispone de gráficos similares a los realizados durante el entrenamiento. La figura 6.15 muestra un ejemplo de validación de una red neuronal Backpropagation.



a) Error porcentual en las salidas de la red neuronal



b)Salidas calculadas vs. salidas deseadas

Figura 6.15. Validación de una red neuronal.

6.3. ENTRENAMIENTO DE REDES NEURONALES EN EL CÁLCULO DE SOBRETENSIONES POR RAYOS

6.3.1. Introducción

El objetivo del entrenamiento es minimizar la diferencia entre la salida real y la calculada por la red neuronal mediante el ajuste de los parámetros de la red. Como resultado se obtiene la configuración de la red mediante pesos (W) y umbrales (B).

A continuación se resume la estrategia general seguida en el entrenamiento

1)Se empieza con una capa oculta y un número determinado de neuronas ocultas. Se fijan los valores de 0.25 para la tasa de aprendizaje y 0.9 para el momento. Se utiliza la función de transferencia sigmoidal logística para todas las neuronas tanto de las capas ocultas como de la capa de salida.

Dentro de este apartado se distinguen las siguientes fases

·Se intentan encontrar unas buenas condiciones iniciales de pesos y umbrales con el objetivo de estimar un primer comportamiento de la red neuronal

·Con las condiciones iniciales anteriores se ajusta el intervalo de las entradas netas y la pendiente de la función de transferencia.

2)Se varía el número de neuronas de la primera capa oculta

3)En caso de ser necesario se añade una segunda capa oculta y se varía el número de neuronas de esta capa

4)Con la arquitectura elegida se terminan de ajustar los parámetros de la tasa de aprendizaje y el momento.

Los valores son normalizados según el método media-desviación tipo, ver apartado 6.2.2.2.3.

El entrenamiento de la red neuronal en el cálculo de sobretensiones originadas por el rayo se realizará de dos formas diferentes según se tenga en cuenta la función de probabilidad que sigue la velocidad de retorno del rayo: distribución uniforme y velocidad en función de la intensidad de pico de la descarga, ver expresión (3.10).

Antes de mostrar el entrenamiento de la red neuronal en el cálculo de sobretensiones originadas por el rayo, se muestra un ejemplo de red neuronal trabajando como clasificador de tipos de impacto del rayo en una línea sin cable de tierra con una altura media de sus conductores de 10 m, ver figura 4.2. No se estudiará el clasificador para líneas con cable de tierra porque siguiendo el mismo ejemplo, tal como muestra la figura 4.4, la probabilidad de que un rayo impacte en un conductor de fase es muy reducida, por lo que el clasificador de descargas se convierte en el estudiado en el caso de líneas sin cable de tierra, aunque con la diferencia de que los impactos pueden ser a tierra o al cable de tierra, en lugar de a los conductores de fase.

6.3.2. Clasificador de descargas atmosféricas. Líneas sin cable de tierra

El objetivo es comprobar que la red neuronal puede funcionar como clasificador y distinguir entre descargas a la línea y descargas a tierra. Además se comprobará que las pautas del entrenamiento a seguir posteriormente son adecuadas. Esta tarea se ha realizado exclusivamente con una línea cuyos conductores se encuentran todos a una altura media de 10 m.

Las variables de entrada y salida de la red neuronal son las siguientes

·Entradas (variables independientes): Intensidad máxima de la descarga (variable I), distancia perpendicular entre la línea y la descarga (variable y)

·Salida de la red neuronal (variable dependiente): Tipo de descarga, directa / indirecta (variable t).

La red neuronal se ha entrenado con un total de 300 patrones generados de forma aleatoria. Cada patrón esta formado por un valor para cada entrada (I,y) y el correspondiente valor para la salida (t).

a)Intensidad de la descarga

Los valores de la intensidad de la descarga se generan siguiendo el procedimiento visto en el apartado 5.3.1 del capítulo 5.

b)Distancia perpendicular entre la línea y la descarga

El valor de esta distancia se genera de forma aleatoria, entre 0 m y 500 m, utilizando una distribución uniforme.

c)Tipo de descarga

Se utiliza el Modelo Electrogeométrico, ver expresión (4.2), para saber si

una descarga, caracterizada por los parámetros que componen la entrada de la red neuronal (I,y), es directa a la línea (t = 1) o directa a tierra (t = 0). La figura 6.16 visualiza la clasificación de descargas de acuerdo con el Modelo Electrogeométrico descrito en el capítulo 4, sección 4.2.

Entrenamiento

La Tabla 6.2 muestra una pequeña parte de los datos utilizados en el entrenamiento. El proceso seguido se detalla a continuación.

1)Se fijan los parámetros iniciales de la red neuronal

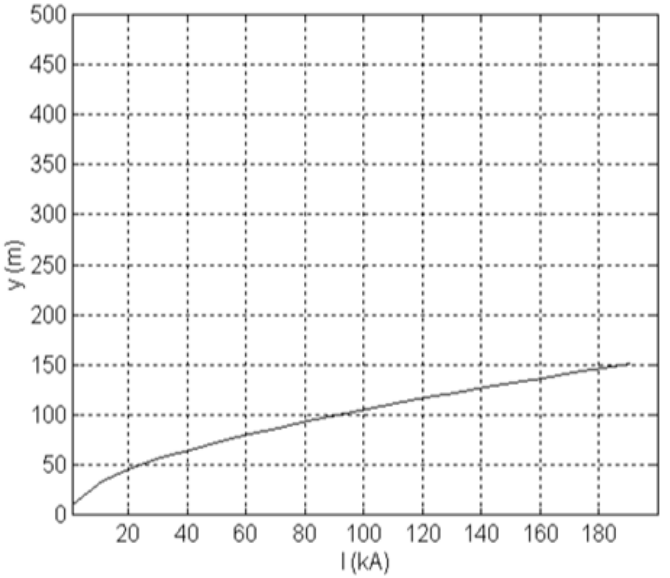
·1 capa de neuronas ocultas

·4 neuronas en la capa oculta

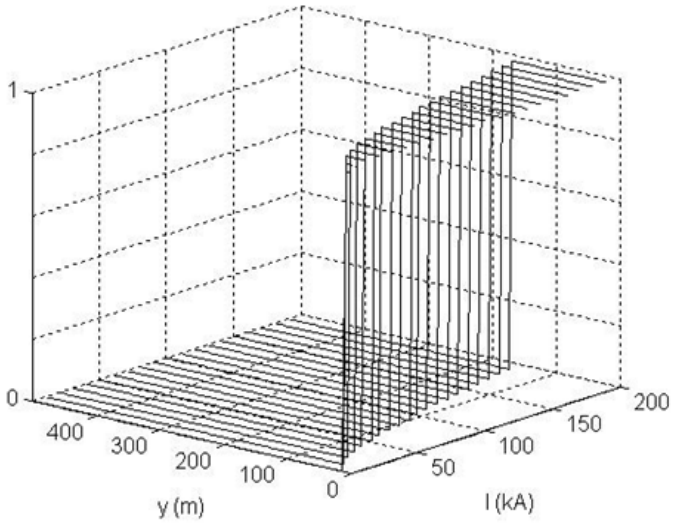
·Funciones de transferencia: log-sigmoidal

·Tasa de aprendizaje variable, $\alpha_0=0.25, \alpha_i=1.01, \alpha_d=0.6$

·Momento $\beta=0.9$.



a) Modelo Electrogeométrico. Vista bidimensional.



b) Modelo Electrogeométrico. Vista tridimensional.

Figura 6.16. Clasificador de descargas atmosféricas de acuerdo con el modelo electrogeométrico (h = 10 m)

Tabla 6.2. Datos del entrenamiento.

ENTRADAS		SALIDA
Intensidad (kA)	Distancia línea - descarga (m)	Tipo descarga
43	68	0
8	27	1
34	416	0
30	114	0
11	21	1
74	433	0
41	66	0
37	497	0
14	374	0
42	242	0
30	4	1
3	219	0
57	340	0
10	299	0
16	467	0
9	331	0
17	119	0
11	5	1
54	481	0
9	98	0
13	69	0
47	288	0
8	499	0
11	235	0
16	292	0
17	254	0
22	366	0
18	397	0
31	39	1
7	246	0
20	395	0
38	178	0
16	329	0
25	9	1
16	203	0
66	388	0
47	232	0
53	36	1
23	123	0
18	128	0
24	112	0
18	160	0
36	16	1
33	449	0
58	18	1
41	310	0
6	435	0
23	427	0
6	82	0
38	117	0
16	269	0

a)Estudio previo. Se pretenden conseguir unas buenas condiciones iniciales para el estudio del intervalo de las entradas netas y el de la pendiente de la función de transferencia. Tras varios entrenamientos, se han escogido las condiciones iniciales de pesos y umbrales que dan el mínimo error global. Los dos próximos estudios se realizarán con 1000 iteraciones.

b)Ajuste del intervalo de las entradas netas de la función de transferencia. Se aumenta el intervalo de dichas entradas hasta que se alcanza un mínimo. Solo ajustando el intervalo de las entradas netas se consiguen muy buenos resultados. Recuérdese que todavía no se han ajustado los parámetros principales de la red. Se acepta que las entradas de la red neuronal se encuentren dentro del intervalo [-32,32].

c)Ajuste de la pendiente de la función de transferencia. Se varía la pendiente de la función de transferencia hasta encontrar el mínimo error. Es importante destacar que aunque hay dos funciones de transferencia, las pendientes de dichas funciones se variarán por igual. Del estudio se deduce que una variación de la pendiente no disminuye el error.

2)Se varía el número de neuronas de la capa oculta. Se han realizado 2 estudios con cada estructura diferente, uno a 1000 iteraciones para buscar unas buenas condiciones iniciales, y otro más largo, a 10000 iteraciones, para encontrar un buen mínimo. Se ha llegado a la conclusión de que un aumento de neuronas ocultas (de 4 a 6) no sólo mejora los resultados sino que además acelera enormemente el proceso. Por ejemplo, mientras que con 4 neuronas ocultas el mínimo global se encuentra a más de 100000 iteraciones, con 6 neuronas ocultas un error bastante más pequeño se consigue en menos de 1000 iteraciones.

3)Se añade una capa más de neuronas ocultas y se mantiene constante el número de neuronas de la primera capa. En este caso, con una capa más de neuronas ocultas el entrenamiento se ralentiza considerablemente. Además, no se consiguen mejores resultados.

4)Del estudio anterior se concluye que una estructura de red neuronal aceptable es la siguiente • primera capa oculta: 6 neuronas

• Funciones de transferencia: log-sigmoidal.

A continuación se intentarán ajustar los parámetros propios de la fase del aprendizaje tales como el momento y la tasa de aprendizaje. Es importante resaltar que con estos ajustes (incluido el del momento) el error puede disminuir, aunque de forma poco sensible.

a)Tasa de aprendizaje. El entrenamiento se realiza con una tasa de aprendizaje variable, ya que después de realizar una serie de pruebas con diferentes valores de este parámetro se admite que una tasa de aprendizaje constante empeora los resultados. Además se llega también a la conclusión de que el valor adoptado durante el entrenamiento por los diferentes parámetros que componen la tasa de aprendizaje variable es óptimo.

b)Ajuste del momento. El valor actual (0.9) es aceptable.

5)La estructura final de la red escogida, ver figura 6.17, con todos sus parámetros ajustados es la siguiente.

·1 capa de 6 neuronas ocultas

·Funciones de transferencia: log-sigmoidal

·Tasa de aprendizaje variable, $\alpha 0=0.25, \alpha i=1.01, \alpha d=0.6$

·Momento: $\beta=0.9$.

Los resultados del entrenamiento con esta red se exponen en la figura 6.18. Se puede observar que el ajuste es muy bueno. No solo la red neuronal no se confunde en ninguno de los casos sino que además ningún error supera el 5%. La Tabla 6.3 muestra los patrones del entrenamiento que han superado el 1 % de error. Se puede ver que tan solo 5 patrones, de una cantidad total de 300, han superado este error.

Validación

A continuación se valida la red neuronal analizando su comportamiento con datos que no han sido utilizados en el proceso de entrenamiento. La validación de esta red neuronal se realiza con 2000 patrones. En los gráficos de la figura 6.19 se puede observar el error cometido en el cálculo de las salidas y el ajuste de las salidas con los valores de los pesos y umbrales óptimos conseguidos en el entrenamiento de la red neuronal. De estos gráficos se deduce que tan solo unos pocos patrones, 13, son confundidos por la red neuronal. Esto significa que la red clasifica correctamente en más de un 99 % de los casos. La Tabla 6.4 muestra las salidas calculadas para estos patrones.

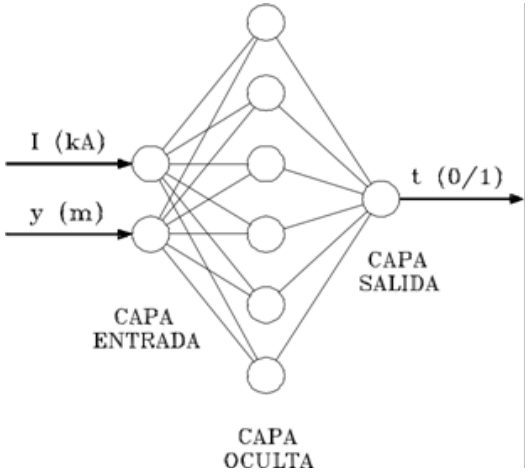


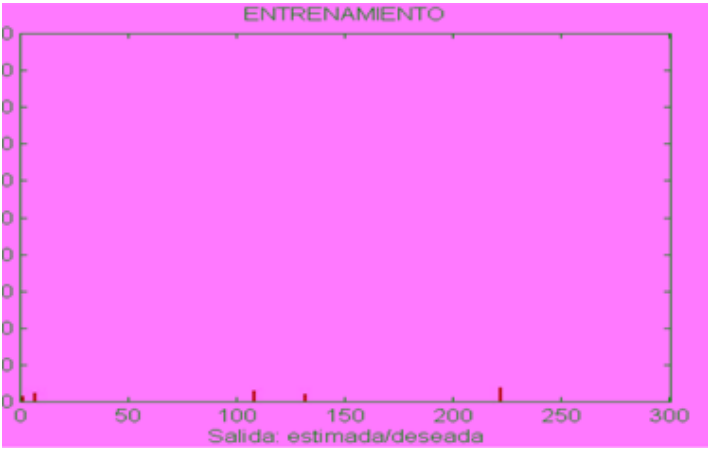
Figura 6.17. Arquitectura de red neuronal.

Tabla 6.3. Resultados del entrenamiento.

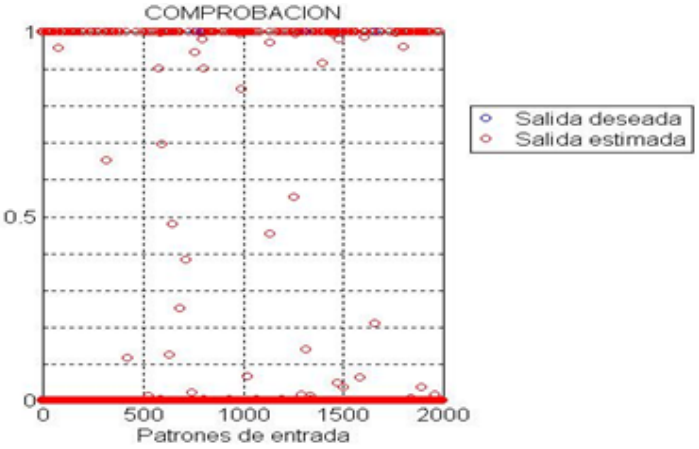
ENTRADAS		SALIDA DESEADA	SALIDA CALCULADA	
Intensidad (kA)	**Distancia línea - descarga (m)**	**Tipo descarga**	**Tipo descarga**	**Error (%)**
43	68	0	0.0155	1.55
41	66	0	0.0234	2.34
7	38	0	0.0292	2.91
36	55	1	0.9797	2.03
23	45	1	0.9623	3.77

Tabla 6.4. Resultados de la validación.

ENTRADAS		SALIDA DESEADA	SALIDA CALCULADA	
Intensidad (kA)	**Distancia línea - descarga (m)**	**Tipo descarga**	**Tipo descarga**	**Error (%)**
13	37	0	0.9551	95.50
9	36	0	0.6519	65.19
12	37	0	0.9005	90.04
7	29	0	0.9976	99.76
8	35	0	0.6960	69.60
38	62	1	0.1242	87.57
157	141	0	1.0000	99.99
24	49	1	0.4525	54.74
104	111	0	0.9699	96.99
32	57	1	0.1364	86.36
3	25	0	0.9987	99.87
35	59	1	0.2095	79.04
39	64	1	0.0352	96.48



a. Error porcentual en las salidas.



b. Ajuste de las salidas.

Figura 6.19. Validación de la red neuronal.

6.3.3.Cálculo de sobretensiones en líneas sin cable de tierra. Velocidad de retorno del rayo con distribución uniforme

6.3.3.1. Método de Rusck

El objetivo es desarrollar una red neuronal que permita calcular sobretensiones originadas por el rayo en líneas sin apantallar, y de acuerdo con el modelo de Rusck cuando se trata de descargas a tierra. Las pautas que se van a seguir en este entrenamiento son las mismas que se comentaron en la aplicación anterior. En este estudio se han escogido las siguientes variables de entrada y salida de la red

·Entradas (variables independientes): Intensidad máxima de la descarga (variable I), distancia perpendicular entre la línea y la descarga (variable y), velocidad de retorno del rayo (variable v), altura de la línea (variable h)

·Salidas (variables dependientes): Tipo de descarga, directa / indirecta (variable t), sobretensión que aparece en la línea (variable V).

La figura 6.20 muestra la geometría general para dos configuraciones de línea, horizontal y vertical. Sin embargo, con el método de Rusck solamente es necesario especificar una altura media para los conductores de fase como entrada de la red neuronal. Además, este conductor equivalente se encontraría situado en el eje del poste, es decir x = 0. Por tanto, en este estudio no se tendrá en cuenta el tipo de configuración de la línea, horizontal o vertical, en los datos de entrada de la red neuronal.

La red neuronal se ha entrenado con un total de 2500 patrones diferentes calculados de forma aleatoria. Cada patrón esta formado por un valor para cada entrada (I,y,v,h) y el correspondiente valor para cada salida (t,V). A continuación se comenta como se genera cada valor de las variables de entrada.

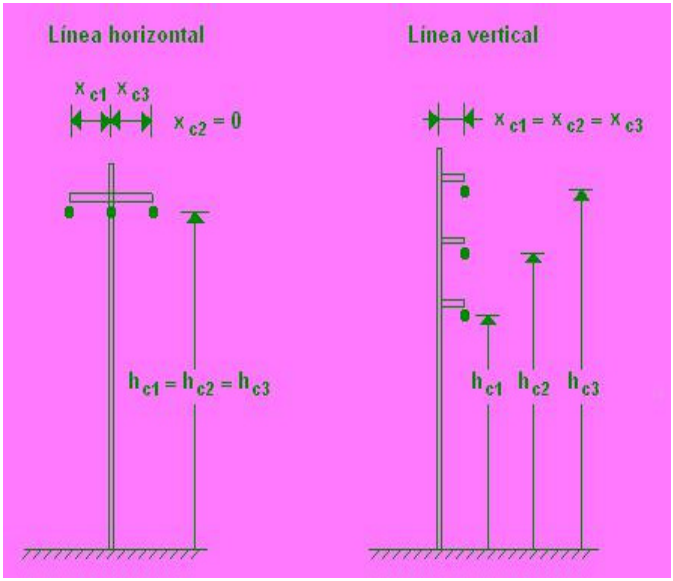


Figura 6.20. Geometría de la línea.

a)Intensidad de la descarga

Los valores de la intensidad de la descarga se generan siguiendo el procedimiento visto en el apartado 5.3.1 del capítulo 5.

b)Distancia perpendicular entre la línea y la descarga

El valor de esta distancia se genera de forma aleatoria, entre 0 m y 500 m, considerando una distribución uniforme.

c)Velocidad de retorno del rayo

El valor de la velocidad de retorno del rayo se genera de forma aleatoria, entre 30000 km/s y 150000 km/s, suponiendo una distribución uniforme.

d)Altura de la línea

Se han utilizado los valores de 5 m, 8 m, 11 m, 14 m y 17 m, en el entrenamiento de la red neuronal. Con los dos valores extremos, 5 m y 17 m, se cubre la totalidad de líneas de distribución.

e)Tipo de descarga

Se utiliza la expresión (4.2) para saber si una descarga, caracterizada por los parámetros que componen la entrada de la red neuronal (I,y,v,h), es directa (t = 1) o indirecta (t = 0).

f)Sobretensión que aparece en la línea

Se utiliza la expresión (4.6) para calcular la tensión que aparece en la línea si la descarga es directa a los conductores de fase, y la expresión (4.20) si la descarga es directa a tierra.

Entrenamiento

La Tabla 6.5 muestra una parte de los datos utilizados en el entrenamiento. El proceso seguido se detalla a continuación.

1)Se fijan los siguientes parámetros

·1 capa de neuronas ocultas

·4 neuronas en la capa oculta

·Funciones de transferencia: tan-sigmoidal (capas ocultas), lineal (capa de salida)

·Tasa de aprendizaje variable, $\alpha 0=0.25, \alpha i=1.01, \alpha d=0.6$

·Momento $\beta=0.9$.

Una diferencia muy importante respecto a la aplicación estudiada en el apartado anterior son las funciones de transferencia utilizadas con esta red neuronal. Puesto que la salida más importante de la red neuronal es la que calcula las tensiones, se ha escogido como función de transferencia global la denominada función de aproximación (función tan-sigmoidal en las diferentes capas ocultas y función lineal en la capa de salida). Se ha demostrado que esta arquitectura de red neuronal es capaz de aproximar cualquier función con un número finito de discontinuidades y precisión arbitraria. Cuanto más compleja sea la función más neuronas serán necesarias para poder aproximarla.

Tabla 6.5. Datos del entrenamiento.

ENTRADAS				SALIDAS	
Altura (m)	**Intensidad (kA)**	**Distancia descarga - línea (m)**	**Velocidad (km/s)**	**Tipo descarga**	**Tensión (kV)**
5	1	0	30000	1	207.23
5	1	0	150000	1	207.23
5	1	500	30000	0	0.32
5	1	500	150000	0	0.41
5	200	0	30000	1	41446.53
8	63	300	141534	0	68.51
8	56	474	134736	0	37.96
8	19	67	77057	0	81.18
8	53	203	78053	0	74.01
8	34	335	149545	0	33.26
8	152	214	68439	0	198.40
11	139	242	124134	0	247.97
11	41	416	139549	0	43.90
11	44	480	59657	0	34.82
11	78	145	60652	0	201.87
11	41	277	132145	0	64.45
11	58	156	51039	0	136.95
11	30	489	131538	0	26.85
11	34	240	34022	0	50.38
11	27	376	82476	0	28.01
11	59	33	127945	1	13596.17
11	6	61	66066	0	39.88
11	10	471	37615	0	7.26
11	92	55	142365	1	21228.74
11	35	494	86894	0	27.93
11	29	95	145569	0	137.88
11	38	90	40431	0	153.91
11	33	69	132751	0	207.96
11	14	205	86288	0	27.15
11	17	179	110321	0	40.86
11	40	400	56453	0	37.70
11	61	49	147171	1	14033.48
11	33	446	61259	0	27.89
11	18	69	130003	0	114.46
11	10	222	32269	0	16.57
14	20	175	122137	0	63.23
14	59	328	56836	0	85.53
14	46	307	43630	0	69.79
14	21	207	93298	0	52.57

a)Estudio previo. Tanto éste como los próximos estudios se realizarán con 1000 iteraciones. Por ahora, el objetivo es conseguir unas buenas condiciones iniciales para el estudio del intervalo de las entradas netas y el de la pendiente de las funciones de transferencia. Se guardan los pesos y los umbrales que dan el mínimo error. Se comprueba que en este punto todavía existe un error considerable en el cálculo de las salidas.

b)Ajuste del intervalo de las entradas netas de las funciones de transferencia. A diferencia del ejemplo anterior, en este caso se concluye que aumentando el intervalo de dichas entradas no se modifica en absoluto el mínimo alcanzado.

c)Ajuste de la pendiente de la función de transferencia. Se varían por separado las pendientes de las funciones de transferencia. En este caso, una variación en el valor de dichas pendientes disminuye notablemente el mínimo conseguido. Se mantienen como valores óptimos un valor unidad para la pendiente de la función tan-sigmoidal, y un valor de 0.1 para la función lineal. Se observa claramente que la capa de neuronas tan-sigmoidales es la que actúa como clasificador, y la capa de neuronas lineales es la que termina adaptando las tensiones.

2)Variación del número de neuronas de la primera capa oculta. Se han realizado 2 estudios con cada estructura diferente, uno a 1000 iteraciones para buscar unas buenas condiciones iniciales, y otro más largo, a 10000 iteraciones, para encontrar un buen mínimo. En este caso, se ha aumentado hasta 10 el número de neuronas y los resultados apenas han mejorado.

3)Variación del número de neuronas de la segunda capa oculta. Se añade una capa más de neuronas ocultas y se mantiene constante en 4 el número de neuronas de la primera capa. Se comienzan a conseguir mejores resultados con el aumento del número de neuronas de esta segunda capa. A la vista de los resultados conseguidos se prefiere aumentar el número de neuronas de la primera capa oculta para así poder aumentar el número de neuronas de la segunda capa. Se prueban diferentes estructuras, la que mejor se adapta tiene 14 neuronas en la primera capa oculta y 12 neuronas en la segunda capa oculta.

4)Del estudio anterior se concluye que la estructura de red neuronal ajustada es la siguiente • primera capa oculta: 14 neuronas

• Segunda capa oculta: 12 neuronas

• Funciones de transferencia: tan-sigmoidal (capas ocultas), lineal (capa de salida).

A continuación se intentarán ajustar los parámetros propios de la fase del aprendizaje tales como el momento y la tasa de aprendizaje. Tal como se comentó durante el entrenamiento del ejemplo, con estos ajustes el error disminuirá poco.

a)Tasa de aprendizaje. Se comprueba que una tasa de aprendizaje adaptativa va mejor que una constante, y que los parámetros actuales son aceptables.

b)Ajuste del momento. Se comprueba que el valor actual (0.9) es aceptable.

5)La estructura final de la red con todos sus parámetros ajustados es la siguiente, ver figura 6.21

172 Aplicación de redes neuronales en el cálculo de sobretensiones y tasa de contorneamientos

·2 capas de neuronas ocultas, con 14 y 12 neuronas respectivamente

·Funciones de transferencia: tan-sigmoidal (capas ocultas), lineal (capa de salida)

·Tasa de aprendizaje variable, $\alpha 0=0.25, \alpha i=1.01, \alpha d=0.6$

·Momento: β=0.9.

Los resultados del entrenamiento con esta red se muestran en la figura 6.22. Los gráficos se han calculado con las salidas normalizadas porque muestran de manera más clara el error cometido por la red neuronal en el entrenamiento. Debido a la enorme diferencia entre los valores extremos de las tensiones, esta red neuronal ha funcionado mucho mejor con datos normalizados. Se puede observar que el ajuste es muy bueno. Por un lado, el error cometido en los valores calculados de las salidas que actúan como clasificador no supera en ningún caso el 1 %. En el cálculo de las tensiones, el error cometido no supera el 2 %, estando en la mayoría de los casos por debajo del 1 %.

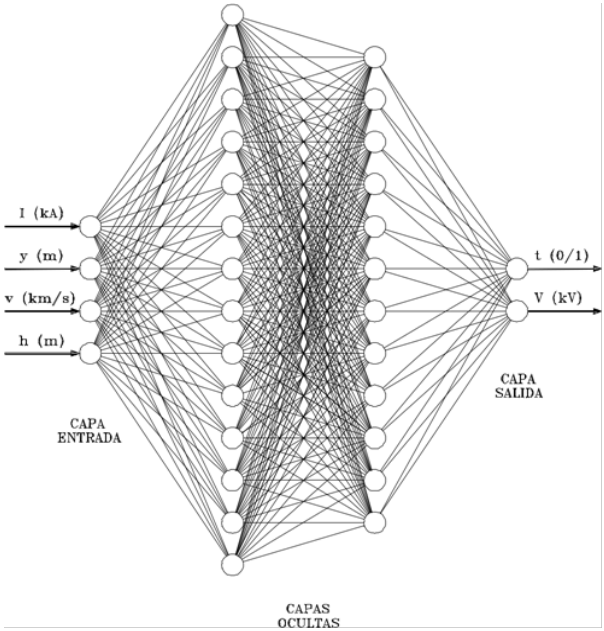
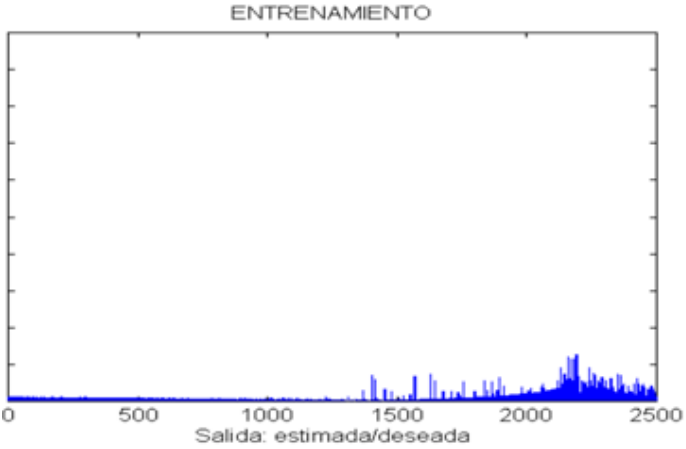


Figura 6.21. Arquitectura de la red neuronal (Método de Rusck con líneas sin apantallar).

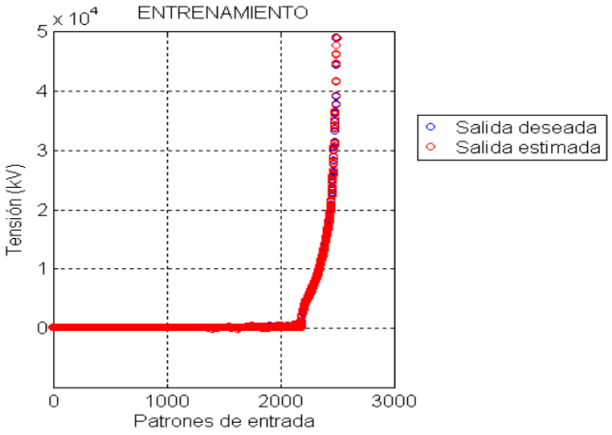
Validación

La validación de esta red neuronal se realiza con 6000 patrones que no han intervenido en el proceso de entrenamiento. Se han utilizado 1000 patrones para cada altura de la línea (5 m, 8 m, 10 m, 11 m, 14 m y17 m). La altura de 10 m no ha sido utilizada durante el entrenamiento de la red neuronal.

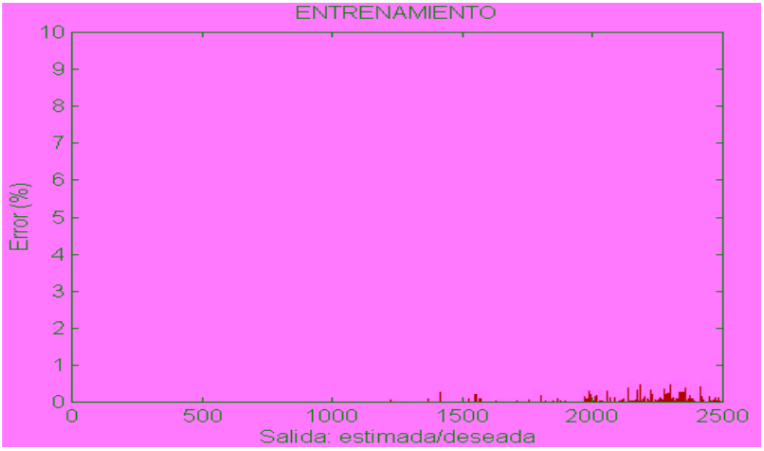
Los gráficos de las figuras 6.23, 6.24, y 6.25 muestran para tres alturas diferentes el error cometido en el cálculo de ambas salidas y el ajuste de las salidas con los valores de los pesos y umbrales óptimos conseguidos en el entrenamiento de la red neuronal. Se puede observar que en más del 99 % de los casos el error cometido en el cálculo de tensiones se encuentra por debajo del 2 %. En unos pocos casos el error puede llegar a alcanzar el 10 %, y esto ocurre cuando la red neuronal muestra confusión en la clasificación de descargas.



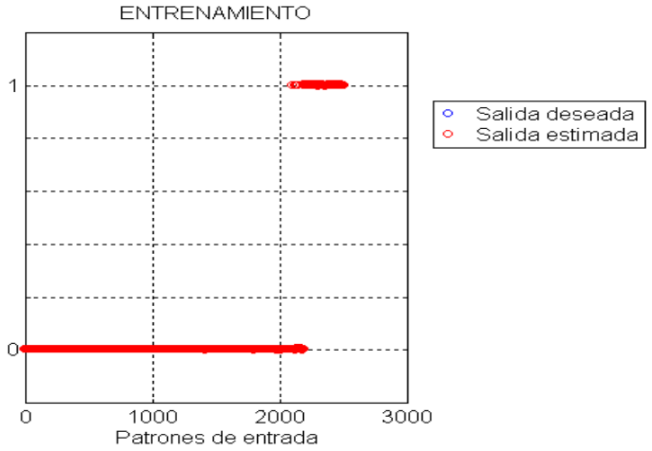
a. Error porcentual en el cálculo de sobretensiones.



b. Ajuste de las salidas (tensiones).

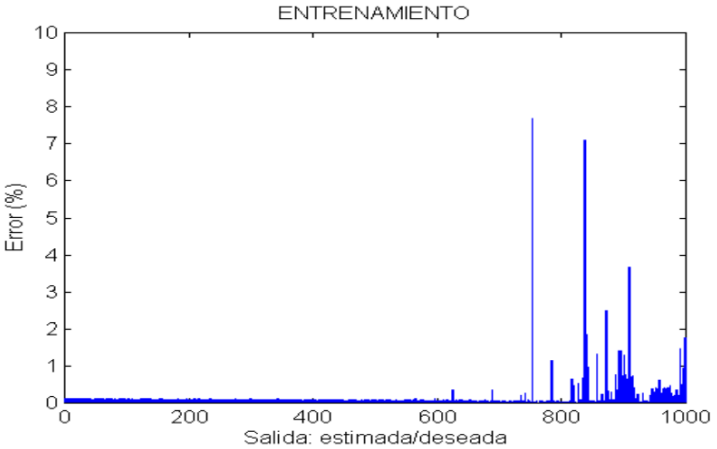


c. Error porcentual en la clasificación de descargas.

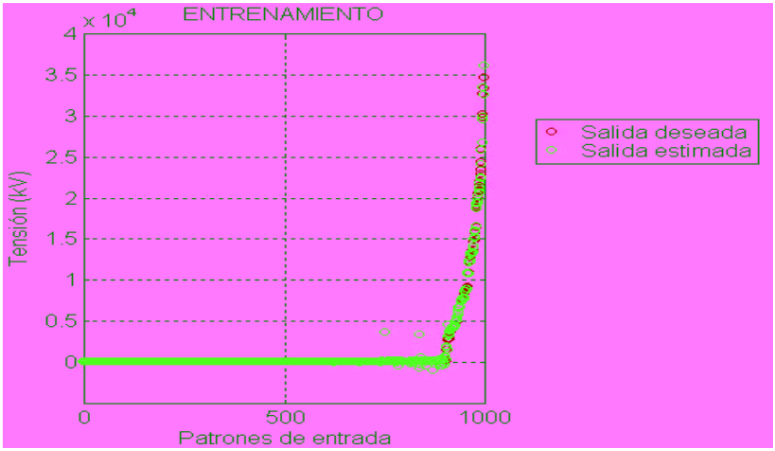


d. Ajuste del clasificador.

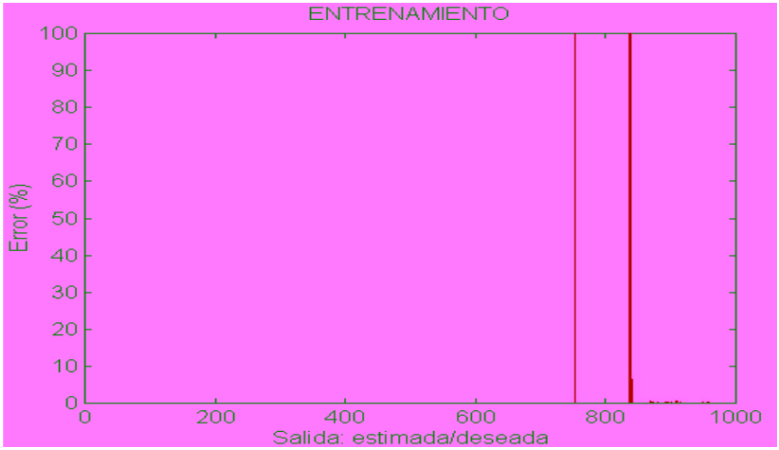
Figura 6.22. Entrenamiento de la red neuronal.



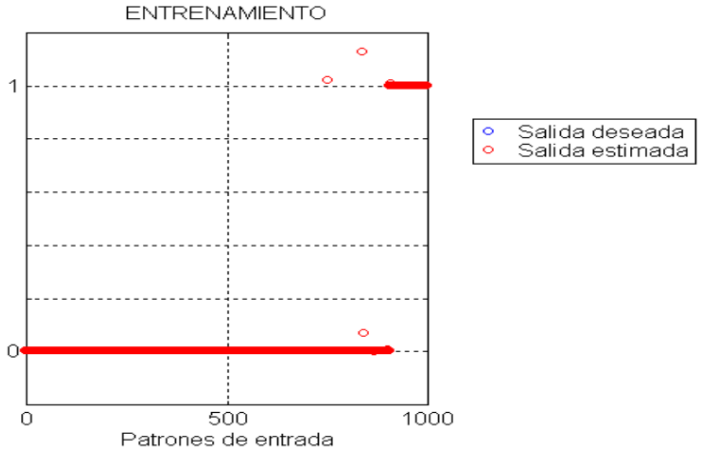
a. Error porcentual en el cálculo de sobretensiones.



b. Ajuste de las salidas (tensiones).

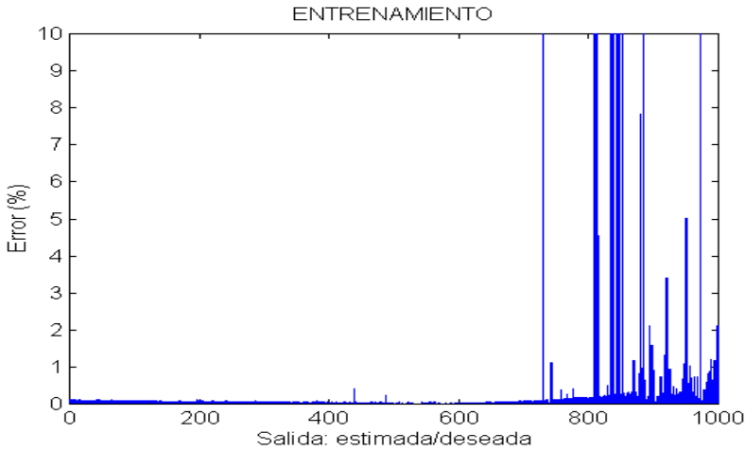


c. Error porcentual en la clasificación de descargas.

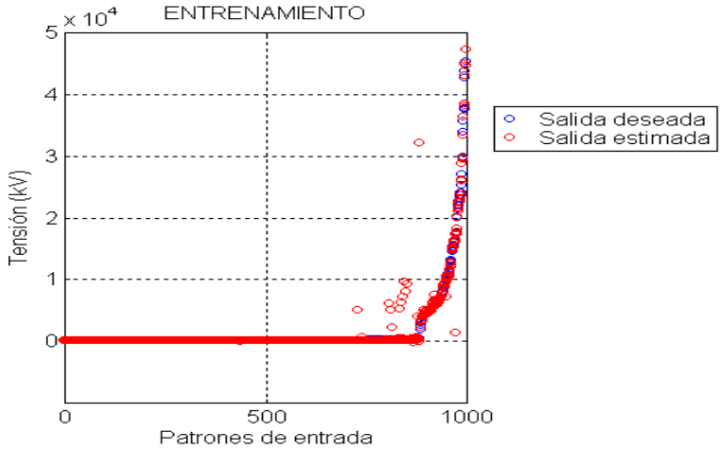


d. Ajuste del clasificador.

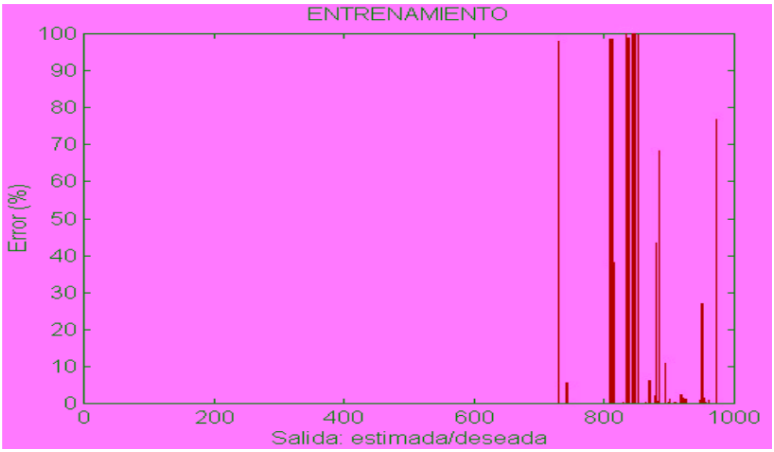
Figura 6.23. Validación de la red neuronal. Altura = 5 m.



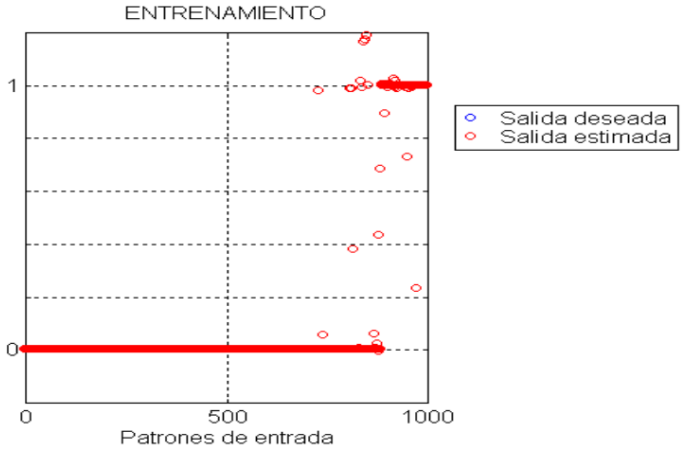
a. Error porcentual en el cálculo de sobretensiones.



b. Ajuste de las salidas (tensiones).

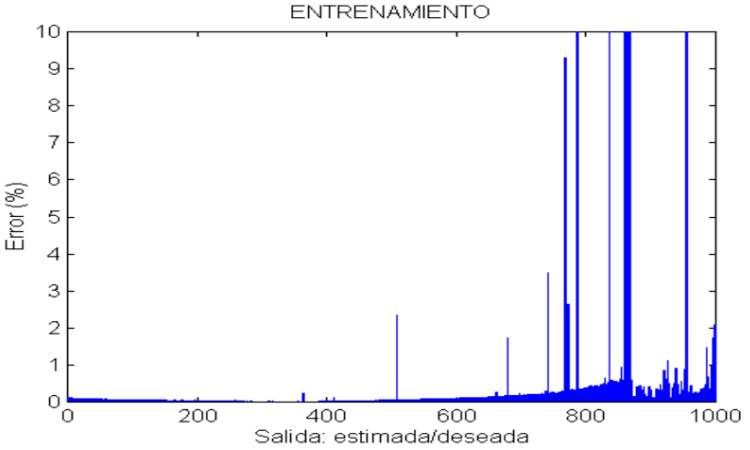


c. Error porcentual en la clasificación de descargas.



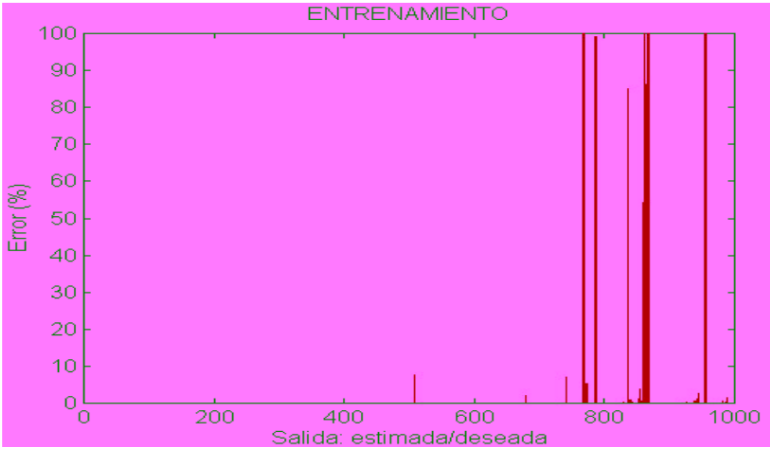
d. Ajuste del clasificador.

Figura 6.24. Validación de la red neuronal. Altura = 10 m.

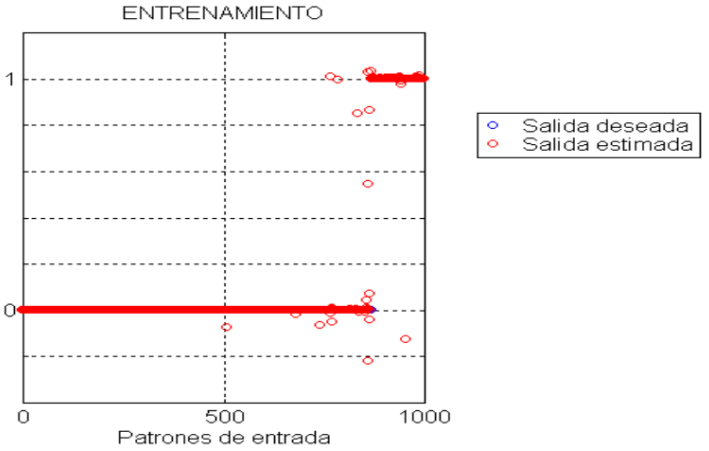


a. Error porcentual en el cálculo de sobretensiones.

b. Ajuste de las salidas (tensiones).



c. Error porcentual en la clasificación de descargas.



d. Ajuste del clasificador.

Figura 6.25. Validación de la red neuronal. Altura = 17 m.

6.3.3.2. Método de Chowdhuri

En este estudio se han escogido las siguientes variables de entrada y salida de la red

·Entradas: Intensidad máxima de la descarga (variable I), tiempo de frente de la onda de corriente del rayo (variable tf), distancia perpendicular entre la línea y la descarga (variable y), velocidad de retorno del rayo (variable v), altura de cada conductor de fase (variable hc1, variable hc2, variable hc3), distancia al eje de cada conductor de fase (variable xc1, variable xc2, variable xc3)

·Salidas: Tipo de descarga, directa / indirecta (variable t), sobretensión que aparece en la línea (variable V).

La red neuronal se ha entrenado con un total de 2800 patrones diferentes calculados de forma aleatoria. Cada patrón esta formado por un valor

para cada entrada (I,tf,y,v,hc1,hc2,hc3,xc1,xc2,xc3) y el correspondiente valor para cada salida (t,V). A continuación se comenta como se genera cada valor de las variables anteriores.

a)Intensidad y tiempo de frente

Los valores de la intensidad y del tiempo de frente de la onda de corriente del rayo se generan siguiendo el procedimiento visto en el apartado 5.3.1 del capítulo 5.

b)Distancia perpendicular entre la línea y la descarga

El valor de esta distancia se genera de forma aleatoria, entre 0 m y 500 m, utilizando una distribución uniforme.

c)Velocidad de retorno del rayo

El valor de la velocidad de retorno del rayo se genera de forma aleatoria, entre 30000 km/s y 150000 km/s, utilizando una distribución uniforme.

d)Geometría de la línea

Se han utilizado 5 configuraciones diferentes para las líneas de tipo horizontal. La altura en cada configuración ha sido de 5 m, 8 m, 11 m, 14 m y 17 m. La figura 6.20 muestra los tipos de línea utilizados en el entrenamiento.

Se han utilizado 4 configuraciones diferentes para las líneas de tipo vertical (con tres alturas diferentes para cada conductor de fase): 5 m, 7 m, 9 m (configuración 1), 8 m, 10 m, 12 m (configuración 2), 11 m, 13 m, 15 m (configuración 3), 14 m, 16 m, 18 m (configuración 4). Para cada configuración de línea horizontal se han utilizado dos tipos de distancias al eje diferentes por parte de cada conductor: -1 m, 0 m, 1 m (distancia entre conductores más externos = 2 m), -3 m, 0 m, 3 m (distancia entre conductores más externos = 6 m). En líneas verticales se ha supuesto que los conductores se encuentran colocados en el eje de la línea, es decir, distancia al eje = 0.

La Tabla 6.6 resume las combinaciones adoptadas en el entrenamiento

de la red neuronal para las variables altura y distancia al eje.

Tabla 6.6. Variables altura y distancia al eje.

	h_{c1} (m)	h_{c2} (m)	h_{c3} (m)	x_{c1} (m)	x_{c2} (m)	x_{c3} (m)
Línea horizontal	5	5	5	-1	0	1
	5	5	5	-3	0	3
	8	8	8	-1	0	1
	8	8	8	-3	0	3
	11	11	11	-1	0	1
	11	11	11	-3	0	3
	14	14	14	-1	0	1
	14	14	14	-3	0	3
	17	17	17	-1	0	1
	17	17	17	0	0	0
Línea vertical	5	7	9	0	0	0
	8	10	12	0	0	0
	11	13	15	0	0	0
	14	16	18	0	0	0

e)Tipo de descarga

Se utiliza la expresión (4.2) para saber si una descarga, caracterizada por los parámetros que componen la entrada de la red neuronal (I,tf,y,v,hc1,hc2,hc3,xc1,xc2,xc3), es directa (t = 1) o indirecta (t = 0).

f)Sobretensión que aparece en la línea

Se utiliza la expresión (4.6) para calcular la tensión que aparece en la línea si la descarga es directa a los conductores de fase, y la expresión (4.46) si la descarga es directa a tierra.

Entrenamiento

El objetivo no es repetir todos los pasos que se realizaron en el entrena-

miento de la red neuronal del apartado anterior (método de Rusck), sino verificar que la estructura final obtenida en aquel estudio es también adecuada en este estudio. La Tabla 6.7 muestra una parte de los datos utilizados en el entrenamiento.

La estructura de red neuronal utilizada ha sido la siguiente

·2 capas de neuronas ocultas, con 14 y 12 neuronas respectivamente

·Funciones de transferencia: tan-sigmoidal (capas ocultas), lineal (capa de salida)

·Tasa de aprendizaje variable, $\alpha 0=0.25$, $\alpha i=1.01$, $\alpha d=0.6$

·Momento $\beta=0.9$.

Tabla 6.7. Datos del entrenamiento.

ENTRADAS										SALIDAS	
Altura conductor 1 (m)	Altura conductor 2 (m)	Altura conductor 3 (m)	Distancia al eje conductor 1 (m)	Distancia al eje conductor 2 (m)	Distancia al eje conductor 3 (m)	Intensidad (kA)	Tiempo de frente (μs)	Distancia descarga - línea (m)	Velocidad (km/s)	Tipo descarga	Tensión (kV)
5	5	5	-1	0	1	84	3.0	377	144526	0	34.69
5	5	5	-1	0	1	53	4.0	328	48847	0	115.21
5	5	5	-1	0	1	22	4.0	94	106585	0	23.07
8	8	8	-3	0	3	26	5.5	146	103137	0	29.43
8	8	8	-3	0	3	26	3.0	467	130097	0	19.69
8	8	8	-3	0	3	35	3.0	81	103224	0	87.67
11	11	11	-1	0	1	56	5.5	148	92466	0	104.59
11	11	11	-1	0	1	14	4.0	81	128948	0	24.54
11	11	11	-1	0	1	38	6.0	1	40498	1	9433.73
11	11	11	-1	0	1	32	6.5	66	132818	0	34.19
14	14	14	-1	0	1	30	4.0	330	148007	0	26.25
14	14	14	-1	0	1	36	2.0	417	70709	0	203.89
14	14	14	-1	0	1	7	3.5	426	71680	0	25.14
14	14	14	-1	0	1	21	8.0	112	94621	0	36.44
17	17	17	-3	0	3	12	4.0	319	78217	0	46.82
17	17	17	-3	0	3	47	6.0	316	78665	0	120.53
17	17	17	-3	0	3	46	6.0	206	110840	0	79.47
5	7	9	0	0	0	19	2.5	185	81739	0	65.51
5	7	9	0	0	0	31	5.0	288	116289	0	23.95
5	7	9	0	0	0	26	3.5	166	50056	0	131.66
8	10	12	0	0	0	10	2.5	20	35702	1	2541.86
8	10	12	0	0	0	13	1.5	327	98577	0	52.52
8	10	12	0	0	0	30	3.0	453	66939	0	101.95
8	10	12	0	0	0	30	7.0	282	136151	0	16.40
11	13	15	0	0	0	25	2.5	215	84226	0	131.32
11	13	15	0	0	0	18	7.0	313	91809	0	24.96
11	13	15	0	0	0	71	1.5	477	119058	0	192.61
11	13	15	0	0	0	61	2.5	145	37554	0	1091.42

El entrenamiento de esta red neuronal queda reducido a los siguientes pasos

1)Se realiza un estudio previo con el objetivo de conseguir unas buenas condiciones iniciales para el estudio posterior de la pendiente de las funciones de transferencia. Se guardan los pesos y los umbrales que dan el mínimo error. Se comprueba que en general existe un error considerable en el cálculo de las salidas.

2)Ajuste de la pendiente de las funciones de transferencia. Se varían por separado las pendientes de las funciones de transferencia. Una variación en el valor de dichas pendientes disminuye notablemente el error conseguido. Se mantiene un valor unidad para la pendiente de la función tan-sigmoidal, y un valor de 0.1 para la función lineal.

3)Se entrena la red neuronal con la estructura comentada anteriormente y con las pendientes ajustadas, y se comprueba que esta estructura entrena correctamente, siendo los errores de cálculo de las salidas muy pequeños. La figura 6.26 muestra la arquitectura de la red, mientras que la figura 6.27 muestra los resultados del entrenamiento obtenidos con esta red. Se puede observar que el ajuste es similar al que se obtuvo con el método de Rusck. Por un lado, el error cometido en los valores calculados de las salidas que actúan como clasificador no supera en ningún caso el 1 %. En el cálculo de las tensiones, en más de un 90 % de los datos el error cometido no supera el 1 %.

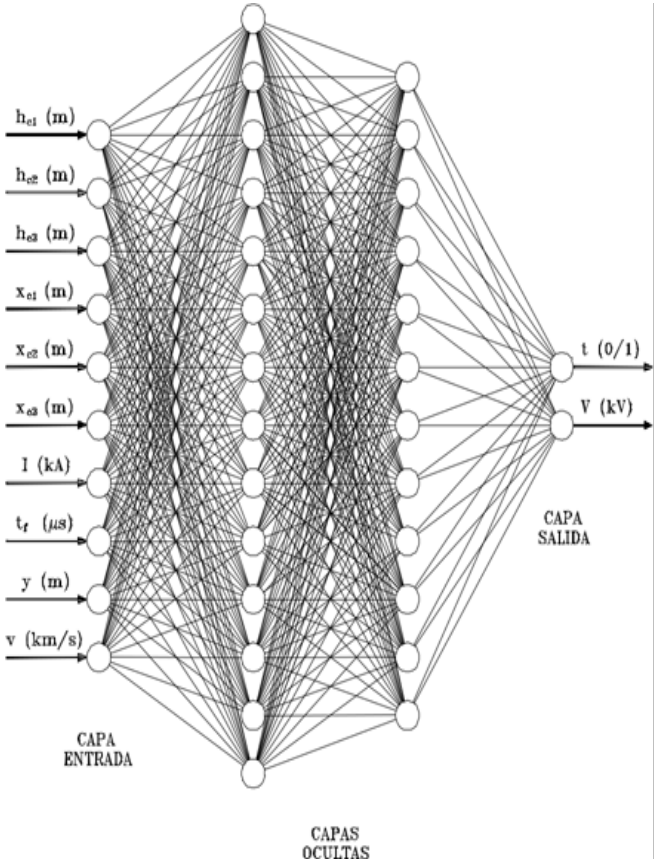


Figura 6.26. Arquitectura de la red neuronal (Método de Chowdhuri con líneas sin apantallar).

Validación

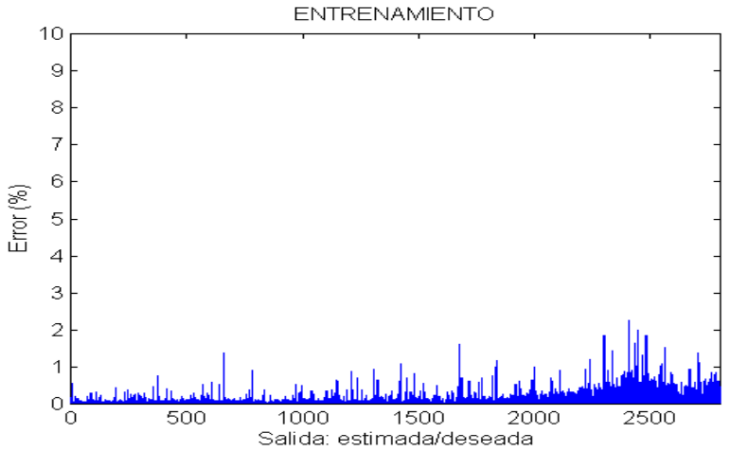
La validación de esta red neuronal se ha realizado con 6000 patrones que no han intervenido en el proceso de entrenamiento. Se han utilizado 1000 patrones para cada una de las configuraciones de línea mostradas en la Tabla 6.8.

Los gráficos de las figuras 6.28, 6.29, y 6.30 muestran los resultados obtenidos para 3 de las configuraciones anteriores. Los resultados obtenidos

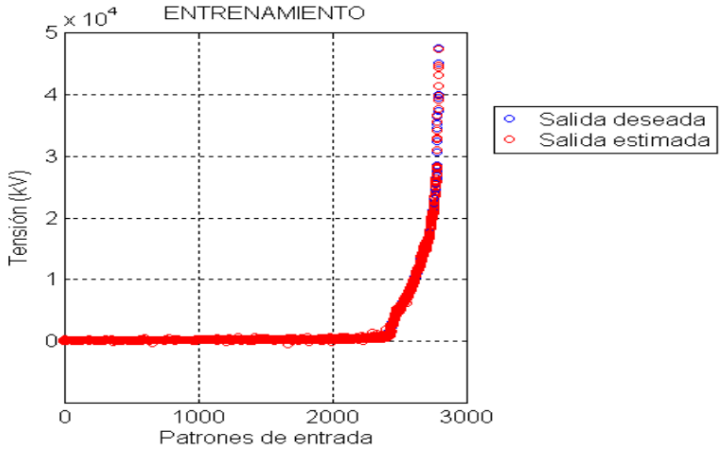
con las otras configuraciones son similares. Las conclusiones de este estudio son similares a las que se obtuvieron con el método de Rusck. En general el error cometido en el cálculo de tensiones se encuentra por debajo del 2 %, aunque cuando la red neuronal presenta confusión en la clasificación de descargas el error es mayor.

Tabla 6.8. Variables altura y distancia al eje de la línea.

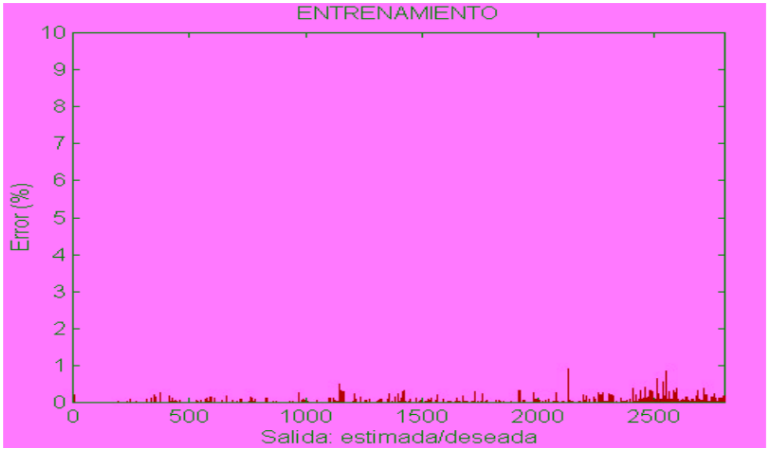
	h_{c1} (m)	h_{c2} (m)	h_{c3} (m)	x_{c1} (m)	x_{c2} (m)	x_{c3} (m)
Línea horizontal	5	5	5	-1.5	0	1.5
	10	10	10	-2	0	2
	17	17	17	-3	0	3
Línea vertical	5	7	9	0	0	0
	10	12	14	0	0	0
	14	16	18	0	0	0



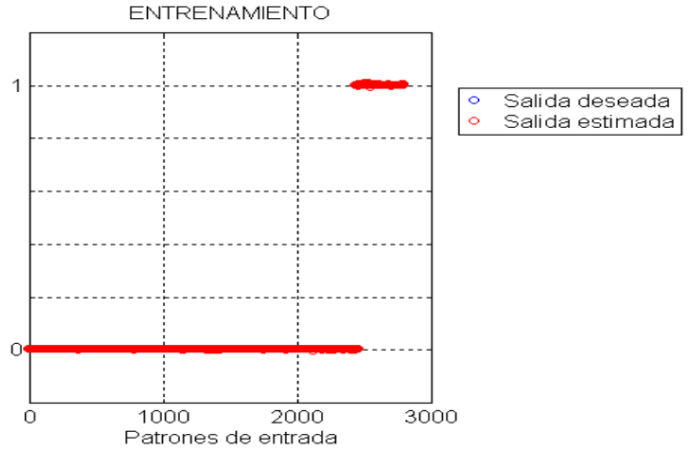
a. Error porcentual en el cálculo de sobretensiones.



b. Ajuste de las salidas (tensiones).

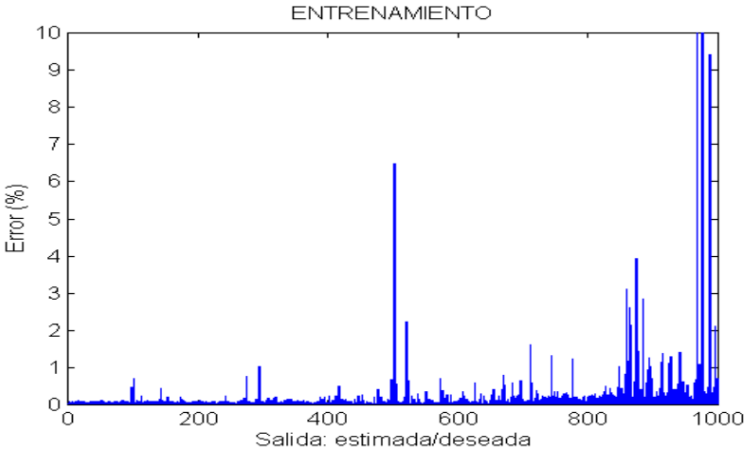


c. Error porcentual en la clasificación de descargas.



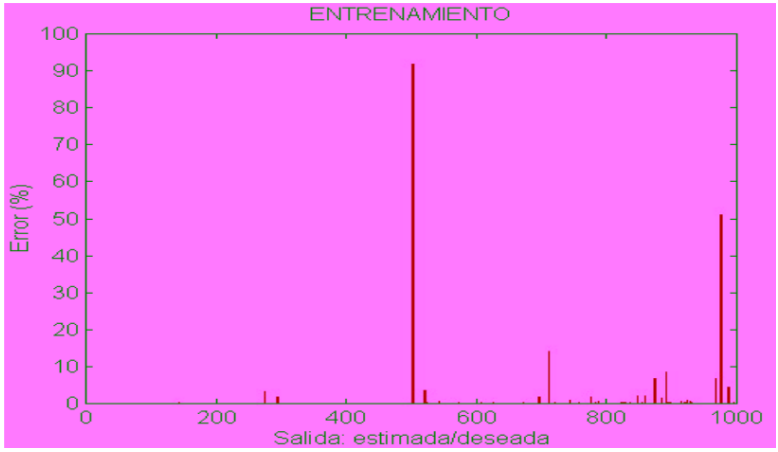
d. Ajuste del clasificador.

Figura 6.27. Entrenamiento de la red neuronal.



a. Error porcentual en el cálculo de sobretensiones.

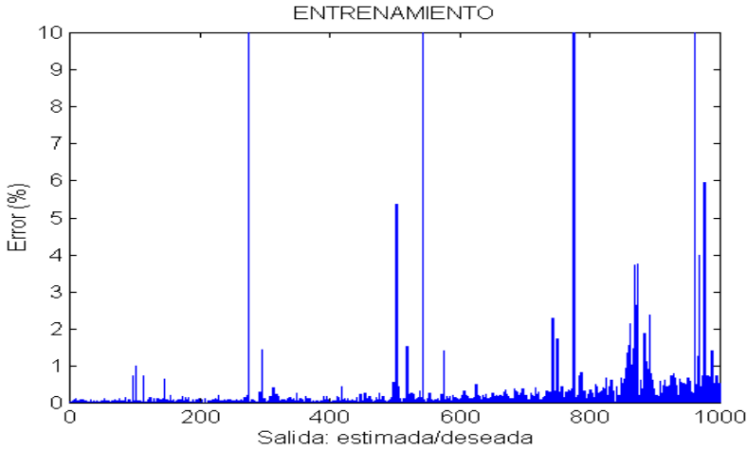
b. Ajuste de las salidas (tensiones).



c. Error porcentual en la clasificación de descargas.

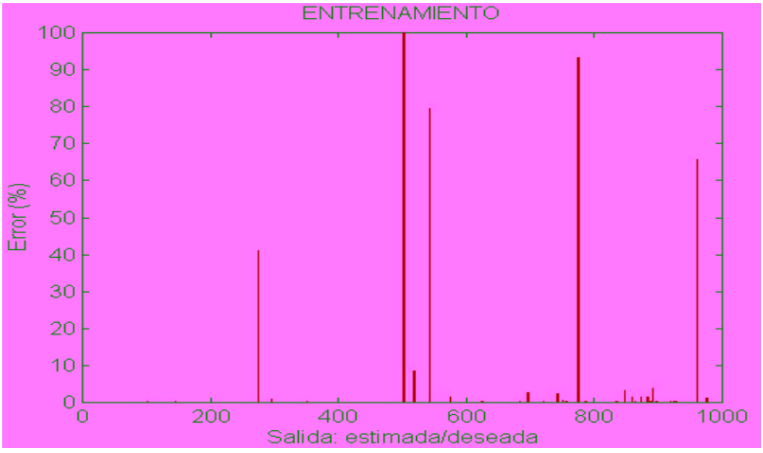
d. Ajuste del clasificador.

Figura 6.28. Validación de la red neuronal. Línea horizontal, altura = 10 m, distancia entre conductores más externos = 4 m.

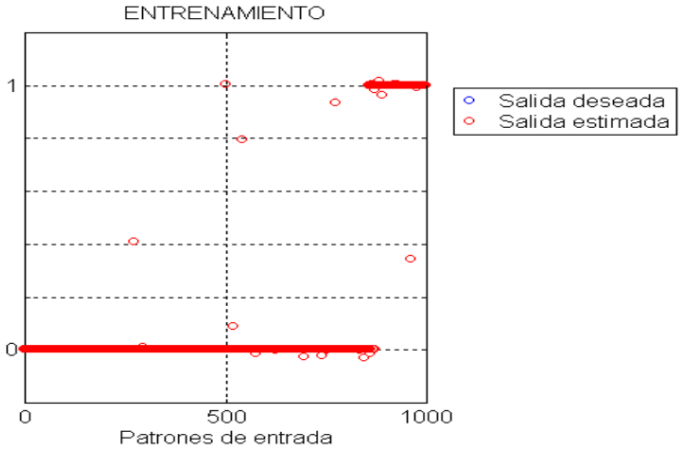


a. Error porcentual en el cálculo de sobretensiones.

b. Ajuste de las salidas (tensiones).

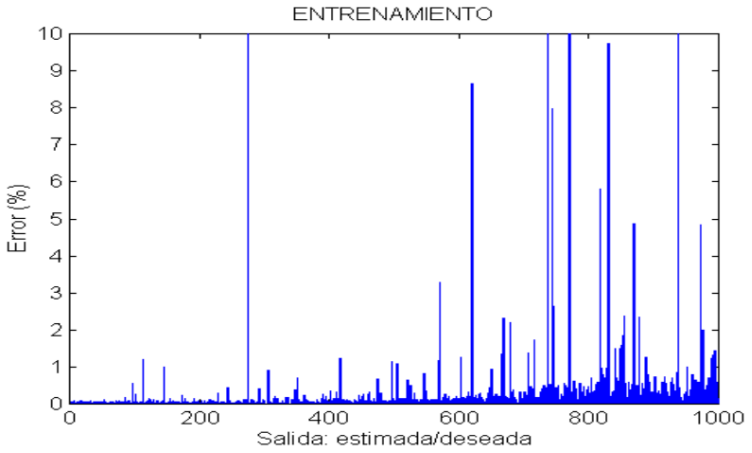


c. Error porcentual en la clasificación de descargas.



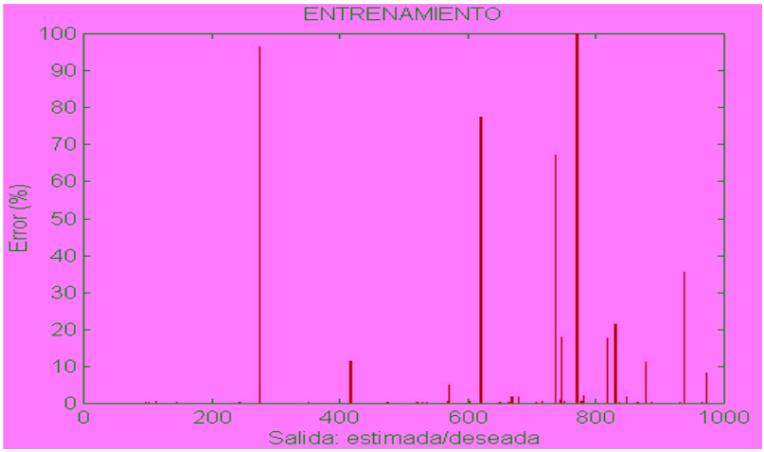
d. Ajuste del clasificador.

Figura 6.29. Validación de la red neuronal. Línea vertical, altura c1 =10 m, altura c2 =12 m, altura c3 =14 m.

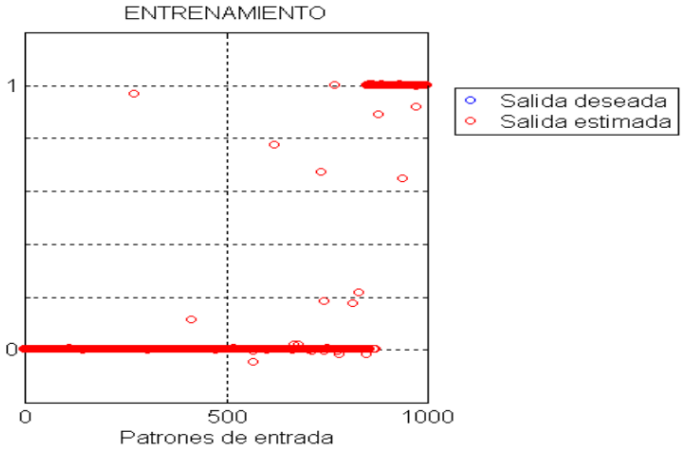


a. Error porcentual en el cálculo de sobretensiones.

b. Ajuste de las salidas (tensiones).



c. Error porcentual en la clasificación de descargas.



d. Ajuste del clasificador.

Figura 6.30. Validación de la red neuronal. Línea vertical, altura c1 =14 m, altura c2 =16 m, altura c3 =18 m

6.3.4.Cálculo de sobretensiones en líneas con cable de tierra. Velocidad de retorno del rayo con distribución uniforme

6.3.4.1. Método de Rusck

En este estudio se han escogido las siguientes variables de entrada y salida de la red

·Entradas: Intensidad máxima de la descarga (variable I), distancia perpendicular entre la línea y la descarga (variable y), velocidad de retorno del rayo (variable v), altura de cada conductor de fase (variable hc1, variable hc2, variable hc3), altura del cable de tierra (variable hct), distancia al eje de cada conductor de fase (variable xc1, variable xc2, variable xc3), resistencia de puesta a tierra de los postes (variable R)

·Salidas: Tipo de descarga, directa al cable de tierra/directa al conductor de fase/indirecta (variable t), sobretensión que aparece en la línea (variable V).

La red neuronal se ha entrenado con un total de 8640 patrones diferentes calculados de forma aleatoria. Cada patrón esta formado por un valor para cada entrada (I,y,v,hc1,hc2,hc3, hct,xc1,xc2,xc3,R) y el correspondiente valor para cada salida (t,V). A continuación se comenta como se genera cada valor de las variables anteriores.

a)Intensidad

La intensidad de la descarga se calcula siguiendo el procedimiento visto en el apartado 5.3.1 del capítulo 5.

b)Distancia perpendicular entre la línea y la descarga

El valor de esta distancia se genera de forma aleatoria, entre 0 m y 500 m, considerando una distribución uniforme.

c)Velocidad de retorno del rayo

El valor de la velocidad de retorno del rayo se genera de forma aleatoria, entre 30000 km/s y 150000 km/s, suponiendo una distribución uniforme.

d)Geometría de la línea

Tanto para las líneas de tipo horizontal como las de tipo vertical se han utilizado las configuraciones comentadas en el apartado 6.3.3, ver figura 6.31. La única diferencia es que la red neuronal se ha entrenado con dos valores diferentes de distancias al eje en el caso de la línea vertical, 0.5 m y 1.5 m, respectivamente, en lugar de estar los conductores colocados sobre el eje de la línea. Esto ha sido así porque en una línea vertical la distancia al eje de los conductores es la distancia horizontal entre conductores y cable de tierra.

El cable de tierra se encuentra situado sobre el eje de la línea, esto es xct = 0, y su altura dependerá de la altura de los conductores, aunque se han utilizado dos alturas diferentes del cable de tierra para cada altura diferente de la línea. Las Tablas 6.9 y 6.10 resumen las combinaciones adoptadas en el entrenamiento de la red neuronal.

Tabla 6.9. Variables altura de los conductores, altura del cable de tierra, distancia al eje de los conductores y resistencia de puesta a tierra. Línea horizontal.

h_{c1} (m)	h_{c2} (m)	h_{c3} (m)	x_{c1} (m)	x_{c2} (m)	x_{c3} (m)	h_{ct} (m)	R (Ω)
5	5	5	-1	0	1	6.5	10
5	5	5	-1	0	1	6.5	40
5	5	5	-1	0	1	6.5	70
5	5	5	-1	0	1	8.0	10
5	5	5	-1	0	1	8.0	40
5	5	5	-1	0	1	8.0	70
5	5	5	-3	0	3	6.5	10
5	5	5	-3	0	3	6.5	40
5	5	5	-3	0	3	6.5	70
5	5	5	-3	0	3	8.0	10
5	5	5	-3	0	3	8.0	40
5	5	5	-3	0	3	8.0	70
8	8	8	-1	0	1	9.5	10
8	8	8	-1	0	1	9.5	40
8	8	8	-1	0	1	9.5	70
8	8	8	-1	0	1	11.0	10
8	8	8	-1	0	1	11.0	40
8	8	8	-1	0	1	11.0	70
8	8	8	-3	0	3	9.5	10
8	8	8	-3	0	3	9.5	40
8	8	8	-3	0	3	9.5	70
8	8	8	-3	0	3	11.0	10
8	8	8	-3	0	3	11.0	40
8	8	8	-3	0	3	11.0	70
11	11	11	-1	0	1	12.5	10
11	11	11	-1	0	1	12.5	40
11	11	11	-1	0	1	12.5	70
11	11	11	-1	0	1	14.0	10
11	11	11	-1	0	1	14.0	40
11	11	11	-1	0	1	14.0	70
11	11	11	-3	0	3	12.5	10
11	11	11	-3	0	3	12.5	40
11	11	11	-3	0	3	12.5	70
11	11	11	-3	0	3	14.0	10
11	11	11	-3	0	3	14.0	40
11	11	11	-3	0	3	14.0	70
14	14	14	-1	0	1	15.5	10
14	14	14	-1	0	1	15.5	40
14	14	14	-1	0	1	15.5	70
14	14	14	-1	0	1	17.0	10
14	14	14	-1	0	1	17.0	40
14	14	14	-1	0	1	17.0	70
14	14	14	-3	0	3	15.5	10
14	14	14	-3	0	3	15.5	40
14	14	14	-3	0	3	15.5	70
14	14	14	-3	0	3	17.0	10
14	14	14	-3	0	3	17.0	40
14	14	14	-3	0	3	17.0	70
17	17	17	-1	0	1	18.5	10
17	17	17	-1	0	1	18.5	40
17	17	17	-1	0	1	18.5	70
17	17	17	-1	0	1	20.0	10
17	17	17	-1	0	1	20.0	40
17	17	17	-1	0	1	20.0	70
17	17	17	-3	0	3	18.5	10
17	17	17	-3	0	3	18.5	40
17	17	17	-3	0	3	18.5	70
17	17	17	-3	0	3	20.0	10
17	17	17	-3	0	3	20.0	40
17	17	17	-3	0	3	20.0	70

Tabla 6.10. Variables altura de los conductores, altura del cable de tierra, distancia al eje de los conductores y resistencia de puesta a tierra. Línea vertical.

h_{c1} (m)	h_{c2} (m)	h_{c3} (m)	x_{c1} (m)	x_{c2} (m)	x_{c3} (m)	h_{ct} (m)	R (Ω)
5	7	9	0.5	0.5	0.5	10.5	10
5	7	9	0.5	0.5	0.5	10.5	40
5	7	9	0.5	0.5	0.5	10.5	70
5	7	9	0.5	0.5	0.5	12.0	10
5	7	9	0.5	0.5	0.5	12.0	40
5	7	9	0.5	0.5	0.5	12.0	70
5	7	9	1.5	1.5	1.5	10.5	10
5	7	9	1.5	1.5	1.5	10.5	40
5	7	9	1.5	1.5	1.5	10.5	70
5	7	9	1.5	1.5	1.5	12.0	10
5	7	9	1.5	1.5	1.5	12.0	40
5	7	9	1.5	1.5	1.5	12.0	70
8	10	12	0.5	0.5	0.5	13.5	10
8	10	12	0.5	0.5	0.5	13.5	40
8	10	12	0.5	0.5	0.5	13.5	70
8	10	12	0.5	0.5	0.5	15.0	10
8	10	12	0.5	0.5	0.5	15.0	40
8	10	12	0.5	0.5	0.5	15.0	70
8	10	12	1.5	1.5	1.5	13.5	10
8	10	12	1.5	1.5	1.5	13.5	40
8	10	12	1.5	1.5	1.5	13.5	70
8	10	12	1.5	1.5	1.5	15.0	10
8	10	12	1.5	1.5	1.5	15.0	40
8	10	12	1.5	1.5	1.5	15.0	70
11	13	15	0.5	0.5	0.5	16.5	10
11	13	15	0.5	0.5	0.5	16.5	40
11	13	15	0.5	0.5	0.5	16.5	70
11	13	15	0.5	0.5	0.5	18.0	10
11	13	15	0.5	0.5	0.5	18.0	40
11	13	15	0.5	0.5	0.5	18.0	70
11	13	15	1.5	1.5	1.5	16.5	10
11	13	15	1.5	1.5	1.5	16.5	40
11	13	15	1.5	1.5	1.5	16.5	70
11	13	15	1.5	1.5	1.5	18.0	10
11	13	15	1.5	1.5	1.5	18.0	40
11	13	15	1.5	1.5	1.5	18.0	70
14	16	18	0.5	0.5	0.5	19.5	10
14	16	18	0.5	0.5	0.5	19.5	40
14	16	18	0.5	0.5	0.5	19.5	70
14	16	18	0.5	0.5	0.5	21.0	10
14	16	18	0.5	0.5	0.5	21.0	40
14	16	18	0.5	0.5	0.5	21.0	70
14	16	18	1.5	1.5	1.5	19.5	10
14	16	18	1.5	1.5	1.5	19.5	40
14	16	18	1.5	1.5	1.5	19.5	70
14	16	18	1.5	1.5	1.5	21.0	10
14	16	18	1.5	1.5	1.5	21.0	40
14	16	18	1.5	1.5	1.5	21.0	70

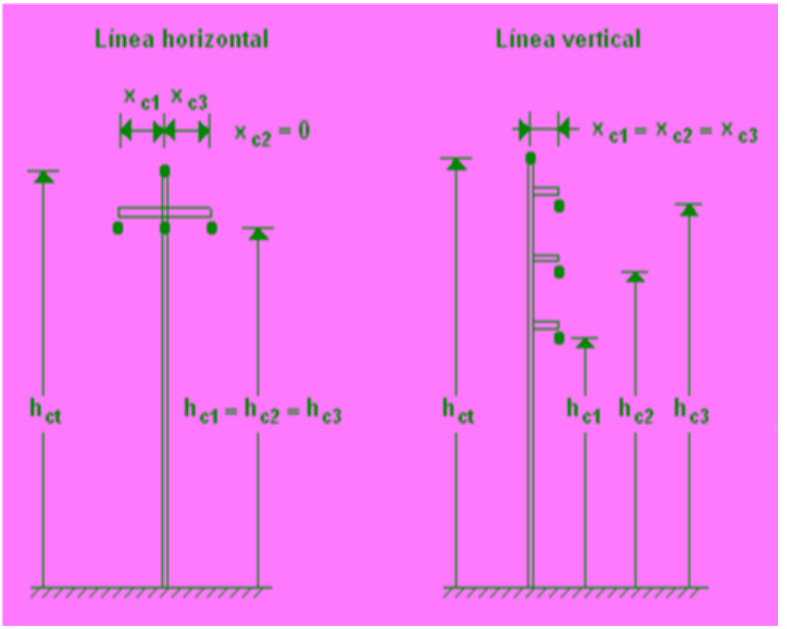


Figura 6.31. Geometría de la línea.

e)Resistencia de puesta a tierra

Se han utilizado tres valores constantes con cada configuración de línea diferente, 10 Ω, 40 Ω y 70 Ω, respectivamente, ver Tablas 6.9 y 6.10.

f)Tipo de descarga

Al igual que en los apartados anteriores, se utiliza el Modelo Electro-geométrico para saber si una descarga, caracterizada por los parámetros que componen la entrada de la red neuronal (I,y,v,hc1,hc2,hc3,hct,xc1,x-c2,xc3,R), es directa al cable de tierra (t = 2), directa a los conductores de fase (t = 1) o indirecta (t = 0).

g)Sobretensión que aparece en la línea

Se utiliza la expresión (4.17) para calcular la tensión que aparece en la

línea si la descarga es directa al cable de tierra, la expresión (4.6) si la descarga es directa a los conductores de fase, y la expresión (4.20) si la descarga es directa a tierra, esta última corregida por el factor de apantallamiento, ver expresión (4.26).

Entrenamiento

Para entrenar la red neuronal se seguirá la metodología utilizada en el apartado 6.3.3, método de Chowdhuri, verificándose de esta manera que la estructura final obtenida cuando la línea no tenía instalado un cable de tierra es también adecuada en este estudio. La Tabla 6.11 muestra una parte de los datos utilizados en el entrenamiento.

Tabla 6.11. Datos del entrenamiento.

ENTRADAS											SALIDAS	
Altura conductor 1 (m)	Altura conductor 2 (m)	Altura conductor 3 (m)	Altura cable de tierra (m)	Distancia al eje conductor 1 (m)	Distancia al eje conductor 2 (m)	Distancia al eje conductor 3 (m)	Resistencia puesta a tierra (Ω)	Intensidad (kA)	Distancia descarga-línea (m)	Velocidad (km/s)	Tipo descarga	Tensión (kV)
5	5	5	6.5	-1	0	1	10	38	27	67302	2	217.06
5	5	5	6.5	-1	0	1	10	46	176	82717	0	32.36
5	5	5	6.5	-1	0	1	10	50	484	105755	0	13.45
8	8	8	9.5	-1	0	1	10	42	144	57624	0	53.68
8	8	8	9.5	-1	0	1	10	70	50	102486	2	385.25
8	8	8	9.5	-1	0	1	10	52	114	44807	0	82.67
11	11	11	12.5	-1	0	1	70	22	419	79720	0	14.68
11	11	11	12.5	-1	0	1	70	54	312	39058	0	45.40
11	11	11	12.5	-1	0	1	70	29	65	58285	0	120.15
14	14	14	15.5	-1	0	1	70	15	269	117063	0	21.61
14	14	14	15.5	-1	0	1	70	6	486	65281	0	3.95
14	14	14	15.5	-1	0	1	70	18	392	110143	0	17.58
14	14	14	15.5	-1	0	1	70	25	456	52464	0	18.23
17	17	17	20.0	-3	0	3	40	23	320	110411	0	33.25
17	17	17	20.0	-3	0	3	40	33	359	82567	0	40.44
17	17	17	20.0	-3	0	3	40	28	325	105604	0	40.29
5	7	9	12.0	0.5	0.5	0.5	10	122	125	83891	0	229.72
5	7	9	12.0	0.5	0.5	0.5	10	24	262	44443	0	19.82
5	7	9	12.0	0.5	0.5	0.5	10	59	250	81682	0	54.80
8	10	12	13.5	1.5	1.5	1.5	40	31	162	69564	0	55.04
8	10	12	13.5	1.5	1.5	1.5	40	18	141	88791	0	38.93
8	10	12	13.5	1.5	1.5	1.5	40	36	145	115639	0	78.91
11	13	15	18.0	0.5	0.5	0.5	70	19	455	35518	0	15.01
11	13	15	18.0	0.5	0.5	0.5	70	29	348	131650	0	37.63
11	13	15	18.0	0.5	0.5	0.5	70	40	399	125241	0	44.07
14	16	18	21.0	0.5	0.5	0.5	40	95	137	87428	0	331.19
14	16	18	21.0	0.5	0.5	0.5	40	23	273	47980	0	36.85
14	16	18	21.0	0.5	0.5	0.5	40	55	261	85219	0	99.70

La estructura de red neuronal utilizada ha sido la siguiente

·2 capas de neuronas ocultas, con 14 y 12 neuronas respectivamente

·Funciones de transferencia: tan-sigmoidal (capas ocultas), lineal (capa de salida)

·Tasa de aprendizaje variable, $\alpha 0=0.25$, $\alpha i=1.01$, $\alpha d=0.6$

·Momento $\beta=0.9$.

El entrenamiento de esta red neuronal consta de los siguientes pasos

1)Estudio previo. Se guardan los pesos y los umbrales que dan el mínimo error.

2)Ajuste de la pendiente de las funciones de transferencia. Se mantiene un valor unidad para la pendiente de la función tan-sigmoidal, y un valor de 0.3 para la función lineal.

3)Se entrena la red neuronal con la estructura comentada anteriormente y con las pendientes ajustadas hasta que el error se estabiliza. La figura 6.32 muestra la arquitectura de la red, mientras que la figura 6.33 muestra los resultados del entrenamiento obtenidos con esta red. Se puede observar que el ajuste es similar al que se obtuvo en el apartado 6.3.3, con línea sin cable de tierra. Tanto en el clasificador como en el cálculo de las tensiones en más de un 90 % de los datos el error cometido no supera el 5 %.

Validación

La validación de esta red neuronal se ha realizado con 6000 patrones que no han intervenido en el proceso de entrenamiento. Se han utilizado 1000 patrones para cada configuración de línea, ver Tabla 6.12.

Los gráficos de las figuras 6.34, 6.35, y 6.36 muestran los resultados obtenidos para 3 de las configuraciones anteriores. Los resultados obtenidos con las otras configuraciones son similares. Se puede observar que en la

mayoría de los casos el error cometido en el cálculo de sobretensiones no supera el 3%. Además, en más del 99 % de los casos la red neuronal no presenta confusiones al clasificar las descargas.

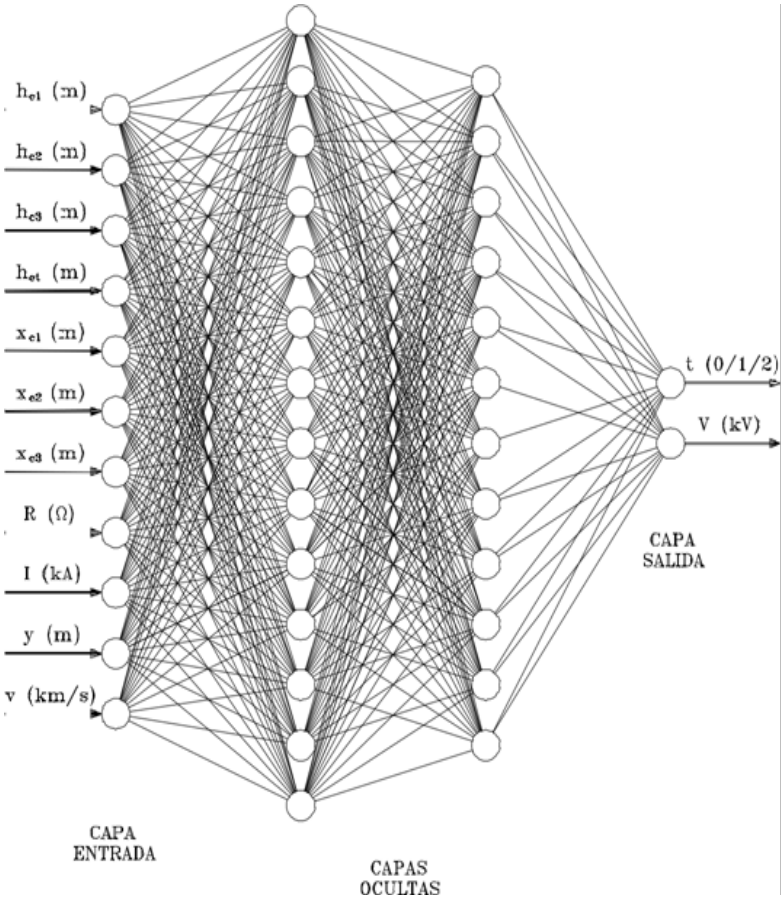
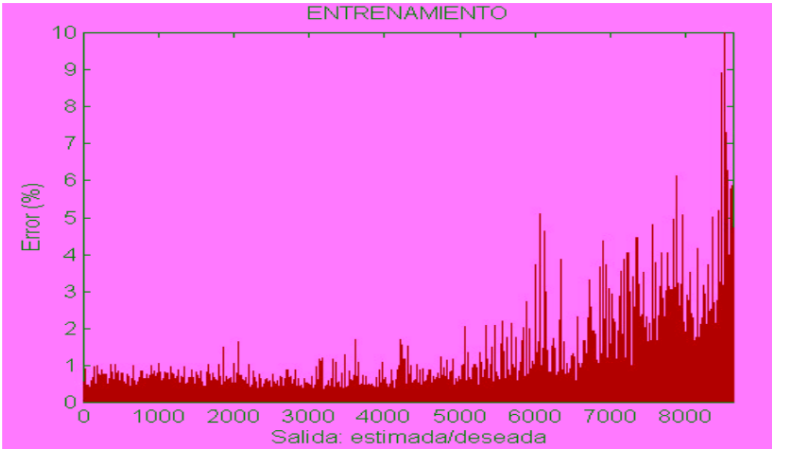


Figura 6.32. Arquitectura de la red neuronal (Método de Rusck con líneas apantalladas).

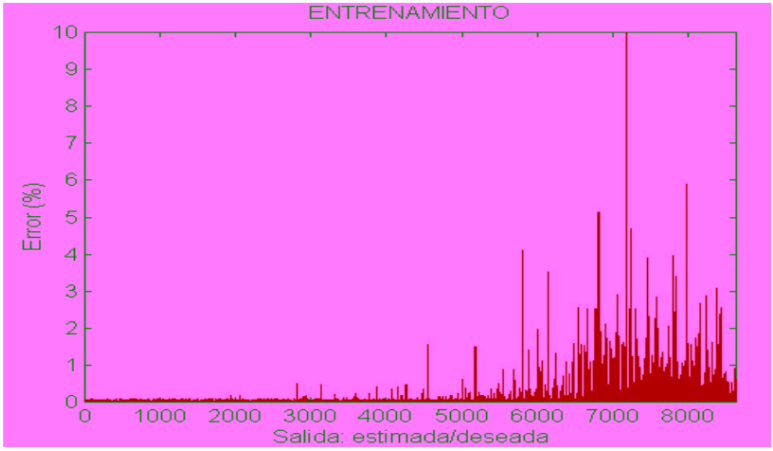
Tabla 6.12. Variables altura de los conductores, altura del cable de tierra, distancia al eje de los conductores y resistencia de puesta a tierra.

	h_{c1} (m)	h_{c2} (m)	h_{c3} (m)	x_{c1} (m)	x_{c2} (m)	x_{c3} (m)	h_{ct} (m)	R (Ω)
Línea horizontal	5	5	5	-1.5	0	1.5	6.5	30
	10	10	10	-2	0	2	11.5	50
	17	17	17	-3	0	3	18.5	10
Línea vertical	5	7	9	1	1	1	10.5	20
	10	12	14	1	1	1	15.5	50
	14	16	18	1.5	1.5	1.5	19.5	70

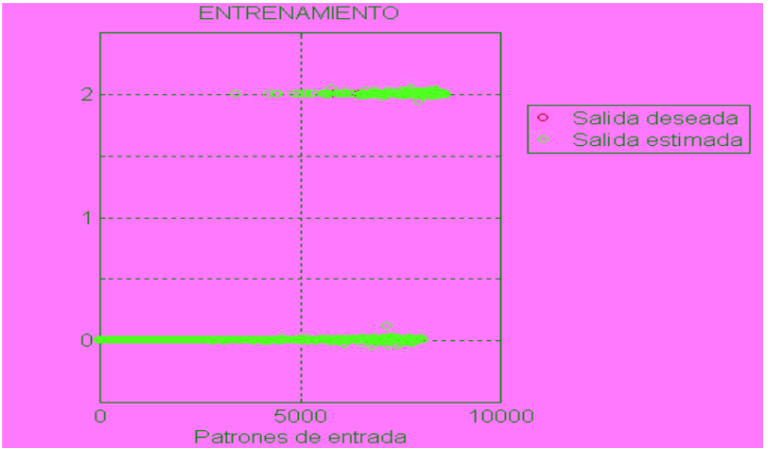


a. Error porcentual en el cálculo de sobretensiones.

b. Ajuste de las salidas (tensiones).

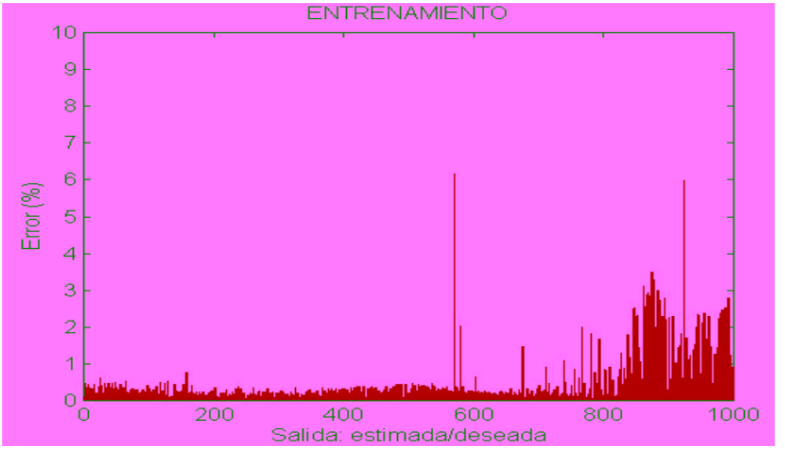


c. Error porcentual en la clasificación de descargas.

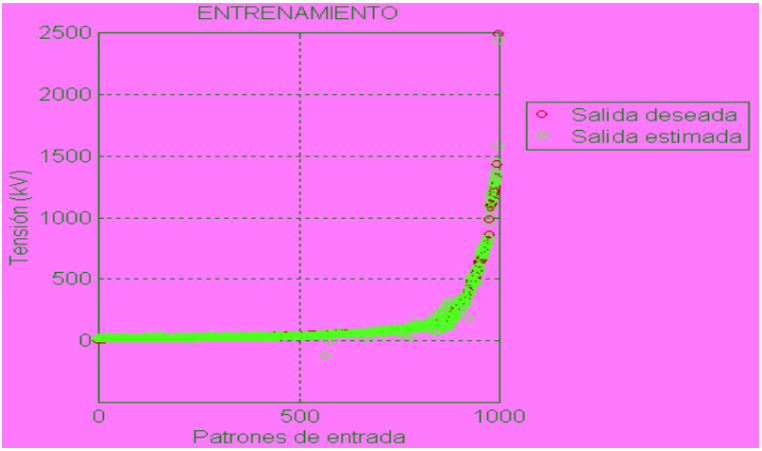


d. Ajuste del clasificador.

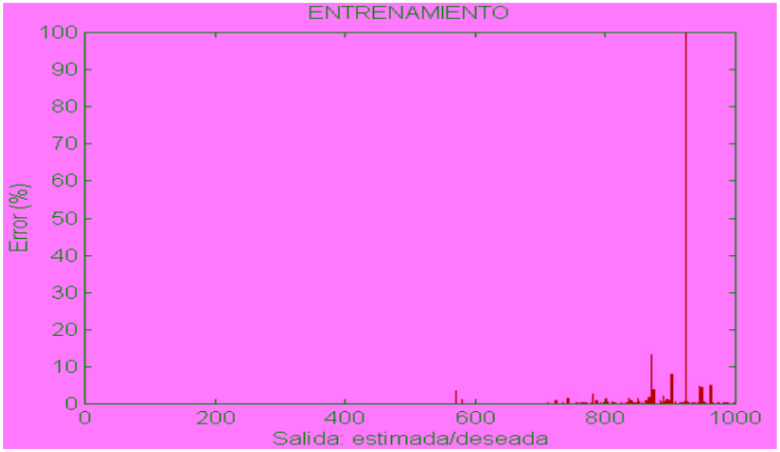
Figura 6.33. Entrenamiento de la red neuronal.



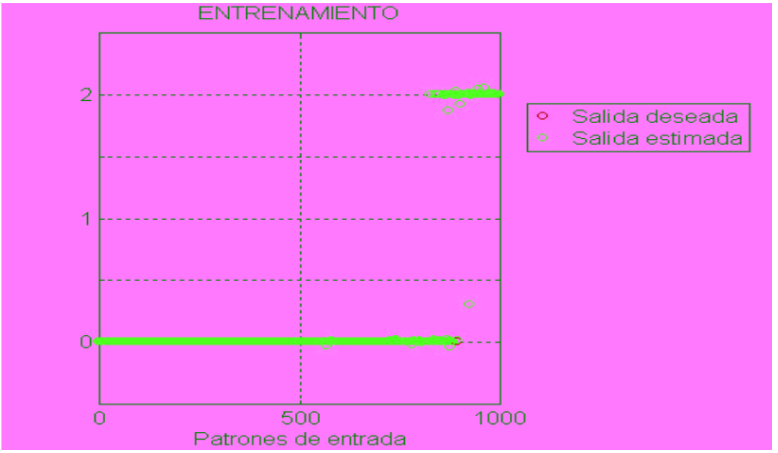
a. Error porcentual en el cálculo de sobretensiones.



b. Ajuste de las salidas (tensiones).



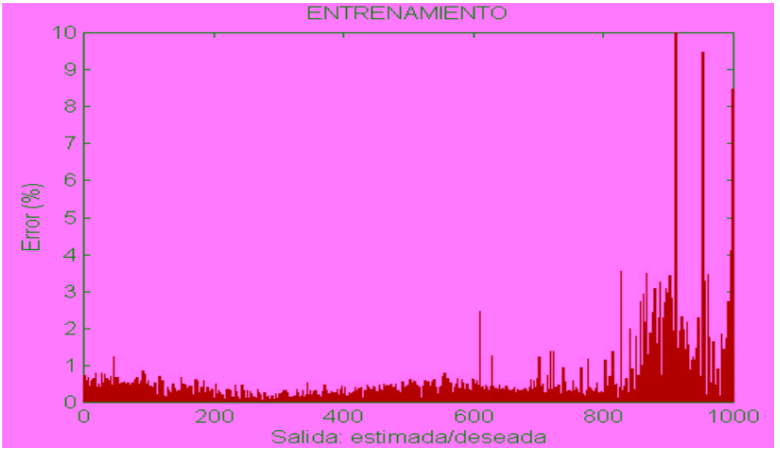
c. Error porcentual en la clasificación de descargas.



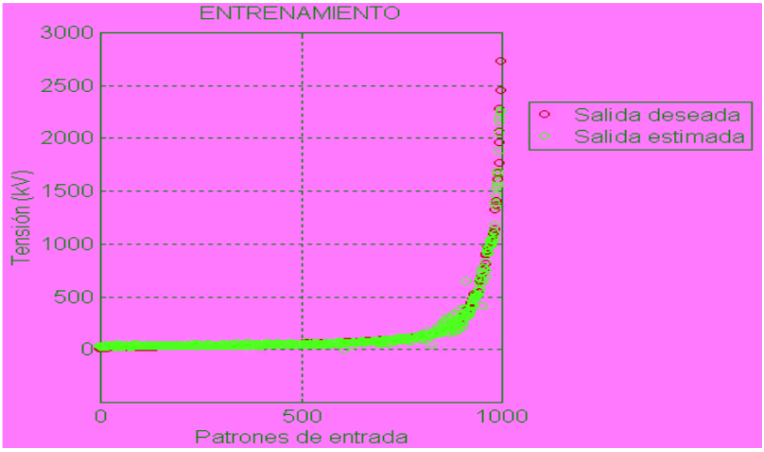
d. Ajuste del clasificador.

Figura 6.34. Validación de la red neuronal. Línea horizontal, altura conductores = 10 m, distancia entre conductores más externos = 4 m,

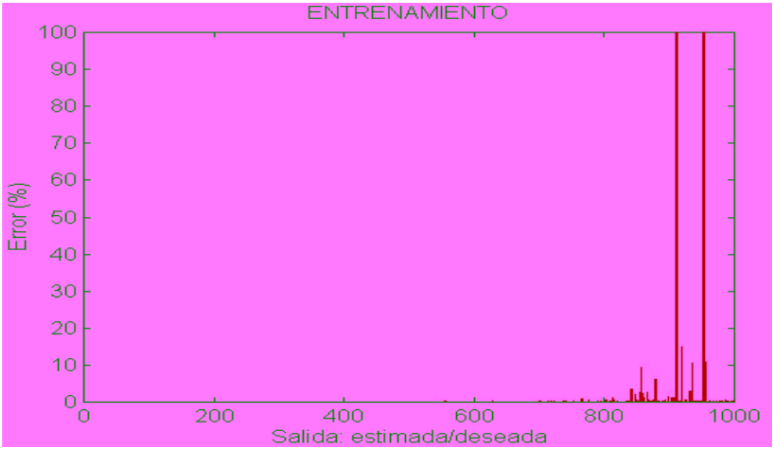
altura cable de tierra = 11.5 m, resistencia de puesta a tierra = 50 Ω.



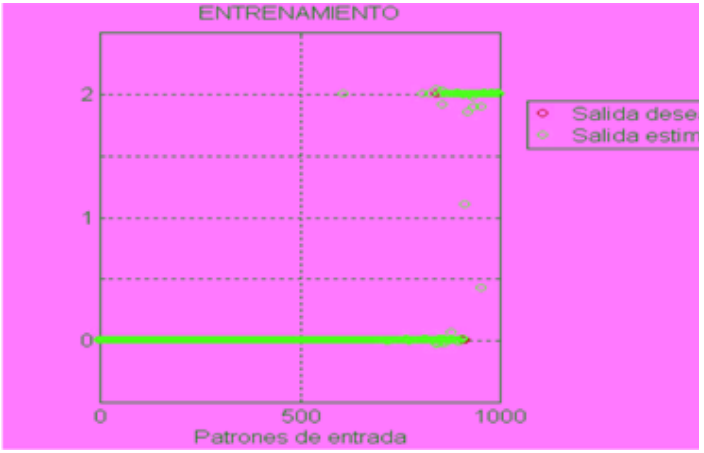
a. Error porcentual en el cálculo de sobretensiones.



b. Ajuste de las salidas (tensiones).

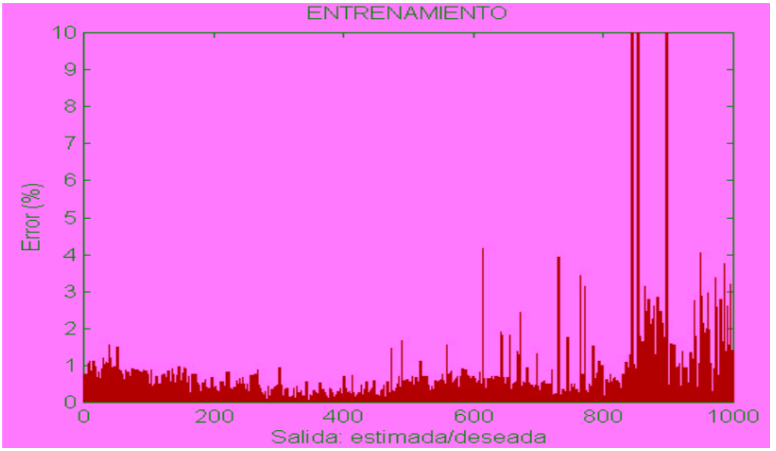


c. Error porcentual en la clasificación de descargas.

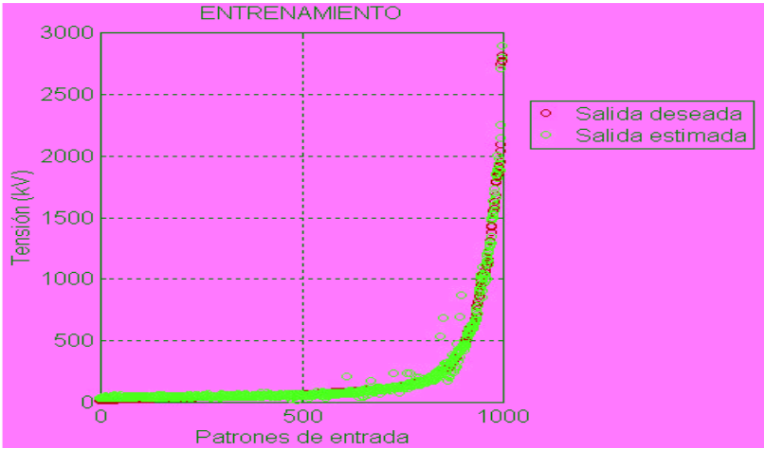


d. Ajuste del clasificador.

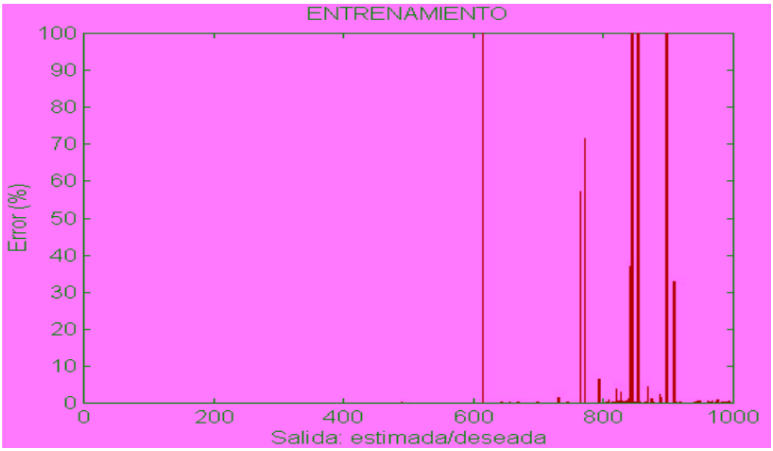
Figura 6.35. Validación de la red neuronal. Línea vertical, altura c1 =10 m, altura c2 =12 m, altura c3 =14 m, distancia horizontal conductores - cable de tierra = 1 m, altura cable de tierra = 15.5 m, resistencia de puesta a tierra = 50 Ω. .



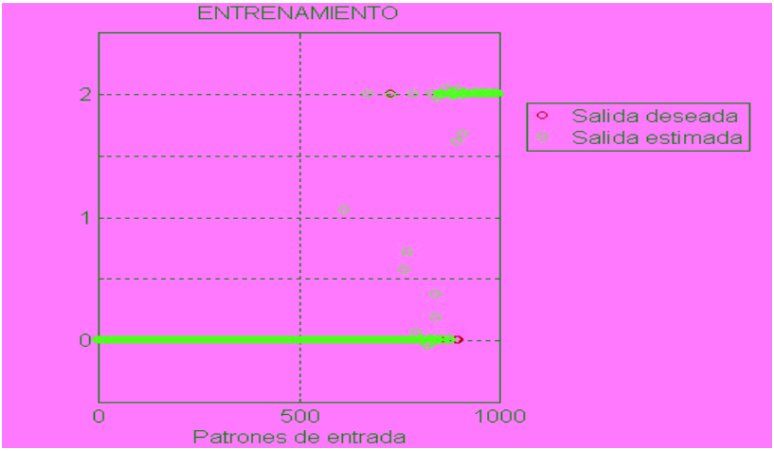
a. Error porcentual en el cálculo de sobretensiones.



b. Ajuste de las salidas (tensiones).



c. Error porcentual en la clasificación de descargas.



d. Ajuste del clasificador.

Figura 6.36. Validación de la red neuronal. Línea vertical, altura c1 =14 m, altura c2 =16 m, altura c3 =18 m, distancia horizontal conductores - cable de tierra = 1.5 m, altura cable de tierra = 19.5 m, resistencia de puesta a tierra = 70 Ω..

6.3.4.2. Método de Chowdhuri

En este estudio se han escogido las siguientes variables de entrada y salida de la red

·Entradas: Intensidad máxima de la descarga (variable I), tiempo de frente de la onda de corriente del rayo (variable tf), distancia perpendicular entre la línea y la descarga (variable y), velocidad de retorno del rayo (variable v), altura de cada conductor de fase (variable hc1, variable hc2, variable hc3), altura del cable de tierra (variable hct), distancia al eje de cada conductor de fase (variable xc1, variable xc2, variable xc3), resistencia de puesta a tierra de los postes (variable R)

·Salidas: Tipo de descarga, directa al cable de tierra/directa al conductor de fase/indirecta (variable t), sobretensión que aparece en la línea (variable V).

Este estudio se ha realizado de manera idéntica al anterior, método de Rusck. La única diferencia es que con el método de Chowdhuri la red neuronal tiene una entrada más, el tiempo de frente, cuyos valores se han generado siguiendo el procedimiento visto en el apartado 5.3.1 del capítulo 5.

Entrenamiento

Para entrenar la red neuronal se seguirá la metodología utilizada en los apartados anteriores. La Tabla 6.13 muestra una parte de los datos utilizados en el entrenamiento.

La estructura de red neuronal utilizada ha sido la siguiente

·2 capas de neuronas ocultas, con 14 y 12 neuronas respectivamente

·Funciones de transferencia: tan-sigmoidal (capas ocultas), lineal (capa de salida)

·Tasa de aprendizaje variable, $\alpha 0=0.25$, $\alpha i=1.01$, $\alpha d=0.6$

·Momento $\beta=0.9$.

El entrenamiento de esta red neuronal tiene los siguientes pasos

1)Estudio previo. Se guardan los pesos y los umbrales que dan el mínimo error.

2)Ajuste de la pendiente de las funciones de transferencia. Se mantiene un valor unidad para la pendiente de la función tan-sigmoidal, y un valor de 0.3 para la función lineal.

3)Se entrena la red neuronal con la estructura comentada anteriormente y con las pendientes ajustadas hasta que el error se estabiliza. La figura 6.37 muestra la arquitectura de la red, mientras que la figura 6.38 muestra los resultados del entrenamiento obtenidos con esta red. Aunque los resultados del entrenamiento han sido algo peores que con el método de Rusck, se puede observar que en general el error cometido en las dos salidas, clasificador y cálculo de tensiones, no supera el 5 %.

Tabla 6.13. Datos del entrenamiento.

ENTRADAS												SALIDAS	
Altura conductor 1 (m)	Altura conductor 2 (m)	Altura conductor 3 (m)	Altura cable de tierra (m)	Distancia al eje conductor 1 (m)	Distancia al eje conductor 2 (m)	Distancia al eje conductor 3 (m)	Resistencia puesta a tierra (Ω)	Intensidad (kA)	Tiempo de frente (μs)	Distancia descarga-línea (m)	Velocidad (km/s)	Tipo descarga	Tensión (kV)
5	5	5	8.0	-3	0	3	70	29	9.0	377	76350	0	12.30
5	5	5	8.0	-3	0	3	70	136	3.0	147	31777	0	776.40
5	5	5	8.0	-3	0	3	70	66	3.5	161	61855	0	135.40
8	8	8	9.5	-1	0	1	10	52	2.0	432	92757	0	83.52
8	8	8	11.0	-1	0	1	10	18	4.5	488	46634	0	37.39
8	8	8	11.0	-1	0	1	40	11	6.5	132	62081	0	18.09
11	11	11	12.5	-1	0	1	40	92	9.0	169	61182	0	134.33
11	11	11	12.5	-3	0	3	10	44	2.0	171	102868	0	144.10
11	11	11	12.5	-3	0	3	10	42	2.0	14	136539	2	243.13
14	14	14	15.5	-1	0	1	40	18	5.0	411	61905	0	41.02
14	14	14	15.5	-1	0	1	40	30	6.5	463	48117	0	70.39
14	14	14	17.0	-1	0	1	70	34	3.0	303	86004	0	97.36
14	14	14	17.0	-1	0	1	70	17	3.5	196	79542	0	54.32
17	17	17	20.0	-1	0	1	40	33	1.5	358	100006	0	143.15
17	17	17	20.0	-1	0	1	40	22	9.0	486	61462	0	33.30
17	17	17	20.0	-1	0	1	40	10	4.0	269	69497	0	37.26
5	7	9	10.5	1.5	1.5	1.5	40	23	10.5	251	33577	0	42.74
5	7	9	12.0	1.5	1.5	1.5	40	21	9.0	418	133333	0	4.28
5	7	9	12.0	1.5	1.5	1.5	70	68	6.0	115	82608	0	89.01
8	10	12	15.0	0.5	0.5	0.5	10	30	2.0	129	141703	0	57.09
8	10	12	15.0	0.5	0.5	0.5	10	150	6.0	95	38035	2	976.40
8	10	12	15.0	0.5	0.5	0.5	10	9	2.0	485	82897	0	22.21
11	13	15	16.5	0.5	0.5	0.5	40	59	1.5	279	109728	0	206.35
11	13	15	16.5	0.5	0.5	0.5	40	20	5.5	205	97807	0	27.84
11	13	15	18.0	0.5	0.5	0.5	40	22	1.5	394	78116	0	103.72
14	16	18	21.0	1.5	1.5	1.5	70	78	5.5	265	50988	0	326.07
14	16	18	21.0	1.5	1.5	1.5	70	142	8.0	450	57135	0	279.95
14	16	18	21.0	1.5	1.5	1.5	70	56	2.5	362	56223	0	364.64

Validación

La validación de esta red neuronal se ha realizado con 6000 patrones que no han intervenido en el proceso de entrenamiento. Se han utilizado las mismas 6 configuraciones de línea que se encuentran definidas en la Tabla 6.12.

Los gráficos de las figuras 6.39, 6.40, y 6.41 muestran los resultados obtenidos para 3 de las configuraciones anteriores. Los resultados obtenidos con las otras configuraciones son similares. Los resultados han sido similares a los obtenidos con el método de Rusck para líneas apantalladas, aunque ahora la red neuronal es algo más compleja. Los gráficos apenas muestran confusión por parte de la red neuronal en la clasificación de descargas, y además, en general el error cometido en el cálculo de las tensiones es inferior al 3 %.

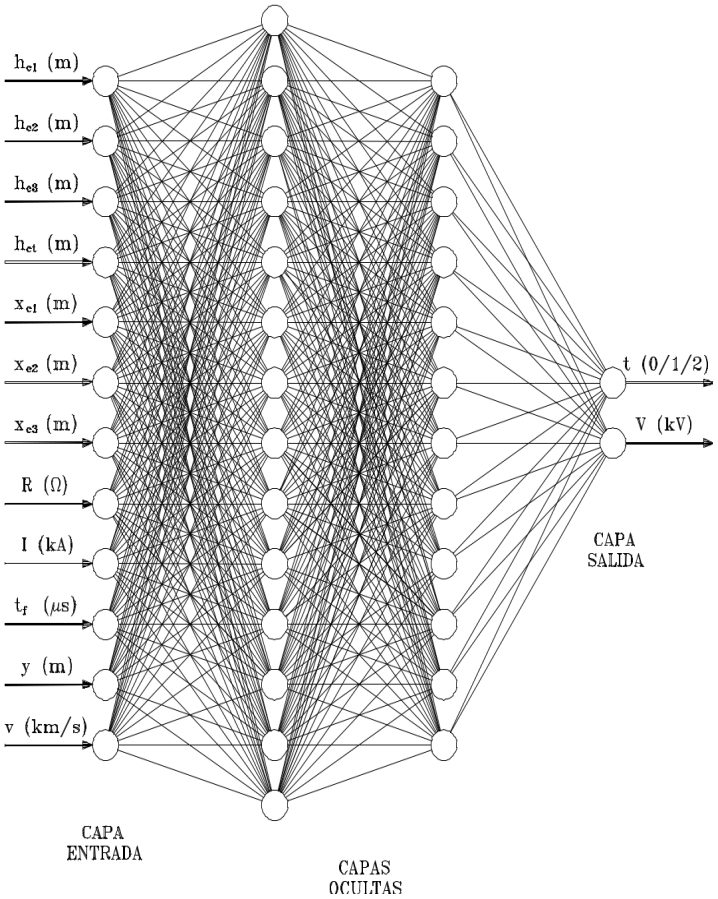
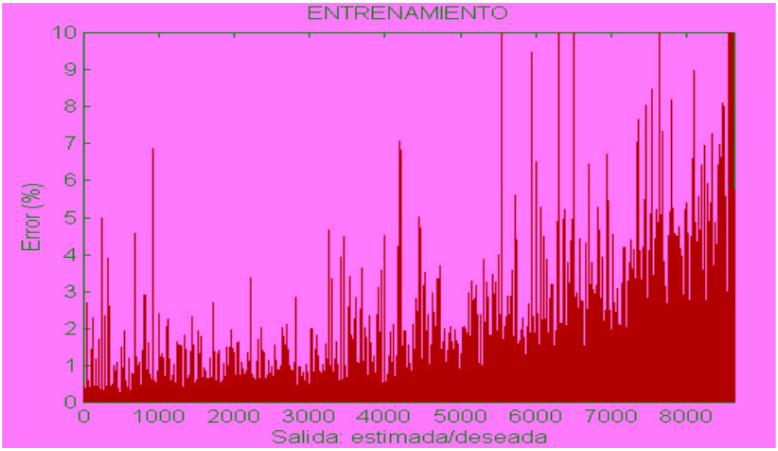
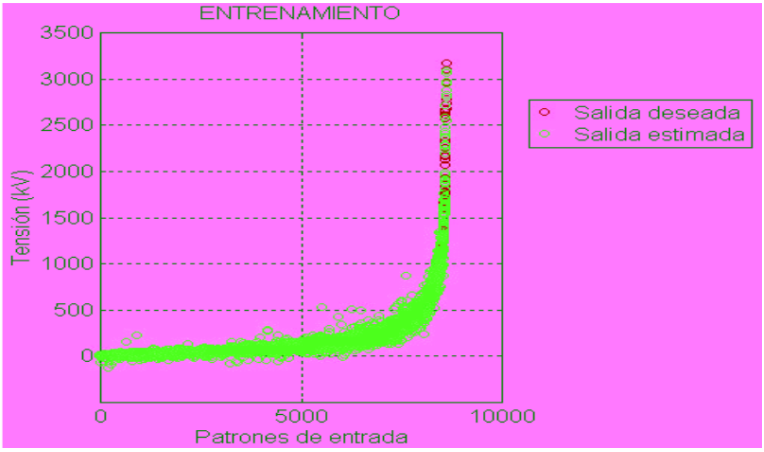


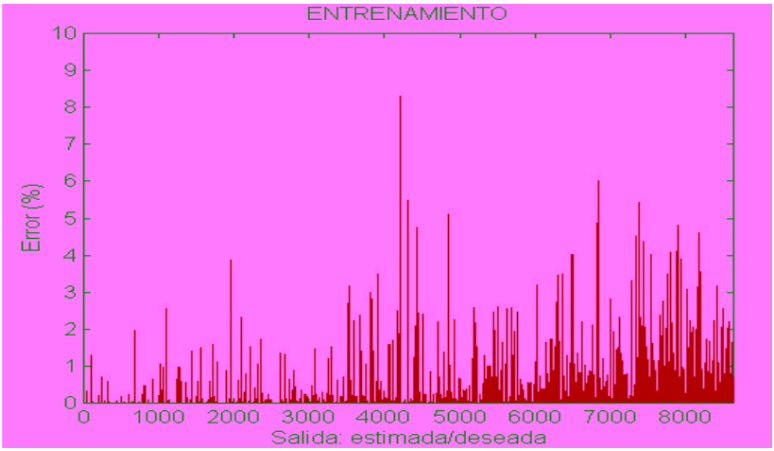
Figura 6.37. Arquitectura de la red neuronal (Método de Chowdhuri con líneas apantalladas).



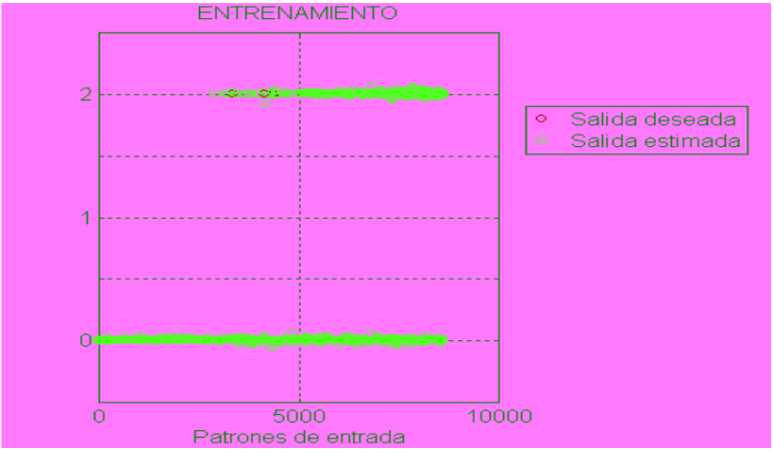
a. Error porcentual en el cálculo de sobretensiones.



b. Ajuste de las salidas (tensiones).

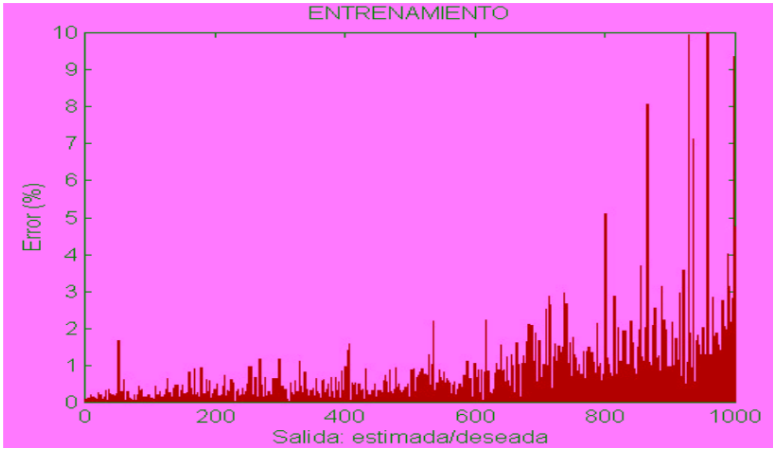


c. Error porcentual en la clasificación de descargas

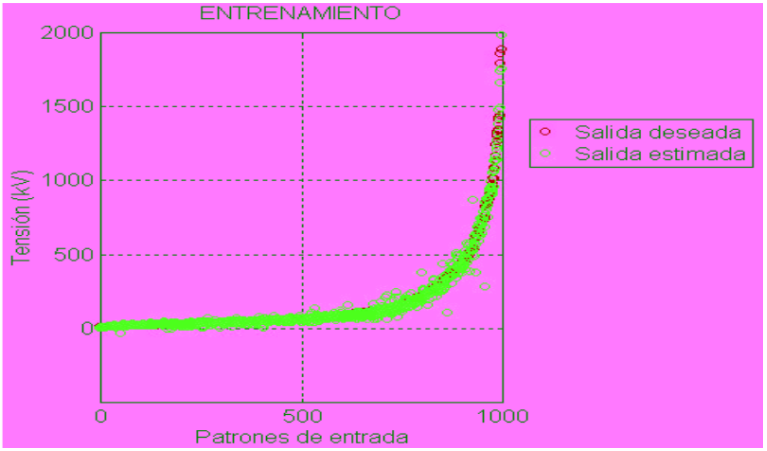


d. Ajuste del clasificador.

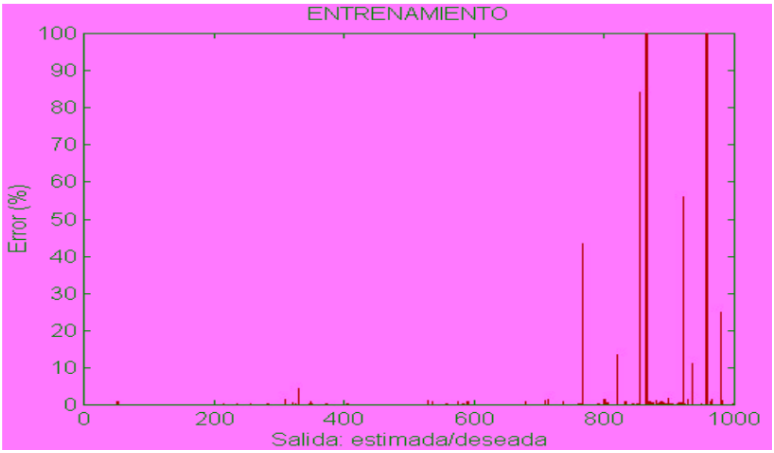
Figura 6.38. Entrenamiento de la red neuronal.



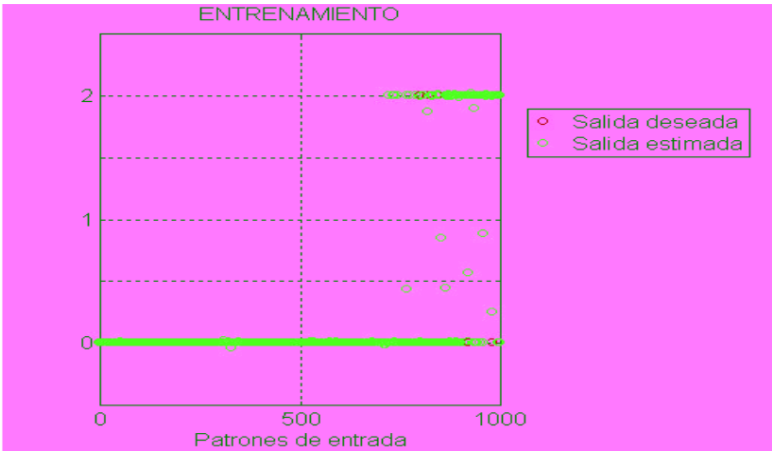
a. Error porcentual en el cálculo de sobretensiones.



b. Ajuste de las salidas (tensiones).



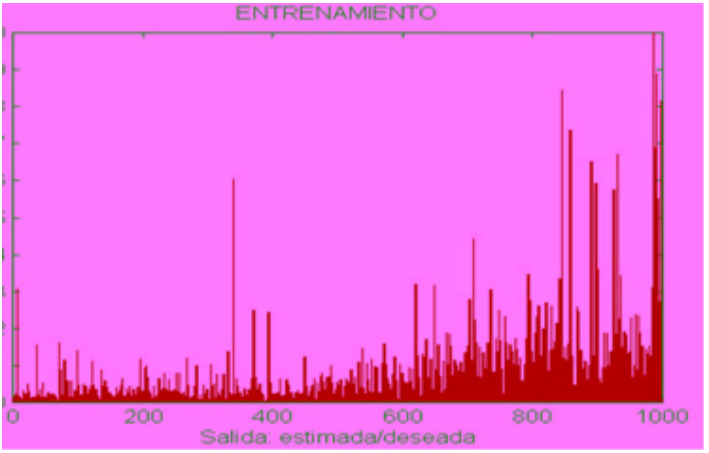
c. Error porcentual en la clasificación de descargas.



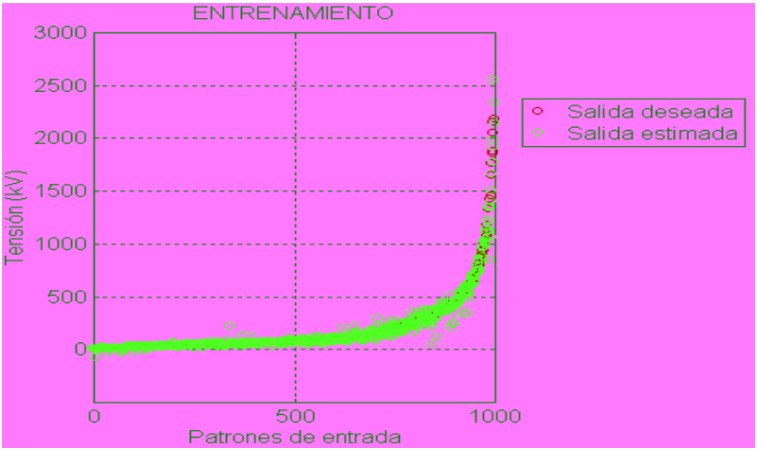
d. Ajuste del clasificador.

Figura 6.39. Validación de la red neuronal. Línea horizontal, altura conductores = 10 m, distancia entre conductores más externos = 4 m,

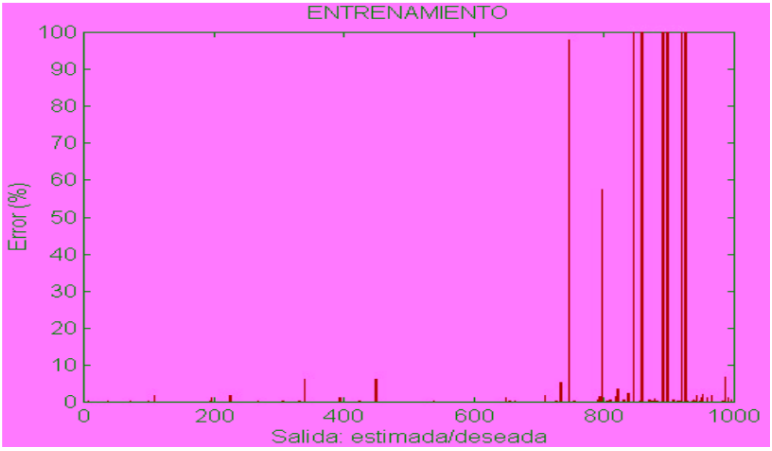
altura cable de tierra = 11.5 m, resistencia de puesta a tierra = 50 Ω.



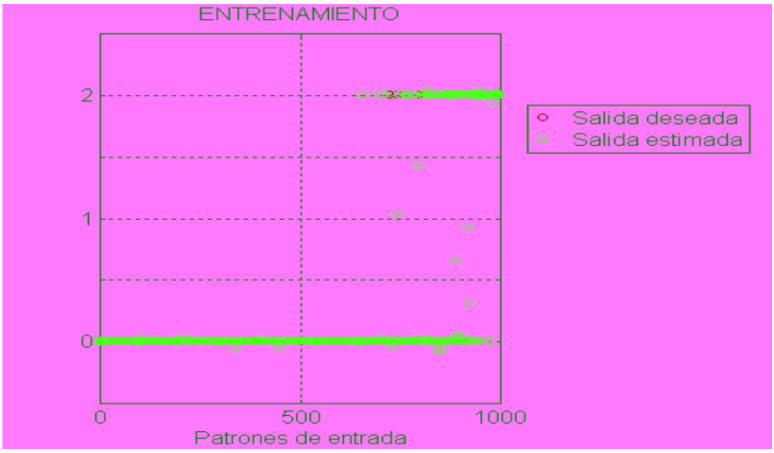
a. Error porcentual en el cálculo de sobretensiones.



b. Ajuste de las salidas (tensiones).

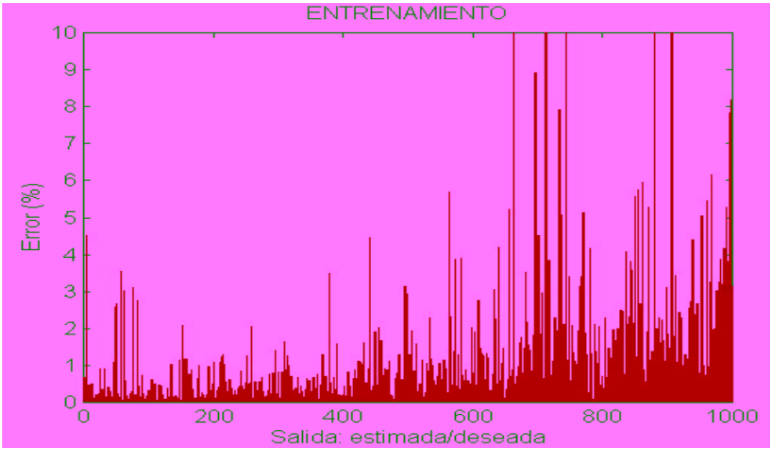


c. Error porcentual en la clasificación de descargas.

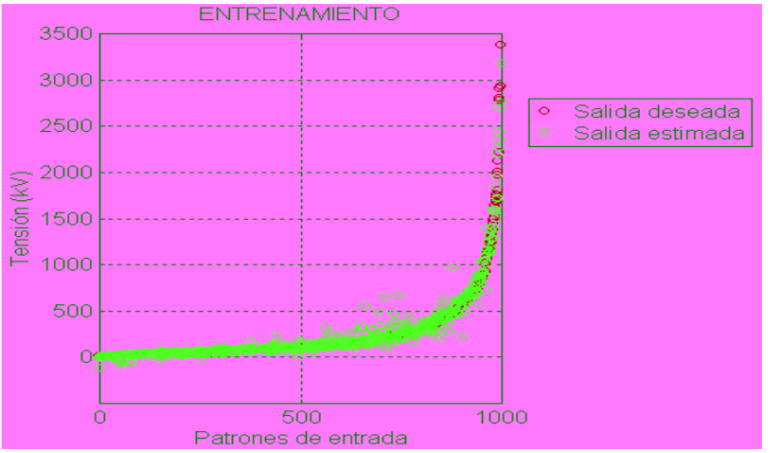


d. Ajuste del clasificador.

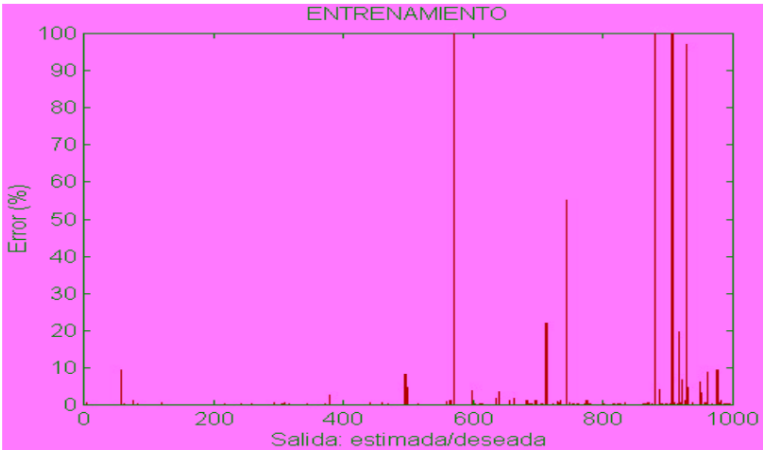
Figura 6.40. Validación de la red neuronal. Línea vertical, altura c1 =10 m, altura c2 =12 m, altura c3 =14 m, distancia horizontal conductores - cable de tierra = 1 m, altura cable de tierra = 15.5 m, resistencia de puesta a tierra = 50Ω.



a. Error porcentual en el cálculo de sobretensiones.



b. Ajuste de las salidas (tensiones).



c. Error porcentual en la clasificación de descargas.

d. Ajuste del clasificador.

Figura 6.41. Validación de la red neuronal. Línea vertical, altura c1 =14 m, altura c2 =16 m, altura c3 =18 m, distancia horizontal conductores - cable de tierra = 1.5 m, altura cable de tierra = 19.5 m, resistencia de puesta a tierra = 70 Ω.

6.3.5.Cálculo de sobretensiones en líneas sin cable de tierra. Velocidad de retorno del rayo en función de la intensidad máxima

6.3.5.1. Método de Rusck

En este estudio se han escogido las siguientes variables de entrada y salida de la red

·Entradas (variables independientes): Intensidad máxima de la descarga (variable I), distancia perpendicular entre la línea y la descarga (variable y), velocidad de retorno del rayo (variable v), parámetro W (variable W), altura de la línea (variable h).

·Salidas: Tipo de descarga, directa al cable de tierra/directa al conductor de fase/indirecta (variable t), sobretensión que aparece en la línea (variable V).

La red neuronal se ha entrenado con un total de 9000 patrones diferentes calculados de forma aleatoria. Este estudio se ha realizado de manera idéntica al utilizado con el método de Rusck en líneas sin apantallar y distribución de velocidad de retorno del rayo uniforme, para más detalles ver el apartado 6.3.3. La diferencia es que ahora la velocidad de retorno del rayo se calcula a partir de la intensidad de pico de la descarga. Por tanto, se añadirá una nueva entrada a la red neuronal, la que corresponde al parámetro W, ver expresión (3.10), el cual tomará cuatro valores diferentes en este entrenamiento: 50, 200, 350 y 500.

Entrenamiento

Para entrenar la red neuronal se seguirá la metodología utilizada en los apartados anteriores. La Tabla 6.14 muestra una parte de los datos utilizados en el entrenamiento.

La estructura de red neuronal utilizada ha sido la siguiente

·2 capas de neuronas ocultas, con 14 y 12 neuronas respectivamente

·funciones de transferencia: tan-sigmoidal (capas ocultas), lineal (capa de salida)

·Tasa de aprendizaje variable, $\alpha 0=0.25$, $\alpha i=1.01$, $\alpha d=0.6$

·Momento $\beta=0.9$.

El entrenamiento de esta red neuronal se puede resumir en los siguientes pasos

1)Estudio previo. Se guardan los pesos y los umbrales que dan el mínimo error.

2)Ajuste de la pendiente de las funciones de transferencia. Se mantiene un valor unidad para la pendiente de la función tan-sigmoidal, y un valor de 0.1 para la función lineal.

3)Se entrena la red neuronal con la estructura comentada anteriormente y con las pendientes ajustadas hasta que el error se estabiliza. La figura 6.42 muestra la arquitectura de la red, y la figura 6.43 muestra los resultados del entrenamiento obtenidos con esta red. Se puede observar que en general el error cometido en las dos salidas, clasificador y cálculo de tensiones, se encuentra acotado por debajo del 10 %. Los resultados han sido algo peores que cuando se supuso que la velocidad seguía una distribución uniforme, lo cual es lógico porque la complejidad de la red neuronal ha aumentado.

Tabla 6.14. Datos de entrenamiento.

ENTRADAS					SALIDAS	
Altura (m)	Intensidad (kA)	Distancia descarga - línea (m)	Velocidad (km/s)	W	Tipo descarga	Tensión (kV)
5	33	180	189164	50	0	40.82
5	61	268	221982	50	0	54.27
5	39	86	197870	50	0	101.66
5	13	45	136277	50	0	57.21
5	19	376	157425	50	0	10.51
5	30	122	182746	50	0	53.11
5	19	99	155906	50	0	38.91
5	33	357	189164	50	0	20.60
5	38	346	197139	50	0	24.88
5	34	250	190863	50	0	30.39
8	57	92	111845	350	0	187.30
8	26	312	78178	350	0	23.17
8	17	130	64567	350	0	35.92
8	13	231	56773	350	0	15.28
8	19	194	67218	350	0	26.36
8	30	440	83642	350	0	19.22
8	96	417	139184	350	0	74.09
8	33	175	88060	350	0	54.53
8	38	164	93326	350	0	66.72
8	34	68	89268	350	0	144.39
11	36	354	116477	200	0	42.36
11	18	416	86204	200	0	17.17
11	23	76	95400	200	0	119.03
11	21	151	91473	200	0	54.39
11	4	457	42008	200	0	3.16
11	22	44	93466	200	1	5408.67
11	13	363	72761	200	0	13.30
11	6	96	49079	200	0	21.07
11	15	427	77999	200	0	13.25
11	25	173	100000	200	0	59.62
14	33	172	74647	500	0	94.67
14	38	161	79730	500	0	117.47
14	34	65	75699	500	1	8800.07
14	46	200	86642	500	0	115.11
14	21	117	59537	500	0	84.04
14	50	264	90453	500	0	96.56
14	40	193	81175	500	0	102.19
14	49	354	89626	500	0	70.56
14	45	428	86204	500	0	53.11
14	17	234	53620	500	0	33.27
17	50	138	211598	50	0	287.43
17	66	25	225918	50	1	17335.63
17	25	99	173205	50	0	185.40
17	58	405	219407	50	0	115.87
17	26	492	174348	50	0	38.22
17	17	311	151115	50	0	38.42
17	14	44	138325	50	1	3572.99
17	19	375	157425	50	0	36.08
17	30	253	182746	50	0	87.48
17	101	97	245354	50	1	26731.23

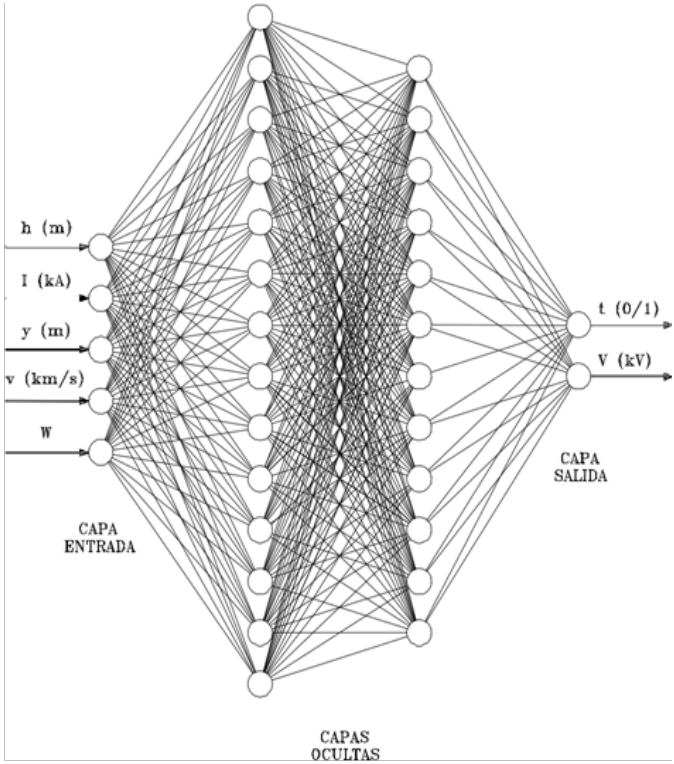


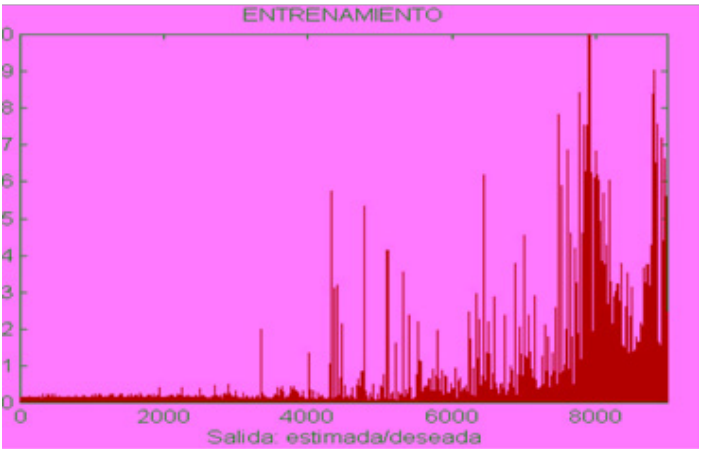
Figura 6.42. Arquitectura de la red neuronal (Método de Rusck con líneas sin apantallar).

Validación

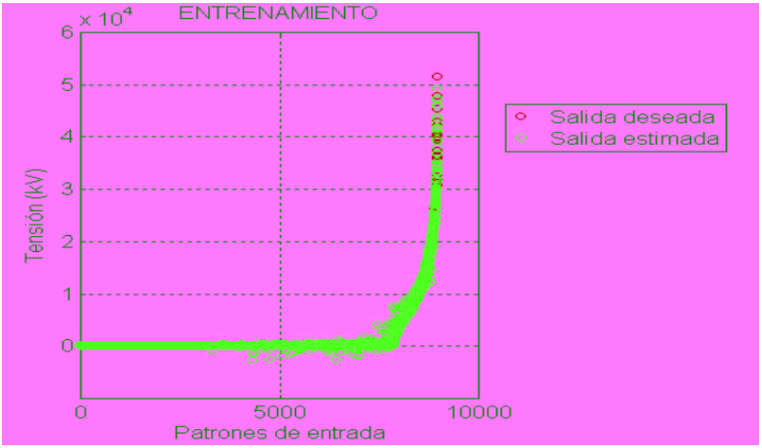
La validación de esta red neuronal se ha realizado con 10000 patrones que no han intervenido en el proceso de entrenamiento. Se han utilizado 1000 patrones para cada combinación de la altura de la línea y del parámetro W mostrada en la Tabla 6.15. Los gráficos de las figuras 6.44, 6.45, y 6.46 muestran los resultados obtenidos para 3 de las configuraciones anteriores. Los resultados obtenidos con las otras configuraciones son similares. Estos resultados muestran que en la inmensa mayoría de los casos el error cometido en el cálculo de tensiones se encuentra por debajo del 3 %, y que la red neuronal apenas muestra confusión en la clasificación de descargas.

Tabla 6.15. Variables altura y parámetro W.

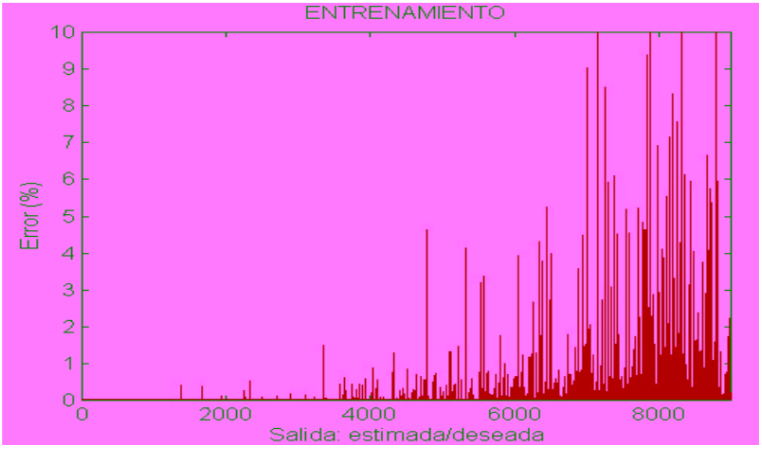
h (m)	W
5	50
8	200
8	500
10	350
10	500
11	50
14	200
14	500
17	350
17	200



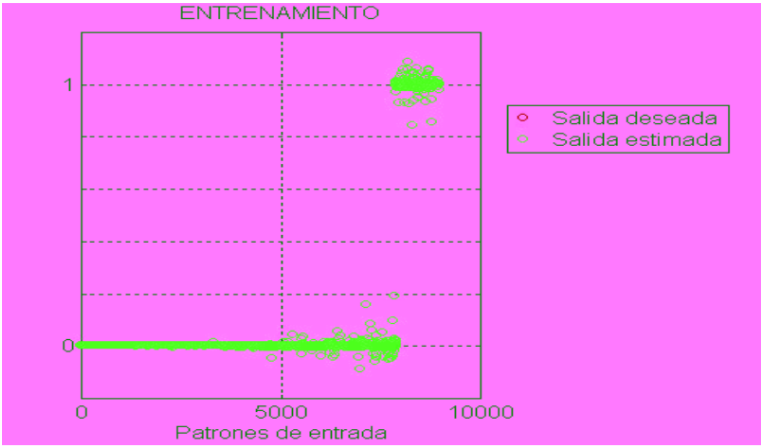
a. Error porcentual en el cálculo de sobretensiones.



b. Ajuste de las salidas (tensiones).

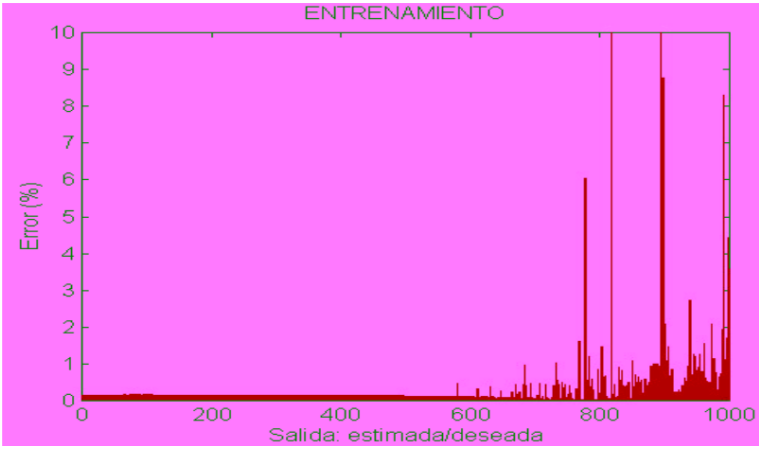


c. Error porcentual en la clasificación de descargas.

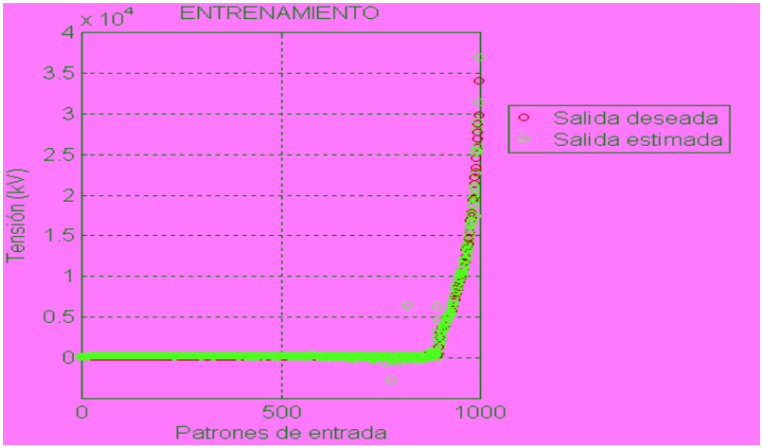


d. Ajuste del clasificador.

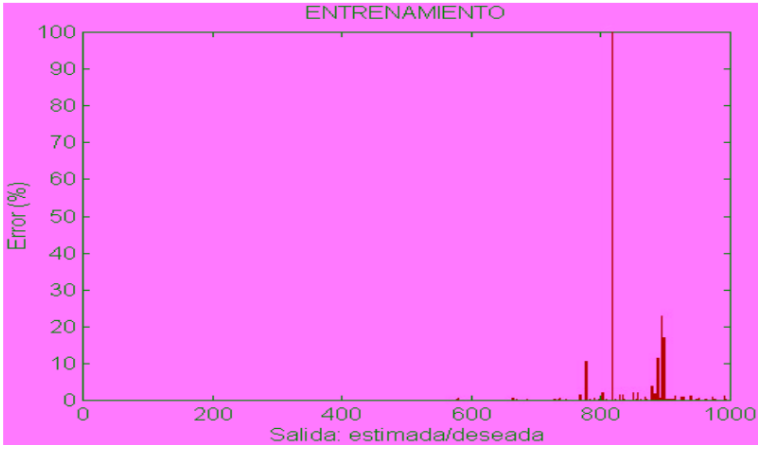
Figura 6.43. Entrenamiento de la red neuronal.



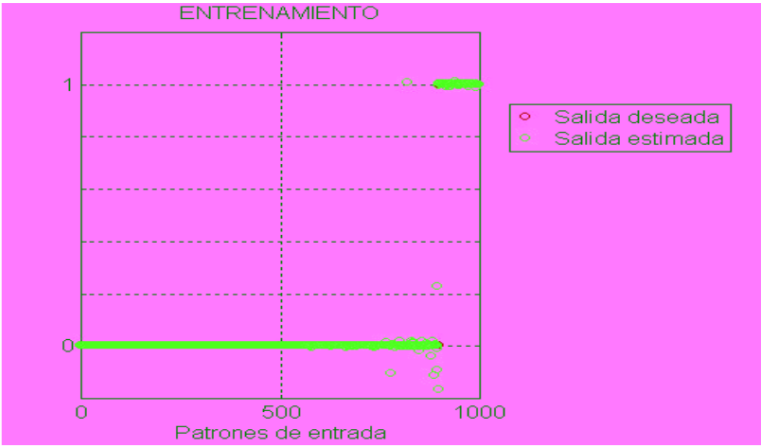
a. Error porcentual en el cálculo de sobretensiones.



b. Ajuste de las salidas (tensiones).

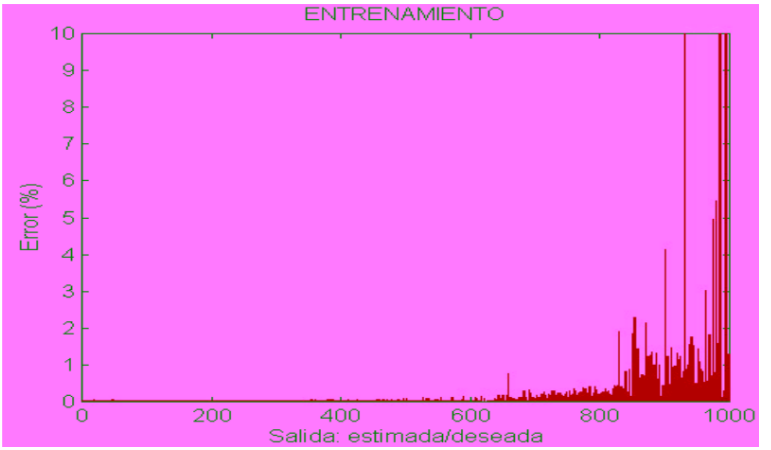


c. Error porcentual en la clasificación de descargas.

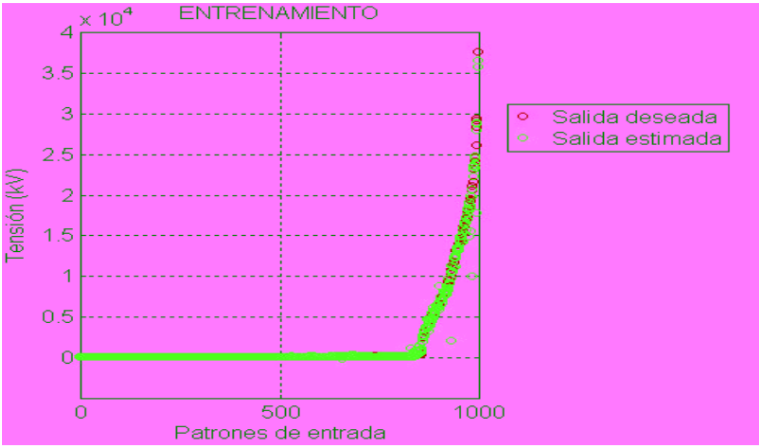


d. Ajuste del clasificador.

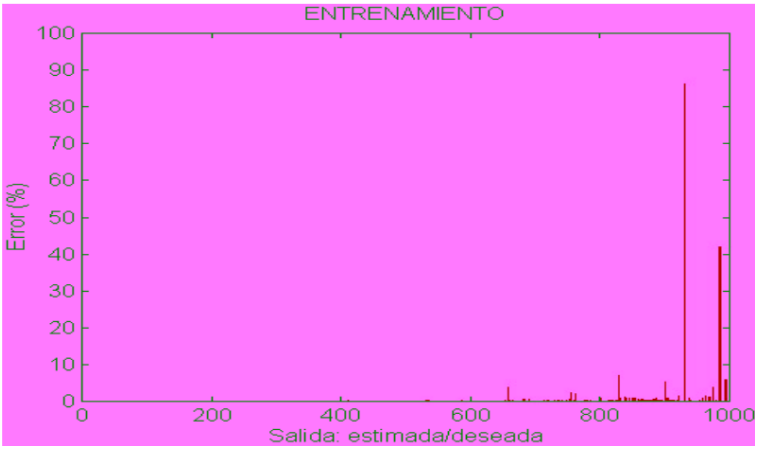
Figura 6.44. Validación de la red neuronal. Altura = 5 m, W = 50.



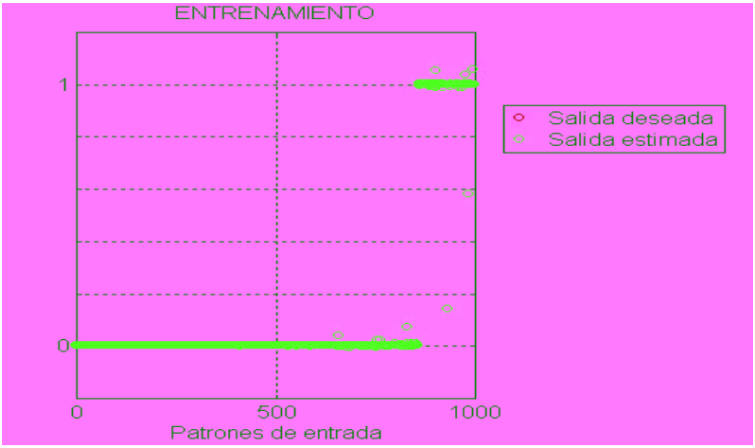
a. Error porcentual en el cálculo de sobretensiones.



b. Ajuste de las salidas (tensiones).

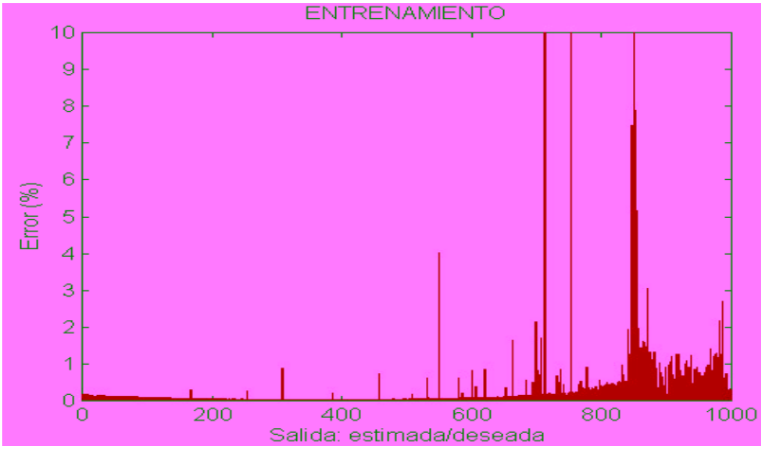


c. Error porcentual en la clasificación de descargas.

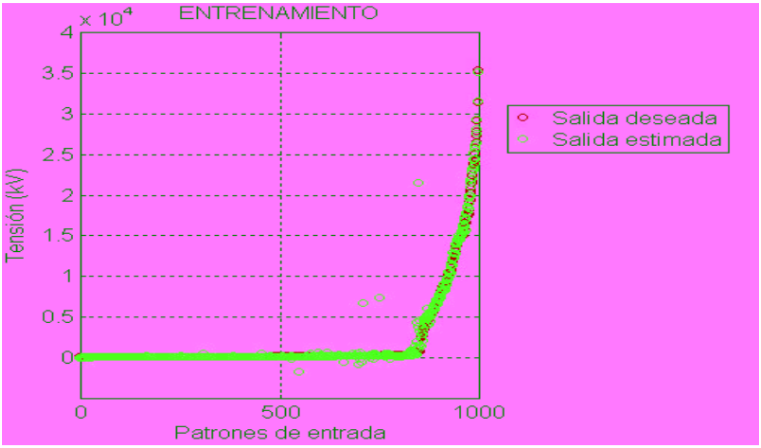


d. Ajuste del clasificador.

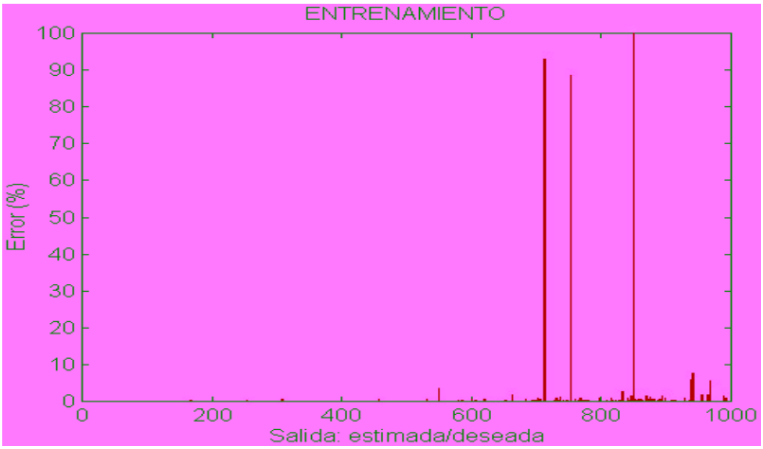
Figura 6.45. Validación de la red neuronal. Altura = 10 m, W = 350.



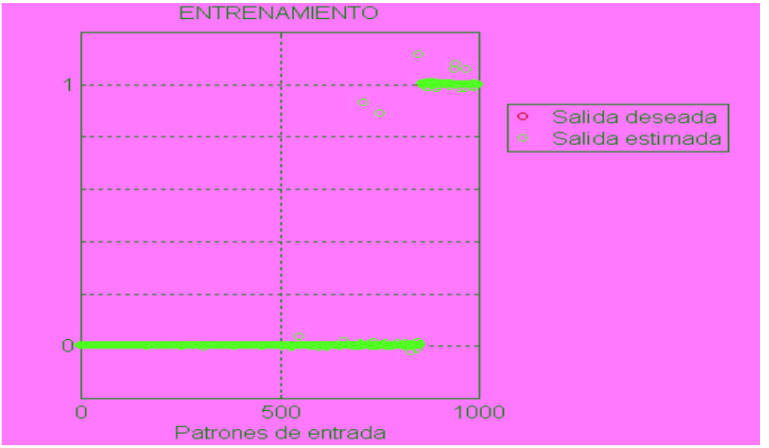
a. Error porcentual en el cálculo de sobretensiones.



b. Ajuste de las salidas (tensiones).



c. Error porcentual en la clasificación de descargas.



d. Ajuste del clasificador.

Figura 6.46. Validación de la red neuronal. Altura = 17 m, W = 200.

6.3.5.2. Método de Chowdhuri

En este estudio se han escogido las siguientes variables de entrada y salida de la red

·Entradas: Intensidad máxima de la descarga (variable I), tiempo de frente de la onda de corriente del rayo (variable tf), distancia perpendicular entre la línea y la descarga (variable y), velocidad de retorno del rayo (variable v), parámetro W (variable W), altura de cada conductor de fase (variable hc1, variable hc2, variable hc3), distancia al eje de cada conductor de fase (variable xc1, variable xc2, variable xc3)

·Salidas: Tipo de descarga, directa al cable de tierra/directa al conductor de fase/indirecta (variable t), sobretensión que aparece en la línea (variable V).

La red neuronal se ha entrenado con un total de 8960 patrones diferentes calculados de forma aleatoria. Este estudio es similar al utilizado con el método de Chowdhuri en líneas sin apantallar y distribución de velocidad

de retorno del rayo uniforme, para más detalles ver el apartado 6.3.3. Al igual que con el método de Rusck, la diferencia es que ahora se debe de añadir una nueva entrada a la red neuronal, la que corresponde al parámetro W, porque la velocidad de retorno del rayo se calcula a partir de la intensidad de pico de la descarga, ver expresión (3.10). La red neuronal se ha entrenado de nuevo con cuatro valores diferentes del parámetro W: 50, 200, 350 y 500.

Entrenamiento

Para entrenar la red neuronal se seguirá la metodología utilizada en los apartados anteriores. La Tabla 6.16 muestra una parte de los datos utilizados en el entrenamiento.

La estructura de red neuronal utilizada ha sido la siguiente

·2 capas de neuronas ocultas, con 14 y 12 neuronas respectivamente

·Funciones de transferencia: tan-sigmoidal (capas ocultas), lineal (capa de salida)

·Tasa de aprendizaje variable, $\alpha_0=0.25, \alpha_i=1.01, \alpha_d=0.6$

·Momento $\beta=0.9$.

El entrenamiento de esta red neuronal tiene los siguientes pasos

1)Estudio previo. Se guardan los pesos y los umbrales que dan el mínimo error.

2)Ajuste de la pendiente de las funciones de transferencia. Se mantiene un valor unidad para la pendiente de la función tan-sigmoidal, y un valor de 0.1 para la función lineal.

3)Se entrena la red neuronal con la estructura comentada anteriormente y con las pendientes ajustadas hasta que el error se estabiliza. La figura 6.47 muestra la arquitectura de la red, y la figura 6.48 muestra los resultados del entrenamiento obtenidos con esta red. Tal y como ocurrió con el método de Rusck, los resultados han sido algo peores que cuando se supuso que la velocidad seguía una distribución uniforme. Ahora el error cometido en las dos salidas, clasificador y cálculo de tensiones, se encuentra acotado por debajo del 5 %.

Tabla 6.16. Datos de entrenamiento.

ENTRADAS											SALIDAS	
Altura conductor 1 (m)	Altura conductor 2 (m)	Altura conductor 3 (m)	Distancia al eje conductor 1 (m)	Distancia al eje conductor 2 (m)	Distancia al eje conductor 3 (m)	Intensidad (kA)	Tiempo de frente (μs)	Distancia descarga-línea (m)	Velocidad (km/s)	W	Tipo descarga	Tensión (kV)
5	5	5	-3	0	3	83	5.0	80	131025	350	1	18791.36
5	5	5	-3	0	3	32	6.5	289	86829	350	0	19.91
5	5	5	-3	0	3	34	3.0	377	88667	350	0	38.33
8	8	8	-1	0	1	62	7.5	258	99285	500	0	44.49
8	8	8	-1	0	1	41	3.5	323	82120	500	0	77.98
8	8	8	-1	0	1	38	3.0	458	79241	500	0	72.21
11	11	11	-3	0	3	54	2.0	424	216173	50	0	10.27
11	11	11	-3	0	3	20	4.0	492	160357	50	0	7.90
11	11	11	-3	0	3	86	5.0	285	238306	50	0	15.79
14	14	14	-3	0	3	29	4.0	177	106759	200	0	71.01
14	14	14	-3	0	3	70	8.5	238	152753	200	0	32.51
14	14	14	-3	0	3	18	5.5	181	86204	200	0	45.20
17	17	17	-1	0	1	24	2.5	20	170848	50	1	6351.99
17	17	17	-1	0	1	40	2.5	153	200000	50	0	35.27
17	17	17	-1	0	1	60	3.5	493	221565	50	0	15.21
5	7	9	0	0	0	9	3.5	247	60573	200	0	29.15
5	7	9	0	0	0	29	2.5	157	105950	200	0	68.44
5	7	9	0	0	0	27	6.0	314	103464	200	0	21.19
8	10	12	0	0	0	16	1.5	315	62725	350	0	127.84
8	10	12	0	0	0	38	3.5	45	93326	350	1	9531.97
8	10	12	0	0	0	25	4.5	318	77460	350	0	57.20
8	10	12	0	0	0	80	6.5	480	129069	350	0	38.39
11	13	15	0	0	0	22	2.0	315	61588	500	0	184.52
11	13	15	0	0	0	67	9.5	15	102786	500	1	17349.94
11	13	15	0	0	0	4	7.0	402	26726	500	0	27.40
14	16	18	0	0	0	25	8.0	240	99105	200	0	36.56
14	16	18	0	0	0	15	2.5	310	79241	200	0	91.04
14	16	18	0	0	0	46	3.0	437	129152	200	0	77.82

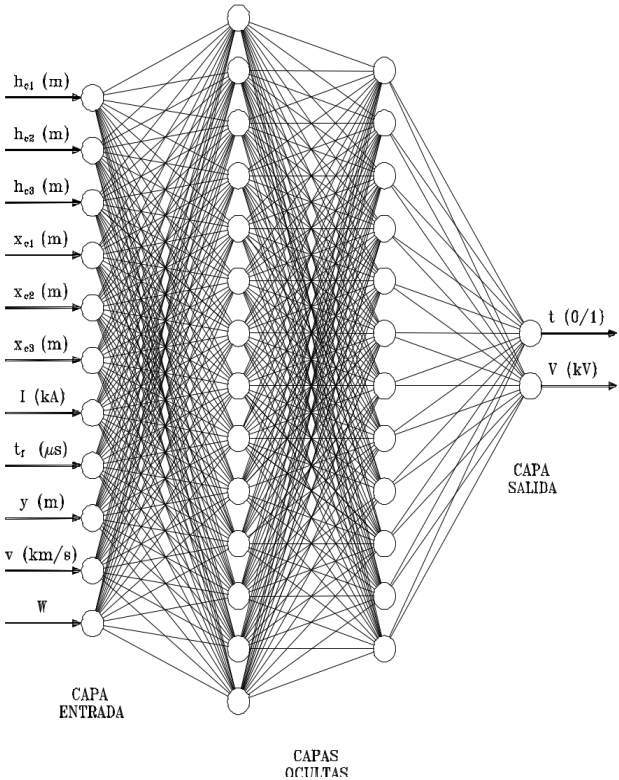


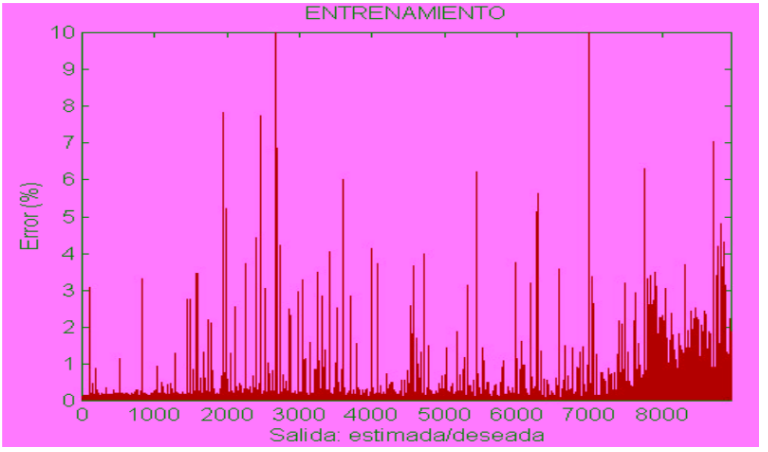
Figura 6.47. Arquitectura de la red neuronal (Método de Chowdhuri con líneas sin apantallar).

Validación

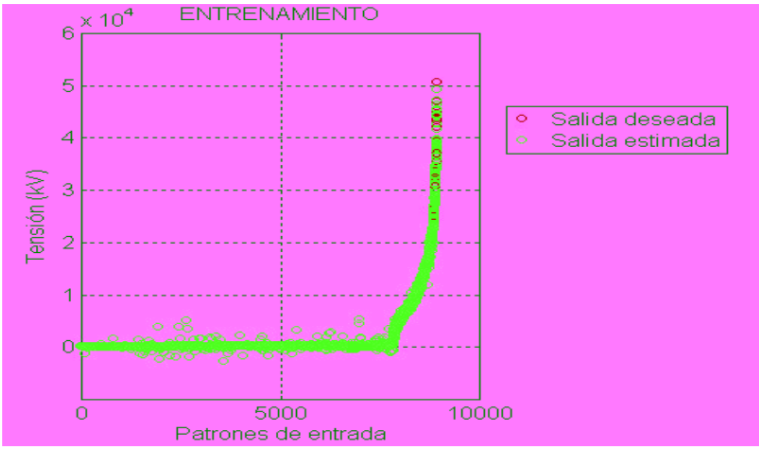
La red neuronal se ha validado con 10000 patrones que no han intervenido en el proceso de entrenamiento, con 1000 patrones para cada una de las configuraciones de línea mostradas en la Tabla 6.17. Los gráficos de las figuras 6.49, 6.50, y 6.51 muestran los resultados obtenidos para 3 de las configuraciones anteriores. Los resultados obtenidos con las otras configuraciones son similares. En general, el error cometido en el cálculo de tensiones se encuentra por debajo del 3 %, y la red neuronal apenas muestra confusión en la clasificación de descargas.

Tabla 6.17. Variables altura, distancia al eje de la línea y parámetro W.

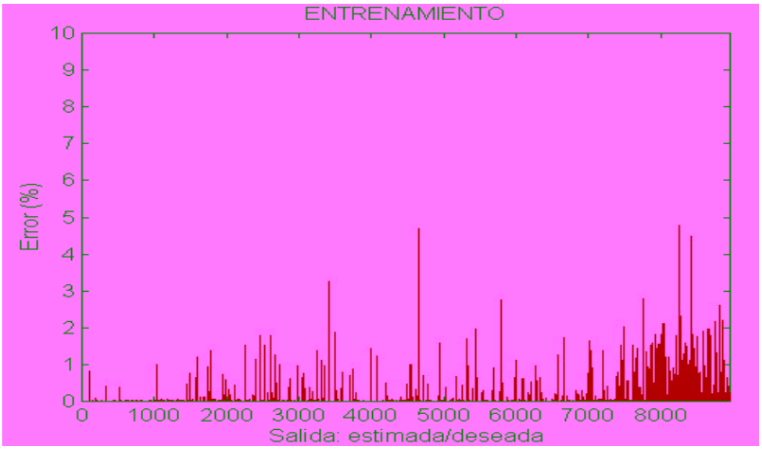
	h_{c1} (m)	h_{c2} (m)	h_{c3} (m)	x_{c1} (m)	x_{c2} (m)	x_{c3} (m)	W
Línea horizontal	5	5	5	-1.5	0	1.5	50
	10	10	10	-2	0	2	200
	10	10	10	-2	0	2	500
	17	17	17	-3	0	3	350
	17	17	17	-3	0	3	500
Línea vertical	5	7	9	0	0	0	50
	5	7	9	0	0	0	200
	10	12	14	0	0	0	500
	10	12	14	0	0	0	350
	14	16	18	0	0	0	200



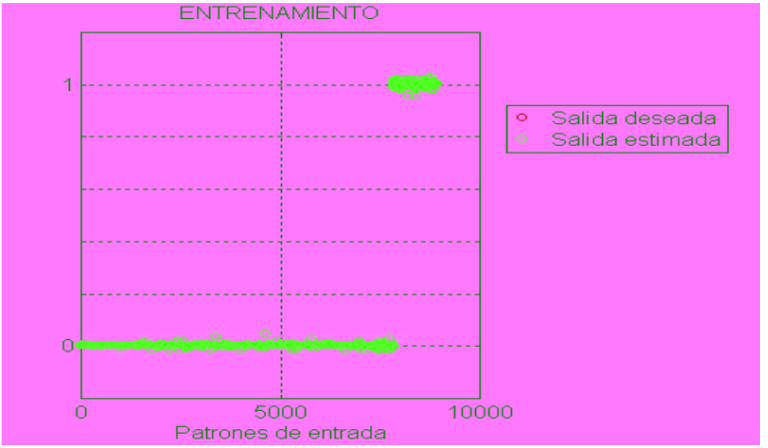
a. Error porcentual en el cálculo de sobretensiones.



b. Ajuste de las salidas (tensiones).

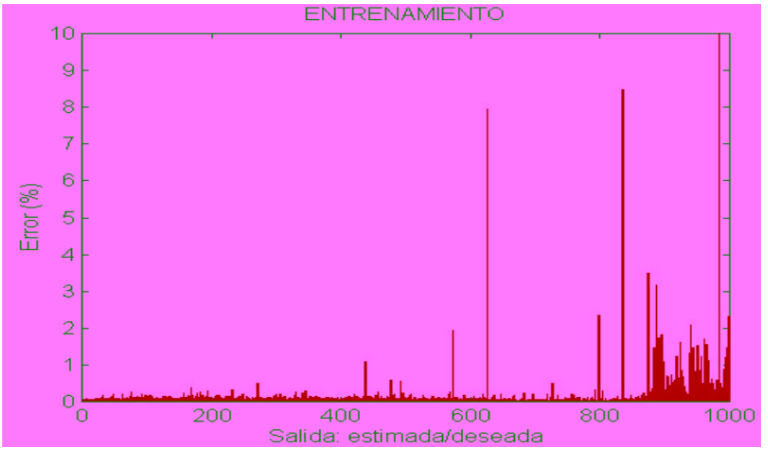


c. Error porcentual en la clasificación de descargas.

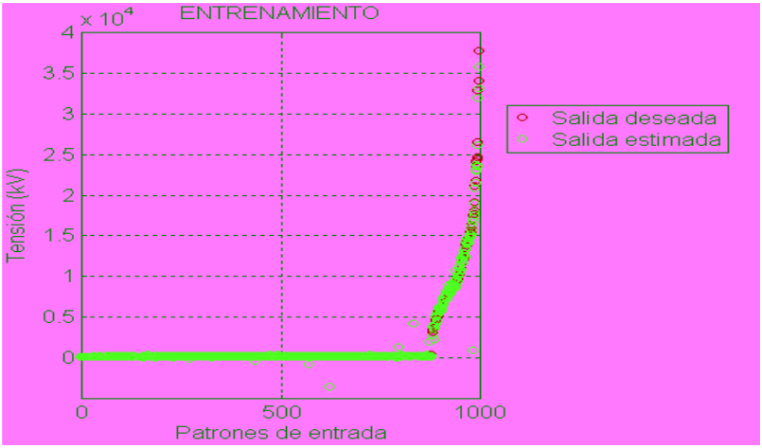


d. Ajuste del clasificador.

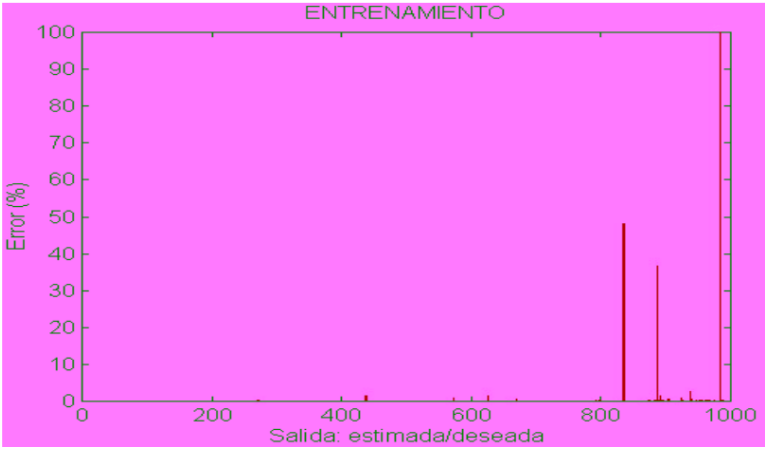
Figura 6.48. Entrenamiento de la red neuronal.



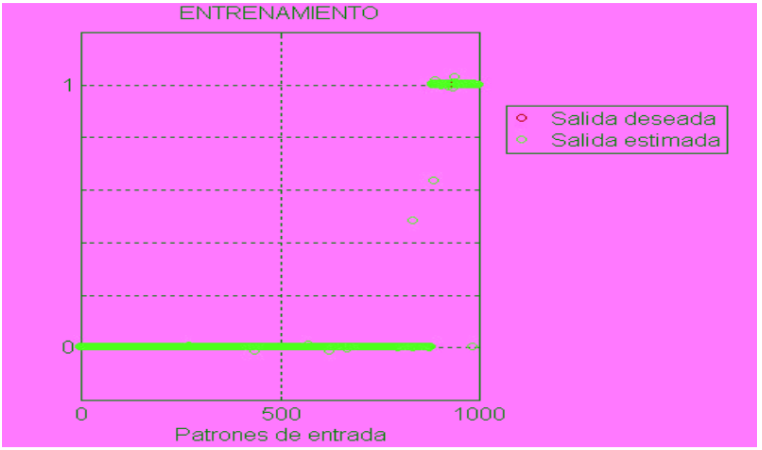
a. Error porcentual en el cálculo de sobretensiones.



b. Ajuste de las salidas (tensiones).

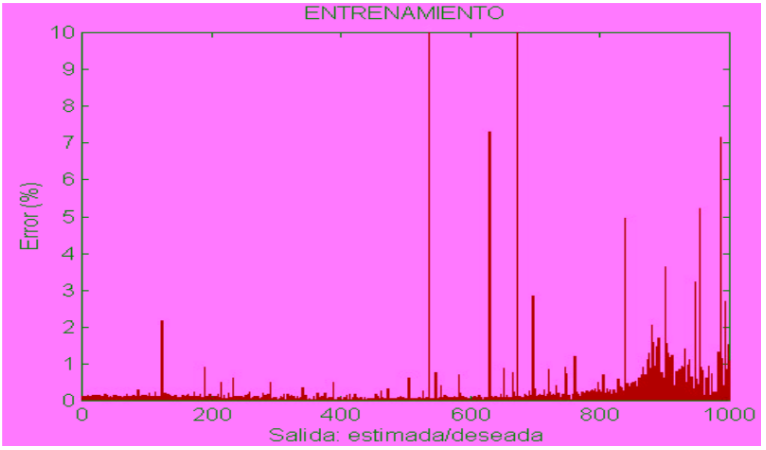


c. Error porcentual en la clasificación de descargas.

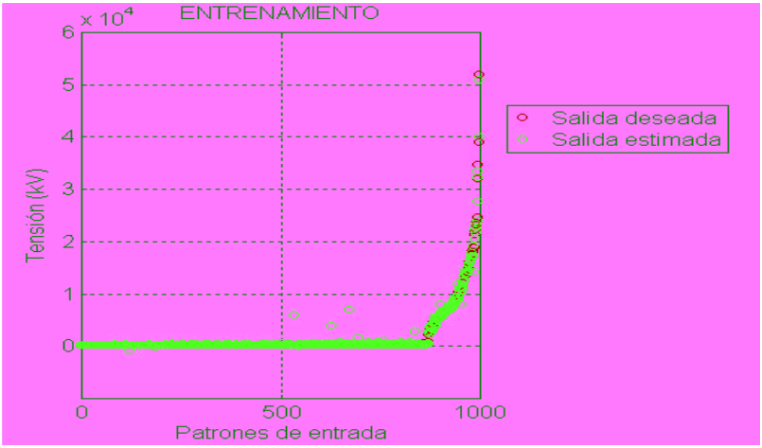


d. Ajuste del clasificador.

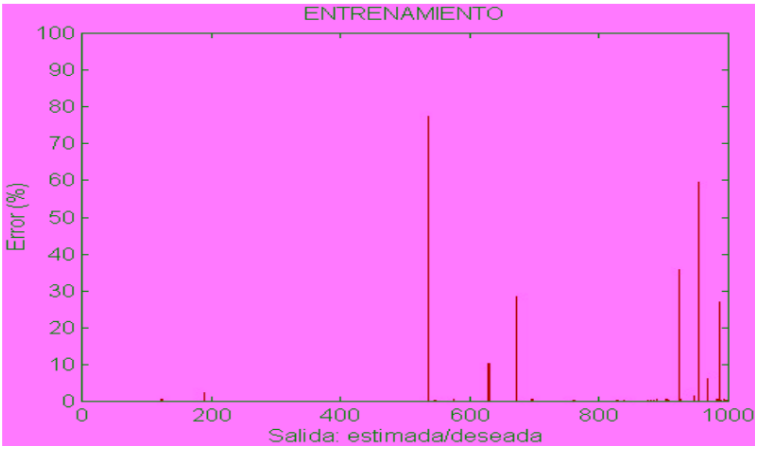
Figura 6.49. Validación de la red neuronal. Línea horizontal, altura = 10 m, distancia entre conductores más externos = 4 m, W = 200.



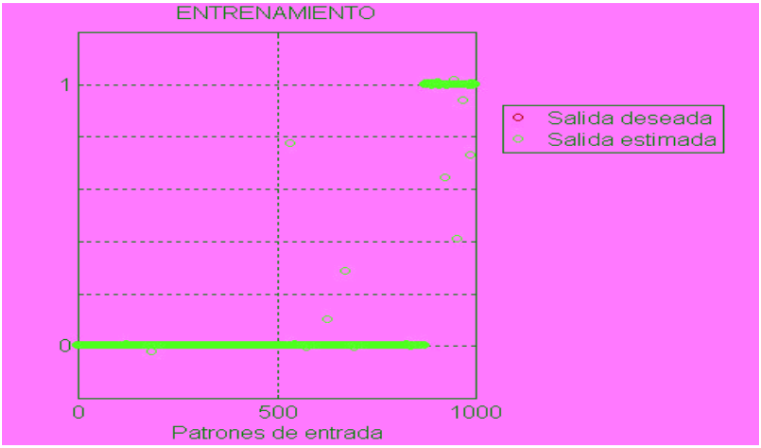
a. Error porcentual en el cálculo de sobretensiones.



b. Ajuste de las salidas (tensiones).

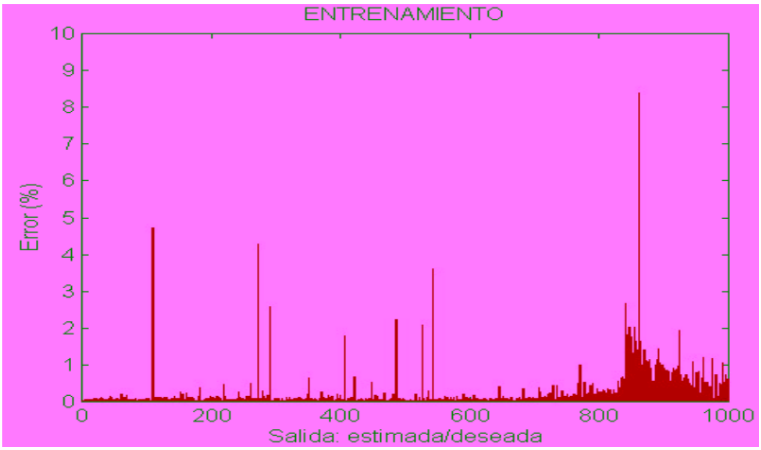


c. Error porcentual en la clasificación de descargas.

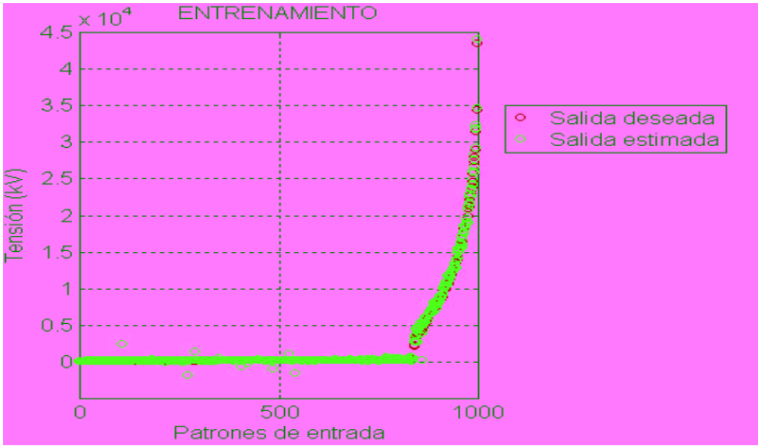


d. Ajuste del clasificador.

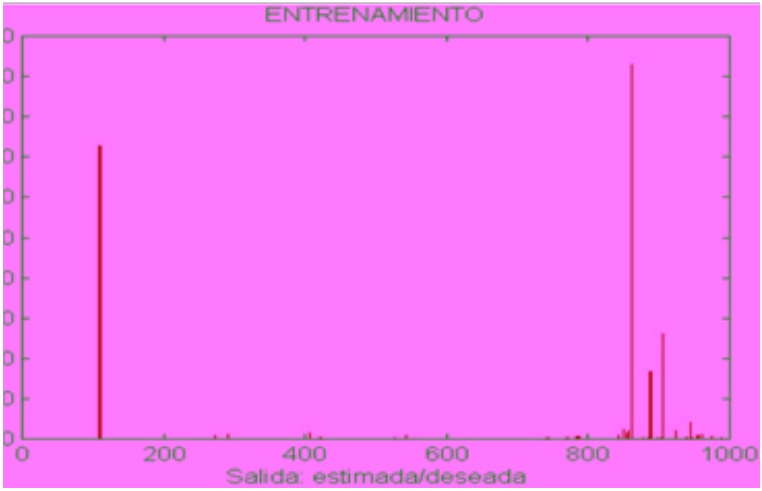
Figura 6.50. Validación de la red neuronal. Línea vertical, altura c1 =10 m, altura c2 =12 m, altura c3 =14 m, W = 500.



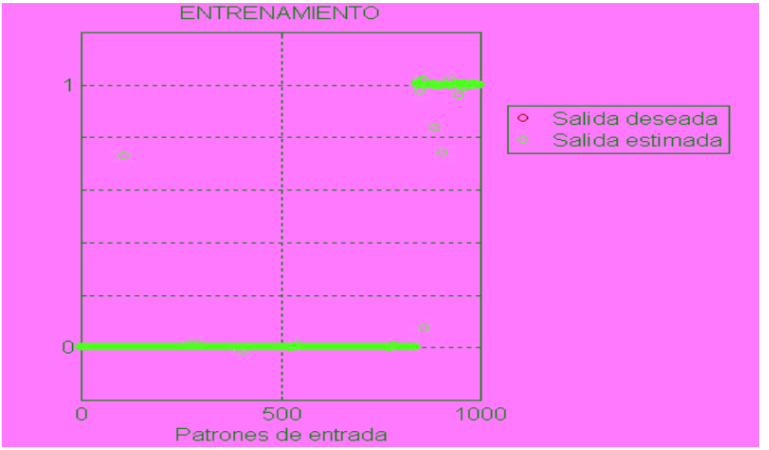
a. Error porcentual en el cálculo de sobretensiones.



b. Ajuste de las salidas (tensiones).



c. Error porcentual en la clasificación de descargas.



d. Ajuste del clasificador.

Figura 6.51. Validación de la red neuronal. Línea vertical, altura c1 =14 m, altura c2 =16 m, altura c3 =18 m, W = 200.

6.3.6.Cálculo de sobretensiones en líneas con cable de tierra. Velocidad de retorno del rayo en función de la intensidad máxima

6.3.6.1. Método de Rusck

En este estudio se han escogido las siguientes variables de entrada y salida de la red

·Entradas: Intensidad máxima de la descarga (variable I), distancia perpendicular entre la línea y la descarga (variable y), velocidad de retorno del rayo (variable v), parámetro W (variable W), altura de cada conductor de fase (variable hc1, variable hc2, variable hc3), altura del cable de tierra (variable hct), distancia al eje de cada conductor de fase (variable xc1, variable xc2, variable xc3), resistencia de puesta a tierra de los postes (variable R)

·Salidas: Tipo de descarga, directa al cable de tierra/directa al conductor

de fase/indirecta (variable t), sobretensión que aparece en la línea (variable V).

La red neuronal se ha entrenado con un total de 10800 patrones diferentes calculados de forma aleatoria. Este estudio es similar al utilizado con el método de Rusck en líneas apantalladas y distribución de velocidad de retorno del rayo uniforme, para más detalles ver el apartado 6.3.4. En este apartado el número de patrones ha sido escogido teniendo en cuenta una duración aceptable del proceso de entrenamiento. Sin embargo, es fácil darse cuenta que el número de patrones utilizados comienza a ser un problema ya que en el mismo estudio pero con distribución de velocidad uniforme se utilizaron casi 9000. La red neuronal se ha entrenado con cuatro valores diferentes del parámetro W: 50, 200, 350 y 500, los mismos que han sido utilizados para líneas sin apantallar.

Entrenamiento

Para entrenar la red neuronal se seguirá la metodología utilizada en los apartados anteriores. La Tabla 6.18 muestra una parte de los datos utilizados en el entrenamiento.

La estructura de red neuronal utilizada ha sido la siguiente

·2 capas de neuronas ocultas, con 14 y 12 neuronas respectivamente

·Funciones de transferencia: tan-sigmoidal (capas ocultas), lineal (capa de salida)

·Tasa de aprendizaje variable, $\alpha 0=0.25, \alpha i=1.01, \alpha d=0.6$

·Momento $\beta=0.9$.

El entrenamiento de esta red neuronal tiene los siguientes pasos

1)Estudio previo. Se guardan los pesos y los umbrales que dan el mínimo error.

2)Ajuste de la pendiente de las funciones de transferencia. Se mantiene un valor unidad para la pendiente de la función tan-sigmoidal, y un valor de 0.3 para la función lineal.

3)Se entrena la red neuronal con la estructura comentada anteriormente y con las pendientes ajustadas hasta que el error se estabiliza. La figura 6.52 muestra la arquitectura de la red, y la figura 6.53 muestra los resultados del entrenamiento obtenidos con esta red. El error cometido en las dos salidas, clasificador y cálculo de tensiones, se encuentra acotado por debajo del 10 %.

Tabla 6.18. Datos de entrenamiento.

ENTRADAS												SALIDAS	
Altura conductor 1 (m)	Altura conductor 2 (m)	Altura conductor 3 (m)	Altura cable de tierra (m)	Distancia al eje conductor 1 (m)	Distancia al eje conductor 2 (m)	Distancia al eje conductor 3 (m)	Resistencia puesta a tierra (Ω)	Intensidad (kA)	Distancia descarga-línea (m)	Velocidad (km/s)	W	Tipo descarga	Tensión (kV)
5	5	5	6.5	-1.0	0.0	1.0	10	27	131	177647	50	0	31.54
5	5	5	6.5	-1.0	0.0	1.0	10	23	147	168393	50	0	23.47
5	5	5	6.5	-1.0	0.0	1.0	10	27	214	177647	50	0	19.26
8	8	8	11.0	-1.0	0.0	1.0	10	24	127	75251	350	0	38.14
8	8	8	11.0	-1.0	0.0	1.0	10	35	305	90453	350	0	24.38
8	8	8	11.0	-1.0	0.0	1.0	10	80	266	129399	350	0	69.35
11	11	11	14.0	-1.0	0.0	1.0	70	33	464	112902	200	0	22.96
11	11	11	14.0	-1.0	0.0	1.0	70	33	428	112163	200	0	24.44
11	11	11	14.0	-1.0	0.0	1.0	70	36	188	116477	200	0	61.38
14	14	14	15.5	-3.0	0.0	3.0	10	16	386	62725	350	0	13.22
14	14	14	15.5	-3.0	0.0	3.0	10	15	30	59835	350	2	83.41
14	14	14	15.5	-3.0	0.0	3.0	10	20	239	69749	350	0	27.14
14	14	14	15.5	-3.0	0.0	3.0	10	31	84	85574	350	0	123.92
17	17	17	20.0	-3.0	0.0	3.0	10	29	53	180763	50	2	166.63
17	17	17	20.0	-3.0	0.0	3.0	10	47	171	208826	50	0	154.07
17	17	17	20.0	-3.0	0.0	3.0	10	82	221	236177	50	0	220.55
5	7	9	12.0	0.5	0.5	0.5	70	15	233	79241	200	0	15.95
5	7	9	12.0	0.5	0.5	0.5	70	25	411	99105	200	0	15.38
5	7	9	12.0	0.5	0.5	0.5	70	44	372	127395	200	0	32.39
8	10	12	15.0	1.5	1.5	1.5	40	31	157	72486	500	0	61.87
8	10	12	15.0	1.5	1.5	1.5	40	23	23	62254	500	2	326.95
8	10	12	15.0	1.5	1.5	1.5	40	20	27	58835	500	2	290.62
11	13	15	16.5	1.5	1.5	1.5	70	45	352	206474	50	0	64.03
11	13	15	16.5	1.5	1.5	1.5	70	46	304	207666	50	0	76.00
11	13	15	16.5	1.5	1.5	1.5	70	28	235	179743	50	0	56.35
14	16	18	21.0	1.5	1.5	1.5	10	60	146	98198	500	0	192.08
14	16	18	21.0	1.5	1.5	1.5	10	7	342	33985	500	0	7.78
14	16	18	21.0	1.5	1.5	1.5	10	75	358	108347	500	0	100.29

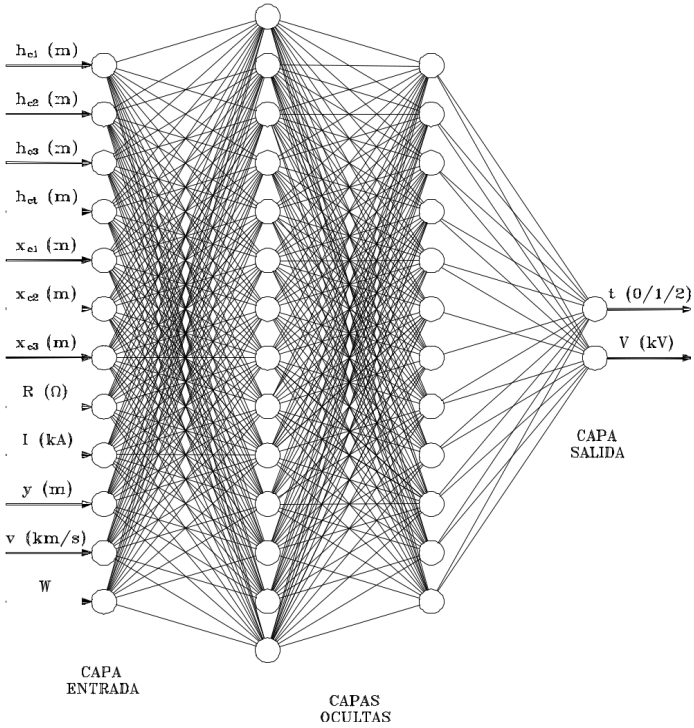


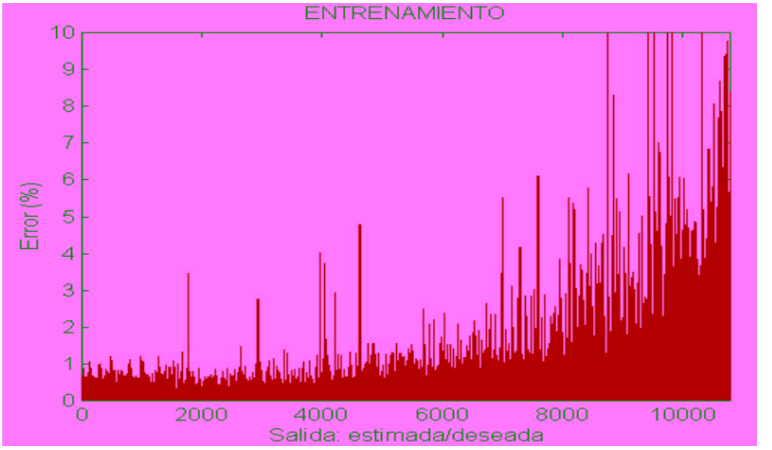
Figura 6.52. Arquitectura de la red neuronal (Método de Rusck con líneas apantalladas).

Validación

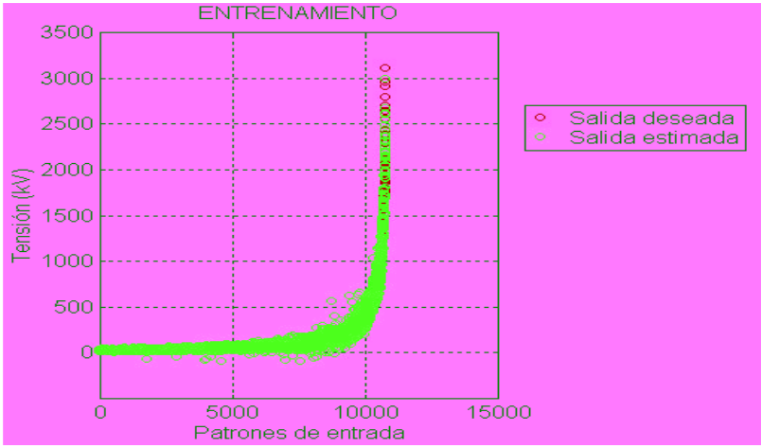
La red neuronal se ha validado con 10000 patrones que no han intervenido en el proceso de entrenamiento, con 1000 patrones para cada una de las configuraciones de línea mostradas en la Tabla 6.19. Los gráficos de las figuras 6.54, 6.55, y 6.56 muestran los resultados obtenidos para 3 de las configuraciones anteriores. Los resultados obtenidos con las otras configuraciones son similares. Al igual que en el entrenamiento, en general los errores correspondientes a ambas salidas de la red neuronal, clasificador y cálculo de tensiones, se encuentran por debajo del 10 %.

Tabla 6.19. Variables altura de los conductores, altura del cable de tierra, distancia al eje de los conductores, resistencia de puesta a tierra y parámetro W.

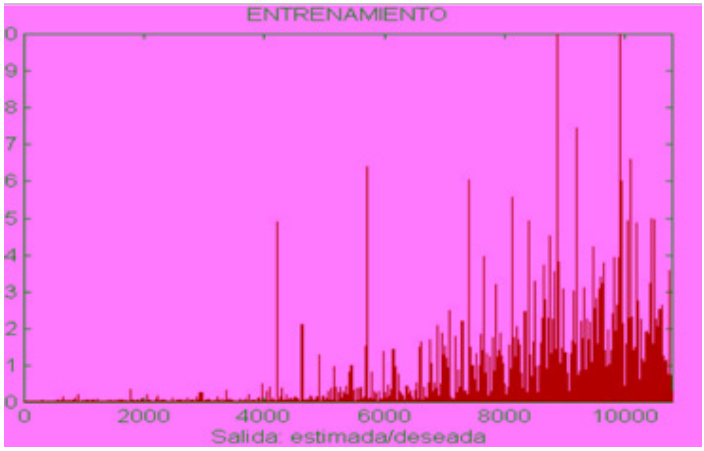
	h_{c1} (m)	h_{c2} (m)	h_{c3} (m)	h_{ct} (m)	x_{c1} (m)	x_{c2} (m)	x_{c3} (m)	R (Ω)	W
Línea horizontal	5	5	5	6.5	-1.5	0	1.5	30	50
	10	10	10	11.5	-2	0	2	50	200
	10	10	10	11.5	-2	0	2	50	500
	17	17	17	18.5	-3	0	3	10	350
	17	17	17	18.5	-3	0	3	10	500
Línea vertical	5	7	9	10.5	0	0	0	20	50
	5	7	9	10.5	0	0	0	20	200
	10	12	14	15.5	0	0	0	50	500
	10	12	14	15.5	0	0	0	50	350
	14	16	18	19.5	0	0	0	70	200



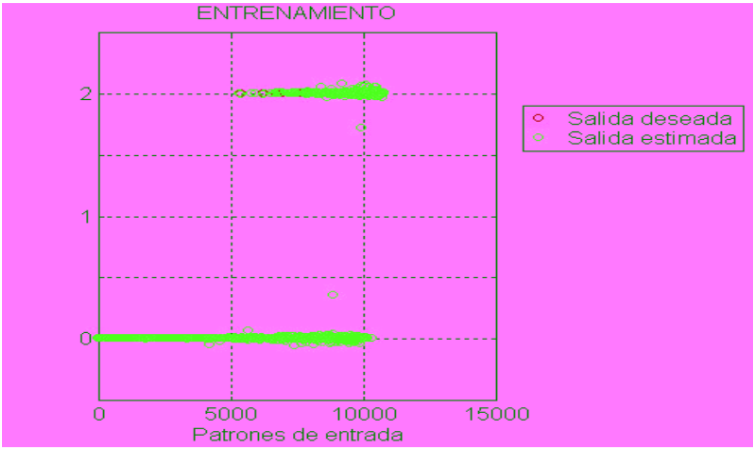
a. Error porcentual en el cálculo de sobretensiones.



b. Ajuste de las salidas (tensiones).

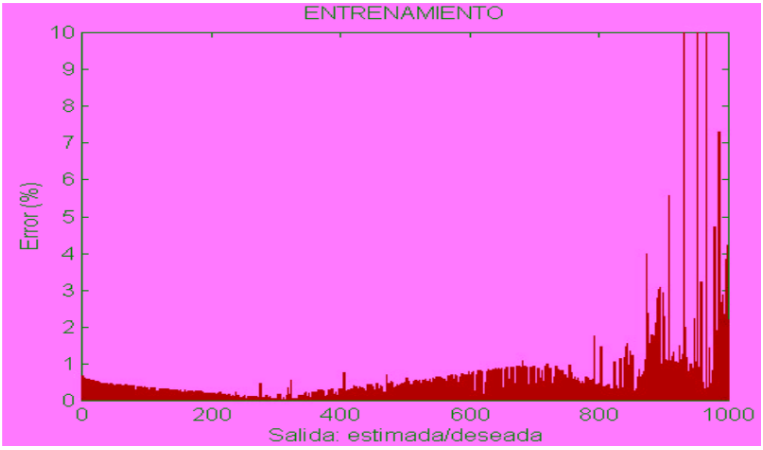


c. Error porcentual en la clasificación de descargas.

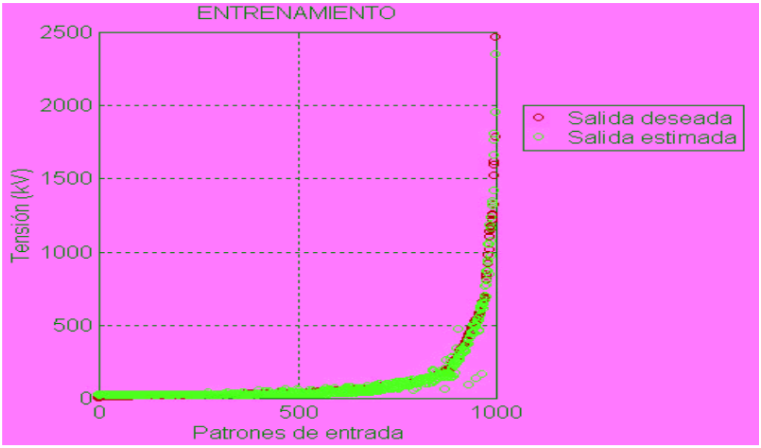


d. Ajuste del clasificador.

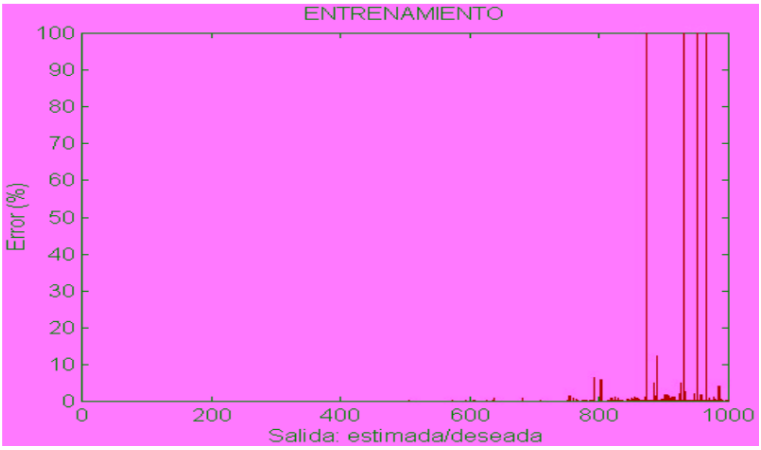
Figura 6.53. Entrenamiento de la red neuronal.



a. Error porcentual en el cálculo de sobretensiones.



b. Ajuste de las salidas (tensiones).

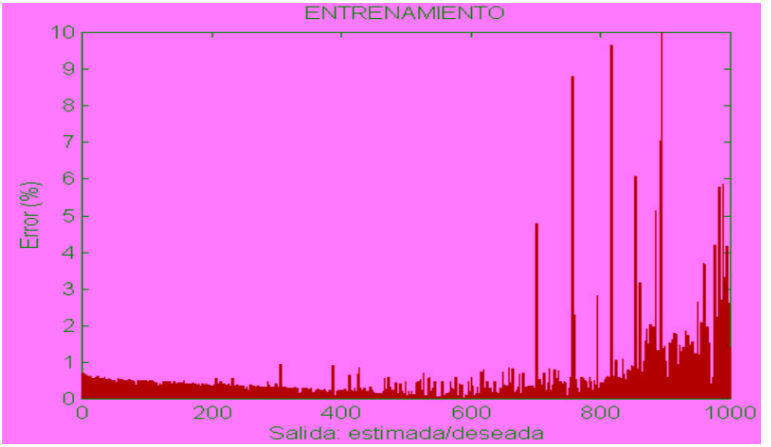


c. Error porcentual en la clasificación de descargas.

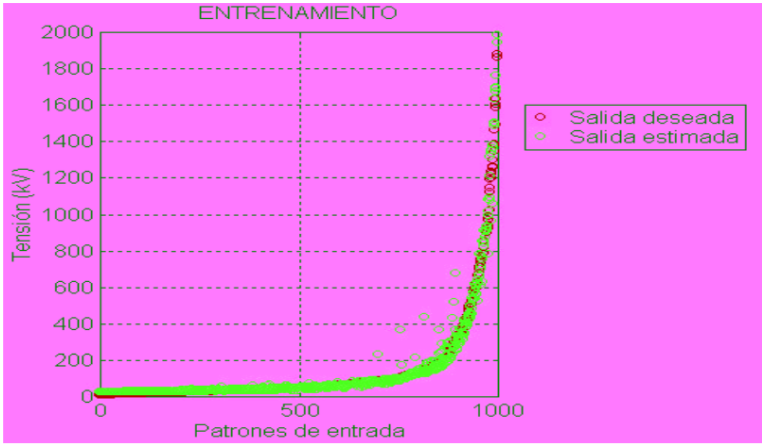
d. Ajuste del clasificador.

Figura 6.54. Validación de la red neuronal. Línea horizontal, altura conductores = 10 m, distancia entre conductores más externos = 4 m,

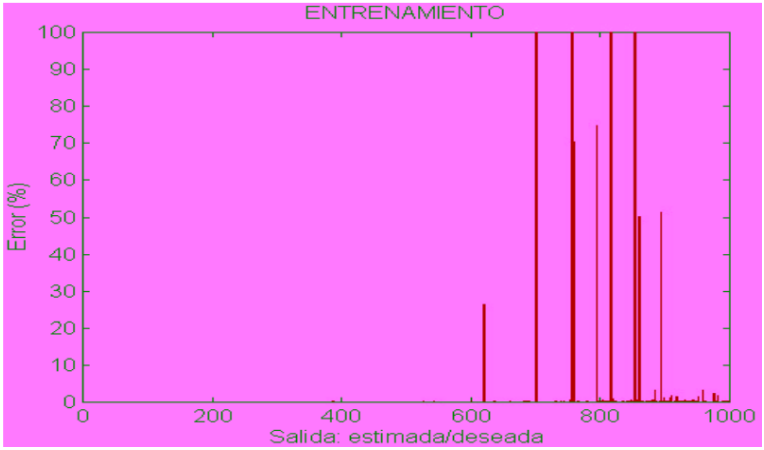
altura cable de tierra = 11.5 m, resistencia de puesta a tierra = 50 Ω, W = 200.



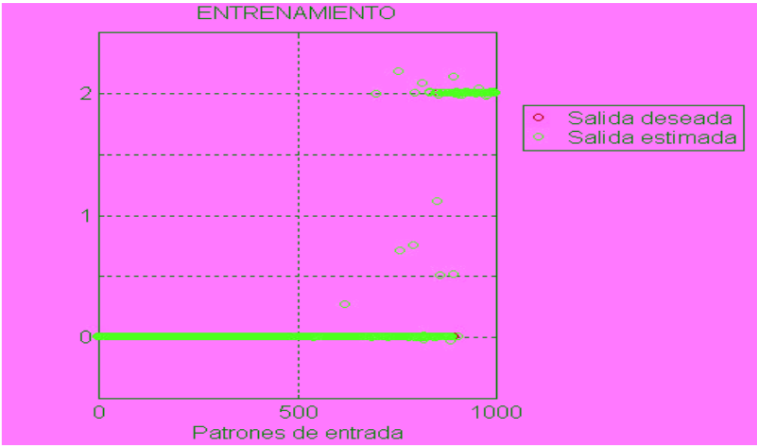
a. Error porcentual en el cálculo de sobretensiones.



b. Ajuste de las salidas (tensiones).

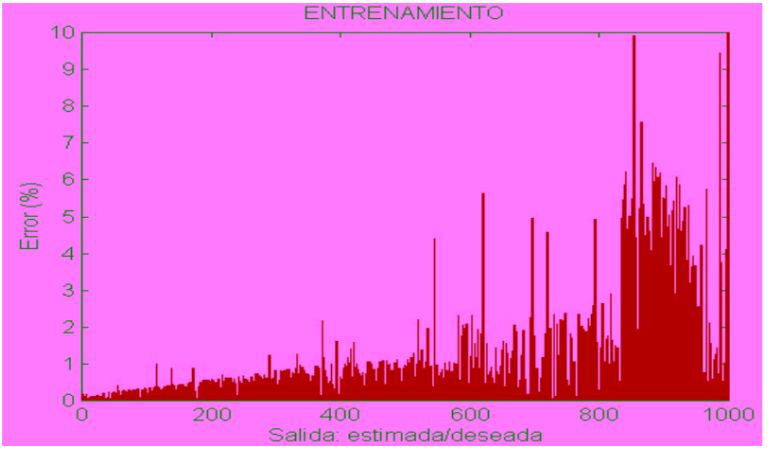


c. Error porcentual en la clasificación de descargas.



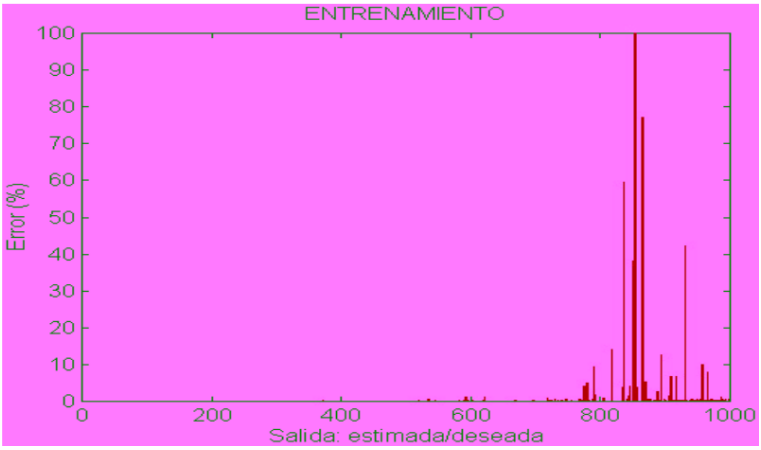
d. Ajuste del clasificador.

Figura 6.55. Validación de la red neuronal. Línea vertical, altura c1 =10 m, altura c2 =12 m, altura c3 =14 m, distancia horizontal conductores - cable de tierra = 1 m, altura cable de tierra = 15.5 m, resistencia de puesta a tierra = 50 Ω, W = 500.

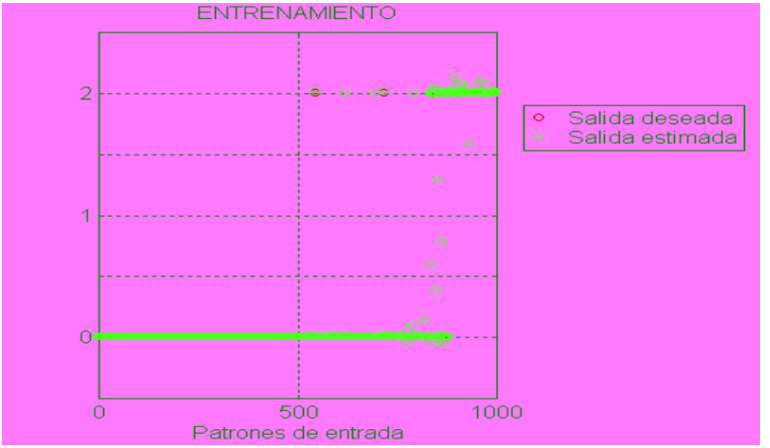


a. Error porcentual en el cálculo de sobretensiones.

b. Ajuste de las salidas (tensiones).



c. Error porcentual en la clasificación de descargas.



d. Ajuste del clasificador.

Figura 6.56. Validación de la red neuronal. Línea vertical, altura c1 =14 m, altura c2 =16 m, altura c3 =18 m, distancia horizontal conductores - cable de tierra = 1.5 m, altura cable de tierra = 19.5 m, resistencia de puesta a tierra = 70 Ω, W = 200.

6.3.6.2. Método de Chowdhuri

En este estudio se han escogido las siguientes variables de entrada y salida de la red

·Entradas: Intensidad máxima de la descarga (variable I), tiempo de frente de la onda de corriente del rayo (variable tf), distancia perpendicular entre la línea y la descarga (variable y), velocidad de retorno del rayo (variable v), parámetro W (variable W), altura de cada conductor de fase (variable hc1, variable hc2, variable hc3), altura del cable de tierra (variable hct), distancia al eje de cada conductor de fase (variable xc1, variable xc2, variable xc3), resistencia de puesta a tierra de los postes (variable R)

·Salidas: Tipo de descarga, directa al cable de tierra/directa al conductor de fase/indirecta (variable t), sobretensión que aparece en la línea (variable V).

La red neuronal se ha entrenado con un total de 10800 patrones diferentes calculados de forma aleatoria. Este estudio es similar al utilizado con el método de Chowdhuri en líneas apantalladas y distribución de velocidad de retorno del rayo uniforme, para más detalles ver el apartado 6.3.4, la única diferencia es que se ha añadido una nueva entrada, la correspondiente al parámetro W. La red neuronal se ha entrenado con cuatro valores diferentes del parámetro W: 50, 200, 350 y 500.

Entrenamiento

Para entrenar la red neuronal se seguirá la metodología utilizada en los apartados anteriores. La Tabla 6.20 muestra una parte de los datos utilizados en el entrenamiento.

La estructura de red neuronal utilizada ha sido la siguiente

·2 capas de neuronas ocultas, con 14 y 12 neuronas respectivamente

·Funciones de transferencia: tan-sigmoidal (capas ocultas), lineal (capa de salida)

·Tasa de aprendizaje variable, α 0=0.25, α i=1.01, α d=0.6

·Momento β=0.9.

El entrenamiento de esta red neuronal tiene los siguientes pasos

1)Estudio previo. Se guardan los pesos y los umbrales que dan el mínimo error.

2)Ajuste de la pendiente de las funciones de transferencia. Se mantiene un valor unidad para la pendiente de la función tan-sigmoidal, y un valor de

0.3 para la función lineal.

3)Se entrena la red neuronal con la estructura comentada anteriormente y con las pendientes ajustadas hasta que el error se estabiliza. La figura 6.57 muestra la arquitectura de la red, y la figura 6.58 muestra los resultados del entrenamiento obtenidos con esta red. Se puede observar que el error cometido por el clasificador se encuentra por debajo del 5 %, y que el error cometido en el cálculo de tensiones es inferior al 10 %.

Tabla 6.20. Datos de entrenamiento.

ENTRADAS													SALIDAS	
Altura conductor 1 (m)	Altura conductor 2 (m)	Altura conductor 3 (m)	Altura cable de tierra (m)	Distancia al eje conductor 1 (m)	Distancia al eje conductor 2 (m)	Distancia al eje conductor 3 (m)	Resistencia puesta a tierra (Ω)	Intensidad (kA)	Tiempo de frente (μs)	Distancia descarga - línea (m)	Velocidad (km/s)	W	Tipo descarga	Tensión (kV)
5	5	5	6.5	-3.0	0.0	3.0	70	29	2.5	358	105950	200	0	25.02
5	5	5	6.5	-3.0	0.0	3.0	70	17	1.5	310	82820	200	0	35.88
5	5	5	6.5	-3.0	0.0	3.0	70	37	3.5	49	118535	200	2	636.78
8	8	8	11.0	-1.0	0.0	1.0	70	49	5.0	219	210511	50	0	7.41
8	8	8	11.0	-1.0	0.0	1.0	70	32	4.5	469	186508	50	0	3.89
8	8	8	11.0	-1.0	0.0	1.0	70	38	2.5	58	196396	50	2	627.44
11	11	11	12.5	-3.0	0.0	3.0	40	161	3.0	393	168214	350	0	67.64
11	11	11	12.5	-3.0	0.0	3.0	40	32	4.0	268	86204	350	0	57.96
11	11	11	12.5	-3.0	0.0	3.0	40	15	4.0	59	60816	350	0	72.16
14	14	14	17.0	-3.0	0.0	3.0	40	32	4.0	425	86204	350	0	59.62
14	14	14	17.0	-3.0	0.0	3.0	40	51	4.0	266	106528	350	0	83.08
14	14	14	17.0	-3.0	0.0	3.0	40	17	8.5	187	63654	350	0	32.74
14	14	14	17.0	-3.0	0.0	3.0	40	26	8.5	11	78889	350	2	343.69
17	17	17	20.0	-3.0	0.0	3.0	70	13	4.5	151	136277	50	0	17.61
17	17	17	20.0	-3.0	0.0	3.0	70	9	3.0	475	114354	50	0	14.12
17	17	17	20.0	-3.0	0.0	3.0	70	43	4.5	436	203350	50	0	10.06
5	7	9	12.0	0.5	0.5	0.5	10	38	6.0	223	93885	350	0	32.32
5	7	9	12.0	0.5	0.5	0.5	10	10	3.0	305	50000	350	0	36.38
5	7	9	12.0	0.5	0.5	0.5	10	48	3.0	251	103705	350	0	64.84
8	10	12	15.0	1.5	1.5	1.5	40	18	6.0	365	85096	200	0	18.93
8	10	12	15.0	1.5	1.5	1.5	40	34	3.0	310	114354	200	0	47.03
8	10	12	15.0	1.5	1.5	1.5	40	23	1.5	207	95400	200	0	101.39
11	13	15	16.5	0.5	0.5	0.5	10	36	5.5	360	77242	500	0	57.04
11	13	15	16.5	0.5	0.5	0.5	10	25	4.0	83	64838	500	0	114.29
11	13	15	16.5	0.5	0.5	0.5	10	53	5.5	317	92477	500	0	66.89
14	16	18	19.5	1.5	1.5	1.5	40	22	2.0	314	165831	50	0	24.67
14	16	18	19.5	1.5	1.5	1.5	40	25	3.5	266	172039	50	0	15.37
14	16	18	19.5	1.5	1.5	1.5	40	64	5.5	466	224393	50	0	11.25

Validación

Para validar esta red neuronal se ha seguido un procedimiento similar al utilizado con el método de Rusck. La red neuronal se ha validado con 10000 patrones que no han intervenido en el proceso de entrenamiento, distribuyendo 1000 para cada una de las configuraciones de línea mostradas en la Tabla 6.19. Los gráficos de las figuras 6.59, 6.60, y 6.61 muestran los resultados obtenidos para 3 de las configuraciones anteriores. Los resultados obtenidos con las otras configuraciones son similares. Los resultados de la validación muestran que los errores correspondientes a ambas salidas de la red neuronal, clasificador y cálculo de tensiones, en general se encuentran acotados por debajo del 10 %.

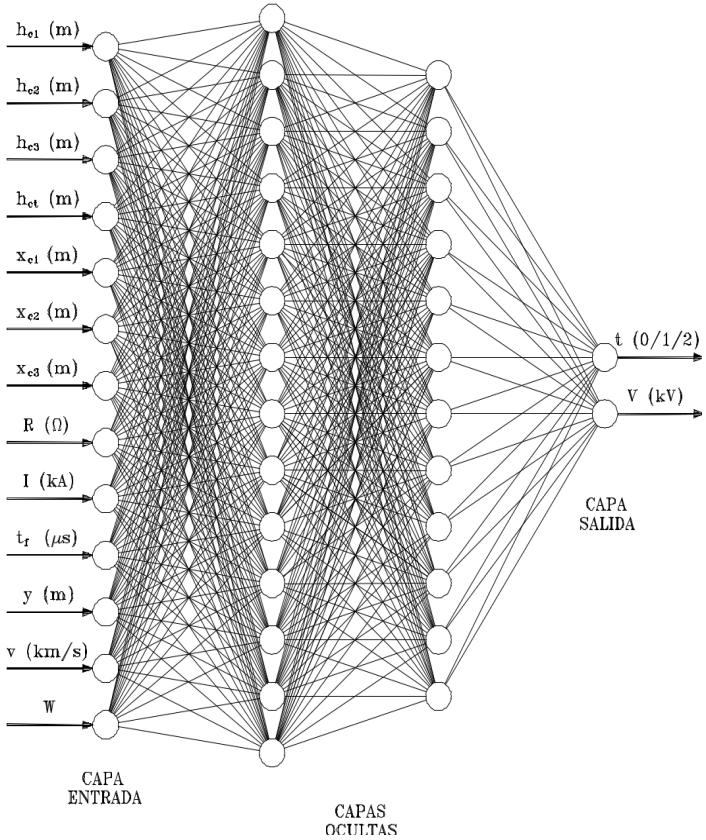
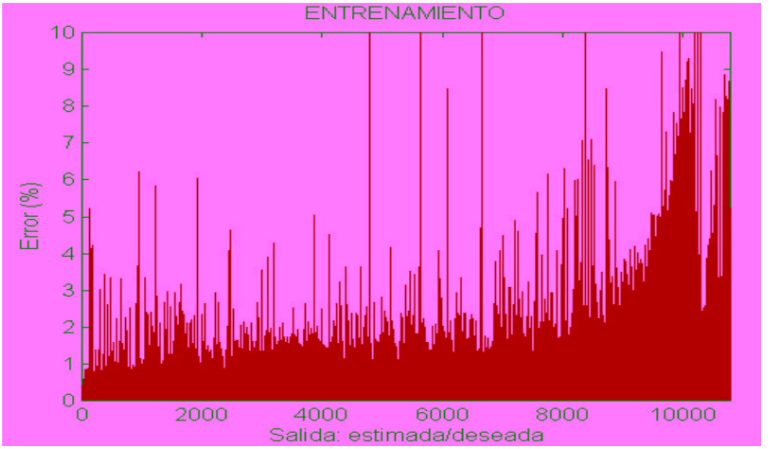
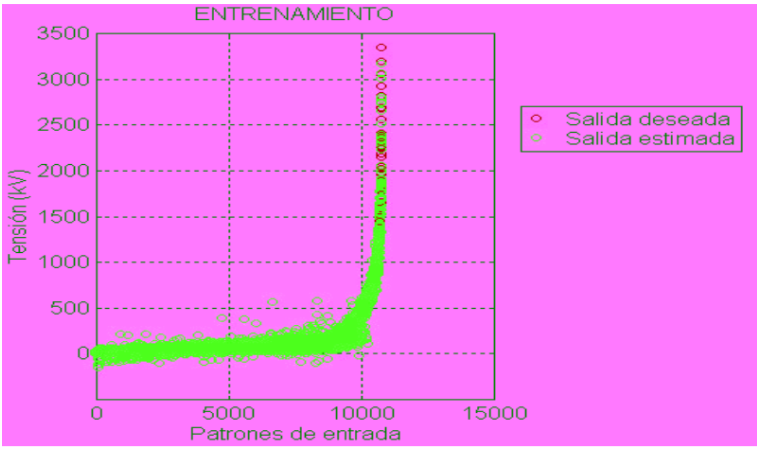


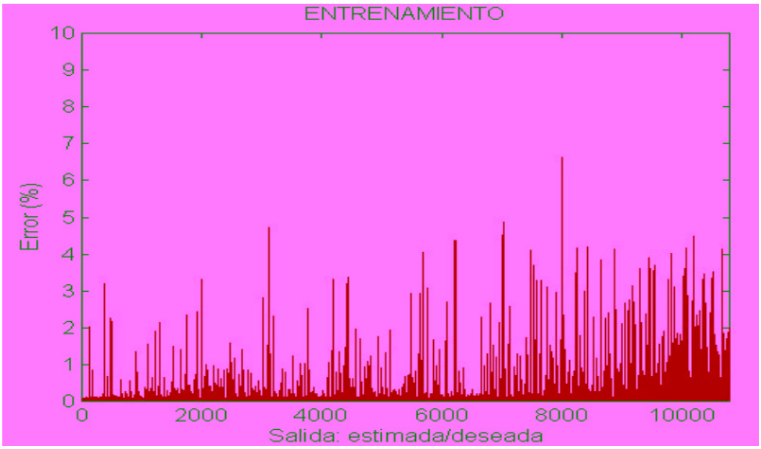
Figura 6.57. Arquitectura de la red neuronal (Método de Chowdhuri con líneas apantalladas).



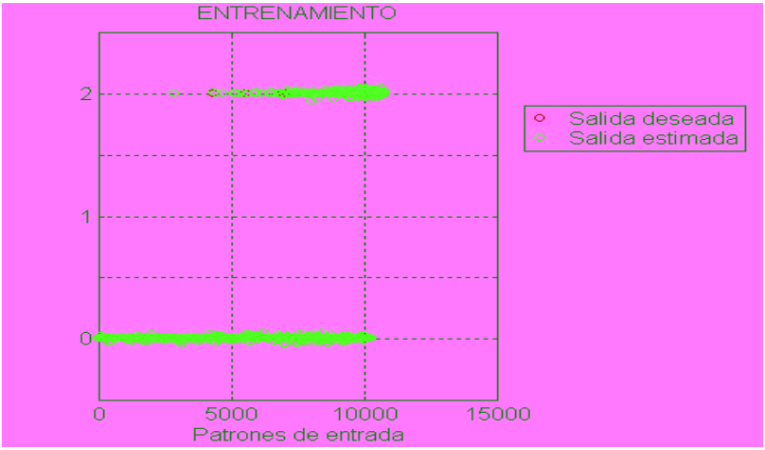
a. Error porcentual en el cálculo de sobretensiones.



b. Ajuste de las salidas (tensiones).

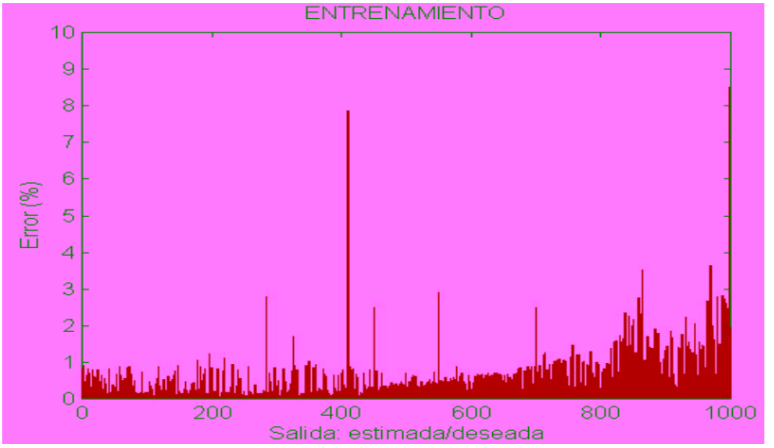


c. Error porcentual en la clasificación de descargas.

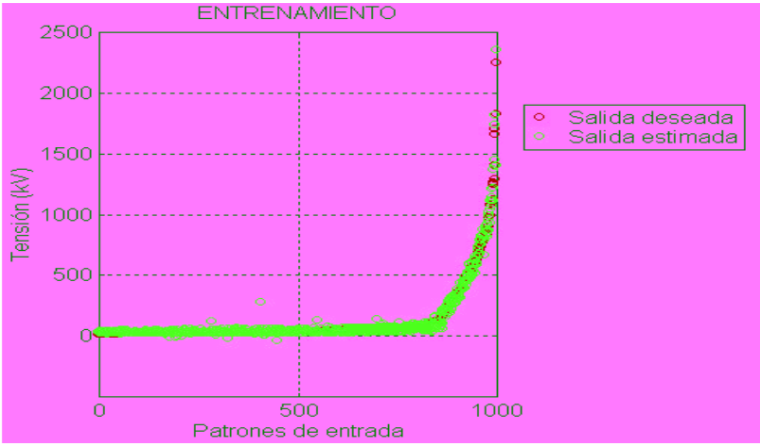


d. Ajuste del clasificador.

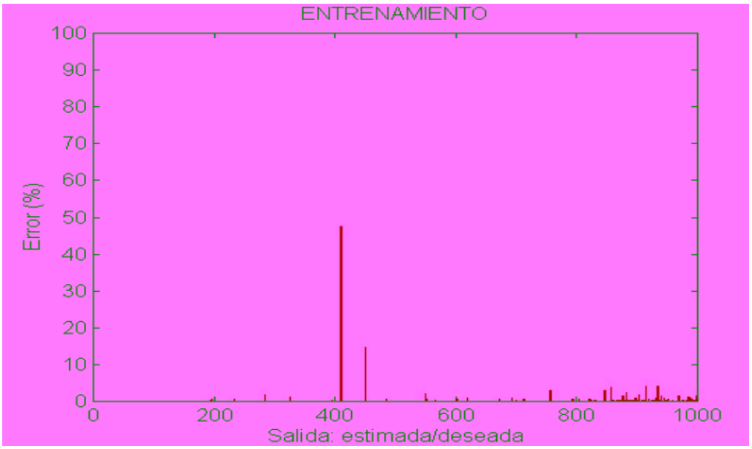
Figura 6.58. Entrenamiento de la red neuronal.



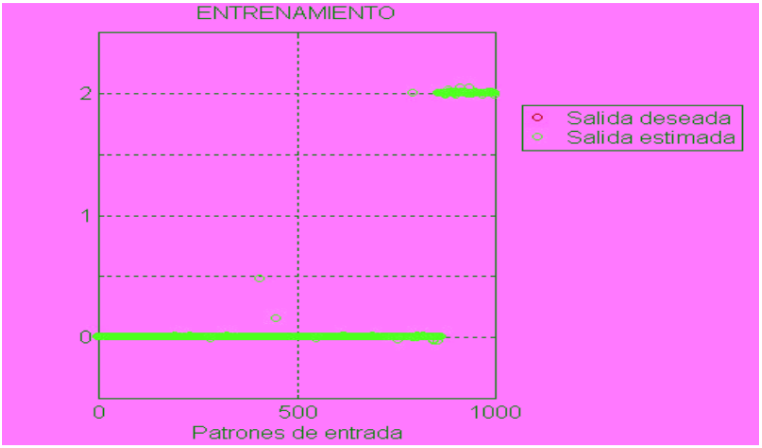
a. Error porcentual en el cálculo de sobretensiones.



b. Ajuste de las salidas (tensiones).



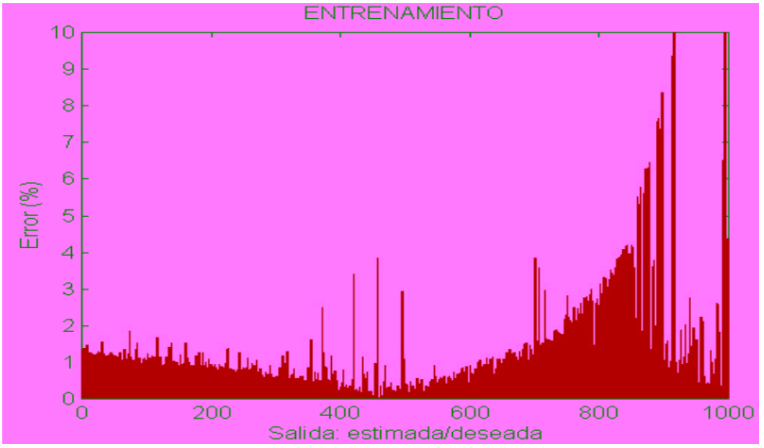
c. Error porcentual en la clasificación de descargas.



d. Ajuste del clasificador.

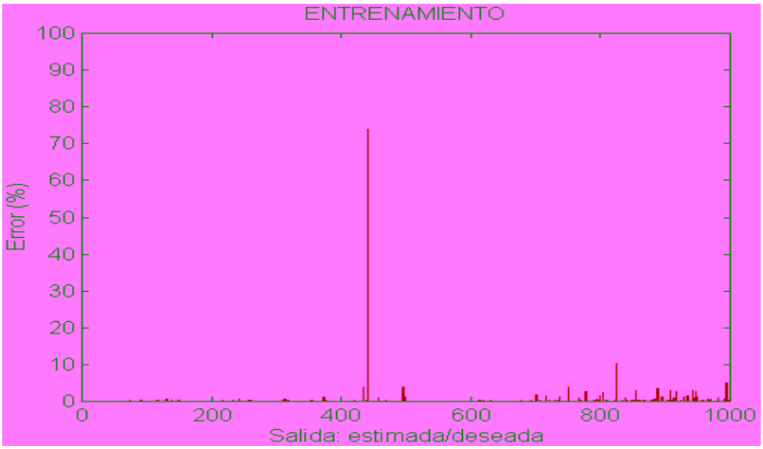
Figura 6.59. Validación de la red neuronal. Línea horizontal, altura conductores = 10 m, distancia entre conductores más externos = 4 m,

altura cable de tierra = 11.5 m, resistencia de puesta a tierra = 50Ω, W = 200.



a. Error porcentual en el cálculo de sobretensiones.

b. Ajuste de las salidas (tensiones).



c. Error porcentual en la clasificación de descargas.

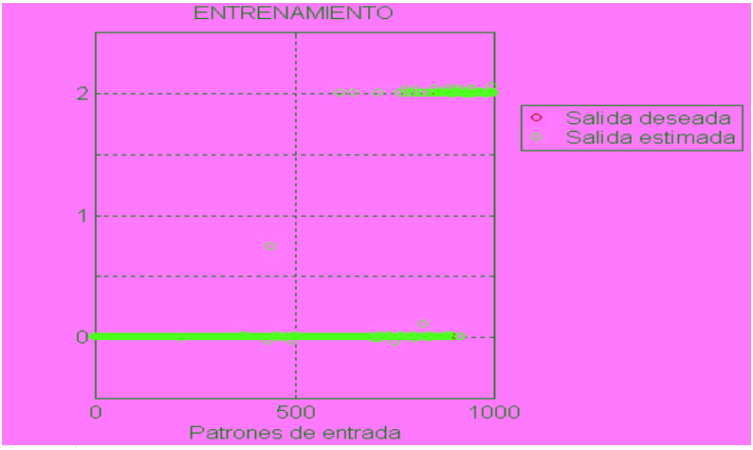
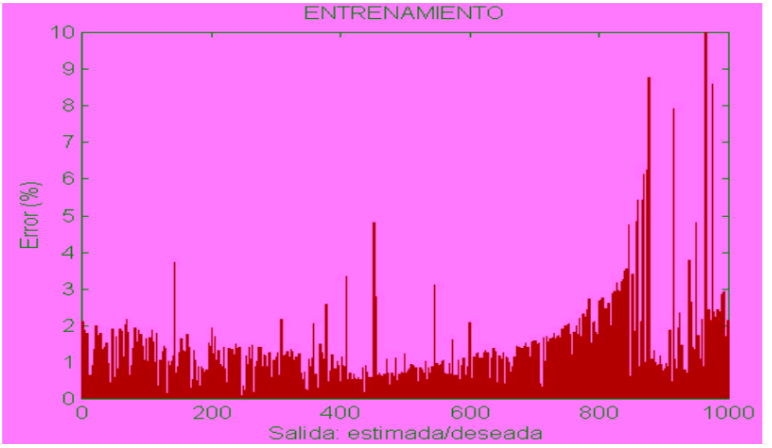
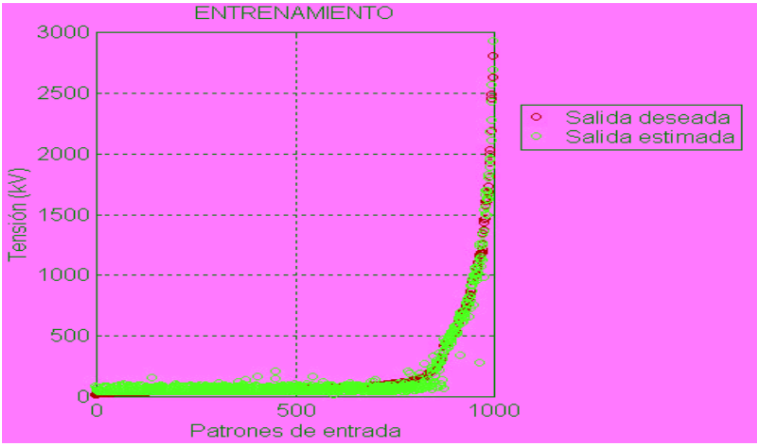


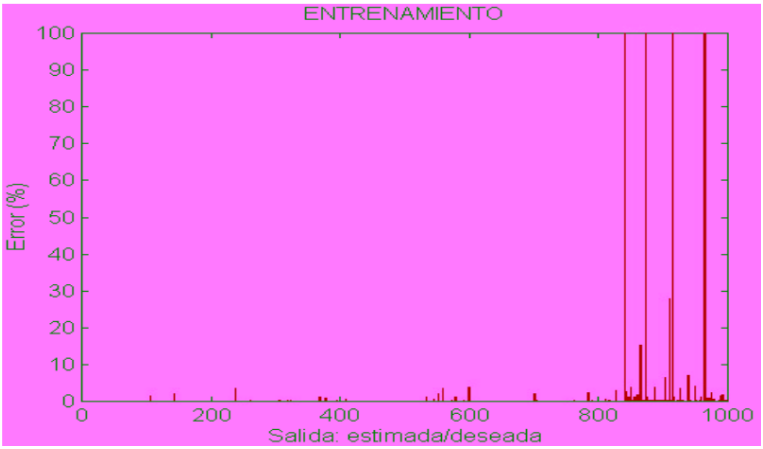
Figura 6.60. Validación de la red neuronal. Línea vertical, altura c1 =10 m, altura c2 =12 m, altura c3 =14 m, distancia horizontal conductores - cable de tierra = 1 m, altura cable de tierra = 15.5 m, resistencia de puesta a tierra = 50Ω, W = 500.



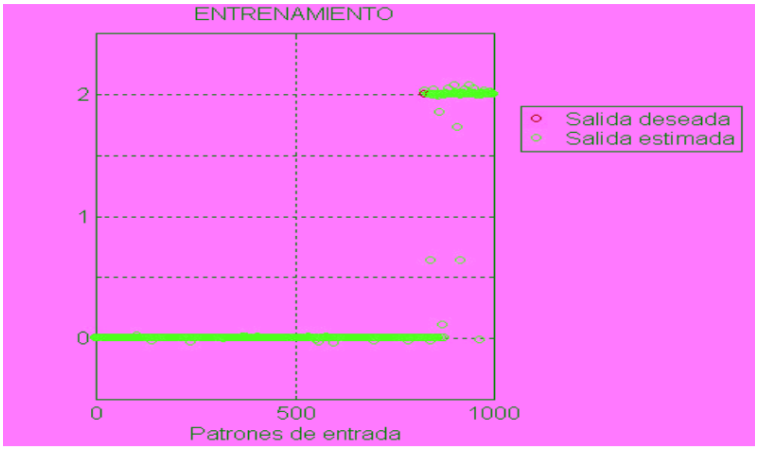
a. Error porcentual en el cálculo de sobretensiones.



b. Ajuste de las salidas (tensiones).



c. Error porcentual en la clasificación de descargas.



d. Ajuste del clasificador.

Figura 6.61. Validación de la red neuronal. Línea vertical, altura c1 =14 m, altura c2 =16 m, altura c3 =18 m, distancia horizontal conductores - cable de tierra = 1.5 m, altura cable de tierra = 19.5 m, resistencia de puesta a tierra = 70 Ω , W = 200.

El libro de "Computación Inteligente y Estados Emocionales" trata sobre los aspectos emocionales que condicionan la conducta humana. En él se toman en cuenta las características sociales, del entorno y además aquellas características propias del individuo como la crianza, las situaciones familiares o las actitudes propias de la personalidad. En este libro los autores reflejan cómo la computación inteligente es capaz de intervenir en el análisis de las emociones humanas.

Este libro ha sido realizado con la finalidad de brindar un aporte a la academia y a aquellos apasionados de las neurociencias y a las aplicaciones médicas como herramientas para el mejoramiento de los procesos. Los autores esperamos que este libro sea de ayuda para investigaciones y revisiones bibliográficas en las áreas consideradas.

ISBN: 978-9942-35-975-9