UNIVERSIDAD DON BOSCO

FACULTAD DE INGENIERÍA

ESCUELA DE ELECTRÓNICA

PROYECTO DE GRADUACIÓN

ARTIFICIAL NEURAL NETWORK SIMULATOR – ANNS V1.1
DISEÑO DE SOFTWARE DE SIMULACIÓN DE REDES NEURONALES ARTIFICIALES
(PARTE II)

PRESENTADO POR:

CLAUDIA GUADALUPE SANDOVAL VÁSQUEZ

CHRISTIAN VLADIMIR MARIN ESCOBAR

PARA OPTAR POR EL TITULO DE **INGENIERO EN ELECTRÓNICA**.

ASESOR:

ING. HÉCTOR RUBÉN CARIAS

Facultad de Ingeniería de la Universidad Don Bosco

CIUDADELA DON BOSCO, JUNIO 2008

UNIVERSIDAD DON BOSCO

FACULTAD DE INGENIERÍA

ESCUELA DE ELECTRÓNICA

AUTORIDADES:

RECTOR

ING. FEDERICO MIGUEL HUGUET RIVERA

VICERRECTOR ACADÉMICO

PBRO. VÍCTOR MANUEL BERMÚDEZ YÁNEZ, SDB

SECRETARIO GENERAL

LIC. MARIO RAFAEL OLMOS ARGUETA

DECANO DE LA FACULTAD DE INGENIERÍA

ING. ERNESTO GODOFREDO GIRÓN

DIRECTOR DE ESCUELA DE ELECTRÓNICA

ING. OSCAR GIOVANNI DURAN VIZCARRA

ASESOR DEL TRABAJO DE GRADUACIÓN

ING. HÉCTOR RUBÉN CARIAS

LECTOR

ING. JUAN CARLOS CRUZ DADA

UNIVERSIDAD DON BOSCO

FACULTAD DE INGENIERÍA

ESCUELA DE ELECTRÓNICA

EVALUADORES DE TRABAJO DE GRADUACIÓN

_____	_____
ING. JUAN CARLOS CRUZ DADA	ING. HÉCTOR RUBÉN CARIAS
LECTOR	**ASESOR**

_____

ING. OSCAR GIOVANNI DURÁN VIZCARRA

TUTOR

AGRADECIMIENTOS

A Dios Todopoderoso, la Virgen María en su advocación de la Virgen de Guadalupe, a mis queridos padres Magdalena y René; y mi hermano Carlos Alexander, ¡Gracias! por ser mi gran apoyo durante todos estos años, para ellos mi más profundo cariño y agradecimiento.
Gracias a mis parientes y amigos por sus palabras de aliento que me animaron y dieron fuerzas para finalizar con esta etapa de mi carrera.

Claudia Guadalupe Sandoval Vásquez

Gracias señor por darme la oportunidad de lograr una meta mas en mi vida, y que esta felicidad la comparta con toda las personas que quiero, tú, has sido el centro principal de este esfuerzo, si no es por ti no lo hubiera logrado.
Agradezco también a mis padres Alicia Guadalupe y Francisco Marin que tanto lucharon para que yo tuviera una formación profesional y llegar a ser un hombre de bien.
A mis hermanos Joselin Guadalupe y koran Marin que me brindaron en todo momento el apoyo para lograr este objetivo tan grande.
A Deira Jocelyn mi amiga, novia y ahora esposa que me acompaño en todos los momentos buenos y malos para alcanzar mis metas, que siempre me tomo de la mano y me brindo todo el apoyo para soportar las pruebas con mucho amor y paciencia, te amo por eso.
Y una fuente de inspiración el tesoro más grande que Dios me pudo regalar a mi pequeño hijo Christian Alessandro. Que también fue punto importante en mi vida para que siempre me recuerde como un modelo de padre, te amo mi pequeño.

Christian Vlaldimir Marin Escobar

A nuestro amigo y compañero Elmer Carias, quien nos brindo su apoyo incondicional en todo momento.

CONTENIDO

INDICE DE FIGURAS

INDICE DE TABLAS

INTRODUCCIÓN

Las líneas de investigación que se han establecido en la facultad de Ingeniería de la Universidad Don Bosco, haciendo énfasis en la Escuela de Ingeniería Electrónica, han permitido que tanto profesores como estudiantes, encuentren las herramientas necesarias para los procesos de investigación.

Muchas de estas herramientas se encuentran encaminadas a fortalecer el proceso de enseñanza – aprendizaje en diferentes áreas de interés para la escuela.

Adicional a esto se han presentado tendencias alternativas que fortalecen el desarrollo de nuevas tecnologías, tales como los principios de Inteligencia Artificial y, más específicamente, el "Estudio de las Redes Neuronales Artificiales" (RNA).

Una Red Neuronal Artificial es un modelo de procesamiento de información que está inspirado por el modo en que el sistema nervioso biológico procesa información. Este modelo se compone de un gran número de elementos de procesamiento (neuronas) interconectados, trabajando en armonía, con el propósito de resolver problemas específicos.

Las Redes Neuronales Artificiales, tal como lo hacen las personas, aprenden con ejemplos.

Toda Red Neuronal Artificial está configurada para una aplicación específica, tal como el reconocimiento de patrones o clasificación de datos, a través de un proceso de aprendizaje.

En sistemas biológicos aprender implica ajustes para las conexiones sinápticas que existen entre las neuronas. Este proceso se desarrolla de igual manera, en las Redes Neuronales Artificiales, las cuales han sido aplicadas a un gran número de problemas reales de complejidad considerable.

La ventaja más importante del uso de las RNA, está referida a la resolución de situaciones con un alto grado de dificultad para tecnologías convencionales. Dificultades para las que no existe un algoritmo de solución o éste es muy difícil de encontrar.

Considerando lo anterior, es lógico plantearse que, el estudio de las RNA, se vuelve fundamental para el entendimiento de nuevas tecnologías aplicadas en la industria y, al interior de las mismas, priorizando aquellas que se encuentran dentro del campo de la electrónica. Así por ejemplo, podemos mencionar: el reconocimiento de imágenes, control y optimización de procesos; motivo por el cual es necesario fortalecer el aprendizaje de las mismas.

En este proyecto se presenta el desarrollo de una herramienta de software cuyo principal objetivo es ayudar a los estudiantes y maestros en el proceso de enseñanza aprendizaje de la Cátedra de Redes Neuronales Artificiales.

El documento está dividido en tres capítulos, siendo el primer capítulo la definición del tema, describiendo el por qué del proyecto y los objetivos que se pretende alcanzar con él. Además se describen los alcances y limitaciones del mismo, así como el tiempo que tomará el desarrollo completo del proyecto. El segundo capítulo hace una introducción a los antecedentes de las redes neuronales artificiales, así como los algoritmos de cada una de las redes que contiene nuestro simulador (ART1, ART2, Hopfield, Kohonen y Contrapropagación).

El tercer capítulo describe la utilización del software desarrollado, los pasos que se deben seguir para poder configurar, entrenar y simular una red neuronal artificial de las diferentes topologías contenidas en el mismo.

Capítulo 1 – DEFINICIÓN DEL TEMA

1.1 PLANTEAMIENTO DEL PROBLEMA

La necesidad de este proyecto surge debido a las limitantes que presenta el software de simulación para la cátedra de Redes Neuronales Artificiales.

A pesar de que el software contiene eficientes algoritmos para la simulación, éste no es suficiente para la ilustración de las prácticas, debido a que se requiere de un tiempo mucho mayor que el asignado en un laboratorio para terminar una práctica completa.

En la actualidad el software que se utiliza en las prácticas de laboratorio es Matlab. Dicha herramienta no ha sido diseñada directa y exclusivamente para ser utilizado en el estudio de Redes Neuronales, Si bien es cierto que posee un ambiente gráfico, en éste no se pueden identificar perfectamente todos los elementos como: neuronas, capas, pesos, etc., lo que dificulta el completo entendimiento del proceso de entrenamiento y simulación.

También es importante considerar que, para su utilización, se requiere de licencia de usuario, y es por esta razón que en el laboratorio solo se cuenta con cinco licencias del Toolbox de Redes Neuronales. Al implementar el simulador, éste podrá instalarse en la cantidad de computadoras que sea necesario sin problemas de licencia y sin incurrir en gastos adicionales.

Para utilizar Matlab, se requiere que el usuario dedique un espacio de tiempo relativamente extenso para poder adaptarse a mismo. Primero se necesita disponer de un tiempo para poder conocer varias instrucciones antes de dedicarse a crear una red y poderla entrenar y simular, por otro lado, resulta difícil para el estudiante acostumbrarse a la notación, nomenclatura y estructura de las redes que Matlab utiliza.

El ambiente gráfico que posee es limitado, es necesario que el usuario digite líneas de comando (algunas funciones especificas) para poder realizar diferentes pasos para la creación, entrenamiento y simulación de la red.

1.2 DESCRIPCIÓN DE LA PROPUESTA DEL PROYECTO

Se diseñará el Software de Simulación de Redes Neuronales Artificiales parte dos para las redes ART1, ART2, Hopfield, Kohonen y Contra propagación.

El software contará con una interfaz de usuario amigable en ambiente gráfico, lo que garantizará que las personas que lo utilicen puedan familiarizarse rápidamente con él, dando como resultado una optimización en el tiempo de ejecución de las prácticas de laboratorio.

Este simulador de RNA parte II, se creará en una plataforma diferente del simulador SG-SIM2006, es decir, que no se diseñará dentro del núcleo del primero.

Con este Software se contribuye de una manera significativa en la optimización del tiempo en el que se realizan las prácticas de laboratorio de los estudiantes que cursan la asignatura de Redes Neuronales.

1.3. VALIDACIÓN DE RESULTADOS

La ratificación de los resultados se realizará ante diversas condiciones; durante el proceso de diseño del software de simulación de Redes Neuronales Artificiales parte II.

Para validar el software, recurriremos en primera instancia a las guías de laboratorio de la cátedra *Redes Neuronales Artificiales*, es decir, que dichas prácticas se ejecutarán en el nuevo simulador y el resultado deberá ser igual o con un error por debajo del 10% del resultado obtenido con el software para el cual fueron diseñadas (MATLAB).

En segunda instancia se tomarán algunos ejemplos de la bibliografía consultada y el resultado que estos presentan será comparado con los obtenidos al ejecutarlos en el simulador.

Al finalizar, el software diseñado se someterá a pruebas en diferentes computadoras a fin de verificar la operatividad del mismo.

Es importante señalar que, para ejecutar la aplicación, se deberá cumplir con ciertos requisitos técnicos, los cuales serán señalados al final en el manual técnico de instalación del simulador de Redes Neuronales parte II.

1.3 OBJETIVOS

1.4.1 OBJETIVO GENERAL:

Desarrollar un software para entrenar y simular las siguientes topologías de Redes Neuronales Artificiales: ART1, ART2, Hopfield, Kohonen y Contra propagación; y de esa manera contribuir al proceso de enseñanza aprendizaje en la cátedra Redes Neuronales Artificiales.

1.4.2 OBJETIVOS ESPECÍFICOS:

- Diseñar un software con una interfaz gráfica amigable para el usuario.
- Crear un software que permita crear cinco topologías diferentes de redes neuronales.
- Permitir al usuario, a través de la interfaz gráfica, la modificación de los parámetros de entrenamiento, para cada tipo de red que se desee entrenar.
- Elaborar, en la interfaz, la opción de guardar las redes ya creadas, así como los parámetros de entrenamiento previamente definidos.
- El simulador será capaz de reconstruir las redes, solamente utilizando los parámetros definidos y guardados con anterioridad.
- La interfaz podrá mostrar los pesos finales cuando el proceso de entrenamiento finalice.
- Crear una ventana de edición de parámetros de entrenamiento para cada topología, la que será diferente e independiente para cada una de ellas.

1.4 ALCANCES.

- En el simulador se incluirán cinco tipos diferentes de Redes: ART1, ART2, KOHONEN, HOPFIELD Y CONTRAPROPAGACIÓN.

- El simulador se creará con Visual Basic.NET debido a la versatilidad que presenta en la realización de aplicaciones. Sin embargo, no se abandona la posibilidad de utilizar algún software adicional que facilite el desarrollo del simulador.

- La introducción de los patrones podrá realizarse en forma matricial, es decir, que para el usuario ya no será necesario ordenar los patrones en forma de vectores.

- Al final de cada proceso de entrenamiento, se mostrarán los pesos alcanzados.

- El simulador contará con la opción de poder definir los pesos iniciales para cada tipo de red creada.

- Durante el entrenamiento, el usuario tendrá la opción de poder detener el proceso, sin perder los pesos calculados hasta ese momento.

- Se podrá introducir varios patrones de entrada simultáneamente para realizar el entrenamiento, esto si la topología de la red lo permite.

- Las funciones de transferencia que podrán afectar la salida de cada neurona podrá ser cualquiera de las siguientes: Escalón, Escalón Bipolar, Sigmoidal Tangencial y Sigmoidal Logarítmica, siempre y cuando la topología de la red así lo permita.

- Para las redes multicapa, será posible aplicar una función de transferencias independientes para cada capa, siempre y cuando la topología de estas redes así lo permita.

1.6 LIMITACIONES

- El simulador ha desarrollar funcionará únicamente en ambiente Windows XP de 32 bits.

- El software únicamente incluirá las redes ART1, ART2, Hopfield, Kohonen y Contra propagación.

- El tiempo de entrenamiento al momento de la ejecución del mismo, estará sujeto a las condiciones del Hardware que se esté utilizando.

- El simulador permitirá la configuración de las redes y realizar su respectivo proceso de entrenamiento además de la simulación de las mismas.

- Para el caso de la redes de contra propagación, se contará con un máximo de tres capas.

- Para las redes ART1, ART2, Kohonen y Hopfield, el número máximo de capas dependerá de la topología en sí misma.

- Los patrones de prueba deberán ser generados dentro de la aplicación, para que tengan el formato necesario.

- El simulador podrá correr sólo una red a la vez.

- Cada una de las neuronas que se encuentren dentro de una capa particular, responderá a la función de transferencia que se le aplicará a toda la capa.

- Las funciones de transferencia que se utilizarán para cada red estarán sujetas a la topología de la misma.

- Si la topología de la red que se esté empleando lo permite, se podrán colocar funciones de transferencia diferentes para cada capa que conforme la red que se está creando.

Capitulo 2 – MARCO TEÓRICO

2.1 PANORAMA HISTÓRICO

Las primeras explicaciones teóricas sobre el cerebro y el pensamiento fueron dadas por algunos antiguos filósofos griegos, como Platón (427-347 a.C.) y Aristóteles (422 – 384 a.C.). Las mismas ideas sobre el proceso mental las mantuvieron Descartes (1596 – 1650) y los filósofos empiristas del siglo X.

Alan Turing en 1936 fue el primero en estudiar el cerebro como una forma de ver la computación; sin embargo los primeros teóricos que concibieron los fundamentos de la computacional neuronal fueron:

Walter Pitts y Warren McCulloch intentaron explicar en 1943 el funcionamiento del cerebro humano, por medio de una red de células conectadas entre sí podían ejecutar operaciones lógicas. Partiendo del menor suceso psíquico (estimado por ellos): el impulso todo o nada, generado por una célula nerviosa. El bucle "sentidos - cerebro - músculos", mediante la retroalimentación producirían una reacción positiva si los músculos reducen la diferencia entre una condición percibida por los sentidos y un estado físico impuesto por el cerebro. También definieron la memoria como un conjunto de ondas que reverberan en un circuito cerrado de neuronas. Actualmente, sabemos que las decisiones conscientes acerca de la verdad de las proposiciones lógicas se producen a un nivel más alto, y quizás participen en ellas millones de células cerebrales.

En 1949, el fisiólogo Donald Hebb expuso en su libro The Organitation of Behavior [HEBB49] la conocida regla de aprendizaje. Su propuesta tenia que ver con la conductividad de la sinapsis, es decir, con las conexiones entre neuronas. Hebb expuso que la repetida activación de una neurona por otra a través de una sinapsis determinada, aumenta su conductividad, y la hacía más propensa a ser activada sucesivamente, induciendo a la formación de un circuito de neuronas estrechamente conectadas entre sí.

Durante el verano de 1951, Minsky y Edmons montaron la primera máquina de redes neuronales, compuesta básicamente de 300 tubos de vacío y un piloto automático de un bombardero B-24. Llamaron a su creación "Sharc", se trataba

nada menos que de una red de 40 neuronas artificiales que imitaban el cerebro de una rata.

Cada neurona hacia el papel de una posición del laberinto y cuando se activaba daba a entender que la "rata" sabia en que punto del laberinto estaba. Las neuronas que estaban conectadas alrededor de la activada, hacían la función de alternativas que seguir por el cerebro, la activación de la siguiente neurona, es decir, la elección entre "derecha" o "izquierda" en este caso estaría dada por la fuerza de sus conexiones con la neurona activada. Por ejemplo, la "rata" completaba bien el recorrido eligiendo a partir de la quinta neurona la opción "izquierda" (que correspondería a la sexta), es entonces cuando las conexiones entre la quinta y sexta se hacen más fuertes (dicha conexión era realizada por el piloto automático), haciendo desde este momento más propensa esta decisión en un futuro. Pero las técnicas Skinnerianas (que eran las que se habían puesto en funcionamiento en esta red neuronal) no podrían llevar muy lejos a este prototipo, debido a que en sí, no se trataba de inteligencia, pues la red neuronal nunca llegaría a trazar un plan. Después de su Red Neuronal, Minsky escribió su tesis doctoral acerca de ésta, en ella describía "cerebros mucho mayores", exponiendo que si se realizaba este proyecto a gran escala, con miles o millones de neuronas más y con diferentes sensores y tipos de retroalimentación…la máquina podría ser capaz de razonar, mas él sabía que la realización de esta Red Neuronal era imposible y decidió buscar otra forma de crear inteligencia.

En 1957, Frank Rosenblatt presentó al Perceptrón, una red neuronal con aprendizaje supervisado cuya regla de aprendizaje era una modificación de la propuesta por Hebb. La verdad es que el Perceptrón fue una buena idea, y es posible que los problemas que obtuvo fueran provocados más que nada por el exagerado entusiasmo de su creador. Más tarde, en 1969, Marvin Minsky y Seymour Paper, escribieron un libro llamado Perceptrons [MINS69], en donde definían a estos como caminos sin salida. También es verdad que indagaron en ellos y sacaron conclusiones muy curiosas e interesantes, pero dado que ambos eran dos personalidades de peso en el mundo de la Inteligencia Artificial., en cuanto se publicó el libro, todas las investigaciones a cerca de Perceptrones se paralizaron y anularon.

En los años 60 se propusieron otros dos modelos, supervisados y basados en el Perceptron de Rosenblatt denominados Adaline y Madaline. En estos, la adaptación de los pesos se realiza teniendo en cuenta el error, calculado como la diferencia entre la salida deseada y la dada por la red, al igual que en el Perceptron. Sin embargo, la regla de aprendizaje empleada es distinta.

La era moderna de las redes neuronales artificiales surge con la técnica de aprendizaje de propagación hacia atrás o Backpropagation. En 1977, James Anderson desarrolló un modelo lineal, llamado Asociador Lineal, que consistía en unos elementos integradores lineales (neuronas) que sumaban sus entradas. En 1982 John Hopfield presentó un trabajo sobre redes neuronales en la Academia Nacional de las Ciencias; en el cual describe con claridad y rigor matemático una red a la que ha dado su nombre, que viene a ser una variación del Asociador Lineal. En este mismo año la empresa Fujitsu comenzó el desarrollo de computadores pensantes para aplicaciones en robótica.

En 1985, el Instituto Americano de Física comenzó la reunión anual Neural Networks for Computing. En 1987 la IEEE celebró su primera conferencia internacional sobre redes neuronales. En este mismo año se formó la International Neural Network Society (INNS) bajo la iniciativa y dirección de Grossberg en USA, Kohonen en Finlandia y Amari en Japón. En 1988, resultó la unión entre la IEEE y de la INNS que produjo la International Joint Conference on Neural Networks. Esta nueva organización realizó 430 artículos de los cuales 63 estaban enfocados a una aplicación. La alternativa europea fue la International Conference on Artificial Neural Networks que comenzó su labor en septiembre de 1981, y actualmente está organizada por la Sociedad Europea de Redes Neuronales. También merece una referencia aparte la reunión anual Neural Information Processing Systems celebrada en Denver (Colorado) desde 1987, y que probablemente represente el nivel más alto de calidad desde el punto de vista científico.

2.2 BREVE INTRODUCCIÓN BIOLÓGICA

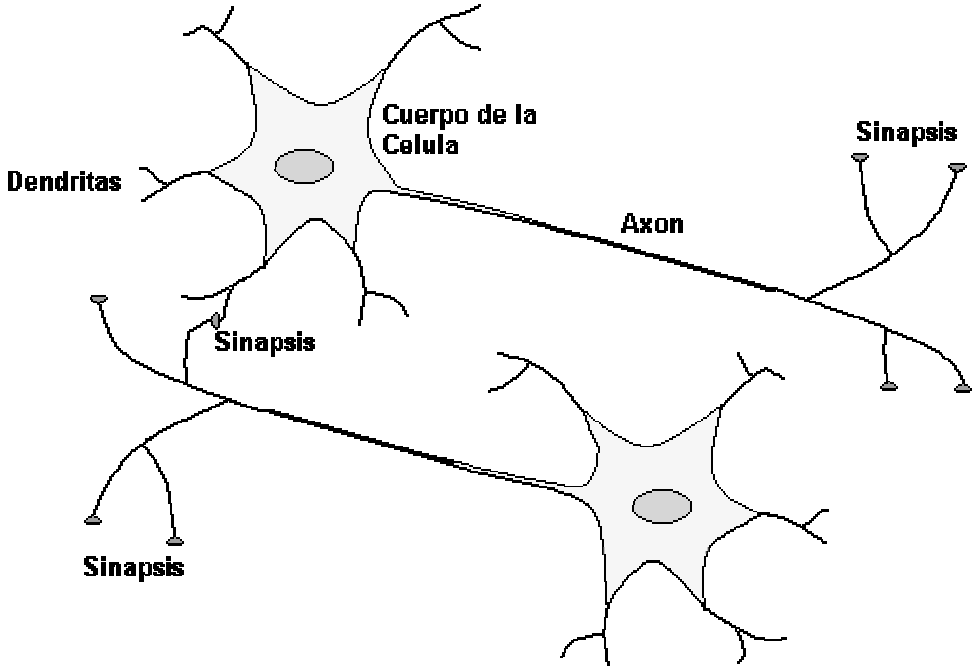


Figura 1. Estructura de una Neurona Biológica.

Una neurona es una célula viva y como tal, contiene los mismos elementos que forman parte de todas las células biológicas. Además, contienen elementos característicos que las diferencian [Hilera José R., Martínez Víctor J., 1994]. Una neurona consta de un cuerpo celular más o menos esférico de 5 a 10 micras de diámetro del que se desprende una rama principal o axón y varias ramas más cortas llamadas dendritas [Figura 1]. A su vez el axón presenta ramas en torno a su punto de arranque, y con frecuencia se ramifica extensamente cerca de su extremo.

Una de las características que diferencian a las neuronas del resto de las células vivas, es la capacidad que tienen éstas de comunicarse. En términos generales, las dendritas y el cuerpo celular reciben señales de entrada, el cuerpo celular las combina, integra y emite señales de salida. El axón transporta estas señales a los terminales axónicos, que se encargan de distribuir información a un nuevo conjunto de neuronas. Por lo general, una neurona recibe información de miles de otras neuronas y envía información a miles de otras más. Se calcula que en el cerebro humano una neurona es capaz de generar alrededor de 10^{15} conexiones.

En la figura 2 se puede observar como las neuronas biológicas se interconectan entre si formando redes neuronales.

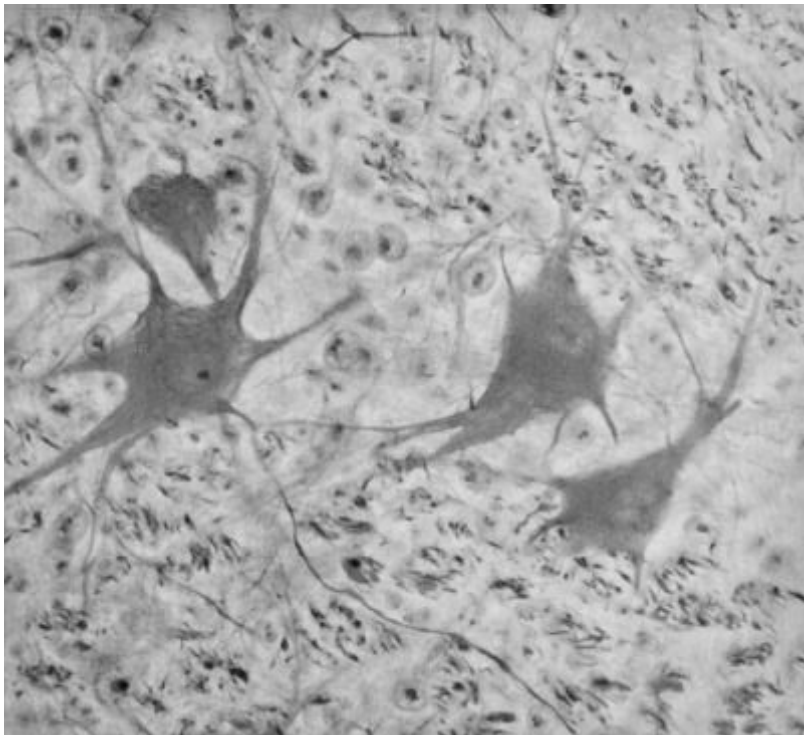


Figura 2. Red Neuronal Biológica

Las señales a las que nos estamos refiriendo son de dos tipos de naturaleza: eléctrica y química. La señal generada por la neurona y transportada a lo largo del axón es un impulso eléctrico, mientras que las señales que se transmiten entre los terminales axónicos de una neurona y las dendritas de otra es de origen químico; concretamente, se realiza mediante moléculas de sustancias transmisoras (neurotransmisores) que fluyen a través de unos contactos especiales, llamados sinapsis, que tienen la función de receptor y están localizados entre los terminales axónicos y las dendritas de las neuronas siguientes.

La generación de las señales eléctricas está íntimamente relacionada con la composición de la membrana celular. El proceso de generación de las señales eléctricas de una neurona se puede simplificar del siguiente modo [Cardinali Daniel P., 1995]: la neurona, como todas las células, es capaz de mantener en su interior un líquido cuya composición defiere marcadamente de la composición del líquido del exterior. La diferencia más notable se da con relación a la concentración de iones sodio y potasio. El medio externo es unas 10 veces más rico en iones sodio que el interno, mientras que el interno tiene una concentración

de iones potasio 10 veces mayor que el externo. Esta diferencia de concentraciones de iones sodio y potasio genera una diferencia de potencial entre el interior y el exterior de la membrana celular del orden de los 70 mV (negativa en el interior de la membrana). Esto es lo que se conoce como el potencial de reposo de la célula nerviosa.

La llegada de señales procedentes de otras neuronas a través de las dendritas (recepción de neurotransmisores) actúa acumulativamente, bajando ligeramente el valor del potencial de reposo. Dicho potencial modifica la permeabilidad de la membrana de manera de que cuando llega a cierto valor crítico comienza una entrada masiva de iones sodio que invierten la polaridad de la membrana.

La inversión del voltaje de la cara interior de la membrana cierra el paso a los iones potasio hasta que se restablece el equilibrio en reposo. La inversión del voltaje, conocida como potencial de acción, se propaga a lo largo del axón y, a su vez, provoca la emisión de los neurotransmisores en los terminales axónicos.
Después de un período refractario, puede seguir un segundo impulso. El resultado de todo esto es la emisión por parte de la neurona, de trenes de impulsos cuya frecuencia varía en función (entre otros factores) de la cantidad de neurotransmisores recibidos. Existen dos tipos de sinapsis: a) las sinapsis excitadoras, cuyos neurotransmisores no permiten las disminuciones de potencial en la membrana de la célula postsináptica, facilitando la generación de impulsos a mayor velocidad, y b) las sinapsis inhibidoras, cuyos neurotransmisores tienden a estabilizar el potencial de la membrana, dificultando la emisión de impulsos. Casi todas las neuronas reciben entradas procedentes de sinapsis excitadoras o inhibidoras. En cada instante algunas de ellas estarán activas y otras se hallarán en reposo; la suma de los efectos excitadores e inhibidores determina si la célula será o no estimulada: es decir, si emitirá o no un tren de impulsos y a qué velocidad.

2.3 ESTRUCTURA DE UN SISTEMA NEURONAL ARTIFICIAL.

Los sistemas neuronales artificiales imitan la estructura del hardware del sistema nervioso, con la intención de construir sistemas de procesamiento de información paralelos, distribuidos y adaptativos, que puedan presentar un cierto comportamiento "inteligente", ya que son capaces de aprender a través de ejemplos.

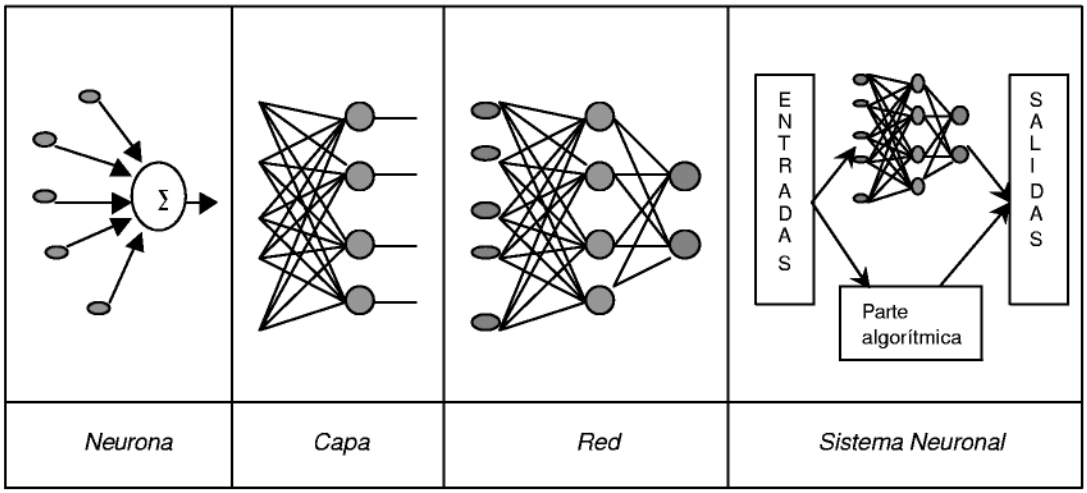


Figura 3. Estructura jerárquica de un sistema basado en Redes Neuronales Artificiales

Cada neurona realiza una función matemática. Las neuronas se agrupan en capas, constituyendo una red neuronal. Una determinada red neuronal esta diseñada y entrenada para llevar a cabo una labor especifica. Finalmente, una o varias redes, mas las interfaces con el entorno, conforman el sistema global.

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

Los modelos neuronales se diferencian entre si por la función que incorpora la neurona, su organización y forma de las conexiones.
Formalmente, un sistema neuronal artificial, está compuesto de los siguientes elementos:

- Un conjunto de procesadores elementales o *neuronas artificiales.*
- Un patrón de conectividad o arquitectura.
- Una dinámica de activaciones.
- Una regla o dinámica de aprendizaje.
- El entorno donde opera.

2.4 ELEMENTOS DE UNA RED NEURONAL ARTIFICIAL.

Las redes neuronales artificiales son modelos que intentan reproducir el comportamiento del cerebro. Como tal modelo, realiza una interpretación, averiguando cuáles son los elementos relevantes del entorno. Una elección adecuada de sus características, más una estructura conveniente, es el procedimiento convencional utilizado para construir redes capaces de realizar una determinada tarea.

Cualquier modelo de red neuronal consta de dispositivos elementales de proceso: las neuronas. A partir de ellas, se pueden generar representaciones específicas, de forma tal que un estado conjunto de ellas puede significar una letra, un número o cualquier otro objeto. Generalmente se pueden encontrar tres tipos de neuronas:

1. Aquellas que reciben estímulos externos, relacionadas con el aparato sensorial, que tomarán la información de entrada.

2. Dicha información se transmite a ciertos elementos internos que se ocupan de su procesado. Es en las sinapsis y las neuronas correspondientes a este segundo nivel donde se genera cualquier tipo de representación interna de la información. Puesto que no tienen relación directa con la información de entrada ni con la de salida, estos elementos se denominan unidades ocultas.

3. Una vez finalizado el período de procesado, la información llega a las unidades de salida, cuya misión es dar la respuesta del sistema.

La neurona artificial [Figura 4] pretende imitar las características más importantes de las neuronas biológicas. Cada neurona j-ésima está caracterizada en cualquier instante por un valor numérico denominado valor o estado de activación $f_i(t)$, una función de activación o de transferencia que transforma el estado actual de activación en una señal de salida Y_i . Dicha señal de salida es enviada a través de canales de comunicación unidireccionales a otras unidades de la red; en estos canales, la señal se modifica de acuerdo con la sinapsis (el peso w_{ij}) asociada a cada uno de ellos según una determinada regla de aprendizaje, que variará dependiendo de la topología de la red. Las señales moduladas que han llegado a la unidad j-ésima se combinan entre ellas, generando así la entrada total Net_j .

$$Net_j = \sum_i y_i w_{ij}$$ **Ecuación 2.1**

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

2.4.1 La Neurona Artificial.

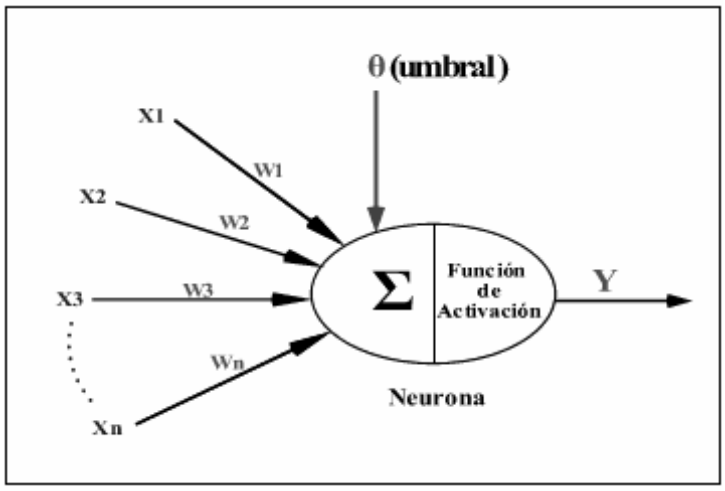


Figura 4. Neurona Artificial.

Si tenemos N unidades (neuronas), podemos ordenarlas arbitrariamente y designar la *j*-ésima unidad como *Uj*. Su trabajo consiste únicamente en recibir las entradas de las neuronas vecinas y calcular un valor de salida, que es enviado a todas las neuronas restantes.

Es útil identificar tres tipos de unidades: entradas, salidas y ocultas.

- Las unidades de entrada reciben señales desde el entorno. Estas entradas (que son entradas a la red) pueden provenir de sensores o de otros sectores del sistema.
- Las unidades de salida envían la señal fuera del sistema. Estas pueden controlar directamente potencias u otros sistemas.
- Las unidades ocultas son aquellas cuyas entradas y salidas se encuentran dentro del sistema, es decir, no tienen contacto con el exterior.

2.4.2 Función de Salida o de Transferencia.

Entre las neuronas que componen la red existe un conjunto de conexiones que las unen. Cada neurona transmite señales a aquellas que están conectadas con su salida.

Asociada a cada unidad *U,* hay una función de salida $F_i(a_i(t))$, que transforma el estado actual de activación $a_i(t)$ en una señal de salida $Y_iF(t)$ *;* es decir:

$$Y_i(t) = F_i(a_i(t)) \qquad \textbf{Ecuación 2.2}$$

En consecuencia, el vector que contiene las salidas de todas las neuronas en un instante t es:

$$Y\ (t)\ =\ (f_1a_1t)), f_2(a_2(t)),..., f_i(a_i(t)),..., f_N(a_N(t))) \qquad \textbf{Ecuación 2.3}$$

En algunos modelos, esta salida es igual al nivel de activación de la unidad, en cuyo caso la función, f_i es la función identidad, $f_i(a_i(t)) = \alpha(t)$. A menudo, f_i, es de tipo sigmoidal, y suele ser *la* misma para todas las neuronas.

Existen cinco funciones de transferencia típicas que determinan los distintos tipos de neuronas:

- Función Lineal o Identidad
- Función Escalón,
- Función lineal - mixta
- Sigmoidal,
- Función gaussiana

La función escalón o umbral únicamente se utiliza cuando las salidas de la red son binarias (0,1) ó (-1,1). La salida de una neurona se activa solo cuando el estado de activación es mayor o igual que cierto valor umbral (la función puede ser desplazada sobre los ejes). La función lineal o identidad equivale a no aplicar función de salida, y se usa muy poco. Las funciones mixta y sigmoidal

son las mas apropiadas cuando queremos como salida una información analógica.

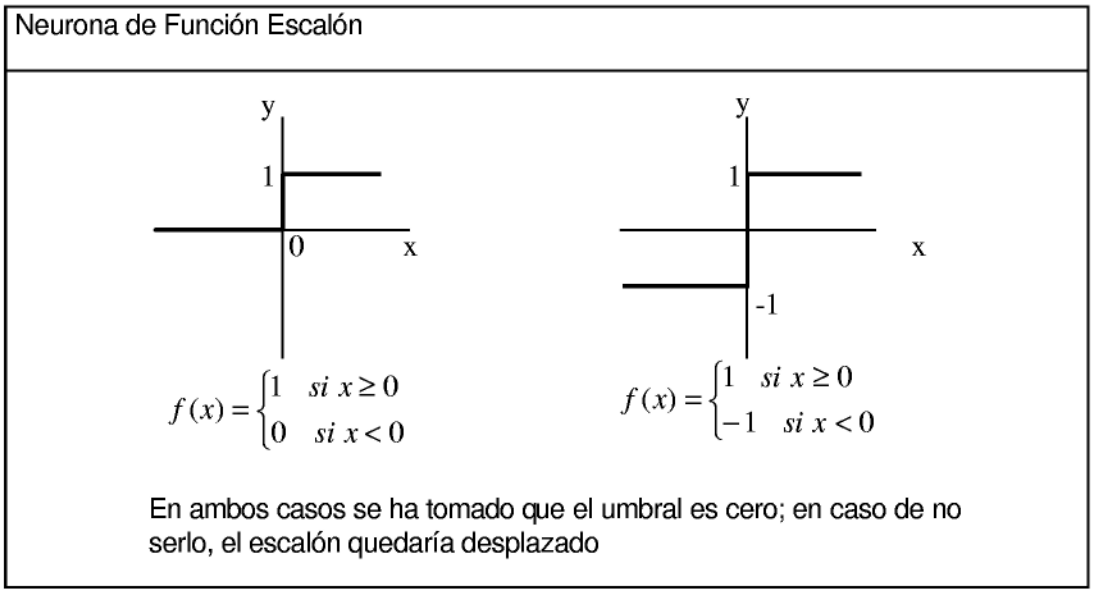


Figura 5. Función Escalón

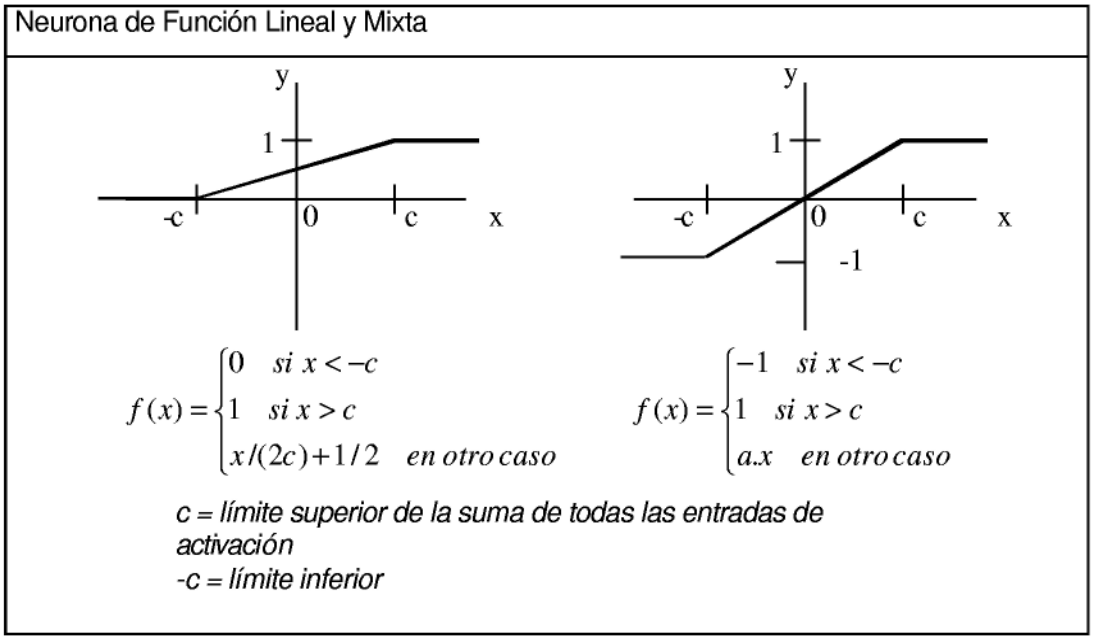
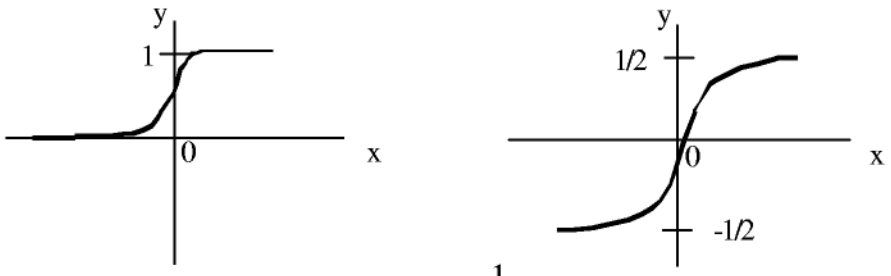


Figura 6. Función Lineal y Mixta

Neurona de Función Continua (sigmoidal)

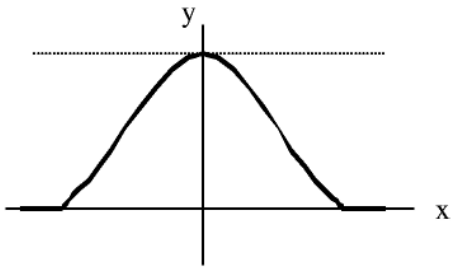


$$f(x) = \frac{1}{1 + e^{-a}}$$

La importancia de la función sigmoidal (o cualquier otra función similar) 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 es particularmente útil para definir métodos de aprendizaje en los cuales se usan derivadas.

Figura 7. Función Sigmoidal

Neurona de Función de transferencia gaussiana



$$y = A.e^{-l}$$

Los centros y anchura de estas funciones pueden ser adaptados, lo cual las hace más adaptativas que las funciones sigmoidales.
Mapeos que suelen requerir dos niveles ocultos (neuronas que se ubican entre las de entrada y las de salida) con neuronas de transferencia sigmoidales, algunas veces se pueden realizar con un solo nivel empleando neuronas de transferencia gaussiana.

Figura 8. Función Gaussiana

2.4.3 Conexiones entre Neuronas.

Las conexiones entre las neuronas de una red tienen asociado un peso, que es el que hace que la red adquiera conocimiento.

Tomemos el valor y_i, como el valor de salida de una neurona *i* en un instante dado. Una neurona recibe un conjunto de señales que le dan información del estado de activación de todas las neuronas con las que se encuentra conectada. Cada conexión *(sinapsis)* entre la neurona *i* y la neurona *j* esta ponderada por un peso w_{ij}. Normalmente, como simplificación, se considera que el efecto de cada señal es aditivo, de forma que la entrada neta que recibe una neurona (potencial post sináptico) Net_j es la suma del producto de cada señal individual por el valor de la sinapsis que conecta ambas neuronas:

$$Net_j = \sum_i^N w_{ij} y_i \qquad \textbf{Ecuación 2.4}$$

Esta regla muestra el procedimiento a seguir para combinar los valores de entrada a una neurona con los pesos de las conexiones que llegan a esa neurona, y es conocida como regla de propagación.

Suele utilizarse una matriz *W* con todos los pesos w_{ji} que reflejan la influencia que la neurona *j* tiene sobre la neurona *i. W* es una matriz de elementos positivos, negativos o nulos. Si w_{ij} es positivo, significa que la interacción entre ambas neuronas es excitadora, es decir, siempre que la neurona *i* este activada, la neurona *j* recibirán una señal proveniente de *i* que tenderá a activarla. Si w_{ij} es negativo, la sinapsis será inhibidora; es decir, si *i* está activada, enviará un mensaje a la neurona *j* que tenderá a desactivarla. Por último, si w_{ij} es cero, significa que no hay conexión entre ambas neuronas.

2.4.4 Regla de Aprendizaje.

El aprendizaje se entiende como la modificación del comportamiento inducido por la interacción con el entorno, y como resultado de experiencias conducente al establecimiento de nuevos modelos de respuesta a estímulos externos.

Lo anterior implica, que la interacción con el medio modifica el comportamiento, por lo tanto la conducta se dará en función de las experiencias con el entorno.
Biológicamente, se suele aceptar que la información memorizada en el cerebro está mas relacionada con los valores sinápticos de las conexiones entre las neuronas, que con ellas mismas; es decir, el conocimiento se encuentra en las sinapsis. [Hilera José y Martínez Víctor, "Redes Neuronales Artificiales; Fundamentos, modelos y aplicaciones"].

En el caso de las redes neuronales artificiales, el conocimiento se encuentra representado en los pesos de las conexiones entre neuronas (estos pesos son los que se encargan de generar la sinapsis). Todo proceso de aprendizaje implica cambios en estas conexiones, es decir, se aprende modificando los valores de los pesos de la red.

Durante el proceso de aprendizaje, los pesos de las conexiones de la red sufren modificaciones, por lo tanto se puede afirmar que la red "ha aprendido" cuando los valores de los pesos permanecen estables:

$$\frac{dw}{dt} = 0 \qquad \textbf{Ecuación 2.5}$$

Un aspecto importante respecto al aprendizaje de las redes neuronales es el conocer como es que se modifican estos valores; es decir, cuáles son los criterios que se siguen para cambiar los valores asignados a las conexiones cuando se pretende que la red aprenda una nueva información.

Estos criterios determinan lo que se conoce como la regla de aprendizaje de la red. De modo general, se distinguen:

- Redes Neuronales con aprendizaje supervisado.
- Redes neuronales con aprendizaje no supervisado o autoorganizado.

La diferencia fundamental entre ambos tipos estriba en la existencia o no de una 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 deshabilitación hasta que el proceso finalice. En el primer caso se trataría de un aprendizaje on line, mientras que el segundo se conoce como aprendizaje off line.

2.4.5 Formas de Conexión entre Neuronas.

La conectividad entre los nodos de una red neuronal está relacionada con la forma en que las salidas de las neuronas están canalizadas para convertirse en entradas de otras neuronas. La señal de salida de un nodo puede ser una entrada de otro elemento de proceso, o incluso ser una entrada de sí mismo en una conexión auto recurrente.

Cuando ninguna salida de las neuronas de una capa es entrada de neuronas del mismo nivel o de niveles precedentes, se dice que la red tiene propagación hacia delante. En caso contrario se dice que la red es de propagación hacia atrás. Las redes de propagación hacia atrás que tienen lazos cerrados se dice que son sistemas recurrentes.

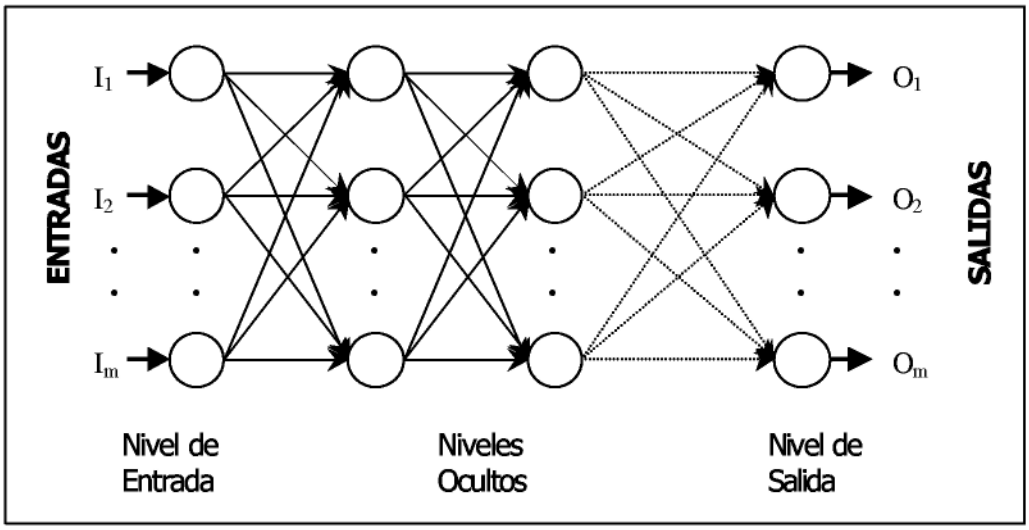


Figura 9. Estructura de una red multicapa con todas las conexiones hacia delante.

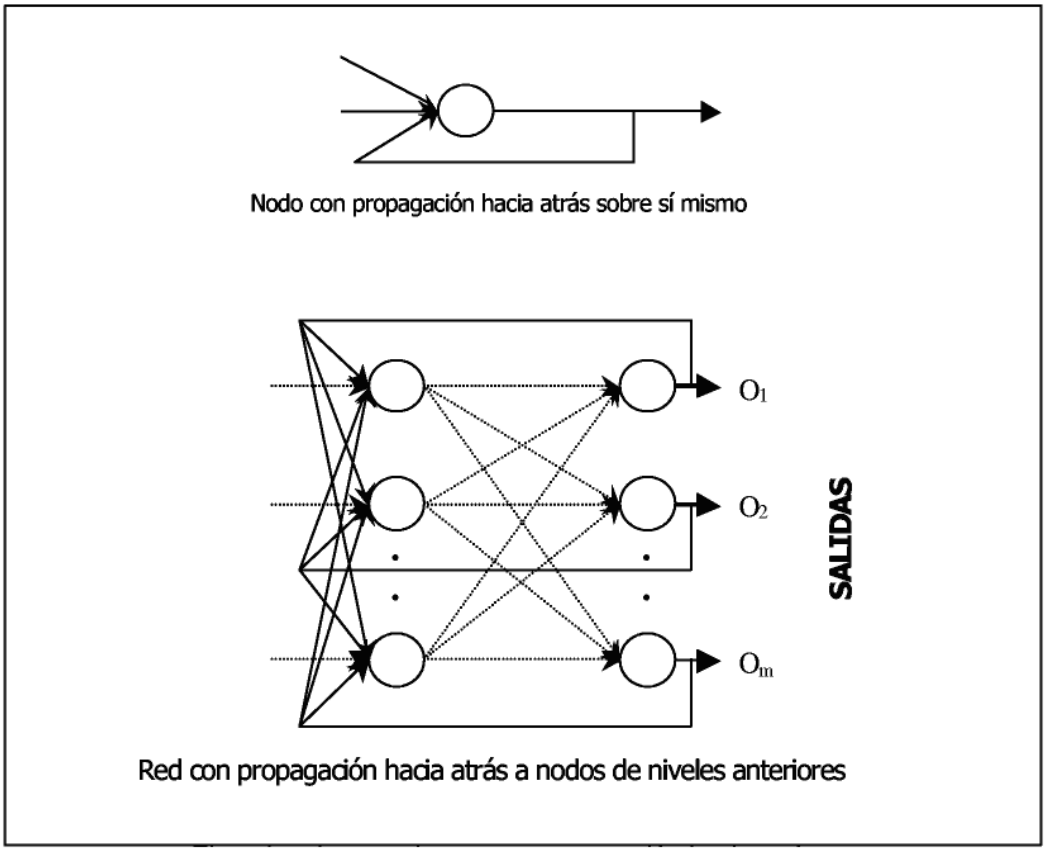


Figura 10. Redes de propagación hacia atrás.

2.5 TOPOLOGÍA DE LAS REDES NEURONALES.

Se denomina topología o arquitectura de la red a la organización y disposición de las neuronas en la red formando capas más o menos alejadas de la entrada y salida de la red. Así, los parámetros fundamentales de la red son: el número de capas (monocapa o multicapa), el número de neuronas por capa, el grado de conectividad y el tipo de conexiones entre neuronas (unidireccionales o recurrentes).

2.5.1 Redes Neuronales Monocapa.

Se utilizan típicamente en tareas relacionadas con la autoasociación; por ejemplo para regenerar informaciones de entrada que se presentan a las redes incompletas o distorsionadas.

TIPOS DE CONEXIONES		MODELO DE RED
Conexiones Laterales Explícitas	Conexiones Autorrecurrentes	Brain-State- In- A- Box
		Additive Grossberg (AG)
		Shunting Grossberg (SG)
		Optimal Linear Associative Memory
	No Autorrecurrentes	Hopfield
		Boltzmann Machine
		Cauchy Machine
Crossbar		Learning Matrix (LM)

Tabla 2.1 Redes Neuronales Monocapa.

2.5.2 Redes Neuronales Multicapa.

Cuando todas las neuronas de una capa reciben señales de entrada de otra capa anterior, mas cercana a la entrada a la red, y envían señales de salida a una capa posterior, estamos ante una red de conexiones hacia delante o feedforward. En las conexiones hacia atrás o feedback las salidas de las neuronas de capas posteriores se conectan a las entradas de capas anteriores.

Estas características permiten distinguir dos tipos de redes entre las multicapa: las redes con conexión hacia delante o redes feedforward, y redes que disponen de conexiones tanto hacia delante como hacia atrás, o redes feedforward/feedback.

Las redes feedforward son especialmente útiles en aplicaciones de reconocimiento o clasificación de patrones.

Por otro lado, la mayoría de las redes multicapa son bicapa. Este tipo de estructura es particularmente adecuada para realizar una asociación de una información o patrón de entrada con otra información o patrón de salida en la segunda capa.

Nº capas	TIPO DE CONEXIONES		MODELO DE RED
2 capas	Conexiones hacia adelante FEEDFORWARD		ADALINE / MADALINE
			PERCEPTRON
			LINEAR /ASSOC REWAR.PENALTY
			LINEAR ASSOCIATIVE MEMORY
			OPTIMAL LINEAR ASSOC. MEM.
			DRIVE-REINFORCEMENT (DR)
		Conexiones laterales implícitas y autorrecurrentes	LEARNING VECTOR QUANTIZER
			TOPOLOGY PRESERVING MAP (TPM)
	Conexiones hacia adelante / atrás (FEDDFORWARD / FEEDBACK)	Sin conexiones laterales	BIDIRECTIONAL ASSOC. MEM. (BAM)
			ADAPTIVE BAM.
			TEMPORAL ASSOC. MEMORY (TAM)
			FUZZY ASSOCIATIVE MEMORY (FAM)
		Con conexiones laterales y autorrecurrentes	COMPETITIVE ADAPTIVE BAM
			ADAPTIVE RESONANCE THEORY (ART) ADAPTIVE HEURISTIC CRITIC (AHC) BOLTZMANN / CAUCHY MACHINE
3 capas	Conexiones hacia delante (FEDDFORWARD)	Sin conexiones laterales	
		Con conexiones laterales	COUNTERPROPAGATION
			BOLTZMANN / CAUCHY MACHINE
	Conexiones adelante / atrás y laterales		BOLTZMANN / CAUCHY MACHINE
N	Conexiones hacia adelante		BACKPROPAGATION (BPN)
	FEEDFORWARD / FEEDBACK (Jerarquía de niveles de capas bidimensionales)		COGNITRON / NEOCOGNITRON

Tabla 2.2 Redes Neuronales Multicapa mas conocidas.

2.6 TOPOLOGÍAS IMPLEMENTADAS.

2.6.1 Red ART1.

La red ART fue Creada por S. Grossberg, en la cual se pretende categorizar los datos que se introducen en la red. Las informaciones similares son clasificadas formando parte de la misma categoría y por tanto deben activar la misma neurona de salida. Las clases o categorías deben ser creadas por la red, puesto que se trata de un aprendizaje no supervisado, a través de las similitudes entre los datos de entrada.

Esta red se basa en la idea de hacer resonar la información de entrada con los representantes o prototipos de las categorías que reconoce la red. Si entra en resonancia con alguno, es decir que es suficientemente similar, la red considera que pertenece a dicha categoría y únicamente realiza una pequeña adaptación del prototipo almacenado representante de la categoría, para que incorpore algunas características del dato presentado. Si no concuerda, no entra en resonancia, la red crea una nueva categoría con el dato prototipo de la misma.

Cada una de las capas existentes de la red realiza tareas diferentes:

Capa1 (Competitiva): Recibe los datos de entradas y realiza la comparación.
Capa 2: Realiza la tarea de reconocimiento y presenta los datos de salidas.

2.6.2 Arquitectura de la Red ART1.

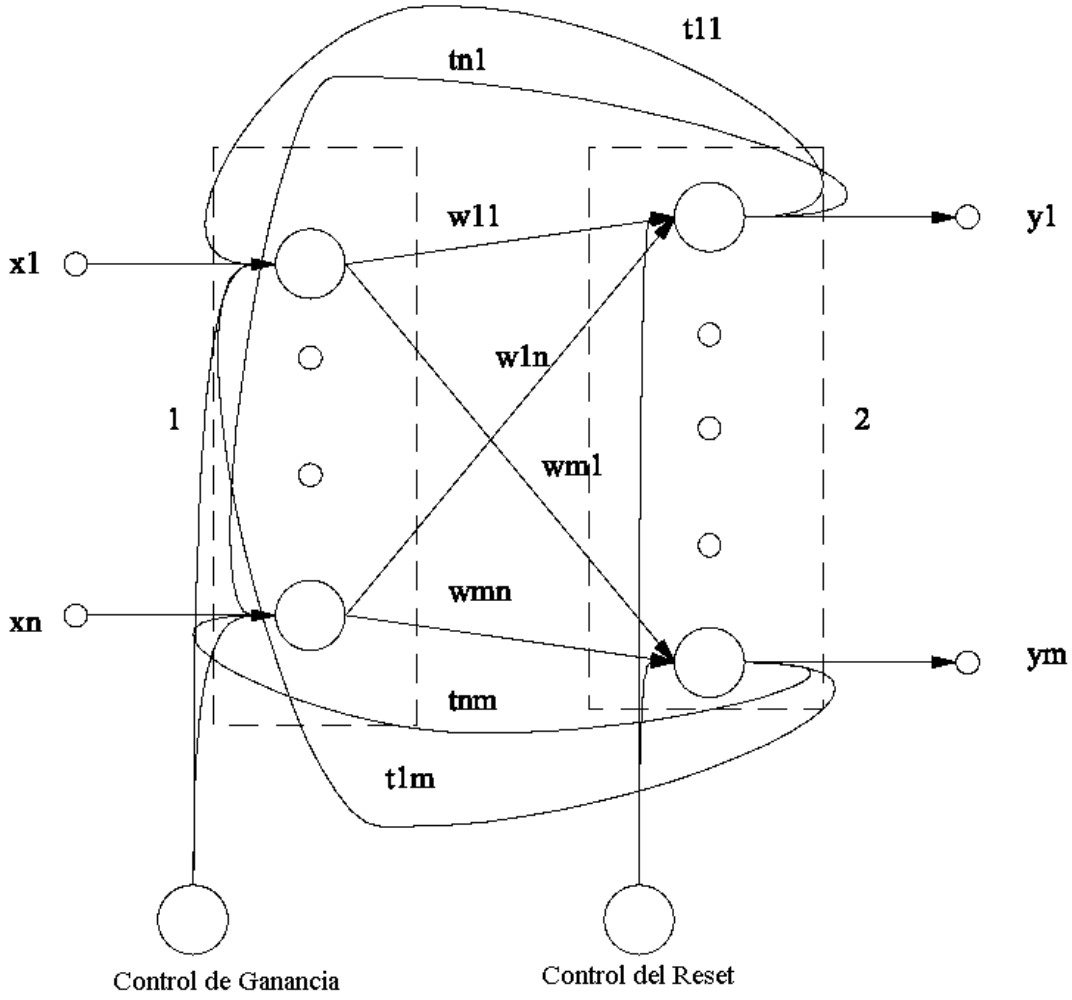


Figura 11. Arquitectura de Red ART1.

Características:

- Aprendizaje no supervisado.
- Aprendizaje on line.
- No requiere una base de datos para entrenar la red.
- Solo requiere las etapas de neurona de entrada y de salida.
- Existen interconexiones entre las neuronas de entrada y las de salida y viceversa.
- Aprendizaje flexible.
- Retener los valores previamente aprendidos.
- Tiene dos subsistemas, uno de reseteo y otro de control de ganancia.

2.6.3 Algoritmo de Aprendizaje.

Paso 1

Inicializar los parámetros:

Los valores permisibles deberán estar dentro de los siguientes rangos:

$$L > 1 \qquad \textbf{Ecuación 2.6}$$

$$0 < \rho \leq 1 \qquad \textbf{Ecuación 2.7}$$

Donde:

L : Es un parámetro de control.

ρ : Es el parámetro de vigilancia que decide que tan parecido es el patrón de entrada a lo que ha aprendido la red.

Paso 2

Inicializar los pesos.

$$0 < b_{ij} < \frac{L}{L-1+n} \qquad \textbf{Ecuación 2.8}$$

$$t_{ji} = 1 \qquad \textbf{Ecuación 2.9}$$

Donde

b_{ij} : Son los pesos que van de la entrada hacia la salida.

t_{ji} : Son los pesos que van de la salida a la entrada.

n: Es la cantidad de elementos que contiene cada vector de entrada.

Paso 3

Realizar los pasos 4 al 14 para todos los patrones de entrada.

Paso 4

Se repiten los pasos 5 al 12 hasta que se hayan deshabilitado todas las salidas.

Paso 5

Poner la entrada en la capa F1:

$$Si = \text{entrada} \qquad \textbf{Ecuación 2.10}$$

Paso 6

Enviar la entrada a la capa *x*:

$$x_i = s_i \qquad \textbf{Ecuación 2.11}$$

Paso 7

Se calcula la salida de la capa *F2*, para aquellas neuronas que no están inhibidas o deshabilitadas.

$$\text{Si } y_j \neq -1 \text{ Entonces } y_j = \sum_i b_{ij} x_i \qquad \textbf{Ecuación 2.12}$$

Paso 8

Mientras el reset sea verdadero, haga los pasos 9 a 12

Paso 9

Buscar cuál de todas las salidas es la mayor (no se toman en cuenta las salidas inhibidas) y colocar en la variable *J* el número de salida que resultó ganadora.
Si todas las salidas son iguales se procede a asignar la primera de las salidas como la neurona ganadora.

Paso 10

Recalcular la capa x:

$$x_i = s_i * t_{Ji} \qquad \textbf{Ecuación 2.13}$$

Paso 11

Calculando Magnitud:

$$\|x\| \|r\| = \sum_i x_i \qquad \textbf{Ecuación 2.14}$$

$$\|s\| = \sum_i s_i \qquad \textbf{Ecuación 2.15}$$

Se compara si:

$$\frac{\|X\|}{\|S\|} < \rho \qquad \textbf{Ecuación 2.16}$$

Entonces se inhibe o desactiva la salida ganadora $y_J = -1$ y se regresa al paso 7 hasta que se termine con todas las salidas. Si han pasado todas las entradas significa que la entrada no puede ser clasificada y se procede al paso 14.

Paso 12

Se compara si

$$\frac{\|X\|}{\|S\|} > \rho \qquad \textbf{Ecuación 2.17}$$

Entonces el patrón se ha identificado, por lo que se procede a poner el reset en falso.

Paso 13

Se modifican los pesos:

$$b_{iJ}(new) = \frac{Lx_i}{L - 1 + \|x\|} \qquad \textbf{Ecuación 2.18}$$

$$t_{Ji} = x_i \qquad \textbf{Ecuación 2.19}$$

Paso 14

Se verifica si ya se mostraron todos los patrones a enseñar. En caso de que no, se regresa al paso 3, en caso de que sí se termina el proceso de aprendizaje.

2.6.4 **Red ART2.**

La red ART 2 es una red heteroasociativa de aprendizaje no supervisado que acepta valores continuos en su entrada. A fin de conseguir clasificar patrones de valor continuo debe realizar una serie de cálculos que vuelven más complejo el algoritmo de entrenamiento. Para comenzar la capa de entrada es una combinación de funciones de normalización y supresión de ruido.

La arquitectura típica de la red ART2 se muestra en la figura 12. La capa F1 está formada por seis tipos de unidades (*W, X, U, V, P y Q*) de modo que por cada una de las entradas de la red hay seis unidades involucradas en la generación de las señales enviadas a la siguiente capa. Un elemento adicional entre *W* y *X* recibe señales de todas las unidades W y calcula la norma o magnitud del vector *W* y envía ese resultado a todas la unidades de la subcapa *X*. Mecanismos similares de normalización se encuentran entre las unidades $P \rightarrow Q$ y $V \rightarrow U$. Las unidades P actúan como interfaces hacia la capa *F2* de esta red, y cada una de las unidades Pi está conectada a todas las neuronas de salida por medio de dos conexiones de pesos, una ascendente (bottom→up) y una descendente (top→down) y es en estas conexiones dobles que se almacenan los prototipos de cada uno de los grupos formados durante el funcionamiento de la red. La acción de la capa *F2* es de competencia, del tipo el ganador-toma-todo es decir que solamente una de las neuronas de salida (la ganadora) tendrá su salida activa a la vez.

2.6.5 Arquitectura de la Red ART2.

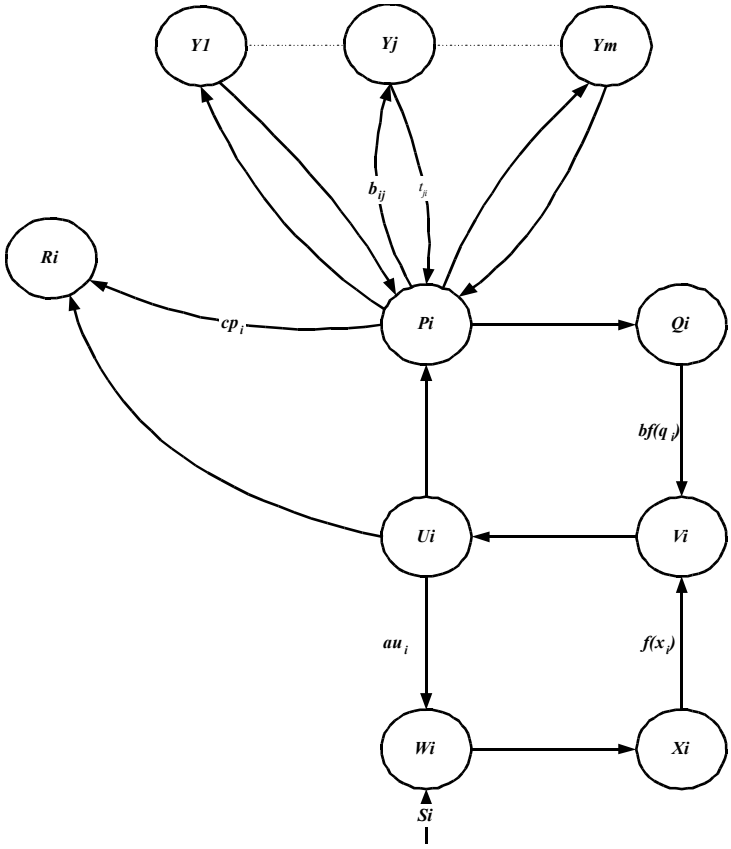


Figura 12 Arquitectura de la red ART 2

2.6.6 Algoritmo de Entrenamiento y Funcionamiento:

El ciclo de aprendizaje para un patrón de entrada inicia con el cálculo de la función de la subcapa *U* (que no es más que la salida normalizada de la subcapa *V*). Cada elemento *Ui* envía una señal a sus respectivos elementos *Wi* y *Pi*, el nivel de activación de estas unidades es entonces calculado. Los elementos *Wi* suman la señal recibida de *Ui* y de la señal de entrada *Si*.

Pi suma las señales provenientes de *Ui* y la señal descendente generada en caso de hallarse activa alguna de las neuronas de la capa F2. La activación de *Xi* y *Qi* son, respectivamente, las versiones normalizadas de *Wi* y *Pi*. Una función de activación, es aplicada a estas unidades y el resultado es enviado a las unidades Vi, donde es realizada la adición de las señales provenientes de la subcapa *X* y *Q*, con lo que se completa el ciclo de actualización de la capa *F1.*

La función de activación utilizada en todas las pruebas realizadas fue la siguiente:

$$f(x)=\begin{cases} x\,;\Leftrightarrow x\geq\theta \\ 0\,;\Leftrightarrow x<\theta \end{cases}$$

Ecuación 2.20

La razón que justifica el uso de esa función es que, de acuerdo a los creadores de esta red (Carpenter & Grossbreg 1987), las unidades de las capas *U* y *P* alcanzan el equilibrio después de tan solo 2 ciclos de actualización de la capa *F1*, y es además la recomendada por los autores que han realizado publicaciones sobre el tema (Fausett 1994, Hilera).

Una vez que la capa *F1* ha alcanzado el equilibrio, las unidades *P* envían sus señales a la capa *F2*, donde tiene lugar la competencia para escoger al candidato a aprender el patrón de entrada.

También las unidades de las subcapas *U* y *P* envían señales hacia las unidades del bloque de control de reset y es en base a la respuesta de ese bloque que la neurona de salida de un grupo o clúster puede ser aceptada o rechazada como no válida por presentar poco grado de semejanza entre los patrones.

Si ese es el caso la unidad rechazada es inhibida, (es decir, su salida es fijada en -1 para excluirla de futuras competencias) y la unidad que tenga la siguiente salida más grande será propuesta como candidata a aprender el patrón de entrada. Ese proceso se repite hasta que el patrón es admitido en un clúster o cuando ya se ha agotado la cantidad máxima de clusters permitidos. El aprendizaje ocurre únicamente si un patrón es aceptado como parte de un clúster de la red.

Descrito en forma más clara el proceso de aprendizaje de una red ART 2 es como sigue:

2.6.7 Algoritmo de Aprendizaje.

Paso 0. Inicializar parámetros de la red:

n: Número de entradas de la red (capa *F1*)

m: Número máximo de clusters (capa *F2*)

a, b: Pesos fijos de la capa *F1*. Inicializarlos a cero produce inestabilidad

c: Peso fijo usado para verificar Reset. Un valor pequeño de *c* proporciona un rango mayor de operación al parámetro de vigilancia *ρ*.

d: Activación de unidad ganadora de la capa *F2*. Los valores escogidos de c y d deben satisfacer la siguiente inecuación:

$$\frac{cd}{1-d} \leq 1$$ **Ecuación 2.21**

e: Un valor muy pequeño que se usa para prevenir la división entre cero.

θ: Parámetro de supresión de ruido utilizado en la función de activación de las neuronas. Un valor típico es:

$$1/\sqrt{n}$$ **Ecuación 2.22**

ρ: Parámetro de vigilancia. Junto con los valores de los pesos ascendentes, b_{ij}, determina la cantidad de clusters que se forman durante el aprendizaje. valores por debajo de 0.7 no difieren de los resultados obtenidos al fijarlo a valores tan bajos como $\rho = 0$.

t_{ji}: Pesos descendentes (F2→F1), deben ser inicializados a 0 para asegurarse que

no ocurrirá un Reset para el primer patrón admitido en un clúster.

b_{ij}: Pesos ascendentes (F1→F2), deben ser escogidos cumpliendo la siguiente desigualdad:

$$b_{ij} \leq \frac{1}{(1-d)\sqrt{n}} \qquad \textbf{Ecuación 2.23}$$

Paso 1.

Realizar pasos del 2 al 10 el número especificado de veces de acuerdo a la cantidad de patrones de ejemplo disponibles.

Paso 2.

Actualizar los valores de las unidades de la capa *F1* con:

$$u_i = q_i = p_i = 0 \qquad \textbf{Ecuación 2.24}$$

$$w_i = s_i + a * u_i \qquad \textbf{Ecuación 2.25}$$

$$\mathrm{x_i} = \frac{\mathrm{w_i}}{(\mathrm{e} + \| \mathrm{w} \|)} \qquad \textbf{Ecuación 2.26}$$

$$v_i = f(x_i) + b * f(q_i) \qquad \textbf{Ecuación 2.27}$$

$$u_i = \frac{v_i}{(e + \| v \|)} \qquad \textbf{Ecuación 2.28}$$

$$p_i = u_i + d * t_{ji} \qquad \textbf{Ecuación 2.29}$$

$$q_i = \frac{p_i}{(e + \| p \|)} \qquad \textbf{Ecuación 2.30}$$

Paso 3.

Actualizar nuevamente los valores en la capa *F1* con los resultados del paso anterior

Paso 4.

Calcular las señales de la capa F2:

$$y_j = \sum_i b_{ij} p_i$$ **Ecuación 2.31**

Paso 5.

Mientras *ResetFlag* sea igual a "verdadero" repetir los pasos 6→7

Paso 6.

Encontrar la unidad ganadora, Y_J, de la capa F2.
Donde el valor de J es tal que $y_J \geq y_j$ para j = 1, 2,… m)

Paso 7.
Verificar el Reset:

$$u_i = \frac{v_i}{(e + \|v\|)}$$ **Ecuación 2.32**

$$p_i = u_i + d * t_{ji}$$ **Ecuación 2.33**

$$r_i = \frac{u_i + c * p_i}{e + \|u\| + c\|p\|}$$ **Ecuación 2.34**

Paso 8.

Se realiza las siguientes condiciones:

$$\text{Si } \|r\| < \rho - e \qquad \textbf{Ecuación 2.35}$$

Entonces:

Resetflag = "verdadero" Y_J = -1. Repetir Paso 6.

Resetflag = "falso" se hace la siguiente condición

$$\text{Si } \|r\| \geq \rho - e \qquad \textbf{Ecuación 2.36}$$

Entonces:

Resetflag = "verdadero" Repetir Paso 3 y continuar con el paso 9

Resetflag = "falso" continuar con el paso 9

Si no se cumple con la ecuación 2.28 se procede a marcar la neurona ganadora con –1 y se hace la pregunta si ya han pasado todas las salidas. Si la respuesta es no se procede al paso 5, en caso contrario se debe desplegar un mensaje anunciando que el patrón no puede ser reconocido y se procede al paso 10

Paso 9.

Actualizar los pesos de la unidad ganadora J

$$T_{\text{ji}} = \frac{u_i}{1 - d} \qquad \textbf{Ecuación 2.37}$$

$$b_{ij} = \frac{u_i}{1 - d} \qquad \textbf{Ecuación 2.38}$$

Paso 10.

Verificar si se ha alcanzado el número de iteraciones previstas.

2.6.8 **Red Hopfield.**

En la década de los 80´s con el fin de estudiar procesos que involucran sistemas gobernados por ecuaciones diferenciales no lineales surge la teoría clásica de control geométrico basada en la geometría diferencial; simultáneamente renace el estudio de las redes neuronales debido al redescubrimiento del algoritmo Backpropagation, este hecho sumado al fracaso de las metodologías tradicionales aplicadas a la inteligencia artificial y a la disponibilidad de herramientas computacionales de bajo costo permitieron el desarrollo las redes neuronales recurrentes cuya principal aplicación es el control e identificación de sistemas no lineales.

Este desarrollo es posible debido a que las propiedades matemáticas de las redes recurrentes están enmarcadas en las mismas propiedades que fundamentan el control geométrico, la primera red neuronal recurrente de naturaleza dinámica fue propuesta por Hopfield en 1984 bajo el contexto de las memorias asociativas.

Las características son:

- Solo tiene una capa.
- Los valores pueden ser binarios o análogos, dependiendo de la función que se utilice a la salida de las neuronas.
- Cada neurona de la red se encuentra conectada a todas las demás neuronas de la red.
- En algunos casos existen conexiones de una neurona hacia si misma.
- Los pesos asociados a las conexiones entre pares de neuronas son simétricos.
- Aprendizaje fuera de línea (off line).
- Es un tipo de red auto asociativa
- La cantidad de neuronas en la red depende del número de entradas que se requiera.
- El número de entradas es igual al número de salidas.

La desventaja principal de esta Red:

Es limitado el número de informaciones que puede aprender (almacenar). Se sugiere que se tome en cuenta que la capacidad de la red está limitada de acuerdo a lo siguiente:

$$capacidad = \begin{cases} 0.138N \\ \dfrac{N}{4 * \ln(N)} \end{cases}$$ **Ecuación 2.39**

Donde:
N: Es la cantidad de neuronas que tiene la red.

NOTA: la primera ecuación es para una buena aproximación, en cambio la segunda es para una recuperación perfecta (idealmente, ya que aunque se cumpla el requisito anterior, no siempre se puede garantizar que la red realice una asociación correcta entre una entrada y una de las informaciones almacenadas)

La red utilizada en nuestra investigación es de tipo digital, cuya característica principal es la función de activación a la salida de la neurona. (Escalón Bipolar o escalón Unitario)

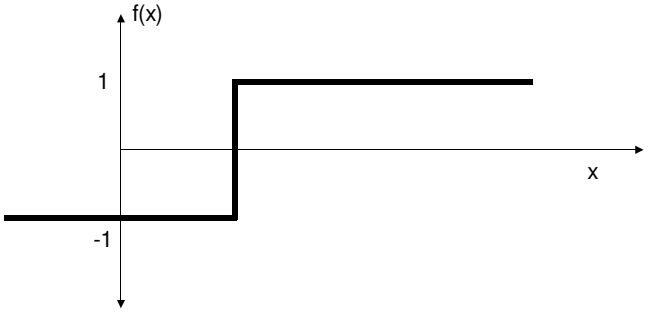


Figura 13. Escalón Bipolar

$$f(x) = \begin{cases} 1 \Rightarrow S(t) < \theta \\ F(t) \Rightarrow S(t) = \theta \\ -1 \Rightarrow S(t) > \theta \end{cases}$$ **Ecuación 2.40**

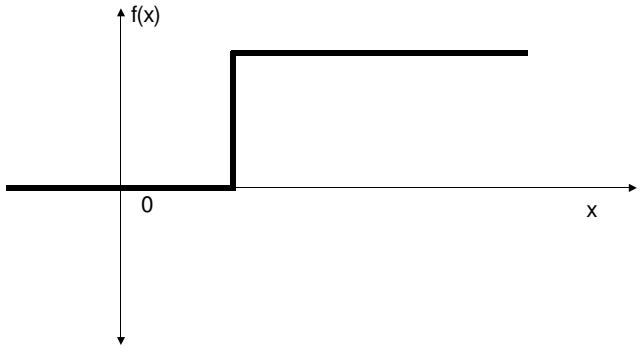


Figura 14. Escalón Unitario

$$f(x)=\begin{cases} 1 \Rightarrow S(t) < \theta \\ F(t) \Rightarrow S(t) = \theta \\ 0 \Rightarrow S(t) > \theta \end{cases}$$ **Ecuación 2.41**

2.6.9 **Arquitectura de la Red Hopfield.**

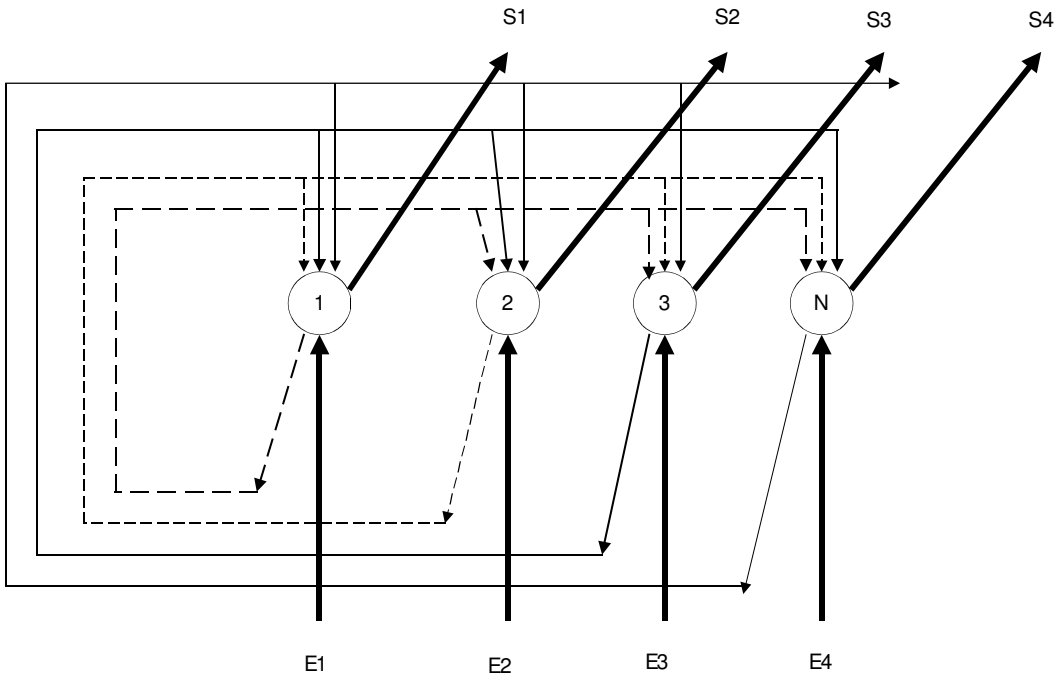


Figura 15 .Arquitectura de la red Hopfield.

2.6.10 **Algoritmo de Aprendizaje.**

Este algoritmo consiste básicamente en determinar el valor ideal de los pesos entre las conexiones de cada neurona, basándose en las siguientes ecuaciones:

$$W_{ij} = \begin{cases} \sum_{k=1}^{M} e_i^{(k)} e_j^{(k)} & 1 \le ij \le N; i \ne j \\ 0 & 1 \le ij \le N; i \ne j \end{cases}$$

Ecuación 2.42

La ecuación 2.42, es válida solo si se toma la función escalón bipolar, en donde:

W_{ij} : Es el peso asociado a la conexión entre la neurona j y la neurona i, y que es exactamente igual a W_{ji}.

$e_i^{(k)}$: Es el valor de la componente i-ésima del vector correspondiente a la información k-ésima que debe aprender la red.

M: Es el número de patrones a aprender.

$$Wij = \begin{cases} \sum_{k=1}^{M} (2 * e_i^{(k)} - 1)(2 * e_j^{(k)} - 1) \Leftrightarrow i \ne j \\ 0 \Leftrightarrow i = j \end{cases}$$

Ecuación 2.43

La ecuación 2.43, es válida para la función escalón unitario.

Esto también se puede representar de forma matricial la que estaría formada por los pesos.

$$W = \begin{bmatrix} W_{11} & W_{21} & W_{31} & \cdots & W_{N1} \\ W_{12} & W_{22} & W_{32} & \cdots & W_{N2} \\ & & \cdots & & \\ W_{1N} & W_{2N} & W_{3N} & \cdots & W_{NN} \end{bmatrix}$$

Ecuación 2.44

Esta matriz es simétrica, al cumplirse que $W_{ij} = W_{ji}$ y tiene una diagonal con valores nulos debido a la no existencia de conexiones autorrecurrentes ($W_{ii} = 0$).

Paso 1.

Ingreso del conjunto de los M Vectores que representan las informaciones que ha de aprender la red:

$$E_1 = \left[e_1^{(1)}, e_2^{(1)}, ..., e_N^{(1)} \right]$$

$$E_2 = \left[e_1^{(2)}, e_2^{(2)}, ..., e_N^{(2)} \right]$$

$$E_M = \left[e_1^{(M)}, e_2^{(M)}, ..., e_N^{(M)} \right] \qquad \textbf{Ecuación 2.45}$$

Paso 2.

Utilizando esta notación, entonces el aprendizaje consistiría en la creación de la matriz de pesos a partir de los M vectores o informaciones de entrada que se enseñan a la red. Esto se expresaría como:

$$W = \sum_{k=1}^{M} \left[E_k^T E_k - I \right] \qquad \textbf{Ecuación 2.46}$$

Donde:

La matriz E_k^T es la transpuesta de la E_k

I: Es la matriz identidad de dimensiones NxN que anula los pesos de las conexiones auto - recurrentes (W_{ii})

Paso 3.

Se toma la salida anterior como entrada y se repite el paso 2 hasta que la salida se mantenga constante o con modificaciones aceptables.

2.6.11 **Red Kohonen.**

T. Kohonen presentó en 1982 un sistema con un comportamiento semejante al del cerebro. Se trataba de un modelo de red neuronal con capacidad para formar mapas de características, de forma similar a como ocurre en el cerebro. En él, hay neuronas que se organizan en muchas zonas, de forma que las informaciones captadas del entorno a través de los órganos sensoriales se representan internamente en forma de mapas bidimensionales.

Este modelo tiene dos variantes, denominadas LVQ (*Learning Vector Quantization*) y TPM (*Topology-Preserving Map*). Ambas forman mapas topológicos para establecer características comunes entre las informaciones de entrada.

2.6.12 Arquitectura de la Red Kohonen.

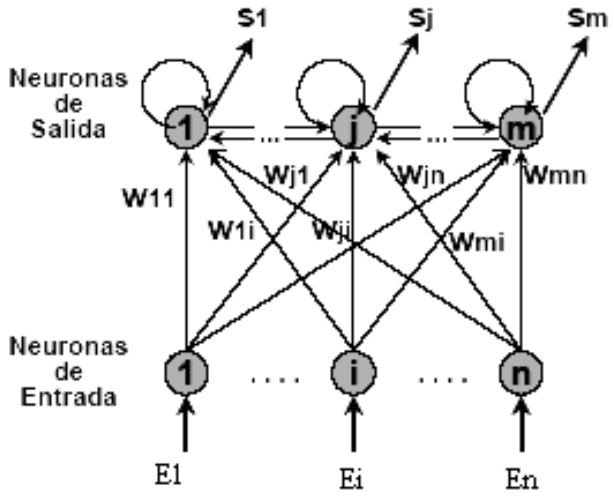


Figura 16. Diagrama de la red Kohonen.

2.6.13 Funcionamiento de la Arquitectura.

El funcionamiento es relativamente simple. Cuando se presenta a la entrada una información *Ek,* cada una de las *M* neuronas de la capa de salida la recibe a través de la conexiones *feedforward* con pesos *wji.* También estas neuronas reciben las correspondientes entradas debidas a las conexiones laterales con el resto de las neuronas de salida y cuya influencia dependerá de la distancia a la que se encuentren.

Se trata de una red de tipo competitivo, ya que al presentar una entrada *Ek,* la red evoluciona a una situación estable en la que se activa una neurona de salida, la vencedora.

En la fase de funcionamiento, lo que se pretende es encontrar el dato aprendido más parecido al de entrada, para averiguar qué neurona se activará y sobre todo, en qué zona del espacio bidimensional de salida se encuentra.

Lo que hace la red en definitiva es realizar una tarea de clasificación, ya que la neurona de salida activada ante una entrada representa la clase a la que pertenece dicha información de entrada. Además, como ante otra entrada parecida se activa la misma neurona de salida, u otra cercana, debido a la semejanza entre las clases, se garantiza que las neuronas topológicamente próximas sean sensibles a entradas físicamente similares. Por tanto, esta red es especialmente útil para establecer relaciones entre conjuntos de datos.

2.6.14 Aprendizaje de Kohonen.

Es de tipo OFF LINE, por lo que se distingue una etapa de aprendizaje y otra de funcionamiento. También utiliza un aprendizaje no supervisado de tipo competitivo. Sólo una neurona de la capa de salida se activa ante la entrada, ajustándose los pesos de las conexiones en función de la neurona que ha resultado vencedora.

Durante la etapa de entrenamiento, se presenta a la red un conjunto de informaciones de entrada para que ésta establezca, en función de la semejanza entre los datos, las diferentes categorías (una por neurona de salida) que servirán

durante la fase de funcionamiento para realizar clasificaciones de nuevos datos que se presenten a la red. Los valores finales de los pesos de las conexiones entre cada neurona de la capa de salida con las de entrada se corresponderán con los valores de los componentes del vector de aprendizaje que consigue activar la neurona correspondiente.

El aprendizaje no concluye después de presentarle una vez todos los patrones de entrada, sino que habrá que repetir el proceso varias veces para refinar el mapa topológico de salida, de tal forma que cuantas más veces se presenten los datos, tanto más se reducirán las zonas de neuronas que se deben activar ante entradas parecidas, consiguiendo que la red pueda realizar una clasificación más selectiva.

2.6.15 Algoritmo de Aprendizaje.

Sea N el número de neuronas de entrada y M el número de neuronas de salida.

Objetivo: establecer los valores de los pesos de las conexiones.

Paso 1.

Inicializar los pesos, *wji*, con valores aleatorios pequeños, fijando la zona inicial de vecindad entre las neuronas de salida.

Paso 2.

Presentar a la red una información de entrada en forma de vector $Ek = (e_1, ..., e_N)$, cuyas componentes, e_i, sean valores continuos.

Paso3.

Determinar la neurona vencedora de la capa de salida: será aquella cuyo vector de pesos, *Wj,* sea el más parecido a la información de entrada Ek. Recordemos que las componentes de *Wj* son los valores de los pesos de las conexiones entre esa neurona, *j*, y cada una de las neuronas de entrada. Para ello se calculan las distancias entre los vectores *Ek* y *Wj* para cada neurona de salida. La expresión matemática sería la siguiente:

$$d_j = \sum_{i=1} (ei - wji)^2 \quad \text{para } 1 \leq j \leq M \qquad \textbf{Ecuación 2.47}$$

Paso 4.

Una vez localizada la neurona vencedora, j*, se *actualizan los pesos de sus conexiones* de entrada y también los de las neuronas vecinas (las que pertenecen a su zona de vecindad, $Zona_j$). Lo que se consigue es asociar la información de entrada con una cierta zona de la capa de salida.

$$w_{ji}(t+1) = w_{ji}(t) + \beta(t)\,[e_i(k) - w_j * i(t)] \qquad \textbf{Ecuación 2.48}$$

Para j perteneciente a $Zona_{j^*}(t)$

El término *ß(t*) es un parámetro de ganancia o coeficiente de aprendizaje, con un valor entre 0 y 1 que decrece con el número de iteraciones del proceso de entrenamiento. Puede utilizarse la expresión:

$$\beta(t) = 1/t \qquad \textbf{Ecuación 2.49}$$

Paso 5.

El proceso se debe *repetir*, volviendo a presentar todo el juego de patrones de aprendizaje E_1, E_2, .., un mínimo de 500 veces (t = 500).

2.6.16 Red Contrapropagación.

La red de contrapropagación (CPN) es una de las redes de más reciente desarrollo. La CPN no es tanto un nuevo descubrimiento sino mas bien una nueva combinación de dos tipos de redes existentes con anterioridad. Hecht-Nielsen (Hecht-Nielsen, 1990) sintetizó la CPN combinando una estructura conocida como red competitiva con la estructura outstar de Grossberg (Grossberg, 1982; Hecht-Nielsen, 1987a; Hecht-Nielsen, 1987b), obteniendo así lo que se llama red de contra propagación.

El funcionamiento de la red se muestra en la figura 17. Dado un conjunto de vectores *(x1, y1), (x2,y2), ...,(xL,yL),* la red puede aprender a asociar a un vector *X* en la capa de entrada con un vector Y en la capa de salida. Si la relación entre *X* e *Y* se puede definir mediante una función continua,ϕ, tal que $Y=\phi(X)$, entonces la red aprenderá a aproximar esa correspondencia para todo valor de *X* en el intervalo especificado por el conjunto de vectores de entrenamiento.

Una red Contra propagación realiza durante su entrenamiento la aproximación de los vectores de entrada, dando como resultado la construcción de una tabla de búsqueda. De esta forma un gran número de puntos de entrenamiento pueden ser comprimidos y hacer más fácil el manejo de los datos de entrada de la tabla.

La red de contrapropagación debe de ser entrenada en dos fases. Durante la primera fase, los vectores de entrada son agrupados. Para realizar estas agrupaciones se utiliza la distancia Euclidea o el producto métrico. Durante la segunda fase de entrenamiento, los pesos desde las neuronas de la segunda capa hasta la capa de salida son actualizados para producir la salida deseada. Existen dos tipos de red contra propagación: la red de contrapropagación completa y la red de contra propagación hacia delante.

La red de contrapropagación hacia delante es una versión simplificada de la red de contrapropagación completa. La contrapropagación hacia delante realiza la aproximación de la función $y=\phi(x)$, la cual no necesariamente debe de poseer su

inversa. Esto significa que la red solamente podrá realizar la aproximación de x hacia y, pero no de *y* hacia *x.*

La red de contra propagación hacia delante difiere de la contrapropagación completa por el hecho que solamente se utilizan los vectores *x* para formar las agrupaciones o clasificaciones de las vectores dentro de las neuronas de capa Kohonen durante la primera fase de entrenamiento. La red de contrapropagación hacia delante calcula la distancia Euclidea entre el vector de entrada x y el vector de pesos de las neuronas de la capa Kohonen, sin embargo también es posible utilizar el producto métrico.

2.6.17 Arquitectura de la Red Contrapropagación hacia Delante.

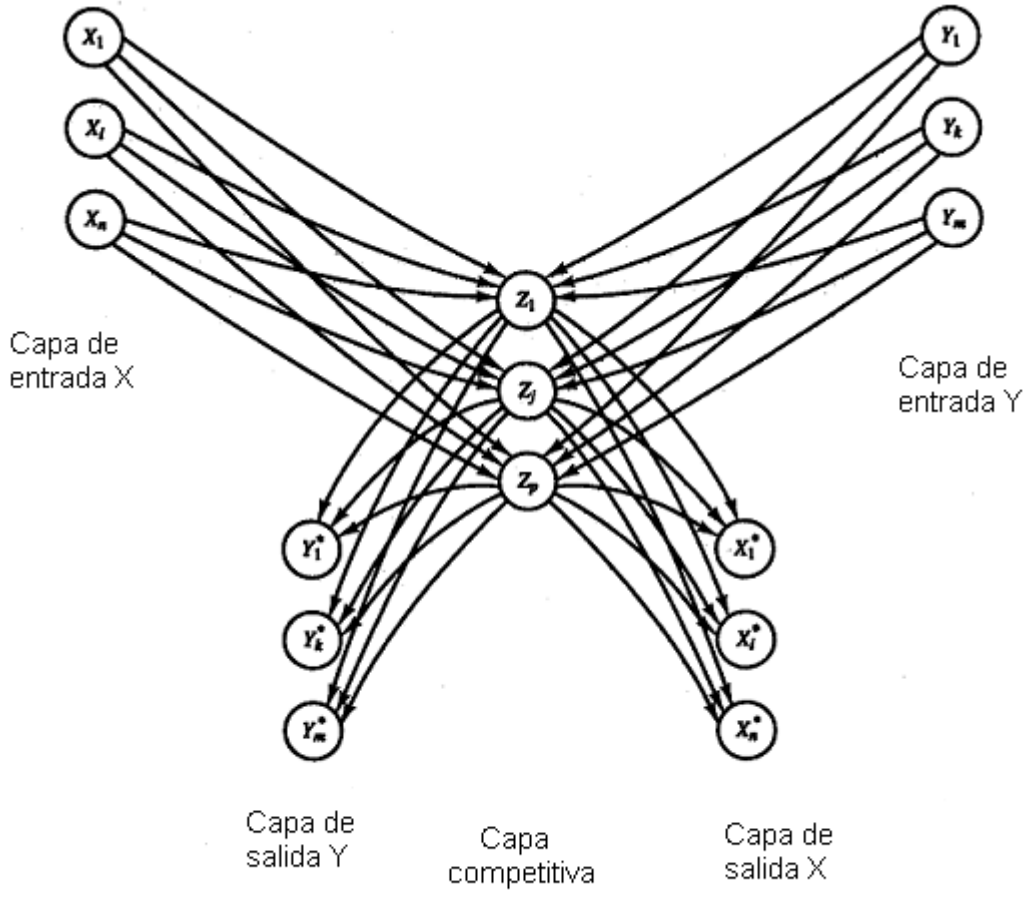


Figura 17. Arquitectura de la red Contrapropagación hacia delante.

2.6.18 Algoritmo de Aprendizaje.

El proceso de entrenamiento de esta red consiste de varios pasos que se describen a continuación:

Paso 0:

Se inicializan los pesos y las razones de aprendizaje: $\alpha = 1/t = \beta$. Sin embargo Hecht-Nielsen sugiere valores iniciales de $\alpha = 0.6 \quad \beta = 0.1$

Paso 1:

Mientras la condición de paro de la primera fase es falsa, realizar los pasos del 2 al 7

Paso 2:

Para cada vector de entrenamiento x, realizar los pasos del 3 al 5

Paso 3:

Seleccionar un vector de entre todos los vectores de entrada que deban ser utilizados para el entrenamiento. La selección deberá ser aleatoria. Se normaliza el vector de entrada y se aplica a la capa de entrada.

Paso 4:

Se determina cual es la neurona ganadora de la competición utilizando la distancia Euclidea entre el vector de entrada *X* y el vector de pesos *V* (gana la menor distancia entre *X* y *V*), y a la neurona ganadora se le asignará el sub-índice *J*

Paso 5:

Se actualiza el peso de la neurona de la capa competitiva (Z_J) que ganó la competición, utilizando la siguiente ecuación:

$$V_{ij}\ (nuevo) = (1-\alpha)\ V_{ij}\ (viejo) + \alpha X_i \qquad \textbf{Ecuación 2.50}$$

$$i=1,\ldots\ldots n$$

Paso 6:

Se reduce la razón de aprendizaje *α.*

Paso 7:

Se repiten los pasos del 3 al 6, hasta que todos los vectores de entrada han sido clasificados correctamente. Cuando esta situación suceda, solo una unidad Z_J ganará la competición para cada grupo de vectores.

Paso 8:

Mientras la condición de paro para la segunda fase sea falsa, realizar los pasos del 9 al 15. (α tendrá un valor pequeño durante la ejecución de la segunda fase de entrenamiento).

Paso 9:

Para cada vector de entrenamiento *X*, *Y*, realizar los pasos del 10 al 13.

Paso 10:

Aplicar un vector de entrada normalizado *X* a la capa de entrada *X*. Aplicar el respectivo vector *Y* normalizado al capa de salida Y.

Paso 11:

Se encuentra la neurona ganadora de la capa competitiva. Por medio de la distancia Euclidea de *X* contra *V.*

Paso 12:

Se actualizan los pesos de la neurona ganadora Z_J (α es pequeño):

$$V_{ij}\ (nuevo) = (1-\alpha)\ V_{ij}\ (viejo) + \alpha X_i \qquad \textbf{Ecuación 2.51}$$

$$i=1,\ldots\ldots n$$

Paso 13:

Se actualizan los pesos desde la neurona ganadora Z_J hacia la neurona de salida.

$$W_{jk}\ (nuevo) = (1-\beta)\ W_{jk}\ (viejo) + \alpha Y_k \qquad \textbf{Ecuación 2.52}$$

$$Z_{\_inj} = \sum_i X_i\ V_{ij} \qquad \textbf{Ecuación 2.53}$$

$$D_j = \sum_i (Xi - Vij)^2 \qquad \textbf{Ecuación 2.54}$$

$$W_{jk}\ (nuevo) = (1-\beta)\ W_{jk}\ (viejo) + \alpha Y_k \qquad \textbf{Ecuación 2.57}$$

$$k=1,\ldots\ldots n$$

Paso 14:

Se reduce la razón de aprendizaje *β*.

Paso 15:

Se verifica si todos los vectores de entrada *X* corresponden con sus salidas satisfactorias.

En los pasos 4 y 11

En caso de haber un empate de las neuronas ganadoras, se tomará a la neurona que presente el índice más pequeño

Para usar el producto interno, y encontrar a la neurona ganadora *Zj* dentro de la capa competitiva con la neurona de entrada más grande de entrada, se utiliza la ecuación 2.58:

$$Z_{\_inj} = \sum_i X_i\ V_{ij} \qquad \textbf{Ecuación 2.58}$$

Para utilizar la distancie Euclidea, y encontrar la neurona ganadora *Zj*, ganará el resultado menor

$$D_j = \sum_i (Xi - Vij)^2 \qquad \textbf{Ecuación 5.59}$$

Capitulo 3 – VALIDACIÓN DE RESULTADOS OBTENIDOS DE LA INVESTIGACIÓN.

3.1 VALIDACIÓN DE LA RED ART1 (Freeman).

Para ver el funcionamiento de este algoritmo, llevaremos a cabo el cálculo paso a paso, de un pequeño ejemplo tomando del Libro "Redes Neuronales, Algoritmos y Aplicaciones" del autor James A. Freeman/David M.Skapura (Pág. 331 – 334).

Seleccionaremos como dimensiones de F_1 y F_2 *a* $M = 5$ *y* $N = 6$, respectivamente. Los demás parámetros del sistema *son* $A_1 = 1$, $B_1 = 1.5$, $C_1 = 5$, $D_1 = 0.9$, $L = 3$, $\rho = 0.9$.

Se dan valores iniciales a los pesos de las unidades de F_1 , añadiendo un pequeño valor positivo (0.2 en este caso) a $(B_1 - 1)/D_1$. Todas las unidades de F_1 tienen entonces el vector de pesos:

$$\mathbf{z}_i = (0.756, 0.756, 0.756, 0.756, 0.756, 0.756,)^t$$

Puesto que: L = 3 y M = 5, los pesos de las unidades de F_2 reciben todos un valor inicial que es ligeramente menor que L/(L – 1 + M). Se asigna a todos los pesos el valor 0.429 – 0.1 = 0.329. Todos los valores de pesos son, por lo tanto:

$$\mathbf{z}_j = (0.329, 0.329, 0.329, 0.329, 0.329,)^t$$

Todas las actividades de F_2 reciben una actividad inicial igual a cero. Las actividades de F_1 reciben un valor inicial de $- B_1/(1 + C_1) = - 0.25$:

$$\mathbf{X}(0) = (-0.25, -0.25, -0.25, -0.25, -0.25,)^t$$

Ahora podemos empezar con el procesamiento en si. Comenzaremos por un sencillo vector de entrada:

$$\mathbf{I}_1 = (0, 0, 0, 1, 0)^t$$

Ahora seguiremos la secuencia fijada por el algoritmo:

1. Una vez aplicado el vector de entrada, las actividades de F_1 pasan a ser:

$$\mathbf{X}_1 = (0, 0, 0, 0.118, 0)^t$$

2. El vector de salida es $S = (0, 0, 0, 1, 0)^t$.

3. Propagando este vector de salida hacia F_2, las entradas netas a todas las unidades de F_2 serán idénticas:

$$\mathbf{T}_j = \mathbf{z}_j * \mathbf{S}$$

Entonces,

$$\mathbf{T} = (0.329, 0.329, 0.329, 0.329, 0.329, 0.329)^t$$

4. Como las actividades de todas las unidades son las mismas, se toma simplemente la primera unidad como ganadora. Entonces la salida de F_2 es:

$$\mathbf{U} = (1, 0, 0, 0, 0, 0)^t$$

5. Se propaga retrocediendo hasta F_1:

$$\mathbf{V}_i = \mathbf{z}_i * \mathbf{U}$$

Entonces:

$$\mathbf{V} = (0.756, 0.756, 0.756, 0.756, 0.756)^t$$

6. Se calculan los nuevos valores de las actividades de F_1:

$$\mathbf{X} = (-0.123, -0.123, -0.123, -0.123, -0.123)^t$$

7. Solo la cuarta unidad tiene una actividad positiva, así que las nuevas salidas son:

$$\mathbf{S} = (0, 0, 0, 1, 0)^t.$$

$$|S|/|\mathrm{I}| = 1 > \rho .$$

8. No hay restauración: se ha alcanzado la resonancia.

9. Se actualizan los pesos ascendentes de la unidad v_1 de F_2, según la regla de aprendizaje rápido. Dado que: z_1 de F_2 pasa a ser $(0, 0, 0, 1, 0)^t$. Por lo tanto, la unidad de v_1 codifica exactamente el vector de entrada. Lo que realmente nos interesa aquí, es la trama de pesos no nulos. Es completamente casual, que el único peso no nulo coincida, con el valor de la entrada, como se observará en breve.

10. Se actualiza los pesos de las conexiones descendentes. Solo el primer peso de cada una de las unidades F_1 resulta modificado. Si cada fila describe los pesos de una unidad, entonces la matriz de pesos para las unidades de F_1 es:

$$\begin{pmatrix} 0 & 0.756 & 0.756 & 0.756 & 0.756 \\ 0 & 0.756 & 0.756 & 0.756 & 0.756 \\ 0 & 0.756 & 0.756 & 0.756 & 0.756 \\ 1 & 0.756 & 0.756 & 0.756 & 0.756 \\ 0 & 0.756 & 0.756 & 0.756 & 0.756 \end{pmatrix}$$

Esto completa el ciclo para la primera trama de entrada. Apliquemos ahora una segunda trama que es el ortogonal a la primera:

$$\mathbf{I}_2 = (0, 0, 1, 0, 1)^t.$$

Se abreviarán un poco los cálculos:
Cuando la trama de entrada se propaga hasta F_2, las actividades resultantes son

$$\mathbf{T} = (0.000, 0.657, 0.657, 0.657, 0.657, 0.657,)^t.$$

Decididamente, la unidad 1 pierde. Se selecciona como ganadora la unidad 2. La salida de F_2 es **U** = $(0, 1, 0, 0, 0, 0)^t$.

Haciendo la propagación hacia atrás hasta F_1 se obtiene V = $(0.756, 0.756, 0.756, 0.756, 0.756)^t$ y X = $(-0.123, -0.123, 0.0234, -0.123, 0.0234)^t$.

La salida resultante coincide con el vector de entrada, $(0, 0, 1, 0, 1)^t$, a si que no hay restauración. (No hubo resonancia)

Los pesos de la unidad ganadora de F_2 reciben el valor $(0, 0, 0.75, 0, 0.75)^t$.
El segundo peso de todas las unidades de F_1 se ajusta de tal manera que la nueva matriz de pesos de F_1 sea:

$$\begin{pmatrix} 0 & 0 & 0.756 & 0.756 & 0.756 & 0.756 \\ 0 & 0 & 0.756 & 0.756 & 0.756 & 0.756 \\ 0 & 1 & 0.756 & 0.756 & 0.756 & 0.756 \\ 1 & 0 & 0.756 & 0.756 & 0.756 & 0.756 \\ 0 & 1 & 0.756 & 0.756 & 0.756 & 0.756 \end{pmatrix}$$

Como la referencia, la matriz de pesos de F_2 es la siguiente:

$$\begin{pmatrix} 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0.75 & 0 & 0.75 \\ 0.329 & 0.329 & 0.329 & 0.329 & 0.329 \\ 0.329 & 0.329 & 0.329 & 0.329 & 0.329 \\ 0.329 & 0.329 & 0.329 & 0.329 & 0.329 \\ 0.329 & 0.329 & 0.329 & 0.329 & 0.329 \end{pmatrix}$$

Veamos ahora lo que sucede cuando se aplica un vector que es un subconjunto de I_2, además, $I_3 = (0, 0, 0, 0, 1)^t$. Cuando se propaga hacia delante esta trama, las actividades de F_2 pasan a ser $(0, 0.75, 0.329, 0.329, 0.329, 0.329)^t$.

Obsérvese que en este caso gana la competición la segunda unidad, así que la salida de F_2 es $(0, 1, 0, 0, 0, 0)^t$. Volviendo a F_1, las entradas netas procedentes de la trama descendente son $V = (0, 0, 1, 0, 1)^t$, en este caso, las actividades de equilibrio son $X = (-0.25, -0.25, -0.087, -0.25, 0.051)^t$, con una única actividad positiva. La nueva trama de salida es $(0, 0, 0, 0, 1)^t$, que coincide exactamente con la trama de entrada, así que no se produce la restauración.

Aun cuando la unidad 2 de F_2 hubiese codificado anteriormente una trama de entrada, ahora se recodificará para adaptarse a la nueva trama de entrada, que es un subconjunto de la original. La nueva matriz de pesos tiene el aspecto siguiente. Para F_1, se tiene:

$$\begin{pmatrix} 0 & 0 & 0.756 & 0.756 & 0.756 & 0.756 \\ 0 & 0 & 0.756 & 0.756 & 0.756 & 0.756 \\ 0 & 0 & 0.756 & 0.756 & 0.756 & 0.756 \\ 1 & 0 & 0.756 & 0.756 & 0.756 & 0.756 \\ 0 & 1 & 0.756 & 0.756 & 0.756 & 0.756 \end{pmatrix}$$

Y para F_2:

$$\begin{pmatrix} 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0.329 & 0.329 & 0.329 & 0.329 & 0.329 \\ 0.329 & 0.329 & 0.329 & 0.329 & 0.329 \\ 0.329 & 0.329 & 0.329 & 0.329 & 0.329 \\ 0.329 & 0.329 & 0.329 & 0.329 & 0.329 \end{pmatrix}$$

Si volvemos al vector del conjunto mayor $(0, 0, 1, 0, 1)^t$, la propagación inicial hacia delante, hasta F_2, proporciona una actividad de (0.000, 1.000, 0.657, 0.657, 0.657, 0.657)^t, así, que vuelve a ganar la unidad 2. Volviendo a F_1, $V = (0, 0, 0, 0, 1)^t$ y las actividades de equilibrio serán $(-0.25, -0.25, -0.071, -0.25, 0.051)^t$. Las nuevas salidas son $(0, 0, 0, 0, 1)^t$. Esta vez llega una señal de restauración, puesto que $\frac{|s|}{|I_2|} = 0.5 < \rho$. Por lo tanto, la unidad 2 de F_2 queda eliminada de la compensación, y se repite el ciclo de búsqueda de coincidencia con el vector de entrada original restaurado en F_1.

La propagación hacia delante, por segunda vez, da lugar a unas actividades de F_2 iguales a $(0.000, 0.000, 0.657, 0.657, 0.657, 0.657)^t$, en donde se ha puesto a cero la actividad de la unidad 2 como consecuencia de la inhibición sostenida generada por el subsistema de orientación.

Ahora seleccionamos la unidad 3 como ganadora, y ésta codifica el vector de entrada. En presentaciones posteriores de vectores de los conjuntos menor y mayor, cada uno de ellos llegará directamente a la unidad apropiada de F_2, sin necesidad de búsqueda. Este resultado se puede verificar por cálculo directo con el ejemplo sencillo que se ha presentado en esta ocasión.

El ejemplo anterior se ejecutó utilizando el Simulador que hemos desarrollado, mostrando que los pesos finales alcanzados coinciden con los pesos que el autor ha definido como finales.

3.1.1 Resultado obtenido con el Simulador ANNS Versión 1.1 para la Red ART1.

Paso 1.

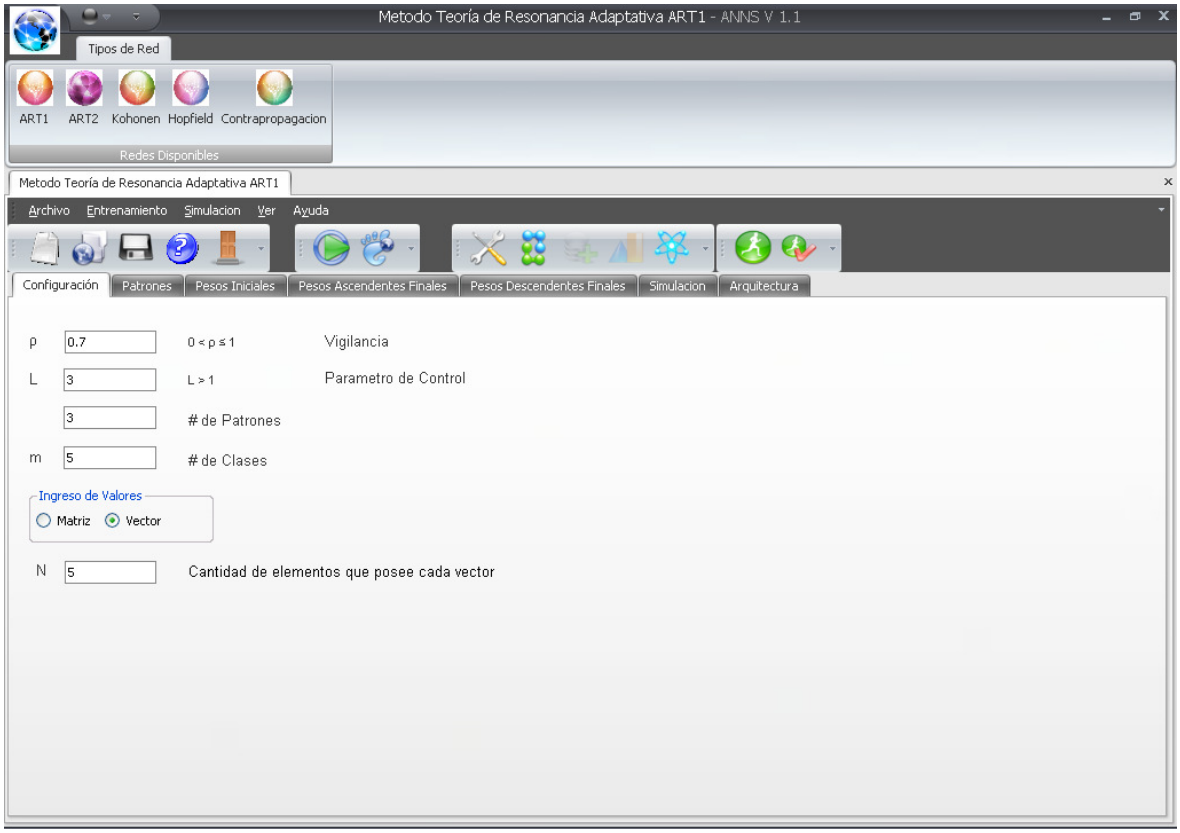


Figura 18. Ventana de configuración de parámetros de la Red Art1 (Freeman) .

Paso 2.

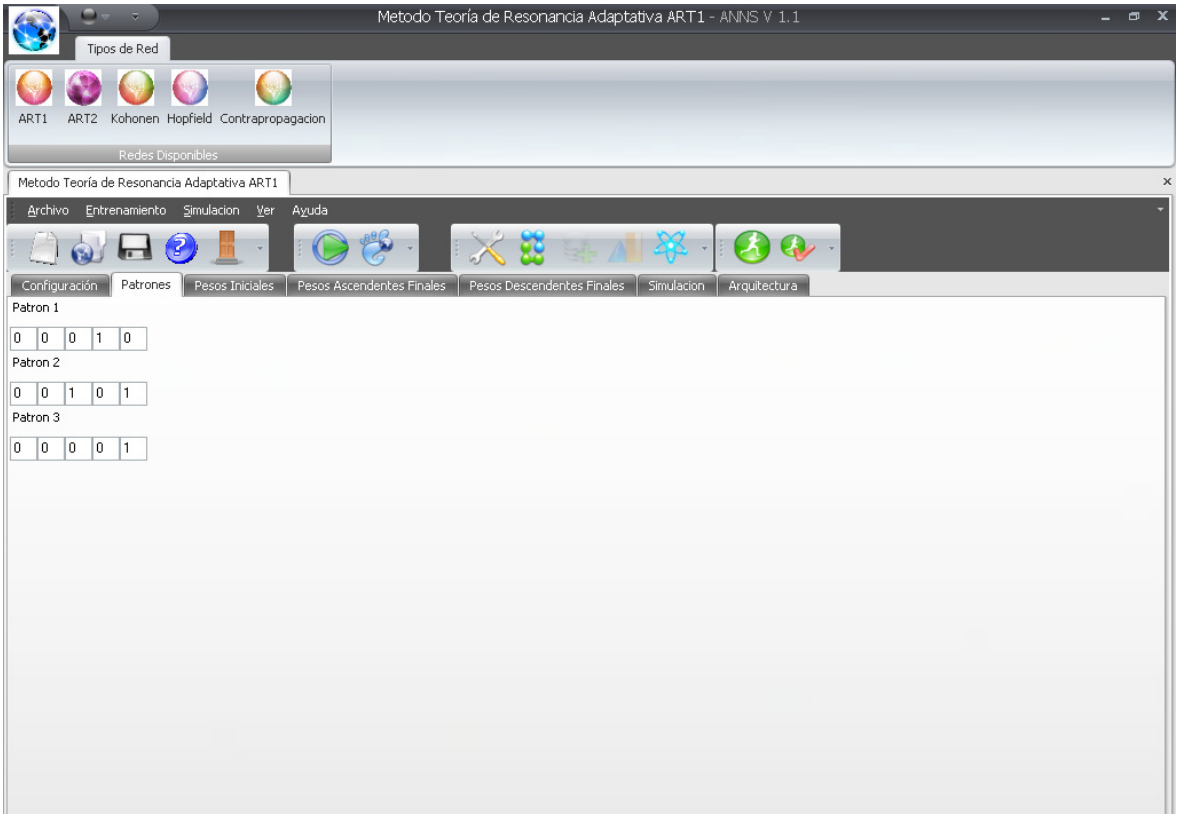


Figura 19. Ventana de patrones a entrenar de la Red Art1 (Freeman).

Paso 3.

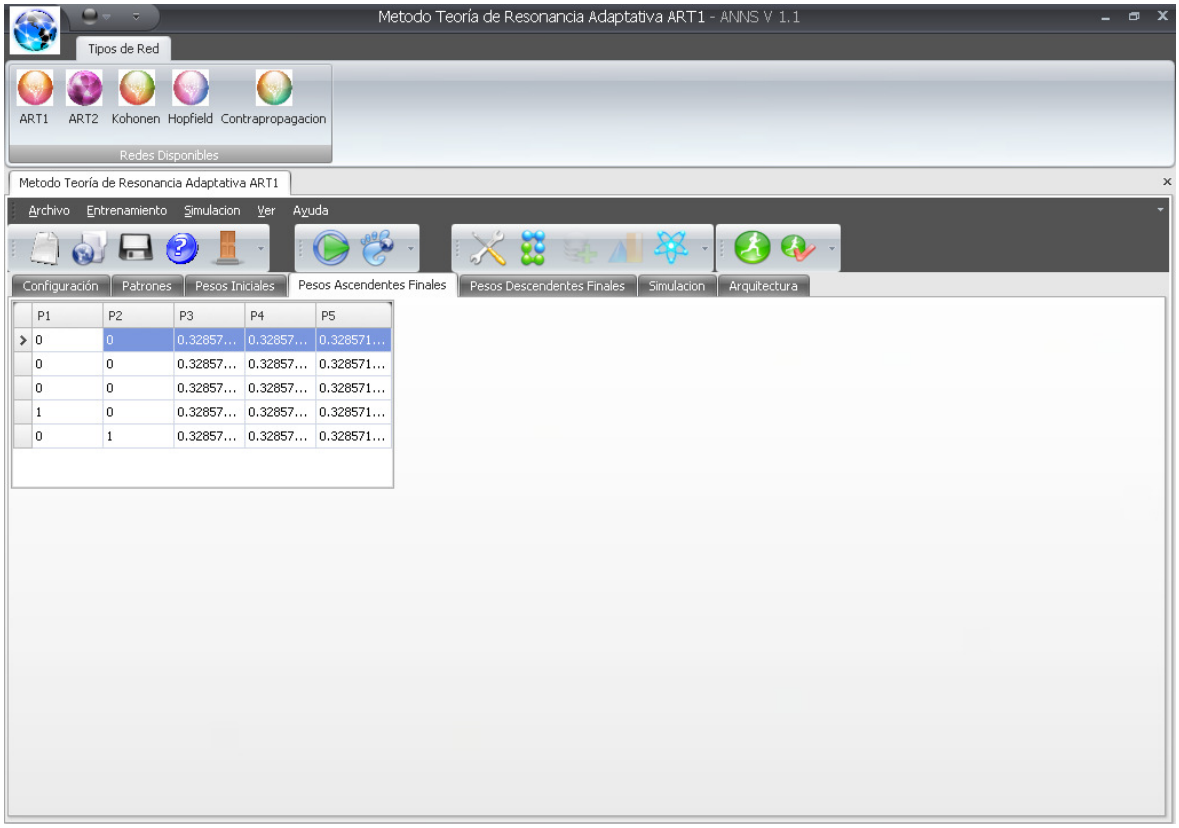


Figura 20. Ventana de pesos ascendentes finales de la Red Art1 (Freeman).

Paso 4.

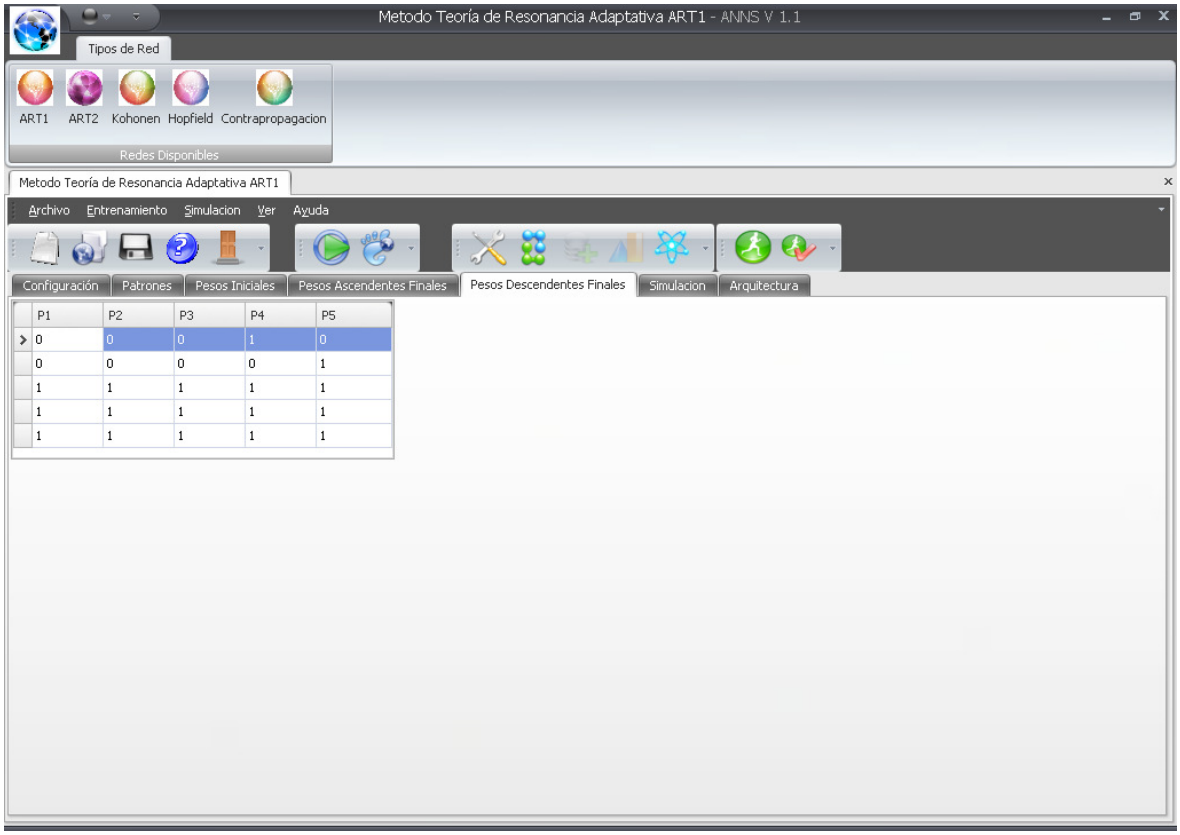


Figura 21. Ventana de pesos descendentes finales de la Red Art1 (Freeman).

3.2 VALIDACION DE LA RED ART1. (Fausett).

Para ver el funcionamiento de este algoritmo, realizaremos los siguientes cálculos un ejemplo del libro "Fundamentals of Neural Networks Architectures, Algorithms, and Applications" del autor Laurene Fausett (Pág. 230 – 236).
Se pide entrenar una red ART1 para que aprenda los elementos:

[1, 1, 0, 0]
[0, 0, 0, 1]
[1, 0, 0, 0]
[0, 0, 1, 1].

Inicialización de parámetros:

n = 4 Numero de componente que posee el vector de entrada
m = 3 Numero máximo de Clases
L = 2
ρ = 0.4

$$b_{ij} = \frac{2}{2-1+4} = 0.4$$

Como:

$$0 < b_{ij} < \frac{L}{L-1+n}$$
$$t_{ji} = 1$$

Entonces tomaremos un valor que esta dentro de ese rango en este caso tomaremos 0.2, para todos los valores:

$$bij = \begin{bmatrix} 0.2 & 0.2 & 0.2 \\ 0.2 & 0.2 & 0.2 \\ 0.2 & 0.2 & 0.2 \\ 0.2 & 0.2 & 0.2 \end{bmatrix}$$

Y los pesos $$t_{ji} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix}$$

Para el primer patrón (1, 1, 0, 0)

$s = (1,1,0,0)$

Luego $x = (1,1,0,0)$

Se calculan las salidas:

$$y_1 = 0.2*1 + 0.2*1 + 0.2*0 + 0.2*0 = 0.2$$
$$y_2 = 0.2*1 + 0.2*1 + 0.2*0 + 0.2*0 = 0.2$$
$$y_3 = 0.2*1 + 0.2*1 + 0.2*0 + 0.2*0 = 0.2$$

Como el reset es verdadero

Podemos ver que las salidas dan el mismo valor, se toma la primera salida como ganadora, J = 1

Calculando x

$x = s * t_{1i}; t_{1i} = (1,1,1,1)$

$x = (1,1,0,0)$

$\|x\| = 1+1+0+0 = 2$

$\|s\| = 1+1+0+0 = 2$

Entonces se realiza la división $\frac{\|x\|}{\|s\|} = \frac{2}{2} = 1$ y se compara con el factor de vigilancia, que para este es 0.4

$\frac{\|x\|}{\|s\|} < 0.4$ Como 1 es mayor que 0.4, entonces no cumple con la condición y se realiza la otra comparación:

$\frac{\|x\|}{\|s\|} \geq 0.4$, para esta condición se cumple que 1 es mayor que 0.4, entonces se actualizarán los pesos.

Sustituyendo valores nos quedaría:

Como J es fijo se realiza un barrido de i:

$$b_{iJ}(new) = \frac{Lx_i}{L - 1 + \|x\|}$$

$$bij(new) = \begin{bmatrix} 0.67 & 0.2 & 0.2 \\ 0.67 & 0.2 & 0.2 \\ 0 & 0.2 & 0.2 \\ 0 & 0.2 & 0.2 \end{bmatrix}$$

Y en el caso de $t_{Ji} = x_i$, entonces:

$$t_{ji} = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix}$$

Para el segundo patrón (0, 0, 0, 1)

$s = (0,0,0,1)$

Luego $x = (0,0,0,1)$

Se calculan las salidas:

$$y_1 = 0.67*0 + 0.67*0 + 0*0 + 0*1 = 0$$
$$y_2 = 0.2*0 + 0.2*0 + 0.2*0 + 0.2*1 = 0.2$$
$$y_3 = 0.2*0 + 0.2*0 + 0.2*0 + 0.2*1 = 0.2$$

Como el reset es verdadero, entonces:

Pasamos a evaluar cuál de las salidas es la mayor. Podemos ver que las salidas 2, 3 y 4 son iguales y mayores que la 1, lo que significa que no es similar a lo que se le enseño, por lo que se debe generar una nueva neurona ganadora, se toma la primera salida mayor como ganadora, J = 2

Calculando x

$$x = s * t_{2i}; t_{2i} = (1,1,1,1)$$
$$x = (0,0,0,1)$$

$$\|x\| = 0+0+0+1 = 1$$
$$\|s\| = 0+0+0+1 = 1$$;

Entonces de realiza la división $\frac{\|x\|}{\|s\|} = \frac{1}{1} = 1$ y se compara con el factor de vigilancia:

$\frac{\|x\|}{\|s\|} < 0.4$ Como el resultado fue 1, y es mayor que 0.4, entonces no se cumple la primera condición, por lo tanto se realiza la otra comparación $\frac{\|x\|}{\|s\|} \geq 0.4$ el resultado de la comparación, cumple con la segunda condición, por lo que se procede a actualizar los pesos,

Sustituyendo valores nos quedaría: como J es fijo se realiza un barrido de i

$$b_{iJ}(new) = \frac{Lx_i}{L-1+\|x\|}$$

$$bij(new) = \begin{bmatrix} 0.67 & 0 & 0.2 \\ 0.67 & 0 & 0.2 \\ 0 & 0 & 0.2 \\ 0 & 1 & 0.2 \end{bmatrix}$$

y en el caso de $t_{Ji} = x_i$ los cuales quedarían como

$$t_{ji} = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix}$$

Para el tercer patrón (1, 0, 0, 0)

$s = (1,0,0,0)$

Luego $x = (1,0,0,0)$

Se calculan las salidas

$$y_1 = 0.67*1 + 0.67*0 + 0*0 + 0*1 = 0.67$$
$$y_2 = 0*1 + 0*0 + 0*0 + 1*0 = 0.0$$
$$y_3 = 0.2*1 + 0.2*0 + 0.2*0 + 0.2*0 = 0.2$$

Como el reset es verdadero

Pasamos a evaluar cuál de las salidas es la mayor, podemos ver que es la salida 1, por lo que J = 1

Recalculando x

$x = s * t_{1i}; t_{1i} = (1,1,0,0)$

$x = (1,0,0,0)$

$\|x\| = 1+0+0+0 = 1$

$\|s\| = 1+0+0+0 = 1$

Entonces de realiza la división $\frac{\|x\|}{\|s\|} = \frac{1}{1} = 1$ y se compara con el factor de vigilancia

$\frac{\|x\|}{\|s\|} < 0.4$ Al comparar el resultado de la división

Se realiza la otra comparación $\frac{\|x\|}{\|s\|} \geq 0.4$ esta resulta que si es cierta, por lo que se procede a actualizar los pesos,

Sustituyendo valores nos quedaría, como J es fijo se realiza un barrido de i

$$b_{iJ}(new) = \frac{Lx_i}{L-1+\|x\|}$$

$$bij(new) = \begin{bmatrix} 1 & 0 & 0.2 \\ 0 & 0 & 0.2 \\ 0 & 0 & 0.2 \\ 0 & 1 & 0.2 \end{bmatrix}$$

Y en el caso de $t_{Ji} = x_i$ los cuales quedarían como

$$t_{ji} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix}$$

Para el cuarto patrón (0, 0, 1, 1)

$s = (0,0,1,1)$

Luego $x = (0,0,1,1)$

Se calculan las salidas

$$y_1 = 1*0+0*0+0*1+0*1 = 0.0$$
$$y_2 = 0*0+0*0+0*1+1*1 = 1.0$$
$$y_3 = 0.2*0+0.2*0+0.2*1+0.2*1 = 0.4$$

Como el reset es verdadero, entonces:

Pasamos a evaluar cuál de las salidas es la mayor, podemos ver que es la salida 2, por lo que J = 2

Calculando x

$x = s * t_{2i}; t_{2i} = (0,0,0,1)$

$x = (0,0,0,1)$

$\|x\| = 0+0+0+1 = 1$

$\|s\| = 0+0+1+1 = 2$

Entonces de realiza la división $\frac{\|x\|}{\|s\|} = \frac{1}{2} = 0.5$ y se compara con el factor de vigilancia.

$\frac{\|x\|}{\|s\|} < 0.4$ El resultado de la comparación no es cierta, entonces, se realiza la otra comparación $\frac{\|x\|}{\|s\|} \geq 0.4$ esta resulta ser cierta, por lo que se procede a actualizar los pesos,

Sustituyendo valores nos quedaría, como J es fijo se realiza un barrido de i

$$b_{iJ}(new) = \frac{Lx_i}{L-1+\|x\|}$$

$$bij(new) = \begin{bmatrix} 1 & 0 & 0.2 \\ 0 & 0 & 0.2 \\ 0 & 0 & 0.2 \\ 0 & 1 & 0.2 \end{bmatrix}$$

y en el caso de $t_{Ji} = x_i$ los cuales quedarían como

$$t_{ji} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix}$$

Las matrices anteriores, representan los pesos finales alcanzados, puesto que no hay más patrones que evaluar.

Como podemos ver el sistema no logró clasificar los cuatro patrones debido a que el factor de vigilancia es muy pequeño,

Hagamos ahora algunos cambios, probemos la red diseñada con el patrón de entrada inicial pero incrementemos el factor de vigilancia a 0.7

Patrón de prueba (0, 0, 1, 1)

$s = (0,0,1,1)$

Luego $x = (0,0,1,1)$

Se calculan las salidas

$$y_1 = 1*0+0*0+0*1+0*1 = 0.0$$
$$y_2 = 0*0+0*0+0*1+1*1 = 1.0$$
$$y_3 = 0.2*0+0.2*0+0.2*1+0.2*1 = 0.4$$

Como el reset es verdadero:

Pasamos a evaluar cuál de las salidas es la mayor, podemos ver que es la salida 2, por lo que J = 2

Calculando x

$x = s * t_{2i}; t_{2i} = (0,0,0,1)$

$x = (0,0,0,1)$

$$\|x\| = 0+0+0+1 = 1$$
$$\|s\| = 0+0+1+1 = 2$$

Entonces de realiza la división $\frac{\|x\|}{\|s\|} = \frac{1}{2} = 0.5$ y se compara con el factor de de vigilancia.

$\frac{\|x\|}{\|s\|} < 0.7$ Como el resultado de la comparación es verdadero, significa que esa salida quedará inhabilitada y se sigue comprobando con el resto de salidas.

Con las salidas se verifica de nuevo cuál es el mayor, como se puede observar las salidas 3 y 4 son iguales, por lo que se toma como salida la primera de ellas por lo tanto la salida es J=3. Esto se interpreta como que no es similar a ninguno de los patrones que aprendió la red.

Se vuelve a calcular x

$x = s * t_{3i}; t_{3i} = (1,1,1,1)$
$x = (0,0,1,1)$

$\|x\| = 0+0+1+1 = 2$
$\|s\| = 0+0+1+1 = 2$

Entonces de realiza la división $\frac{\|x\|}{\|s\|} = \frac{2}{2} = 1$ y se compara con el factor de de vigilancia.

$\frac{\|x\|}{\|s\|} < 0.7$ Como la condición no se cumple entonces, se realiza la otra comparación $\frac{\|x\|}{\|s\|} \geq 0.7$ esta resulta que si es cierta, por lo que se procede a actualizar los pesos,

Sustituyendo valores nos quedaría: como J es fijo se realiza un barrido de i

$$b_{iJ}(new) = \frac{Lx_i}{L-1+\|x\|}$$

$$bij(new) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0.67 \\ 0 & 1 & 0.67 \end{bmatrix}$$

y en el caso de $t_{Ji} = x_i$ los cuales quedarían como

$$t_{ji} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 \end{bmatrix}$$

Lo que significa que la red aprendió el nuevo patrón gracias al cambio que se realizó en el factor de vigilancia.

El ejemplo anterior se ejecuto utilizando el Simulador ANNS v1.1 que hemos desarrollado, mostrando que los pesos finales alcanzados coinciden con los pesos finales que el autor ha presentado.

3.2.1 Resultado obtenido con el Simulador ANNS Versión 1.1 para la Red ART1 (Fausett).

Paso 1.

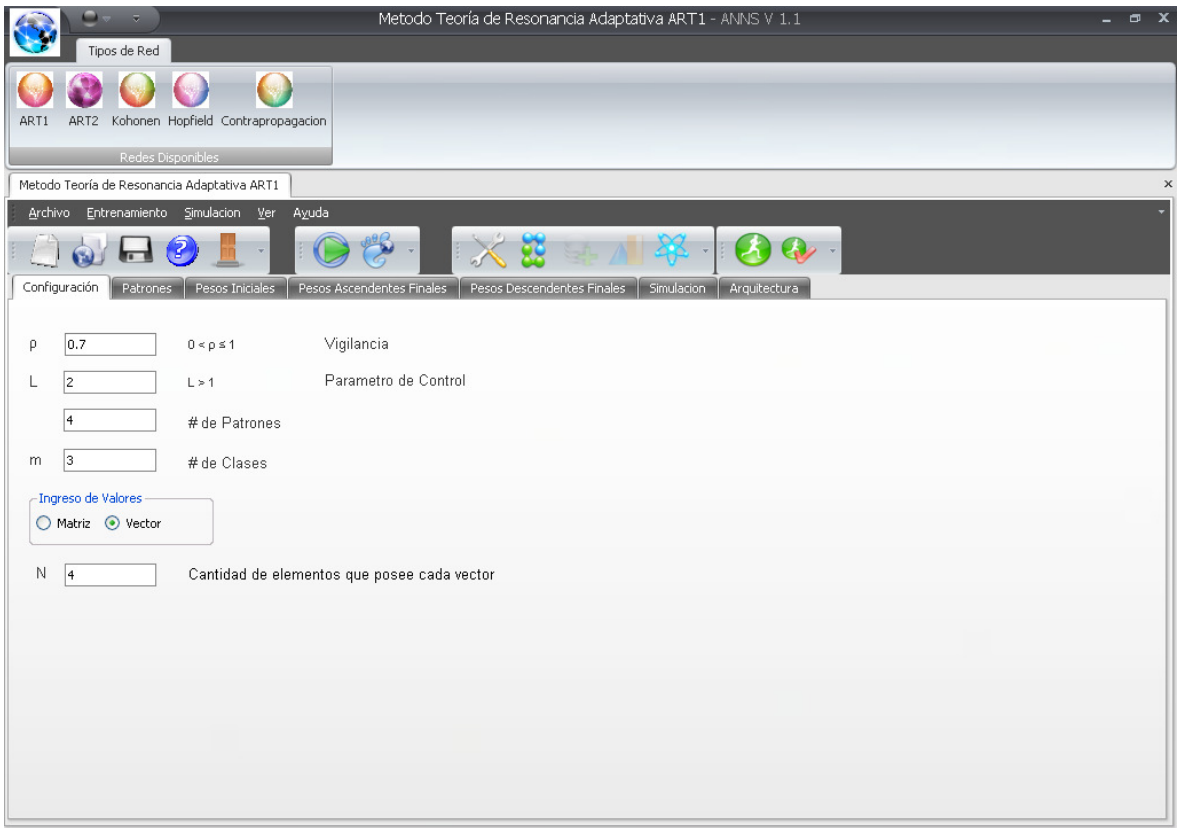


Figura 22. Ventana de configuración de parámetros de la Red Art1 (Fausett).

Paso 2.

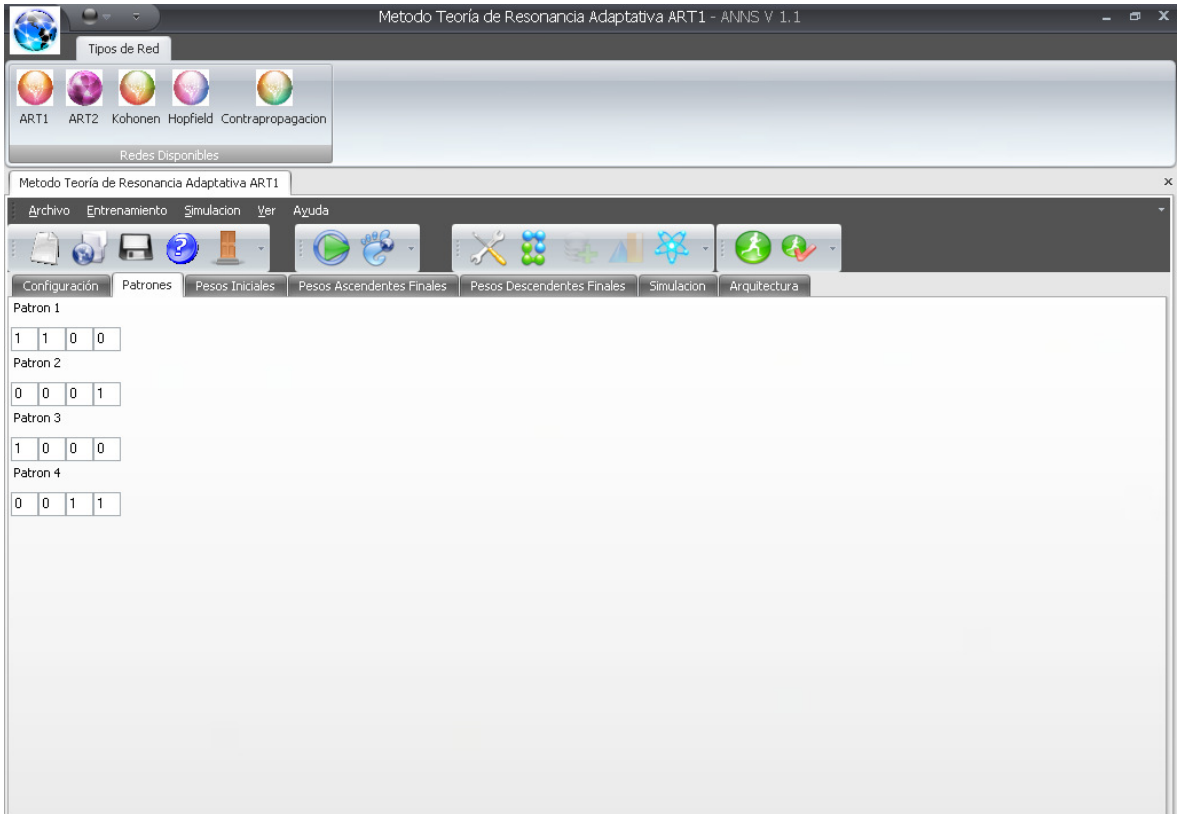


Figura 23. Ventana de patrones a entrenar de la Red Art1 (Fausett).

Paso 3.

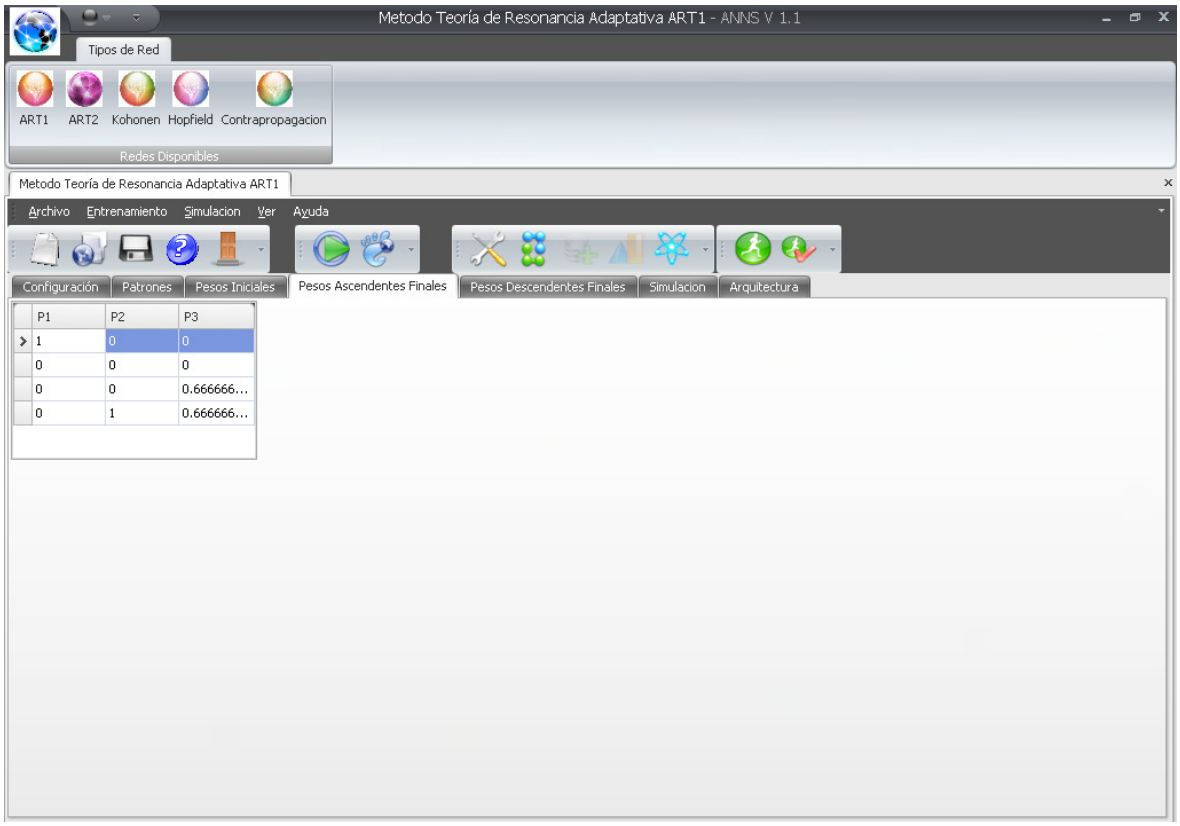


Figura 24. Ventana de pesos ascendentes finales de la Red Art1 (Fausett).

Paso 4.

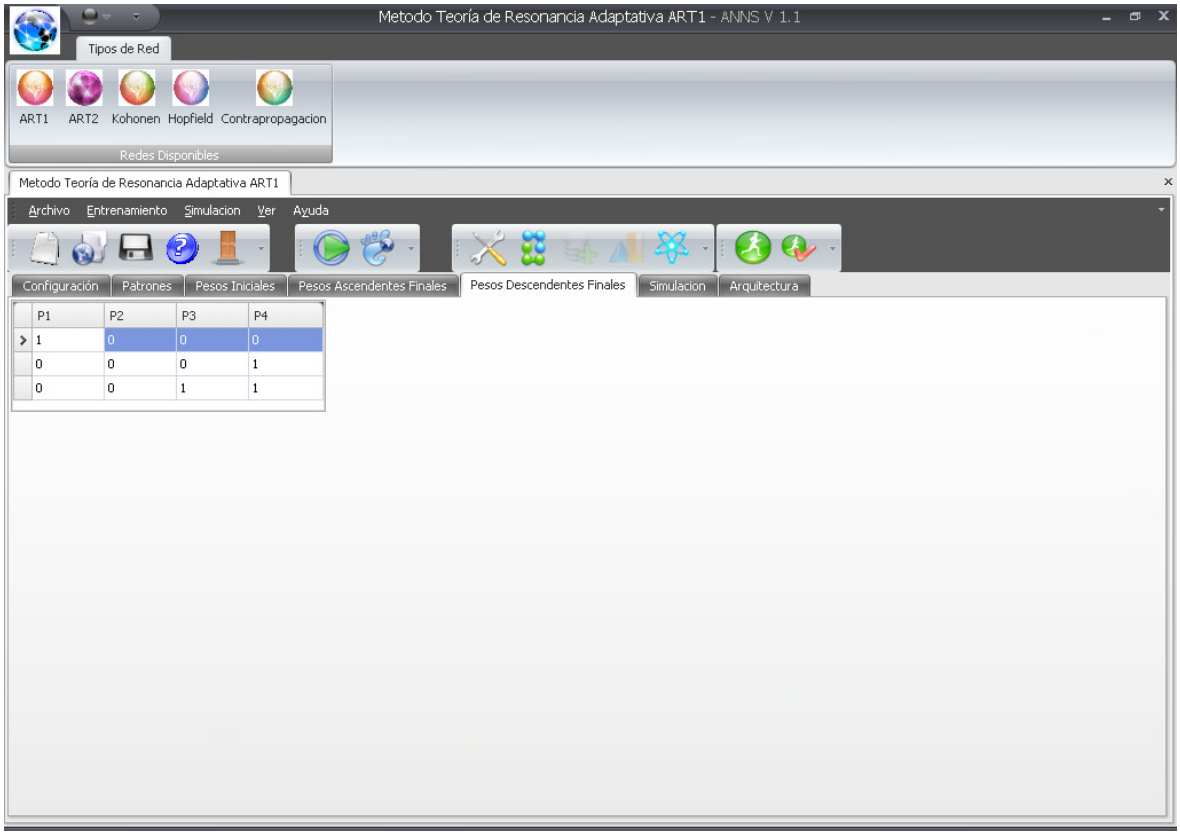


Figura 25. Ventana de pesos descendentes finales de la Red Art1 (Fausett).

3.3 RED ART2.

Para ver el funcionamiento de este algoritmo, realizaremos el calculo de un ejemplo tomado del Libro “Redes Neuronales Algoritmo y Aplicación” del autor James A. Freeman/David M.Skapura (Pág. 338 – 346).

Antes de mostrar el resultado de los cálculos, resumiremos los paramentos de la red, y se mostraran los pesos iniciales.

Anteriormente se habían fijado los siguientes parámetros:
a = *10; b = 10; c = 0.1;* θ *= 0.2*. A esta lista añadimos el parámetro adicional *d = 0.9.* Emplearemos N = 6 unidades en la capa F_2.

Los pesos descendente reciben todos ellos valores iniciales iguales a cero, así que $Z_{ij} = 0$
Los pesos ascendentes reciben valores iniciales:

$$z_{ji} = \frac{0.5}{(1-d)\sqrt{M}} = 2.236$$ Puesto que *M = 5.*

El primer vector de entrada es I_1 $(0.2, 0.7, 0.1, 0.5, 0.4)^t$
Propagaremos este vector por las subcapas siguiendo el orden de ecuaciones dado.
Dado que en este momento no hay realimentación procedente de u, entonces w pasa a ser una copia del vector de entrada.

$$\mathbf{w} = (0.2, 0.7, 0.1, 0.5, 0.4)^t$$

x es una versión normalizada del mismo vector:

$$\mathbf{x} = (0.205, 0.718, 0.103, 0.513, 0.410)^t$$

En ausencia de realimentación procedente de q, v es igual a $f(x)$:

$$\mathbf{v} = (0.205, 0.718, 0, 0.513, 0.410)^t$$

Obsérvese que la tercera componente es ahora cero, puesto que su valor cayó por debajo del umbral θ. Como F_2 está inactiva en este instante, no hay una señal descendente que llegue a F_1. En este caso, las tres subcapas restantes de F_1 pasan a ser copias de **v**:

$$\mathbf{u} = (0.205, 0.718, 0, 0.513, 0.410)^t$$
$$\mathbf{p} = (0.205, 0.718, 0, 0.513, 0.410)^t$$
$$\mathbf{q} = (0.205, 0.718, 0, 0.513, 0.410)^t$$

No podemos detenernos aquí, sin embargo, por que tanto **u** como **q** no son nulas en este momento. Comenzamos de nuevo en w, entonces:

$$\mathbf{w} = (2.263, 7.920, 0.100, 5.657, 4.526)^t$$
$$\mathbf{x} = (0.206, 0.722, 0.009, 0.516, 0.413)^t$$
$$\mathbf{v} = (2.269, 7.942, 0.000, 5.673, 4.538)^t$$

en donde ahora v tiene contribuciones del vector **x** actual y del vector **u** del instante anterior. Tal como sucedía antes, las tres capas restantes van a ser idénticas:

$$\mathbf{u} = (0.206, 0.723, 0.000, 0.516, 0.413)^t$$
$$\mathbf{p} = (0.206, 0.723, 0.000, 0.516, 0.413)^t$$
$$\mathbf{q} = (0.206, 0.723, 0.000, 0.516, 0.413)^t$$

Ahora podemos detenernos por que más iteraciones a través de las subcapas no cambiaran los resultados. En general, bastan dos iteraciones para estabilizar las salidas de las unidades que están en las subcapas.

Durante la primera iteración por F_1, se supuso que no había una señal descendente de F_2 que fuera a contribuir con la activación de la subcapa **p** de F_1.

Ahora se hará la parte de procesamiento y comenzaremos la propagación ascendente del vector **p** hasta F_2, se obtiene un vector de actividades a lo largo de las unidades de F_2 que esta dado por:

$$\mathbf{T} = (4.151, 4.151, 4.151, 4.151, 4.151, 4.151)^t$$

Dado que todas las actividades son iguales, la primera unidad pasa ser la ganadora y el vector de actividad pasa a ser:

$$\mathbf{T} = (4.151, 0, 0, 0, 0, 0)^t$$

Y la salida de la capa F_2 es el vector: $(0.9, 0, 0, 0, 0, 0)^t$.

Ahora se propaga este vector de salida de vuelta hasta F_1 y se vuelve a pasar por las capas. Como todos los pesos descendentes tienen pesos iniciales nulos, no hay cambios en las subcapas de F_1. Se demostró anteriormente que esta situación no va a dar lugar a una restauración procedente del subsistema de orientación; en otras palabras, que ya se ha alcanzado un estado resonante. Los vectores de pesos se actualizan, ahora de acuerdo con las ecuaciones adecuadas que se proporcionaron, se obtiene que la matriz de pesos ascendente es:

$$\begin{pmatrix} 2.063 & 7.220 & 0.000 & 5.157 & 4.126 \\ 2.236 & 2.236 & 2.236 & 2.236 & 2.236 \\ 2.236 & 2.236 & 2.236 & 2.236 & 2.236 \\ 2.236 & 2.236 & 2.236 & 2.236 & 2.236 \\ 2.236 & 2.236 & 2.236 & 2.236 & 2.236 \\ 2.236 & 2.236 & 2.236 & 2.236 & 2.236 \end{pmatrix}$$

Y la matriz de pesos descendente es:

$$\begin{pmatrix} 2.06284 & 0 & 0 & 0 & 0 & 0 \\ 7.21995 & 0 & 0 & 0 & 0 & 0 \\ 0.00000 & 0 & 0 & 0 & 0 & 0 \\ 5.15711 & 0 & 0 & 0 & 0 & 0 \\ 4.12568 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

Se Observa la similitud, ya esperada, entre la primera fila de la matriz ascendente y la primera columna de la matriz descendente. El ejemplo anterior se ejecuto utilizando el Simulador que hemos desarrollado, mostrando que los pesos finales alcanzados coinciden con los pesos que el autor ha definido como finales.

3.3.1 Resultado obtenido con el Simulador ANNS Versión 1.1 para la Red ART2.

Paso 1.

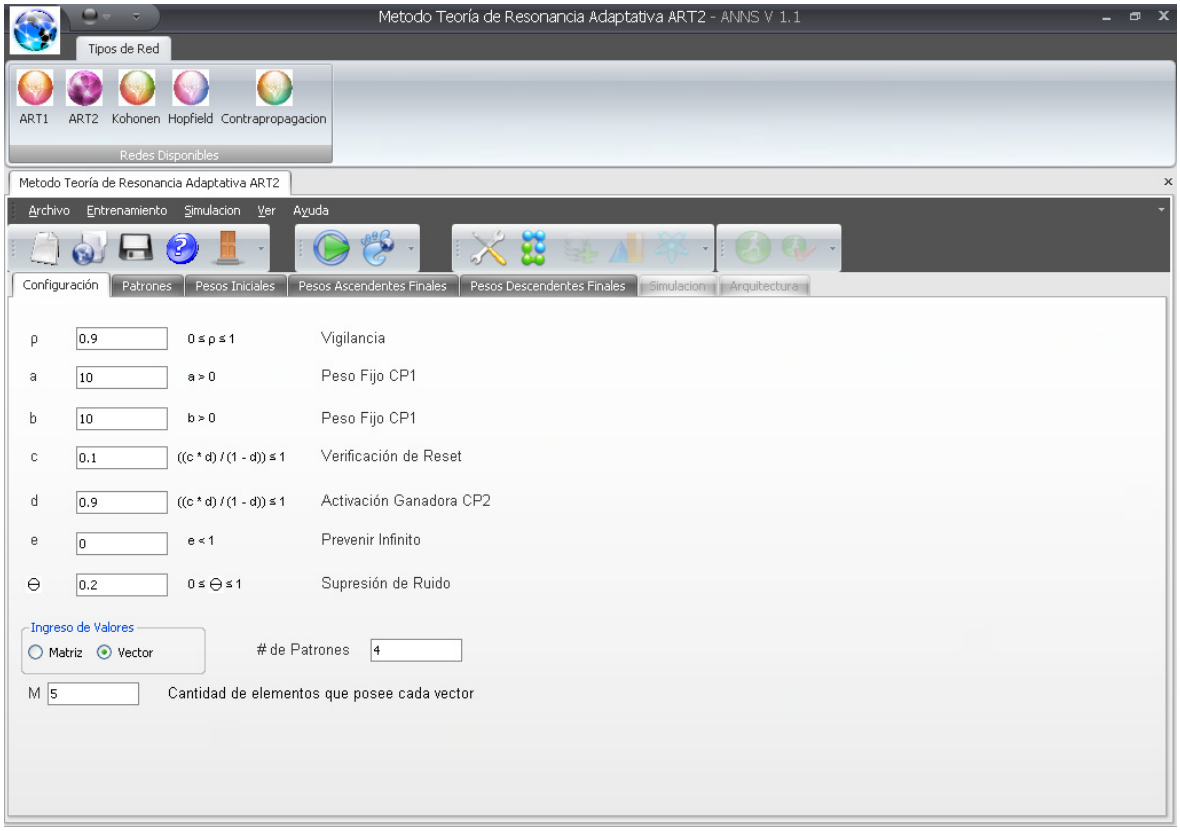


Figura 26. Ventana de configuración de parámetros de la Red Art2 (Freeman).

Paso 2.

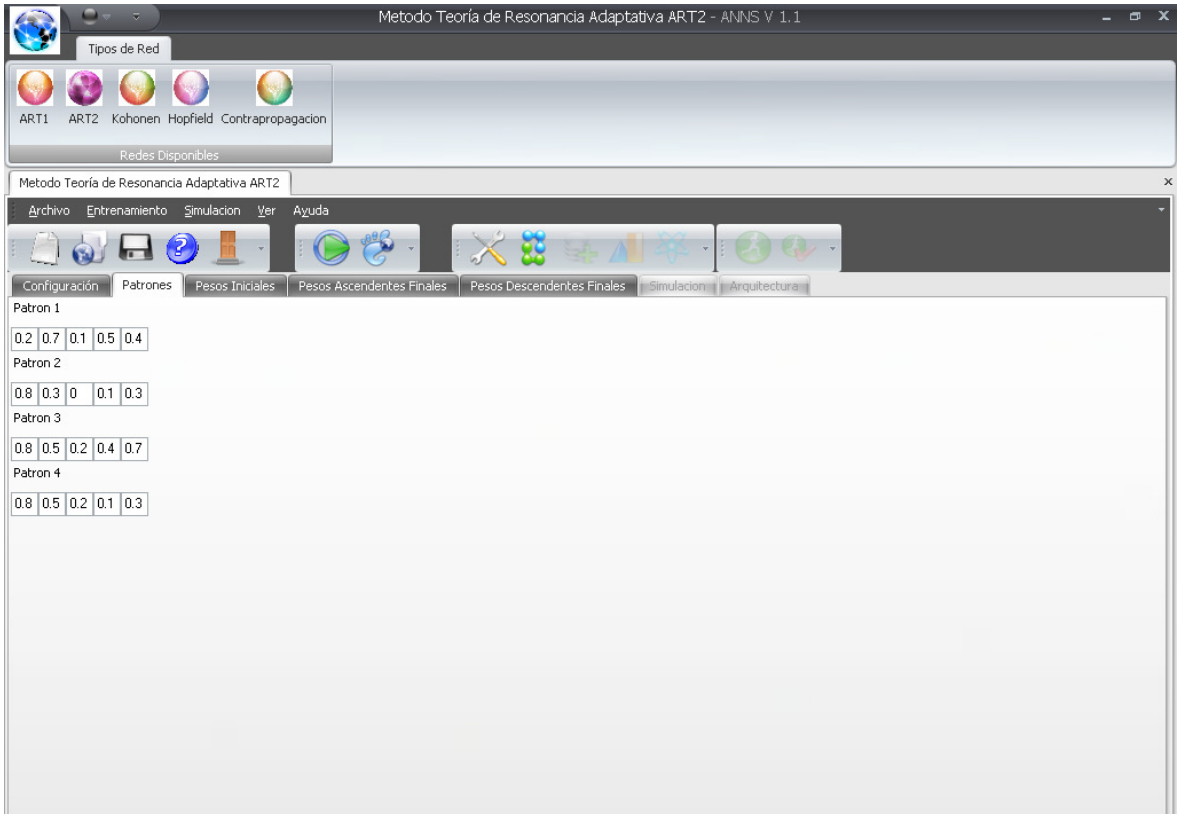


Figura 27. Ventana de patrones a entrenar de la Red Art2 (Freeman).

Paso 3.

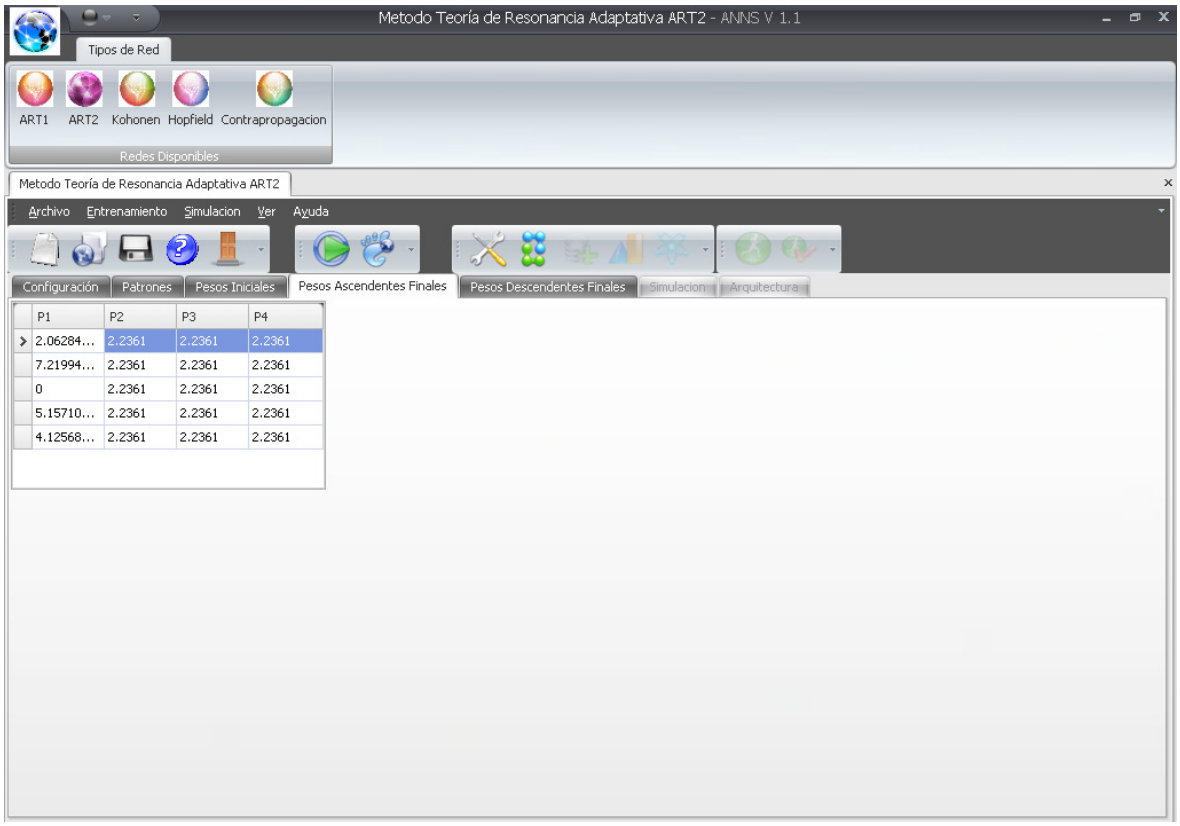


Figura 28. Ventana de pesos ascendentes finales de la Red Art2 (Freeman).

Paso 4.

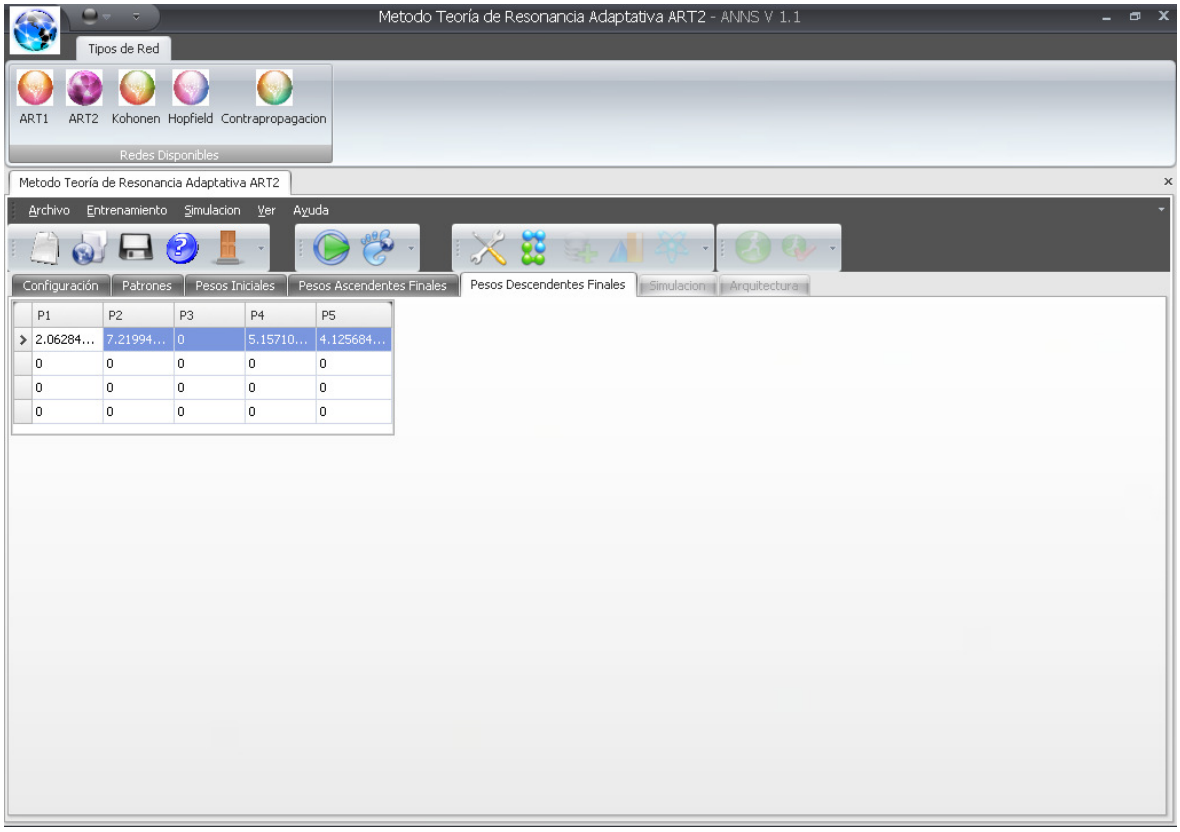


Figura 29. Ventana de pesos descendentes finales de la Red Art2 (Freeman).

3.4 RED KOHONEN.

Para ver el funcionamiento de este algoritmo, se realizará el calculo de los resultados de un ejercicio tomado del Libro "Fundamentals of Neural Networks Architectures, Algorithms, and Applications" del autor Laurene Fausett (Pág. 172 – 175).

Sean los vectores (1, 1, 0, 0); (0, 0, 0, 1); (1, 0, 0, 0); (0, 0, 1, 1)
Las neuronas de la capa competitiva: m=2
α= 0.6

La Matriz de Pesos:

$$\begin{pmatrix} 0.2 & 0.8 \\ 0.6 & 0.4 \\ 0.5 & 0.7 \\ 0.9 & 0.3 \end{pmatrix}$$

Paso 1. Comienza el entrenamiento:

Paso 2. Para el primer vector: (1, 1, 0, 0)

Paso 3: Se calculan las distancias Euclideas:

$$D(1) = (0.2-1)^2 + (0.6-1)^2 + (0.5-0)^2 + (0.9-0)^2 = 1.86$$

$$D(2) = (0.8-1)^2 + (0.4-1)^2 + (0.7-0)^2 + (0.3-0)^2 = 0.98$$

Paso 4:
La neurona ganadora es J=2, ya que la menor distancia fue D (2)

Paso 5:
Se actualizan los pesos de la neurona ganadora:

$$w_{12}(nuevo) = w_{12}(viejo) + 0.6\left[x_i - w_{12}(viejo)\right]$$

$$w_{12}(nuevo) = 0.8 + 0.6\left[1 - 0.8\right] = 0.92$$

$$w_{22}(nuevo) = 0.4 + 0.6\left[1 - 0.4\right] = 0.76$$

$$w_{32}(nuevo) = 0.7 + 0.6[0 - 0.7] = 0.28$$

$$w_{42}(nuevo) = 0.3 + 0.6[0 - 0.3] = 0.12$$

La nueva matriz de peso es:

$$\begin{pmatrix} 0.2 & 0.92 \\ 0.6 & 0.76 \\ 0.5 & 0.28 \\ 0.9 & 0.12 \end{pmatrix}$$

Paso 2. Para el segundo vector: (0, 0, 0, 1)

Paso 3:

$$D(1) = (0.2 - 0)^2 + (0.6 - 0)^2 + (0.5 - 0)^2 + (0.9 - 1)^2 = 0.66$$

$$D(2) = (0.92 - 0)^2 + (0.76 - 0)^2 + (0.28 - 0)^2 + (0.12 - 1)^2 = 2.2768$$

Paso 4:

La neurona ganadora es J=1

Paso 5:

Se actualizan los pesos de la neurona ganadora:

$$\begin{pmatrix} 0.08 & 0.92 \\ 0.24 & 0.76 \\ 0.20 & 0.28 \\ 0.96 & 0.12 \end{pmatrix}$$

Paso 2: Para el tercer vector: (1, 0, 0, 0)

Paso 3:

$$D(1) = (0.08 - 1)^2 + (0.24 - 0)^2 + (0.2 - 0)^2 + (0.96 - 0)^2 = 1.8656$$

$$D(2) = (0.92 - 1)^2 + (0.76 - 0)^2 + (0.28 - 0)^2 + (0.12 - 0)^2 = 0.6768$$

Paso 4:

La neurona ganadora es J=2

Paso 5:

$$\begin{pmatrix} 0.08 & 0.968 \\ 0.24 & 0.304 \\ 0.20 & 0.112 \\ 0.96 & 0.048 \end{pmatrix}$$

Paso 2:

Para el cuarto vector (0, 0, 1, 1)

Paso 3:

$$D(1) = (0.08-0)^2 + (0.24-0)^2 + (0.2-1)^2 + (0.96-1)^2 = 0.7056$$

$$D(2) = (0.968-0)^2 + (0.304-0)^2 + (0.112-1)^2 + (0.048-1)^2 = 2.724$$

Paso 4:

La neurona ganadora es J=1

Paso 5:

$$\begin{pmatrix} 0.032 & 0.968 \\ 0.096 & 0.304 \\ 0.680 & 0.112 \\ 0.984 & 0.048 \end{pmatrix}$$

Paso 6:

Se reduce la razón de entrenamiento α

La matriz de pesos después de la segunda época es:

$$\begin{pmatrix} 0.016 & 0.980 \\ 0.047 & 0.360 \\ 0.630 & 0.055 \\ 0.999 & 0.024 \end{pmatrix}$$

La matriz de pesos se va modificando después de cada iteración de tal forma que para:

Iteración 50: Matriz de pesos:

$$\begin{pmatrix} 1.9e-19 & 1.0000 \\ 5.7e-15 & 0.4700 \\ 0.5300 & 6.6e-15 \\ 1.0000 & 2.8e-15 \end{pmatrix}$$

Iteración 100: Matriz de pesos:

$$\begin{pmatrix} 6.7e-17 & 1.0000 \\ 2.0e-16 & 0.4900 \\ 0.5100 & 2.3e-16 \\ 1.0000 & 1.0e-16 \end{pmatrix}$$

La matriz de pesos finales es:

$$\begin{pmatrix} 0.0 & 1.0 \\ 0.0 & 0.5 \\ 0.5 & 0.0 \\ 1.0 & 0.0 \end{pmatrix}$$

El ejemplo anterior se desarrollo utilizando el Simulador ANNS v1.1, mostrando que los pesos finales alcanzados coinciden con los pesos que el autor ha definido como finales.

3.4.1 Resultado obtenido con el Simulador ANNS Versión 1.1 para la Red Kohonen.

Paso 1.

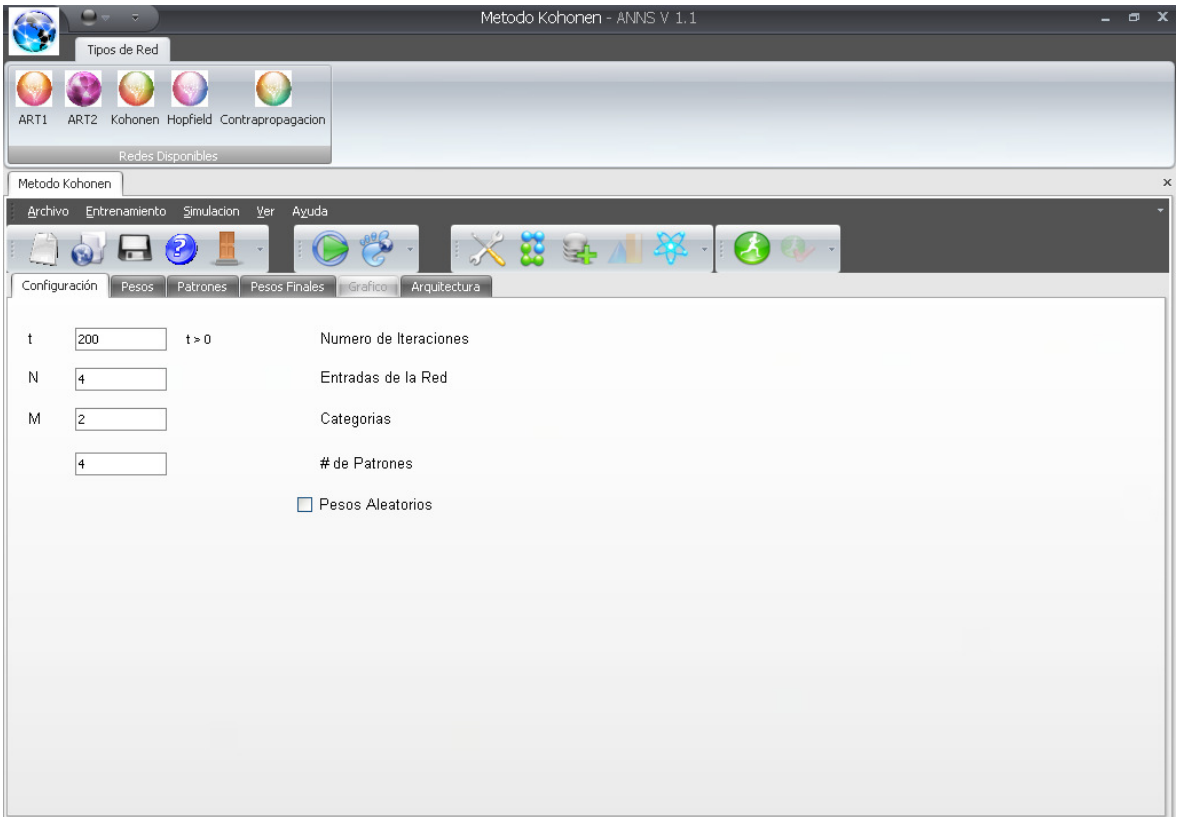


Figura 30. Ventana de configuración de parámetros de la Red Kohonen (Fausett).

Paso 2.

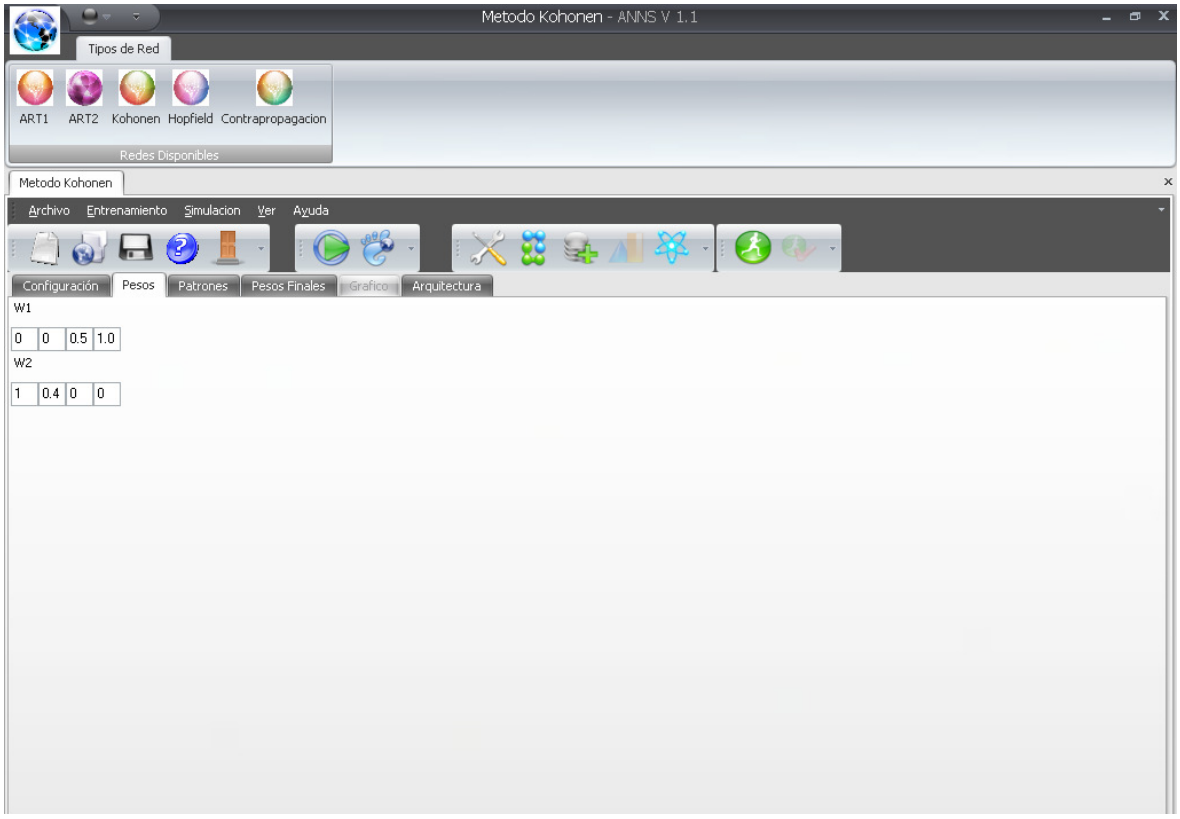


Figura 31. Ventana de pesos de la Red Kohonen (Fausett).

Paso 3.

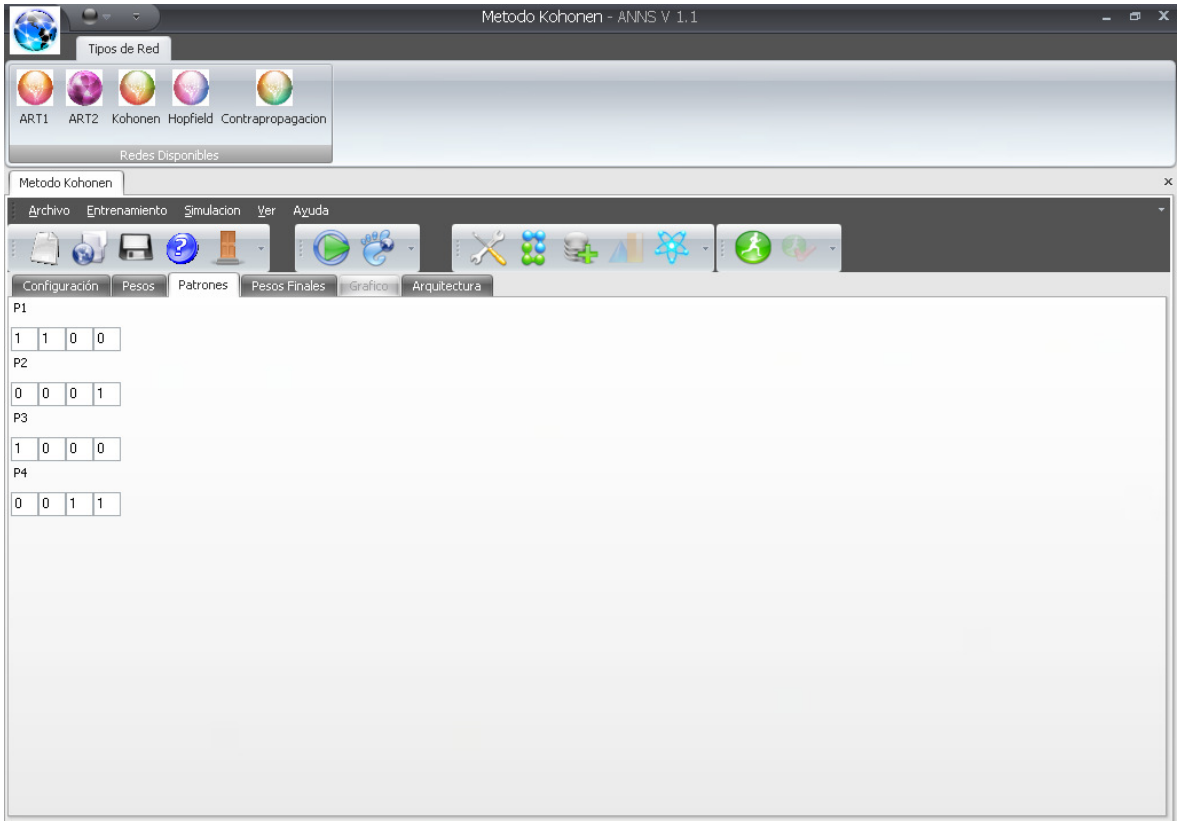
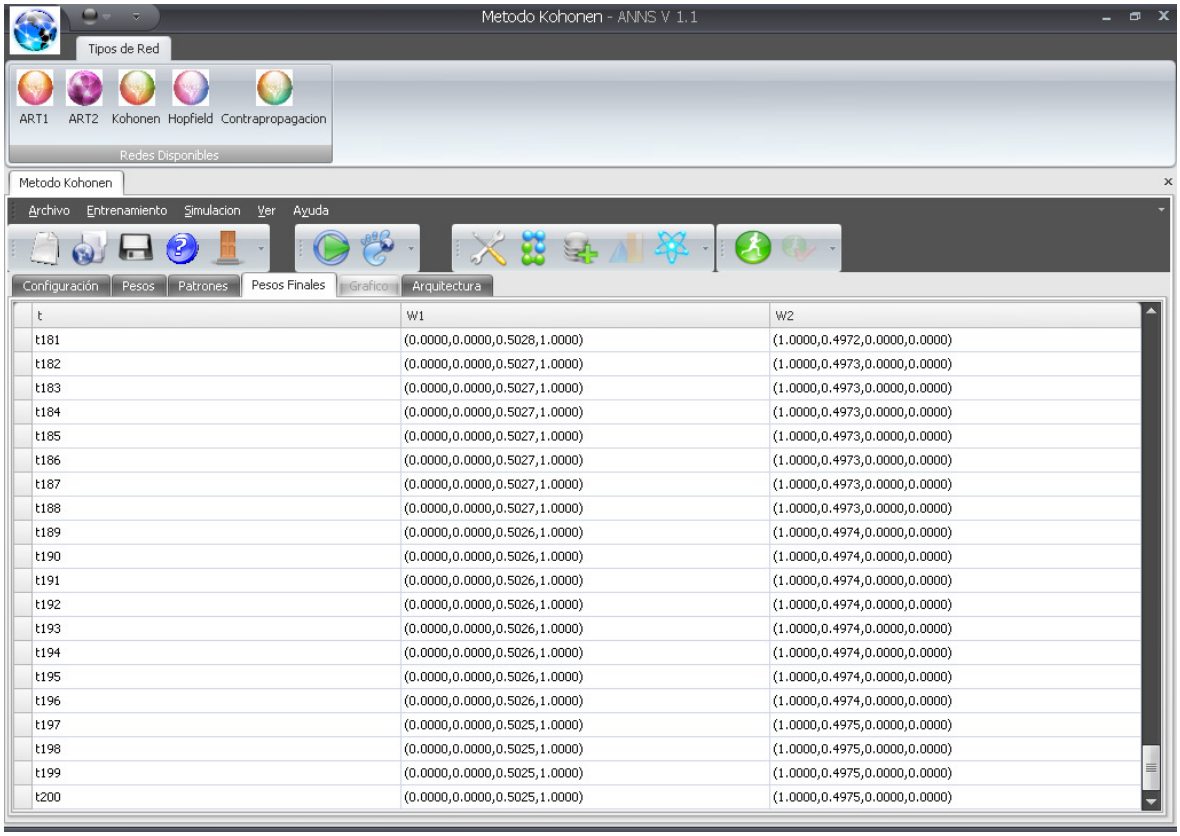


Figura 32. Ventana de patrones a entrenar de la Red Kohonen (Fausett).

Paso 4.



t	W1	W2
t181	(0.0000,0.0000,0.5028,1.0000)	(1.0000,0.4972,0.0000,0.0000)
t182	(0.0000,0.0000,0.5027,1.0000)	(1.0000,0.4973,0.0000,0.0000)
t183	(0.0000,0.0000,0.5027,1.0000)	(1.0000,0.4973,0.0000,0.0000)
t184	(0.0000,0.0000,0.5027,1.0000)	(1.0000,0.4973,0.0000,0.0000)
t185	(0.0000,0.0000,0.5027,1.0000)	(1.0000,0.4973,0.0000,0.0000)
t186	(0.0000,0.0000,0.5027,1.0000)	(1.0000,0.4973,0.0000,0.0000)
t187	(0.0000,0.0000,0.5027,1.0000)	(1.0000,0.4973,0.0000,0.0000)
t188	(0.0000,0.0000,0.5027,1.0000)	(1.0000,0.4973,0.0000,0.0000)
t189	(0.0000,0.0000,0.5026,1.0000)	(1.0000,0.4974,0.0000,0.0000)
t190	(0.0000,0.0000,0.5026,1.0000)	(1.0000,0.4974,0.0000,0.0000)
t191	(0.0000,0.0000,0.5026,1.0000)	(1.0000,0.4974,0.0000,0.0000)
t192	(0.0000,0.0000,0.5026,1.0000)	(1.0000,0.4974,0.0000,0.0000)
t193	(0.0000,0.0000,0.5026,1.0000)	(1.0000,0.4974,0.0000,0.0000)
t194	(0.0000,0.0000,0.5026,1.0000)	(1.0000,0.4974,0.0000,0.0000)
t195	(0.0000,0.0000,0.5026,1.0000)	(1.0000,0.4974,0.0000,0.0000)
t196	(0.0000,0.0000,0.5026,1.0000)	(1.0000,0.4974,0.0000,0.0000)
t197	(0.0000,0.0000,0.5025,1.0000)	(1.0000,0.4975,0.0000,0.0000)
t198	(0.0000,0.0000,0.5025,1.0000)	(1.0000,0.4975,0.0000,0.0000)
t199	(0.0000,0.0000,0.5025,1.0000)	(1.0000,0.4975,0.0000,0.0000)
t200	(0.0000,0.0000,0.5025,1.0000)	(1.0000,0.4975,0.0000,0.0000)

Figura 33. Ventana de pesos finales de la Red Kohonen (Fausett).

3.5 RED HOPFIELD.

A continuación se presenta un ejemplo del algoritmo de entrenamiento para la red Hopfield, tomado del libro: “Redes Neuronales Artificiales, Fundamentos, modelos y aplicaciones” del autor Hilera, José y Martínez, Víctor. Paginas: 191-193.

A continuación se muestran dos figuras que corresponden a los patrones que la red a entrenar deberá de aprender:

FIGURA 1

1 2

3 4

FIGURA 2

1 2

3 4

Los píxeles grises podrían representarse por el valor binario +1 y los blancos con el valor -1. En este caso las informaciones serian dos vectores (M=2) de 4 elementos (N=4) que contienen los valores de los píxeles. La red, por tanto, contendrá 4 neuronas para que cada una de ellas reciba el valor de un píxel.

Los valores de los vectores de entrada que representan a las figuras A y B son los siguientes:

$$E_1 = [1,1,-1,-1] \qquad E_2 = [-1,-1,1,1]$$

El aprendizaje de estas dos informaciones consistiría en la obtención de los pesos de la red (matriz W):

$$E_1^T E_1 - I = \begin{pmatrix} 1 \\ 1 \\ -1 \\ -1 \end{pmatrix} \begin{pmatrix} 1 & 1 & -1 & -1 \end{pmatrix} - \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 0 & 1 & -1 & -1 \\ 1 & 0 & -1 & -1 \\ -1 & -1 & 0 & 1 \\ -1 & -1 & 1 & 0 \end{pmatrix}$$

$$E_2^T E_2 - I = \begin{pmatrix} -1 \\ -1 \\ 1 \\ 1 \end{pmatrix} \begin{pmatrix} -1 & -1 & 1 & 1 \end{pmatrix} - \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 0 & 1 & -1 & -1 \\ 1 & 0 & -1 & -1 \\ -1 & -1 & 0 & 1 \\ -1 & -1 & 1 & 0 \end{pmatrix}$$

$$W = \sum_{K=1}^{2} \left[E_k^T E_k - I \right] = \begin{pmatrix} 0 & 2 & -2 & -2 \\ 2 & 0 & -2 & -2 \\ -2 & -2 & 0 & 2 \\ -2 & -2 & 2 & 0 \end{pmatrix}$$

El ejemplo anterior se ejecutó utilizando el Simulador ANNS v1.1, mostrando que los pesos finales alcanzados coinciden con los pesos mostrados en el libro.

3.5.1 Resultado obtenido con el Simulador ANSS Versión 1.1 para la Red Hopfield.

Paso 1.

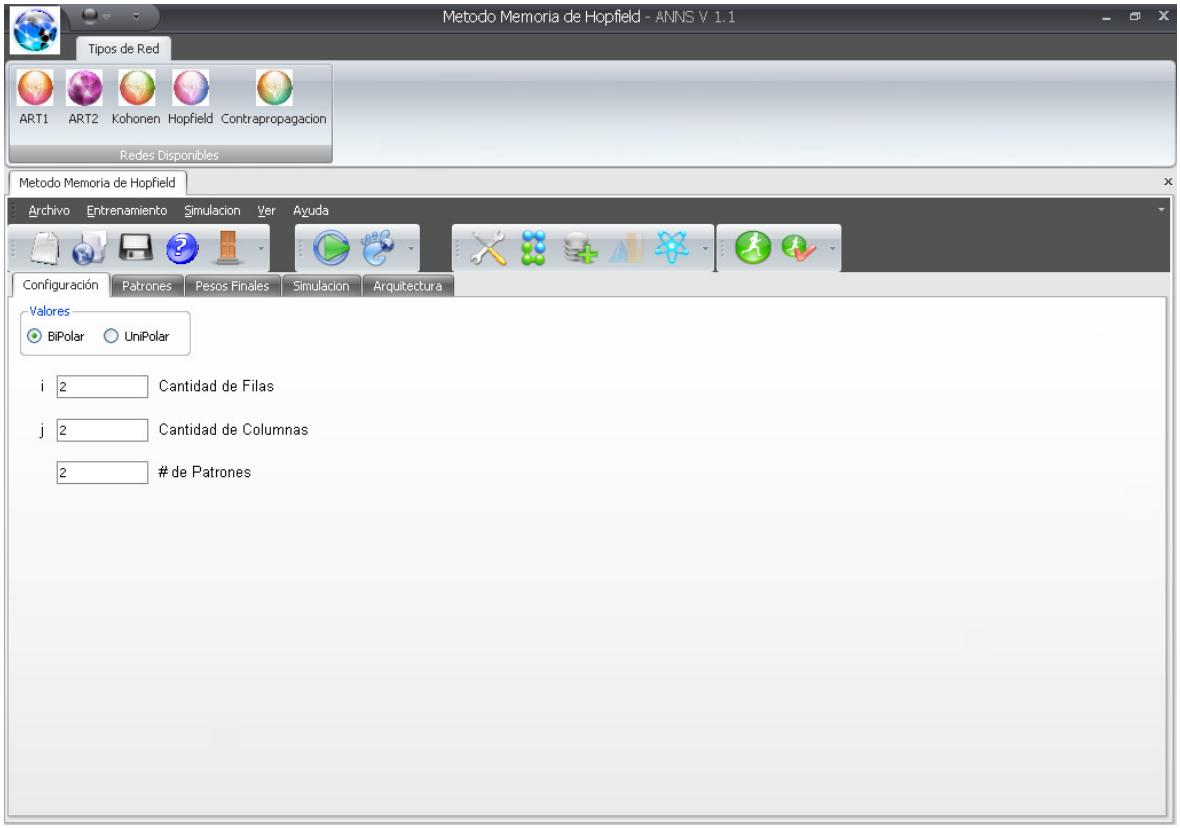


Figura 34. Ventana de configuración de parámetros de la Red Hopfield (Hilera).

Paso 2.

Introducción de los patrones a entrenar

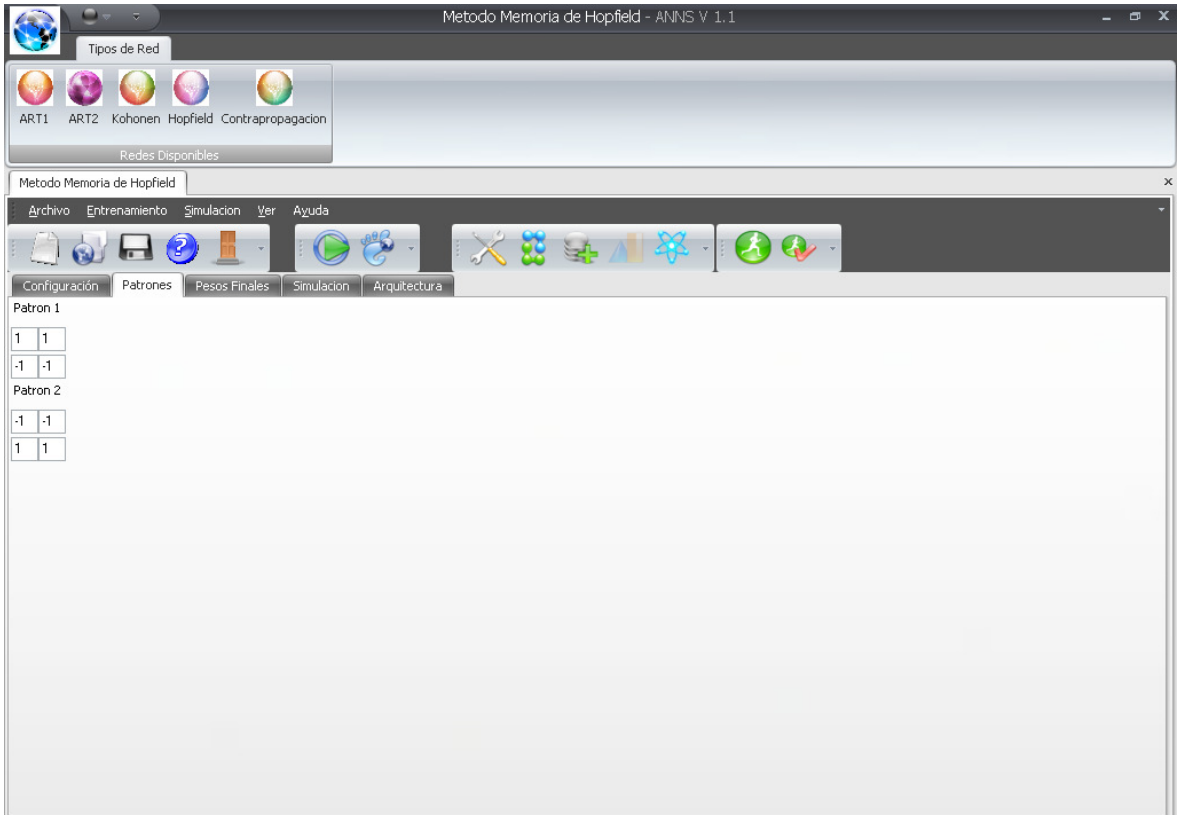


Figura 35. Ventana de patrones a entrenar de la Red Hopfield (Hilera).

Paso 3.

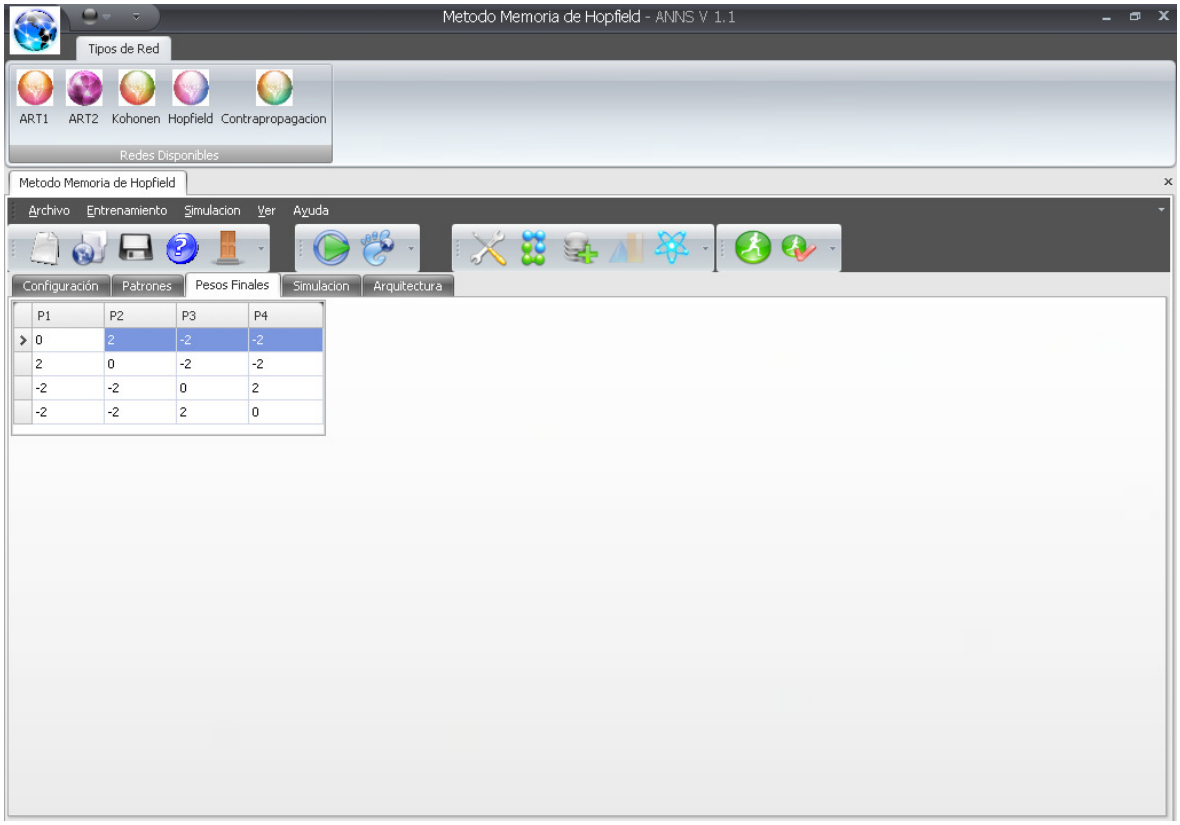


Figura 36. Ventana de pesos finales de la Red Hopfield (Hilera).

3.6 RED CONTRAPROPAGACIÓN.

A continuación se realiza un ejemplo que servirá para comprobar el proceso de entrenamiento de la red de contrapropagación.

La arquitectura de la red a entrenar, tendrá dos neuronas en la capa de entrada, 2 neuronas en la capa competitiva, y dos neuronas en la capa de salida.
El entrenamiento para esta red se divide en dos fases la primera fase entrena la capa de entrada y la capa competitiva. La segunda fase es para la capa competitiva y la de salida.

Fase 1 de entrenamiento.

Sean los vectores de entrenamiento:

a = (3, 4) b = (1, 2) c= (5, 3)

y los vectores de pesos iniciales:

v_1 = (2, 3) v_2 =(4, 3)

Para el correcto funcionamiento de la red es necesario que la capa de entrada normalice los vectores de entrenamiento.

Normalizando los vectores de entrada:

Primer vector de entrenamiento:

$$a = \left(\frac{3}{\sqrt{3^2 + 4^2}}, \frac{4}{\sqrt{3^2 + 4^2}} \right)$$

$$a = (0.6, 0.8)$$

Segundo vector de entrenamiento:

$$b = \left(\frac{1}{\sqrt{1^2 + 2^2}}, \frac{2}{\sqrt{1^2 + 2^2}} \right)$$

$$b = (0.448, 0.857493)$$

Tercer vector de entrenamiento:

$$c = \left(\frac{5}{\sqrt{5^2 + 3^2}}, \frac{3}{\sqrt{5^2 + 3^2}} \right)$$

$$c = (0.857, 0.514496)$$

Normalizando los vectores de peso:

Primer vector de Peso:

$$\upsilon_1 = \left(\frac{2}{\sqrt{2^2+3^2}}, \frac{3}{\sqrt{2^2+3^2}} \right)$$

$$\upsilon_1 = (0.5547, 0.83205)$$

Segundo vector de peso:

$$\upsilon_2 = \left(\frac{4}{\sqrt{4^2+3^2}}, \frac{3}{\sqrt{4^2+3^2}} \right)$$

$$\upsilon_2 = (0.8, 0.6)$$

Las razones de aprendizaje se inicializan con los valores $\alpha = 0.6$ y $\beta = 0.1$ sugeridos por el autor Hecht – Nielsen (1988), tomado del libro "Fundamentals of Neural Networks. Architectures, Algorithms and applications" by Laurene Fausett [Pág. 208].

- Primera iteración:

  Para $\alpha = 0.6$

Primer vector:

Se procede a calcular la distancia entre el vector de entrada a entrenar y los vectores de peso:

$$D_{11} = (0.6 - 0.5547)^2 = 0.002057$$

$$D_{12} = (0.6 - 0.8)^2 = 0.04$$

La neurona ganadora es J=1, por tanto se realiza la actualización de los pesos de la neurona ganadora:

$$\upsilon_{11}(nuevo) = (1-\alpha)\upsilon_{ij}(Viejo) + \alpha x_i$$

$$\upsilon_{11}(nuevo) = (1-0.6)(0.5547) + (0.6)(0.6)$$

$$\upsilon_{11}(nuevo) = 0.58188$$

Segundo vector:

Se calcula la distancia entre el vector de entrada y los vectores de peso

$$D_{21} = (0.448 - 0.58188)^2 = 0.017924$$

$$D_{22} = (0.448 - 0.8)^2 = 0.12$$

La neurona ganadora es J=1, se actualizan los pesos de la neurona ganadora:

$$v_{21}(nuevo) = (1-\alpha)\, v_{ij}(Viejo) + \alpha x$$

$$v_{21}(nuevo) = (1-0.6)(0.58188) + (0.6)(0.448)$$

$$v_{21}(nuevo) = 0.501552$$

Tercer vector:

Se calcula la distancia entre el vector de entrada y los vectores de peso:

$$D_{31} = (0.857 - 0.501552)^2 = 0.126343$$

$$D_{32} = (0.857 - 0.8)^2 = 0.003249$$

La neurona ganadora es J=2, se actualiza el vector de peso de la neurona vencedora:

$$v_{32}(nuevo) = (1-\alpha)\, v_{ij}(Viejo) + \alpha x_i$$

$$v_{32}(nuevo) = (1-0.6)(0.8) + (0.6)(0.857)$$

$$v_{32}(nuevo) = 0.8342$$

- Segunda iteración

  Para $\alpha = 0.5$

Primer vector:

$$D_{11} = (0.6 - 0.501552)^2 = 0.009692$$

$$D_{12} = (0.6 - 0.8342)^2 = 0.05485$$

Neurona Ganadora J=1

$$v_{11}(nuevo) = (1-\alpha)\, v_{ij}(Viejo) + \alpha x_i$$

$$v_{11}(nuevo) = (1-0.5)(0.501552) + (0.5)(0.6)$$

$$v_{11}(nuevo) = 0.550776$$

Segundo vector:

$$D_{21} = (0.448 - 0.550776)^2 = 0.010563$$

$$D_{22} = (0.448 - 0.8342)^2 = 0.14915$$

Neurona Ganadora J=1

$$\upsilon_{21}(nuevo) = (1-\alpha)\,\upsilon_{ij}(Viejo) + \alpha x_i$$

$$\upsilon_{21}(nuevo) = (1-0.5)(0.50776) + (0.5)(0.448)$$

$$\upsilon_{21}(nuevo) = 0.499388$$

Tercer vector:

$$D_{31} = (0.857 - 0.499388)^2 = 0.127886$$

$$D_{32} = (0.857 - 0.8342)^2 = 0.00052$$

Neurona ganadora J=2

$$\upsilon_{32}(nuevo) = (1-\alpha)\,\upsilon_{ij}(Viejo) + \alpha x_i$$

$$\upsilon_{32}(nuevo) = (1-0.5)(0.8342) + (0.5)(0.857)$$

$$\upsilon_{32}(nuevo) = 0.8456$$

- Después de 25 iteraciones:

  $\alpha = 0.00001$

  Los pesos finales obtenidos son los siguientes:

$$\upsilon_{11}(nuevo) = 0.514199$$

$$\upsilon_{21}(nuevo) = 0.514199$$

$$\upsilon_{32}(nuevo) = 0.853651$$

El primer y segundo vector pertenecen a la primera categoría y el peso final asociado a la neurona J=1 es:

$$\upsilon_{1}(nuevo) = 0.514199$$

El tercer vector pertenece a la segunda categoría y el peso final de la neurona J=2 es:

$$\upsilon_{2}(nuevo) = 0.853651$$

Fase II de entrenamiento.

Sean los vectores de entrenamiento:

$$a=(0.6,0.8) \qquad b=(0.448,0.857493) \qquad c=(0.857,0.514496)$$

Los nuevos vectores de peso:

$$\upsilon_1=(0.514199,0.83205) \qquad \upsilon_2=(0.853651,0.6)$$

Las razones de aprendizaje se inicializan para la fase 2 en los siguientes valores: $\alpha=0.00001$**,** debido a que en la primera fase este fue el valor final para α. $\beta=0.1$

Se procede a encontrar la neurona ganadora de la capa competitiva para cada vector de entrada:

- Primer vector de entrada: $a=(0.6,0.8)$

$$D_{11}=(0.6-0.514199)^2-(0.8-0.83205)^2=0.008389$$

$$D_{12}=(0.6-0.853651)^2+(0.8-0.6)^2=0.104339$$

La neurona Ganadora es Z_1

Se deben actualizar los pesos de la neurona vencedora:

$$\upsilon_{11}=(1-\alpha)\upsilon_{11}(viejo)+\alpha x_1$$

$$\upsilon_{11}=(1-0.00001)(0.514199)+(0.00001)(0.6)$$

$$\upsilon_{11}=0.514199$$

Se actualizan los pesos desde la neurona ganadora Z_1 hacia la capa de salida:

$$w_{jk}(Nuevo)=(1-\beta)w_{jk}(Viejo)+\beta y_k$$

$$w_{11}(Nuevo)=(1-0.1)(0.83205)+(0.1)(0.8)$$

$$w_{11}(Nuevo)=0.828845$$

- Segundo vector de entrada: $b=(0.448,0.857493)$

$$D_{21}=(0.448-0.514199)^2+(0.857493-0.828845)^2=0.005203$$

$$D_{22}=(0.448-0.853651)^2+(0.857493-0.6)^2=0.230855$$

La neurona Ganadora es Z_1

Se actualizan los pesos de la neurona ganadora:

$$v_{21} = (1-\alpha)\, v_{21}\,(viejo) + \alpha x_2$$

$$v_{21} = (1-0.00001)(0.514199) + (0.00001)(0.448)$$

$$v_{21} = 0.514198$$

Se actualizan los pesos desde la neurona ganadora Z_1 hacia la capa de salida:

$$w_{jk}\,(Nuevo) = (1-\beta)\, w_{jk}\,(Viejo) + \beta y_k$$

$$w_{21}\,(Nuevo) = (1-0.1)(0.828845) + (0.1)(0.857493)$$

$$w_{21}\,(Nuevo) = 0.83171$$

- Tercer patrón de entrada: $c = (0.857, 0.514496)$

Se calcula cual es la neurona ganadora para el tercer vector de entrada:

$$D_{31} = (0.857 - 0.514198)^2 + (0.514496 - 0.83171)^2 = 0.218138$$

$$D_{32} = (0.857 - 0.853651)^2 + (0.514496 - 0.6)^2 = 0.007322$$

La neurona vencedora fue J=2, por lo tanto, se actualizan los pesos de la neurona ganadora:

$$v_{31} = (1-\alpha)\, v_{31}\,(viejo) + \alpha x_2$$

$$v_{31} = (1-0.00001)(0.853651) + (0.00001)(0.857)$$

$$v_{31} = 0.853651$$

Se actualiza el peso desde la neurona ganadora Z_2 hacia la capa de salida:

$$w_{jk}\,(Nuevo) = (1-\beta)\, w_{jk}\,(Viejo) + \beta y_k$$

$$w_{32}\,(Nuevo) = (1-0.1)(0.6) + (0.1)(0.514496)$$

$$w_{32}\,(Nuevo) = 0.59145$$

Se decrementa el valor de β:

$\beta = 0.09$

Después de 35 iteraciones:

Desde la capa de entrada hacia la capa competitiva los pesos alcanzados son los siguientes:

$$\upsilon_{11} = 0.5229$$

$$\upsilon_{21} = 0.5229$$

$$\upsilon_{32} = 0.8575$$

El primer y segundo vector pertenecen a la primera categoría y el peso final asociado a la neurona J=1 es:

$$\upsilon_1 = 0.5229$$

El tercer vector pertenece a la segunda categoría y el peso final de la neurona J=2 es:

$$\upsilon_2 = 0.8575$$

Desde la capa competitiva hacia la capa de salida los pesos alcanzados son los siguientes:

$$w_{11} = 0.8477$$

$$w_{21} = 0.8477$$

$$w_{32} = 0.5154$$

Por lo tanto, el peso final alcanzado desde la capa competitiva Z_1 hasta la capa de salida es $w_1 = 0.8477$ y el peso final alcanzado desde Z_2 hacia la capa de salida es $w_2 = 0.5154$.

El ejemplo anterior se realizó utilizando el Simulador ANNS v1.1 que hemos desarrollado, mostrando que los pesos finales alcanzados con él coinciden con los pesos encontrados en el procedimiento anterior.

3.6.1 Resultado obtenido con el Simulador ANNS Versión 1.1 para la Red Contrapropagación.

Paso 1.

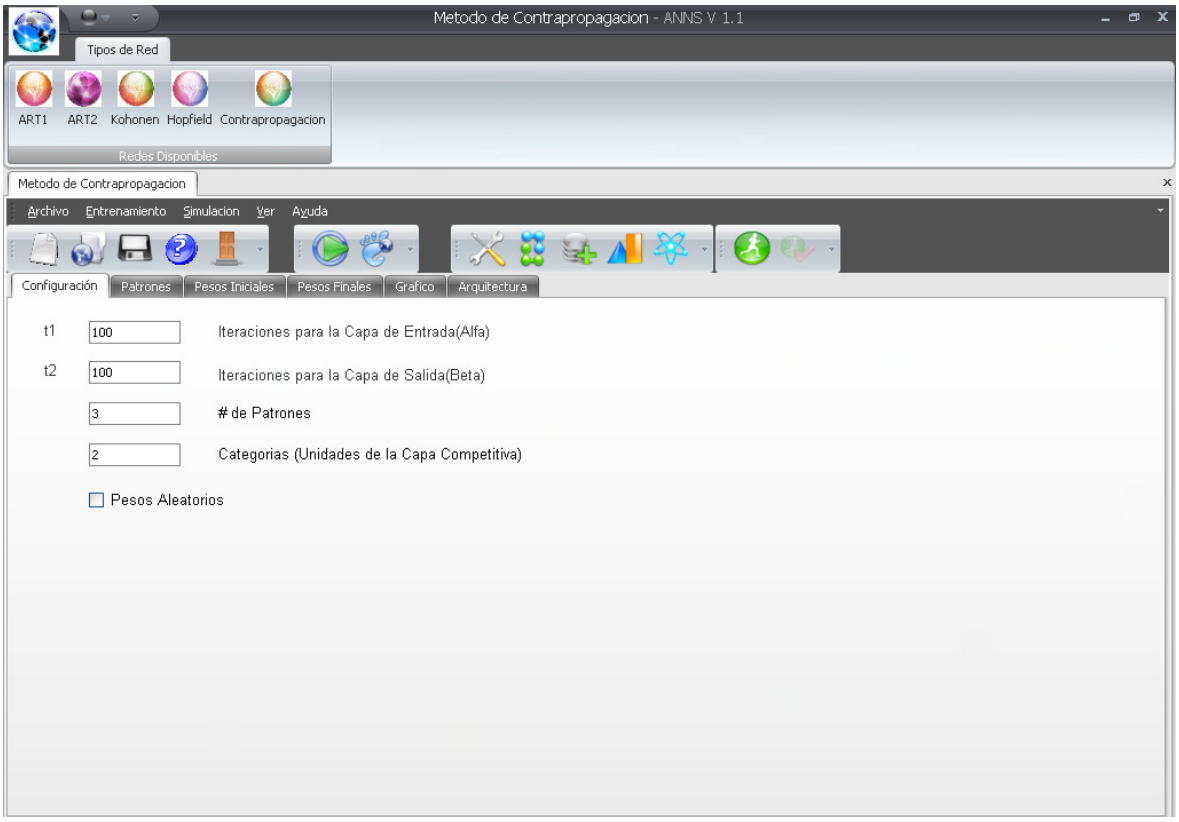


Figura 37. Ventana de configuración de parámetros de la Red Contrapropagación.

Paso 2.

Introducción de los vectores de entrenamiento:

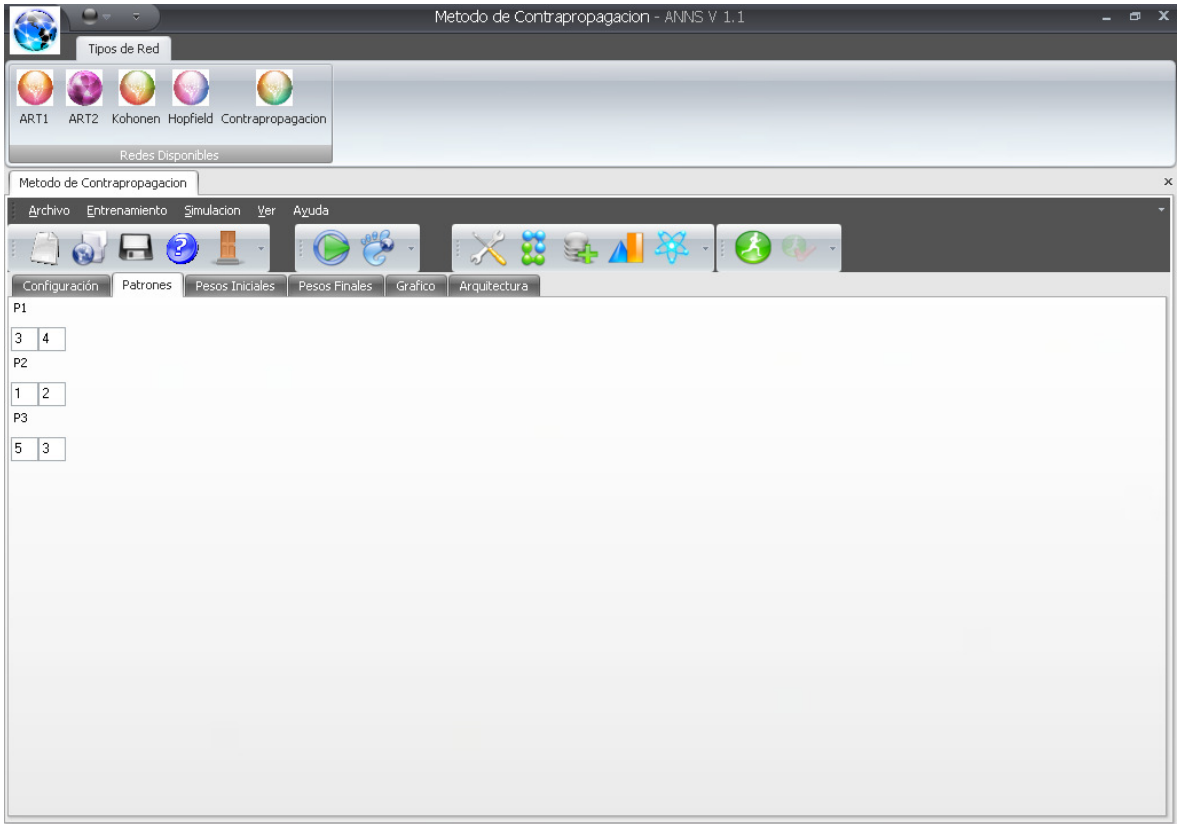


Figura 38. Ventana de patrones a entrenar de la Red Contrapropagación.

Paso 3.

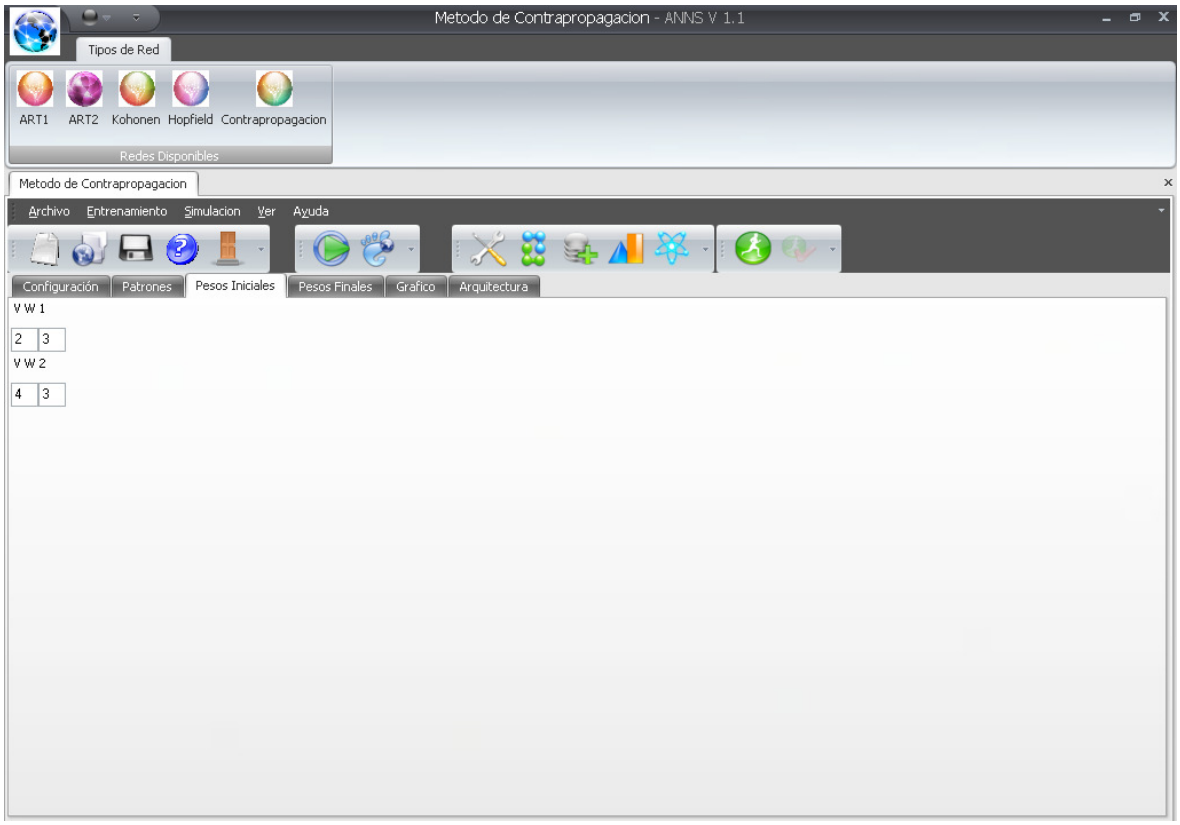
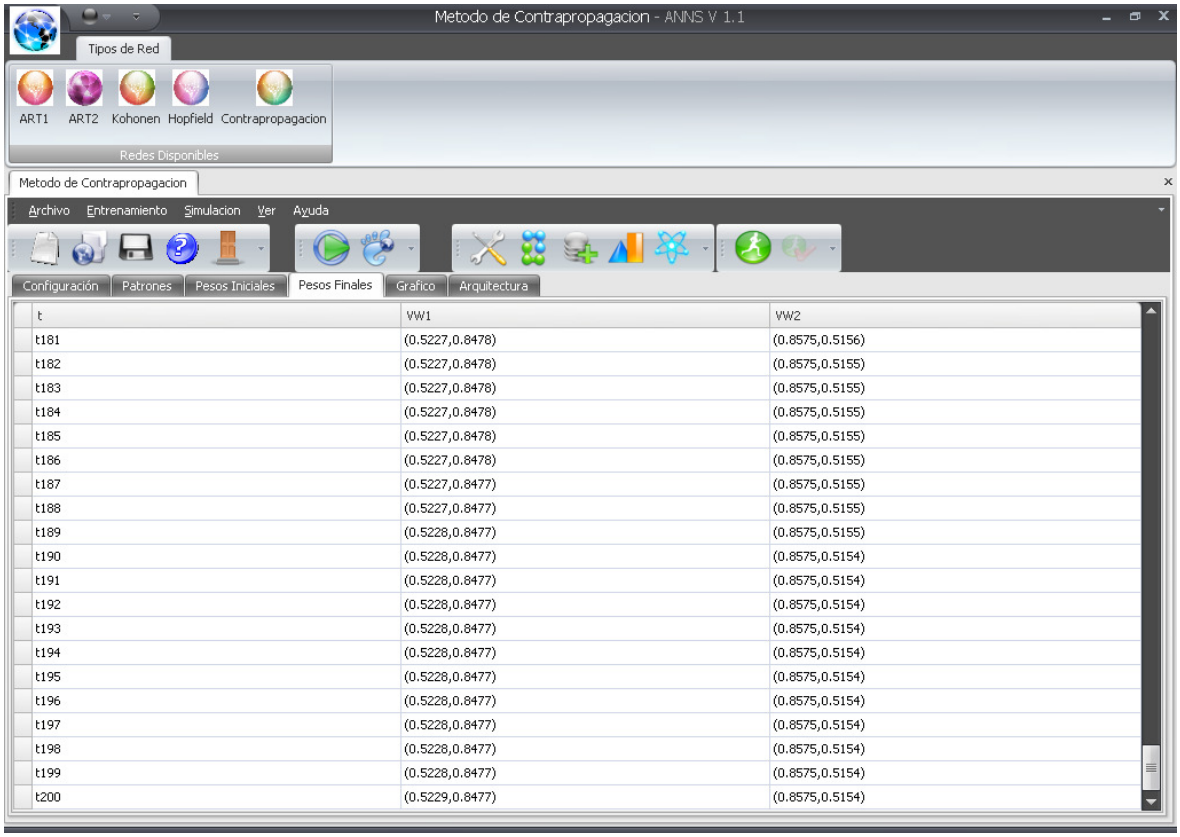


Figura 39. Ventana de vectores de pesos de la Red Contrapropagación.

Paso 4.



t	VW1	VW2
t181	(0.5227,0.8478)	(0.8575,0.5156)
t182	(0.5227,0.8478)	(0.8575,0.5155)
t183	(0.5227,0.8478)	(0.8575,0.5155)
t184	(0.5227,0.8478)	(0.8575,0.5155)
t185	(0.5227,0.8478)	(0.8575,0.5155)
t186	(0.5227,0.8478)	(0.8575,0.5155)
t187	(0.5227,0.8477)	(0.8575,0.5155)
t188	(0.5227,0.8477)	(0.8575,0.5155)
t189	(0.5228,0.8477)	(0.8575,0.5155)
t190	(0.5228,0.8477)	(0.8575,0.5154)
t191	(0.5228,0.8477)	(0.8575,0.5154)
t192	(0.5228,0.8477)	(0.8575,0.5154)
t193	(0.5228,0.8477)	(0.8575,0.5154)
t194	(0.5228,0.8477)	(0.8575,0.5154)
t195	(0.5228,0.8477)	(0.8575,0.5154)
t196	(0.5228,0.8477)	(0.8575,0.5154)
t197	(0.5228,0.8477)	(0.8575,0.5154)
t198	(0.5228,0.8477)	(0.8575,0.5154)
t199	(0.5228,0.8477)	(0.8575,0.5154)
t200	(0.5229,0.8477)	(0.8575,0.5154)

Figura 40. Ventana de pesos finales de la Red Contrapropagación.

Capitulo 4 – MANUAL DE USUARIO

OBJETIVO DEL MANUAL*:*

Que por medio de la lectura de este manual cualquier persona con conocimientos básicos acerca de Redes Neuronales Artificiales, pueda familiarizarse rápidamente con el uso del Software ANNS V1.1.

CONOCIMIENTOS PREVIOS:

Para que el software ANNS V1.1 sea utilizado como una herramienta de estudio de las topologías de Redes Neuronales Artificiales contenidas en el mismo, es necesario que el usuario reconozca conceptos básicos relacionados con las redes que el simulador posee. Estos conceptos son:

- ✓ ¿Qué es una Red Neuronal Artificial?
- ✓ ¿Cómo funciona una Red Neuronal Artificial?
- ✓ Conocer las Topologías de las siguientes redes: ART1, ART2, Kohonen, Hopfield, Contrapropagación.
- ✓ Dominar las limitantes que posee cada una de las redes que integran el software.
- ✓ Identificar las partes que conforman cada una de las redes contenidas en el simulador: Entrada de la Red, Salida de la Red, Neurona, Capas Ocultas, Pesos o sinapsis, Vectores de entrada o patrones de entrenamiento.
- ✓ Identificar los diferentes parámetros de inicialización para cada una de las redes.

4.1 INSTALACIÓN DEL ANNS VERSIÓN 1.1

En este capitulo se presenta la información necesaria acerca de los requerimientos del equipo, para asegurar el correcto funcionamiento del software ANNS v1.1, así como los pasos a seguir para la instalación del mismo.

- ✓ Requerimientos del sistema
- ✓ Instalación y lanzamiento del Software

4.1.1 Instalación y Lanzamiento del Software.

Paso 1.

Introduzca el CD que contiene el software de instalación del programa ANNS Versión 1.1 a su unidad de CD ROM, ubique el icono del archivo de instalación y haga doble click sobre él, ver figura 41.

Figura 41. Icono del Instalador de ANNS V 1.1

Inmediatamente después aparecerá una ventana en la que se muestra, el proceso de instalación del .NET Framework 2.0, ver figura 42.

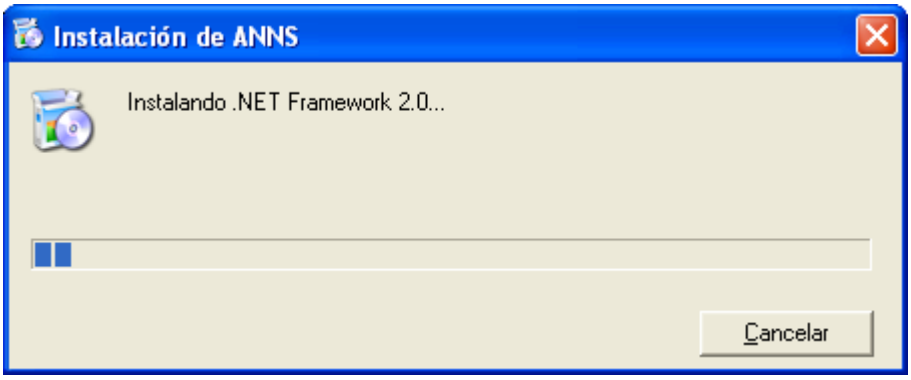


Figura 42. Ventana que muestra la instalación del .NET Framework 2.0

Paso 2.

Aparecerá el cuadro de diálogo de instalación del programa ANNS Versión 1.1. Presione el botón “Siguiente” para iniciar la instalación, ver figura 43.

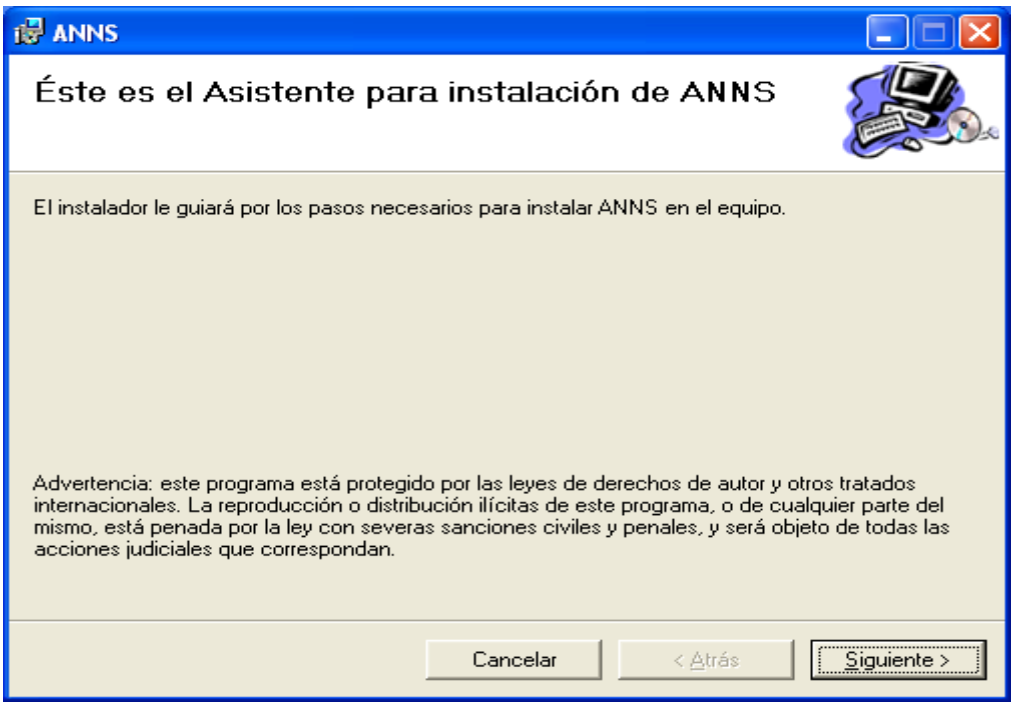


Figura 43. Ventana del Asistente de Instalación del ANNS V1.1.

Paso 3.

Seleccione la ruta en donde desee instalar el software ANNS Versión 1.1. Por defecto la ruta asignada será: “C:\ Archivos de programa \ ANNS v1.1”, presione Siguiente> para continuar, ver figura 44.

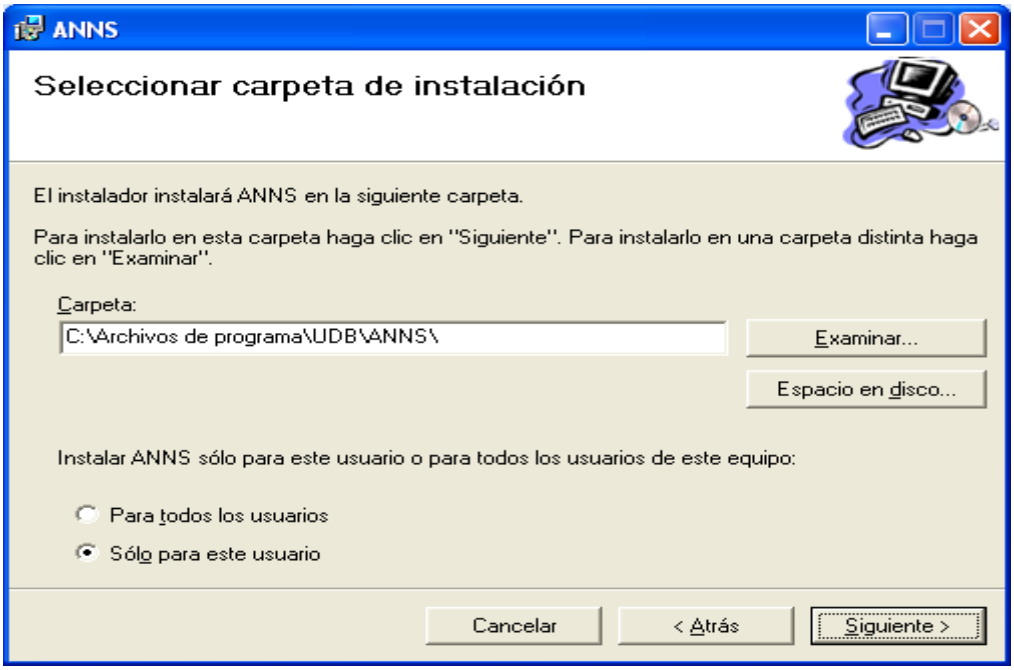


Figura 44. Ventana que muestra la ubicación del Software.

El simulador está listo para ser instalado en su computadora. Para continuar presione el botón Siguiente>, ver figura 45.

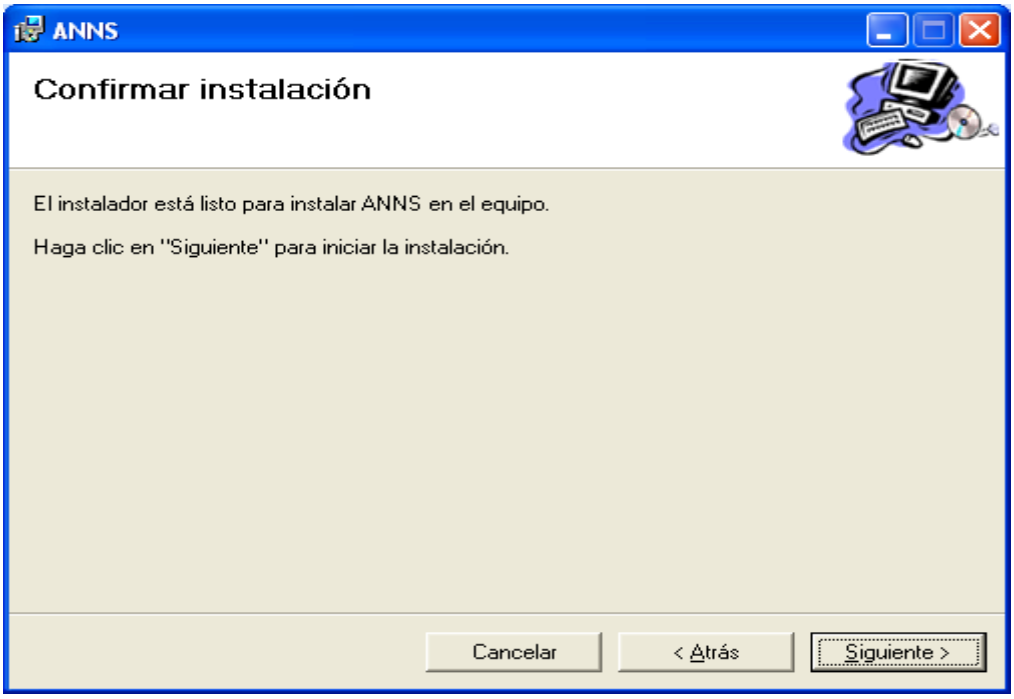


Figura 45. Ventana de confirmación para la instalación.

Paso 4

El software ANNS V1.1 iniciará el proceso de instalación, ver figura 46.

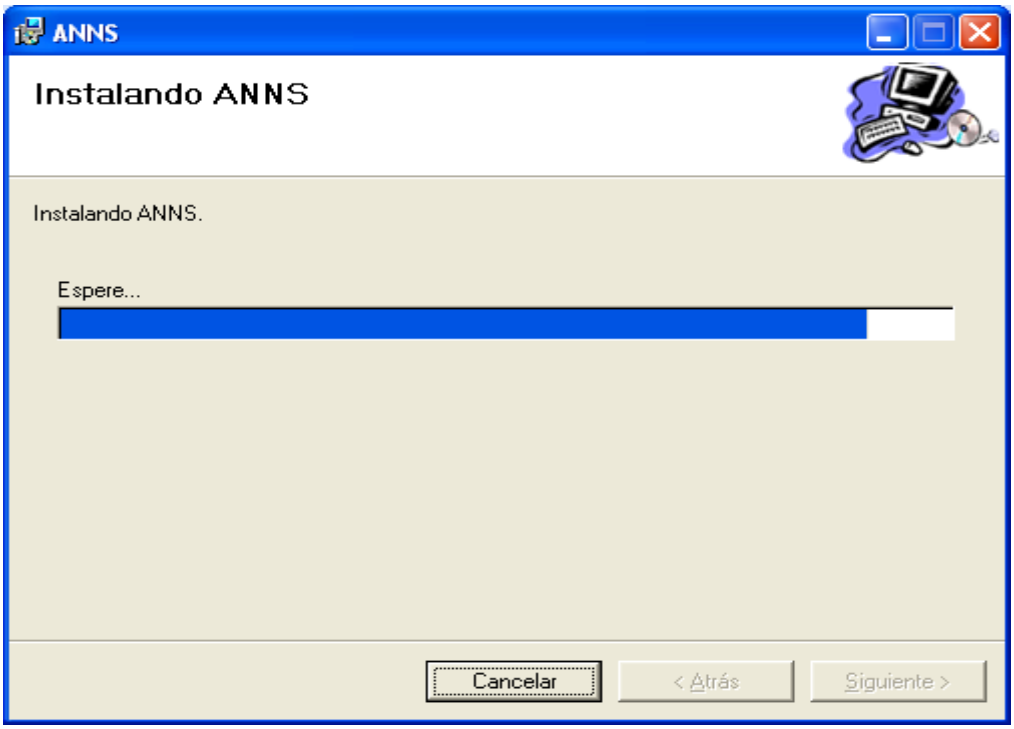


Figura 46. Ventana de Instalación del ANNS V1.1.

Paso 5

Una vez completado el proceso anterior, presione el botón “Cerrar” para salir del cuadro de diálogo, ver figura 47.

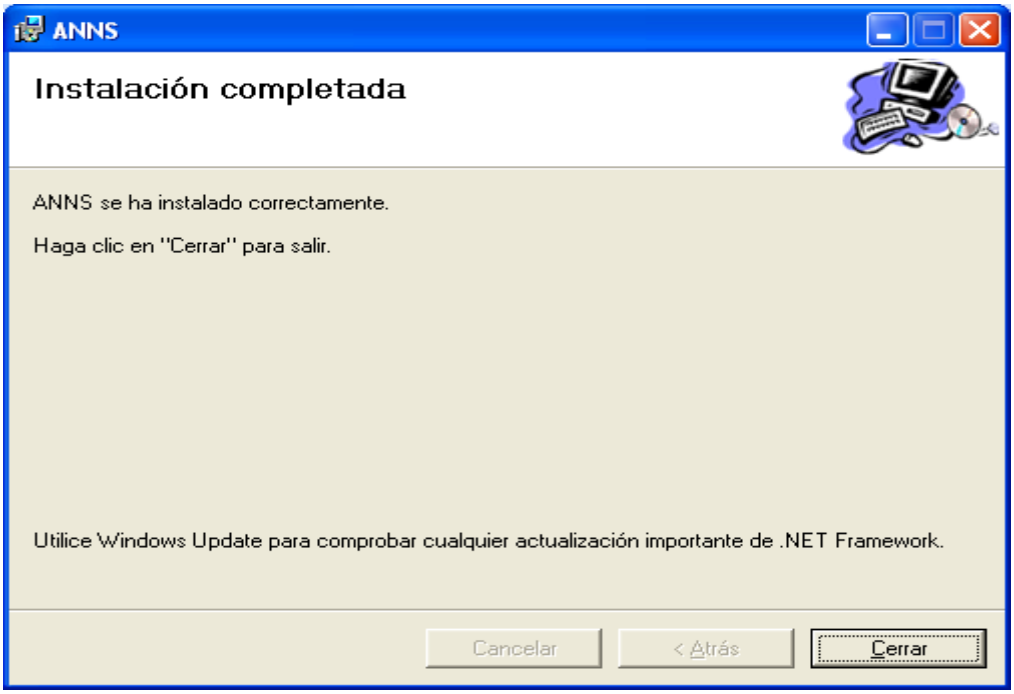


Figura 47. Ventana que muestra que la instalación ha sido completada.

Paso 6

Presione el botón de inicio, ver figura 48.

Figura 48. Botón de Inicio.

Busque la carpeta que contiene el simulador ANNS V1.1, colocado en el menú de programas, ver figura 49.

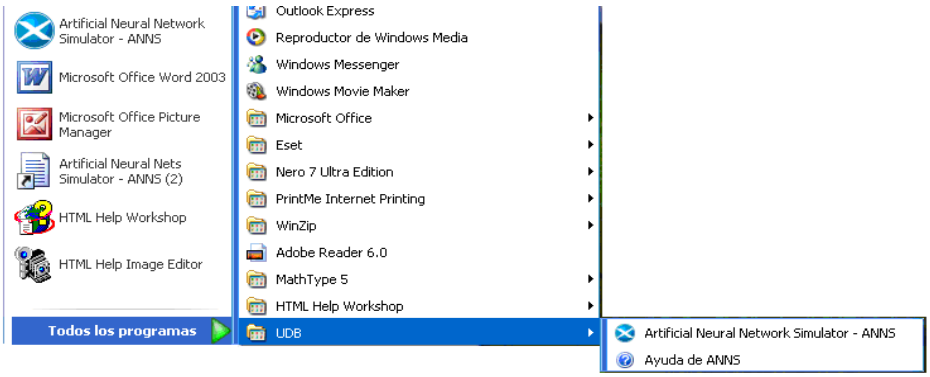


Figura 49. Menú desplegable del botón de inicio.

4.1.2 Requerimientos del Sistema.

El programa ANNS v1.1 necesita los siguientes requerimientos de Hardware, ver tabla 4.1.

RECOMENDADO		MINIMO	
PROCESADOR	PENTIUM 4	*PROCESADOR*	PENTIUM233MHZ
RAM	1 GB	*RAM*	512 MB
VIDEO	32 MB	*VIDEO*	16 MB
CD-ROM	No aplica	*CD-ROM*	No aplica
MONITOR	No aplica	*MONITOR*	No aplica
TECLADO	No aplica	*TECLADO*	No aplica
MOUSE	No aplica	*MOUSE*	No aplica
DISCO DURO	20GB	*DISCO DURO*	1.105GB para x86
			1.435 GB para 64 bits

Tabla 4.1. Características del Hardware

En la tabla 4.2 se muestran los requerimientos del sistema operativo y software que el fabricante recomienda para la instalación del Framework.NET 2.0.

RECOMENDADO		MINIMO	
SISTEMA OPERATIVO	Windows 2000 (Service Pack 3); Windows 98; Windows 98 Second Edition; Windows ME; Windows Server 2003; Windows XP (todas las versiones)Service Pack 2	**SISTEMA OPERATIVO**	Windows 98
SOFTWARE NECESARIO	Windows Installer 3.0 (excepto para Windows 98/ME, que requiere Windows Installer 2.0 o una versión superior). Windows Installer 3.1 o superior	**SOFTWARE NECESARIO**	Windows Installer 2.0 o una versión superior

Tabla 4.2 Características del Software.

El Programa ANNS v1.1 tiene los siguientes requisitos de espacio en el disco duro en la tabla 4.3.

RECOMENDADO	
Programa ANNS v1.1	25MB.
Framework 2.0	280 MB (x86), 610 MB (64 bits).
Espacio en disco para guardar archivos generados por la aplicación.	800MB.

Tabla 4.3 Espacio que ocupa el ANNS en el disco duro.

4.2 VENTANAS PRINCIPALES.

Este capitulo contiene información acerca de cada una de las ventanas que el simulador despliega durante su ejecución. A continuación se realiza una descripción de las características más importantes de cada una de ellas:

- ✓ Ventana de Lanzamiento del Sistema.
- ✓ Principal.
- ✓ Configuración de la Red.
- ✓ Introducción de Vectores o patrones de entrenamiento.
- ✓ Asignación de Pesos.
- ✓ Pesos finales.
- ✓ Simulación.
- ✓ Topología de la Red.

4.2.1. Ventana de Lanzamiento del Sistema.

La ventana de lanzamiento del sistema aparece cada vez que se inicia el software Artificial Neural Network Simulator – ANNS v1.1, ver figura 50.

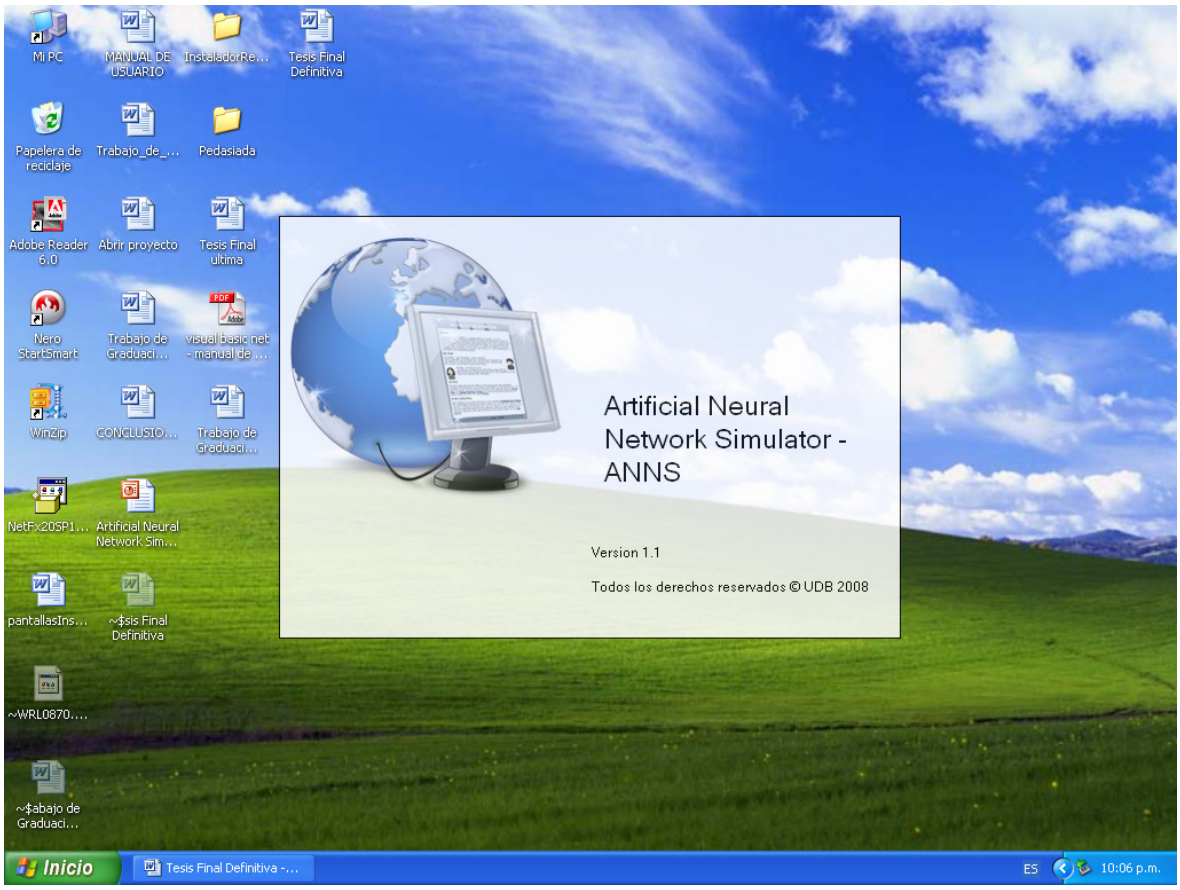


Figura 50. Ventana de lanzamiento del sistema.

4.2.2. Ventana Principal o de Inicio.

Es la primera de las ventanas con las que el usuario tiene contacto, en esta ventana se observan cinco pestañas, por medio de ellas se pueden alcanzar cualquiera de las cinco topologías con las que el simulador cuenta, ver figura 51.

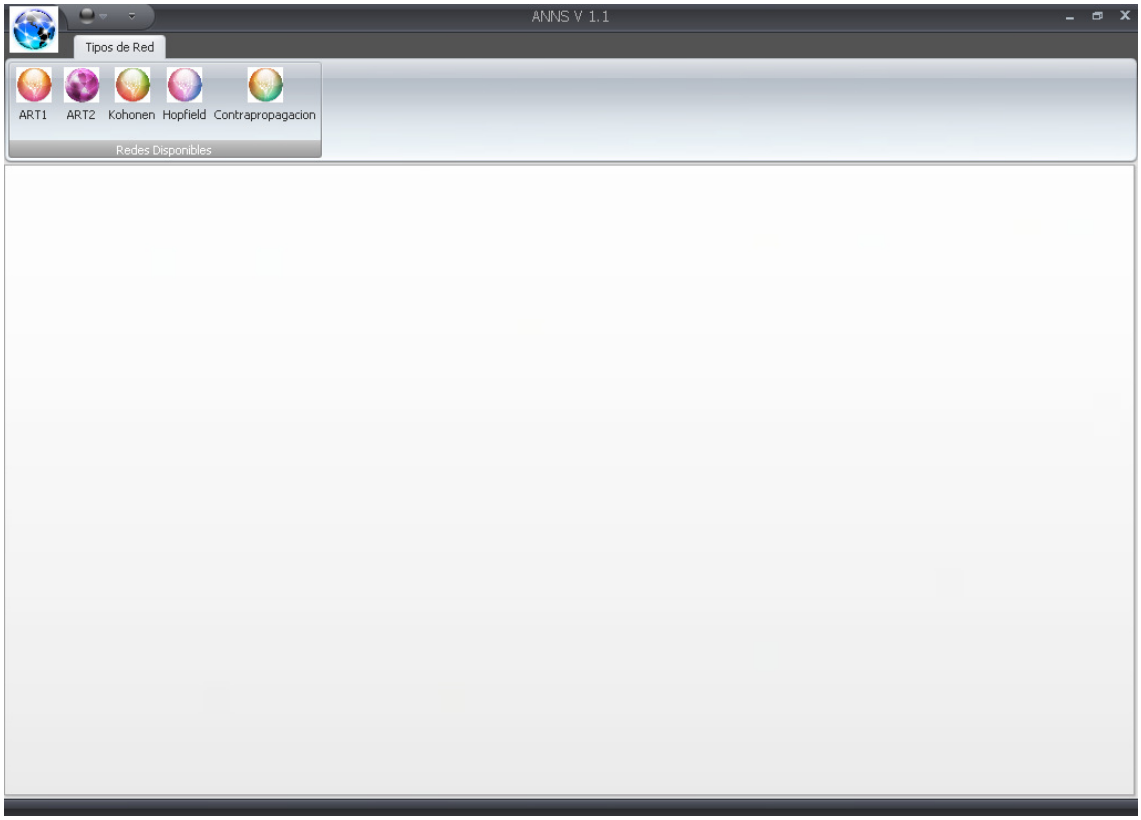


Figura 51. Ventana principal o de inicio.

4.2.3. Ventana de Configuración de la Red.

Se presenta cuando el usuario ha escogido la topología de la red que utilizará y decide introducir los parámetros de inicialización necesarios, previos a la etapa de entrenamiento. Dichos parámetros pueden variar dependiendo del tipo de red que se ha elegido.

En la ventana de configuración para cada tipo de red, se incluye una "Barra de Menú" con la cual, se pueden alcanzar todas las funciones principales del simulador y éstas, se encuentran clasificadas según el área de efecto: Archivo, Entrenamiento, Simulación, Ver y Ayuda.

Además se cuenta con una "Barra de Accesos Directos", donde se observa un grupo de botones para realizar diferentes procedimientos y así poder crear, simular y entrenar la red. Esta barra de accesos directos se describe mas adelante en el apartado 4.3.1," Creación de una Red Neuronal Artificial."

En esta misma ventana, el usuario tendrá contacto con un grupo de pestañas, que le ayudaran a ingresar los parámetros iniciales, patrones de entrenamiento y pesos iniciales de manera más cómoda. Además dentro de este grupo se incluye una pestaña para poder observar los pesos que se van obteniendo durante el entrenamiento y / o después de finalizado el entrenamiento, así como otra pestaña que muestra el diagrama de la Red (topología creada).

En la figura 52 se muestra la ventana de configuración para la Red Contrapropagación, para la cual los parámetros de configuración son:

- t_1: Numero de iteraciones para la primera fase de entrenamiento (desde la capa de entrada hasta la capa competitiva α).
- t_2: Numero de iteraciones para la segunda fase de entrenamiento (desde la capa competitiva hasta la capa de salida β).
- # de patrones: Cantidad de vectores de entrenamiento
- Categorías: Número de neuronas que formarán la capa competitiva (cantidad de categorías que se establecerán)

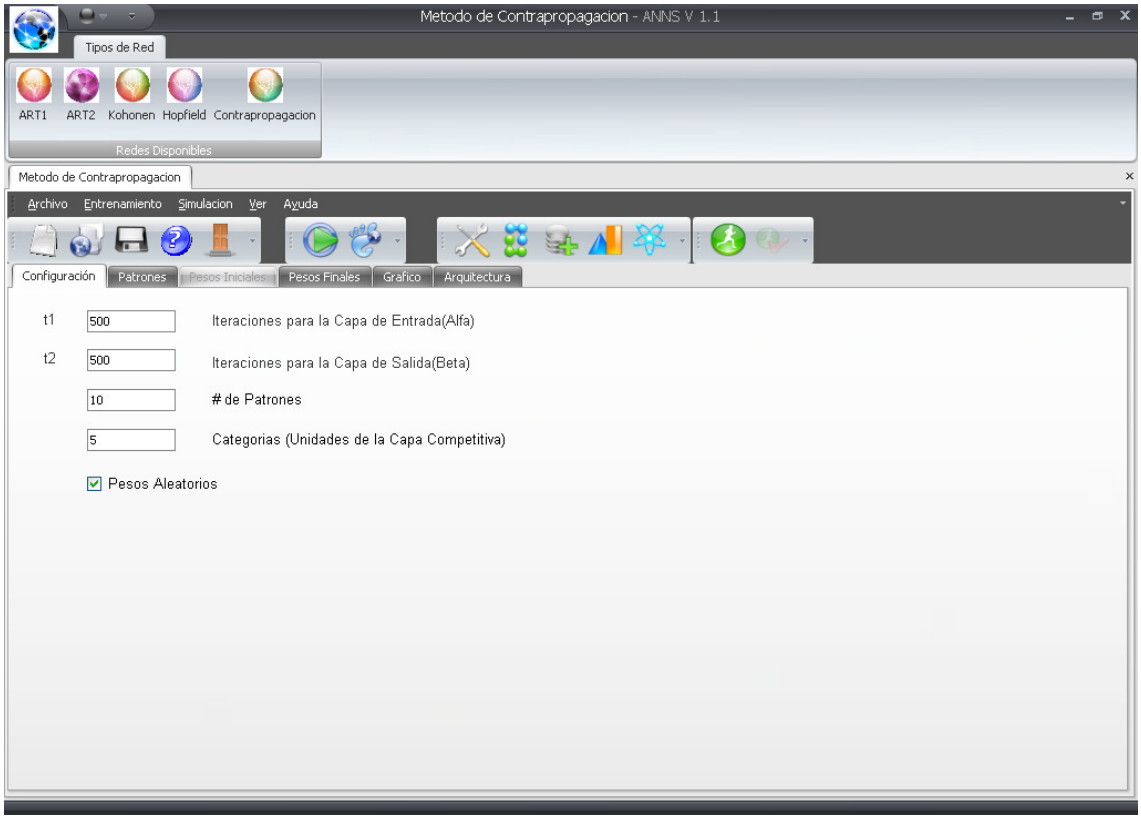


Figura 52. Pantalla de configuración.

Para la selección de los pesos iniciales aparece la opción de Pesos Aleatorios, si se hace clic sobre ella, entonces el software es capaz de generar los pesos necesarios de manera aleatoria y el usuario no debe preocuparse por ingresar de forma manual los vectores de peso. Si esta opción no es utilizada entonces se habilita una pestaña llamada “Pesos”, ver figura 54. Lo anterior equivale a una opción personalizada, entonces, desde ahí se introducirán los pesos iniciales, haciendo un click sobre la celda en la que se desea ingresar el valor. Otra forma es utilizando la tecla “tab” para desplazarse sobre las celdas y así, poder digitar en ellas los valores deseados.

4.2.4 Ventana de Ingreso de Patrones a Entrenar.

A través de la ventana, el usuario podrá ingresar los patrones que la red deberá aprender, ver figura 53.

El formato de los mismos puede ser en forma de vector o en forma de matriz, lo cual dependerá de la topología de red que se quiere entrenar.

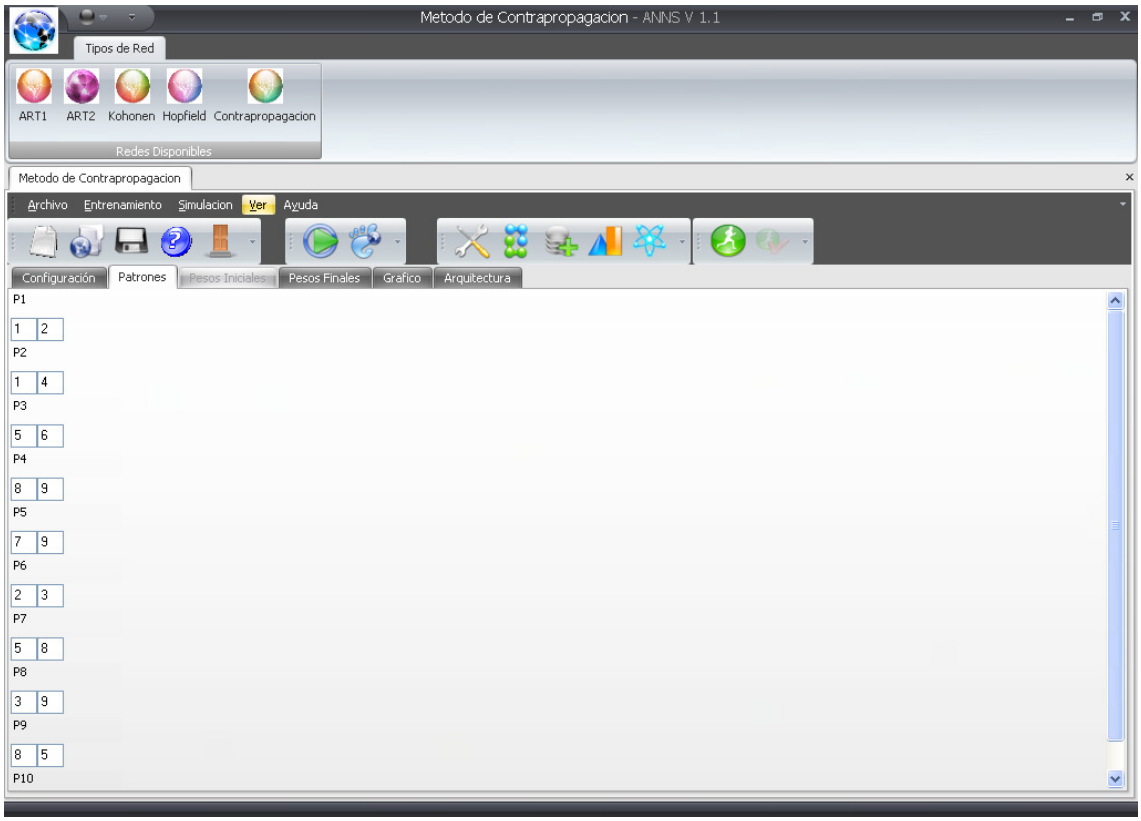


Figura 53. Ventana de patrones.

Para poder ingresar los vectores o patrones de entrenamiento, el usuario deberá seleccionar la pestaña “patrones”. Una vez abierta la ventana, el usuario deberá posicionarse sobre la celda donde se ingresará el dato y hará un click sobre la misma. Con lo anterior, la celda quedará habilitada y lista para digitar sobre ella el valor deseado el cual podrá ser de tipo unipolar, bipolar o continuo, dependiendo de la topología de red seleccionada. También utilizando la tecla “tab” para desplazarse sobre las celdas, las que se activarán y permitirán el ingreso de valores.

4.2.5. Ventana de Asignación de Pesos.

En la ventana de Asignación de Pesos es posible introducir los valores necesarios para cada una de las sinapsis correspondientes. Esta pestaña se activará después que el usuario, aún dentro de la pantalla de configuración no elija la opción asignación de pesos aleatorios, ver figura 54.

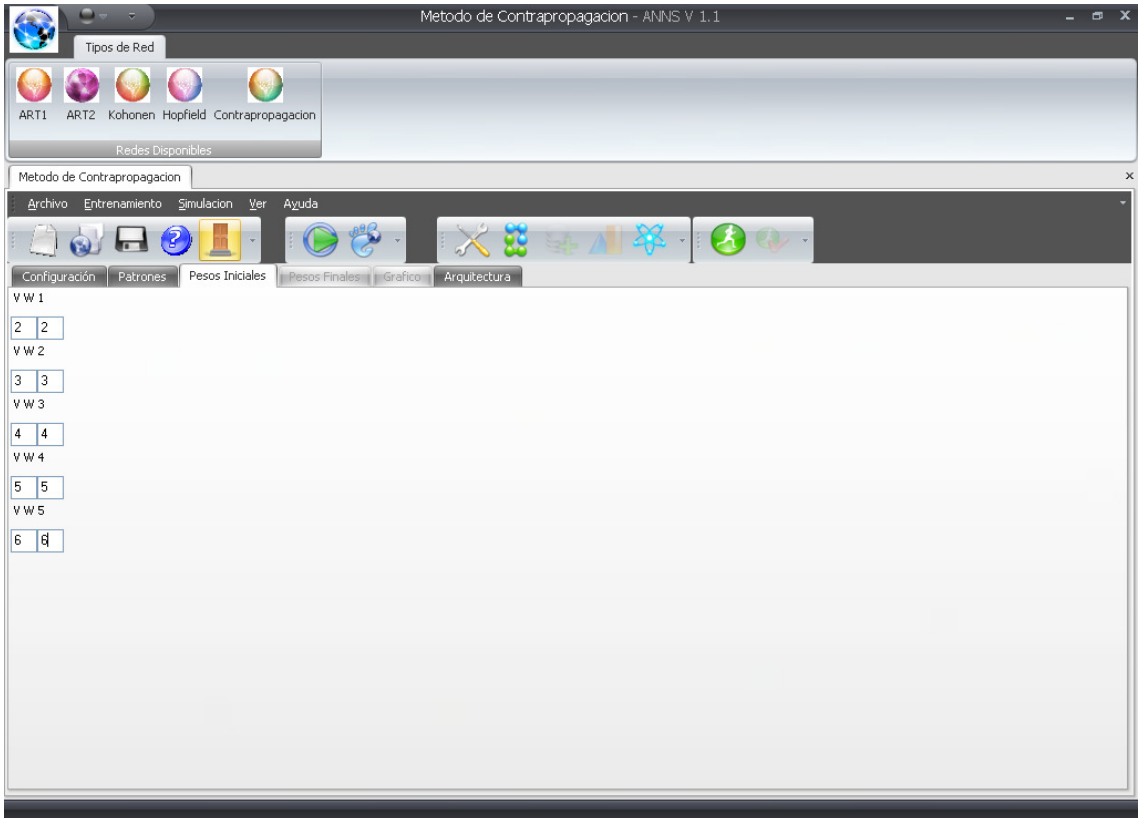
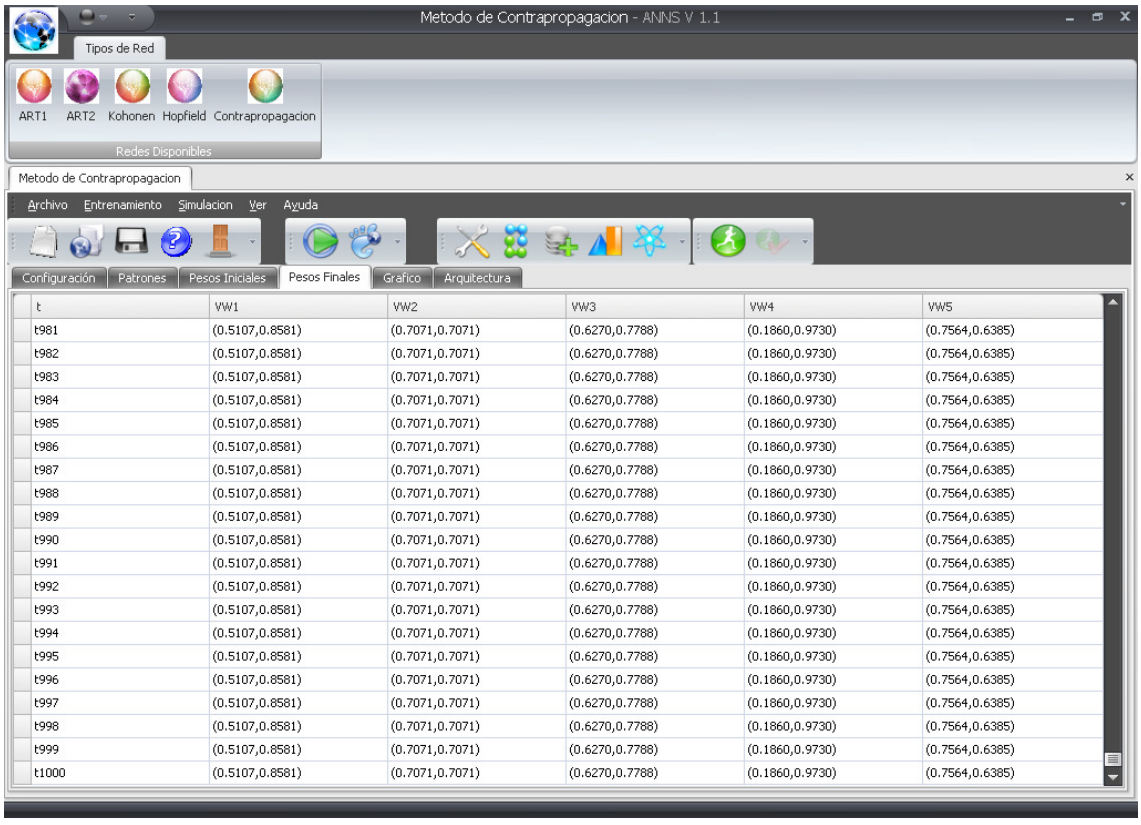


Figura 54. Ventana de asignación de Pesos.

En esta venta el usuario tendrá la opción de digitar los valores de los pesos, ubicándose sobre las celdas correspondientes. Para ingresar el dato se deberá hacer un click sobre ella; la celda se habilitará y se podrá digitar el dato deseado.

4.2.6. Ventana de Pesos Finales.

En esta pantalla se muestra la evolución de los pesos finales durante el proceso de entrenamiento. Para acceder a ésta información el usuario deberá seleccionar la pestaña llamada “Pesos finales”. Después de realizado lo anterior, aparecerá la ventana donde se observa la evolución de los pesos finales, ver figura 55. Por medio de un scroll, el usuario podrá desplazarse a lo largo y ancho de la ventana y de ese modo, accederá a todos los pesos que se van obteniendo durante el proceso de entrenamiento, hasta que el mismo finalice. La ventana de pesos finales se activa tanto en la modalidad de entrenamiento completo como en el entrenamiento paso a paso.



t	VW1	VW2	VW3	VW4	VW5
t981	(0.5107,0.8581)	(0.7071,0.7071)	(0.6270,0.7788)	(0.1860,0.9730)	(0.7564,0.6385)
t982	(0.5107,0.8581)	(0.7071,0.7071)	(0.6270,0.7788)	(0.1860,0.9730)	(0.7564,0.6385)
t983	(0.5107,0.8581)	(0.7071,0.7071)	(0.6270,0.7788)	(0.1860,0.9730)	(0.7564,0.6385)
t984	(0.5107,0.8581)	(0.7071,0.7071)	(0.6270,0.7788)	(0.1860,0.9730)	(0.7564,0.6385)
t985	(0.5107,0.8581)	(0.7071,0.7071)	(0.6270,0.7788)	(0.1860,0.9730)	(0.7564,0.6385)
t986	(0.5107,0.8581)	(0.7071,0.7071)	(0.6270,0.7788)	(0.1860,0.9730)	(0.7564,0.6385)
t987	(0.5107,0.8581)	(0.7071,0.7071)	(0.6270,0.7788)	(0.1860,0.9730)	(0.7564,0.6385)
t988	(0.5107,0.8581)	(0.7071,0.7071)	(0.6270,0.7788)	(0.1860,0.9730)	(0.7564,0.6385)
t989	(0.5107,0.8581)	(0.7071,0.7071)	(0.6270,0.7788)	(0.1860,0.9730)	(0.7564,0.6385)
t990	(0.5107,0.8581)	(0.7071,0.7071)	(0.6270,0.7788)	(0.1860,0.9730)	(0.7564,0.6385)
t991	(0.5107,0.8581)	(0.7071,0.7071)	(0.6270,0.7788)	(0.1860,0.9730)	(0.7564,0.6385)
t992	(0.5107,0.8581)	(0.7071,0.7071)	(0.6270,0.7788)	(0.1860,0.9730)	(0.7564,0.6385)
t993	(0.5107,0.8581)	(0.7071,0.7071)	(0.6270,0.7788)	(0.1860,0.9730)	(0.7564,0.6385)
t994	(0.5107,0.8581)	(0.7071,0.7071)	(0.6270,0.7788)	(0.1860,0.9730)	(0.7564,0.6385)
t995	(0.5107,0.8581)	(0.7071,0.7071)	(0.6270,0.7788)	(0.1860,0.9730)	(0.7564,0.6385)
t996	(0.5107,0.8581)	(0.7071,0.7071)	(0.6270,0.7788)	(0.1860,0.9730)	(0.7564,0.6385)
t997	(0.5107,0.8581)	(0.7071,0.7071)	(0.6270,0.7788)	(0.1860,0.9730)	(0.7564,0.6385)
t998	(0.5107,0.8581)	(0.7071,0.7071)	(0.6270,0.7788)	(0.1860,0.9730)	(0.7564,0.6385)
t999	(0.5107,0.8581)	(0.7071,0.7071)	(0.6270,0.7788)	(0.1860,0.9730)	(0.7564,0.6385)
t1000	(0.5107,0.8581)	(0.7071,0.7071)	(0.6270,0.7788)	(0.1860,0.9730)	(0.7564,0.6385)

Figura 55. Ventana de pesos finales de la Red Contrapropagacion.

4.2.7. Ventana de Simulación.

Es la ventana en la cual el usuario podrá realizar las simulaciones después de finalizado el proceso de entrenamiento.

Si el entrenamiento aún no ha terminado, entonces esta ventana no estará habilitada. La ventana se habilita solamente cuando el entrenamiento ha finalizado por completo. Para poder realizar el proceso de simulación de cualquiera de las cinco topologías de las que consta el simulador ANNS v 1.1, se deberá hacer click sobre la pestaña llamada “Simulación”.

El estilo de la simulación variará dependiendo del tipo de red que se utilice, ver figura 56.

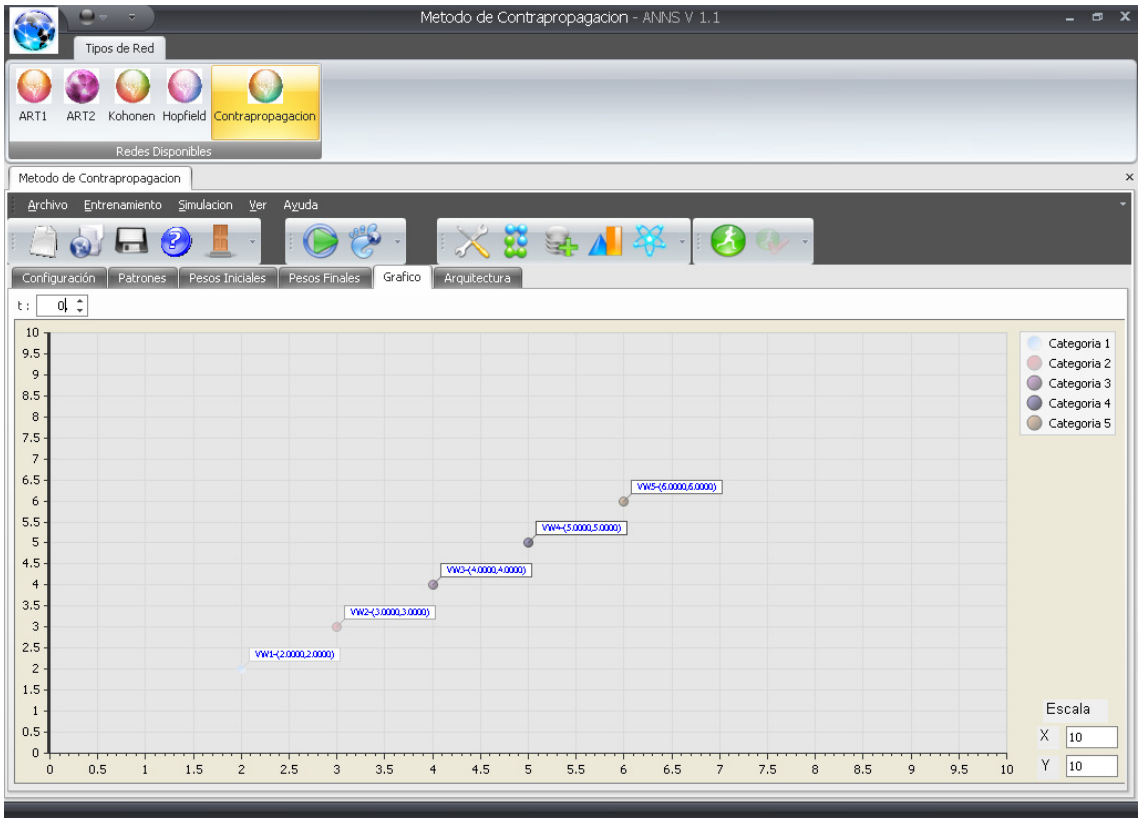


Figura 56. Ventana de simulación para la red Contrapropagación.

4.2.8. Ventana de Arquitectura.

En está ventana se muestra la arquitectura de cada una de las redes creadas en el simulador. Aquí el software ANNS v1.1, genera el diagrama de la red creada con el número de neuronas y conexiones entre ellas, las que dependerán de la configuración realizada cuando se crea la red. La ventana se muestra en la figura 57.

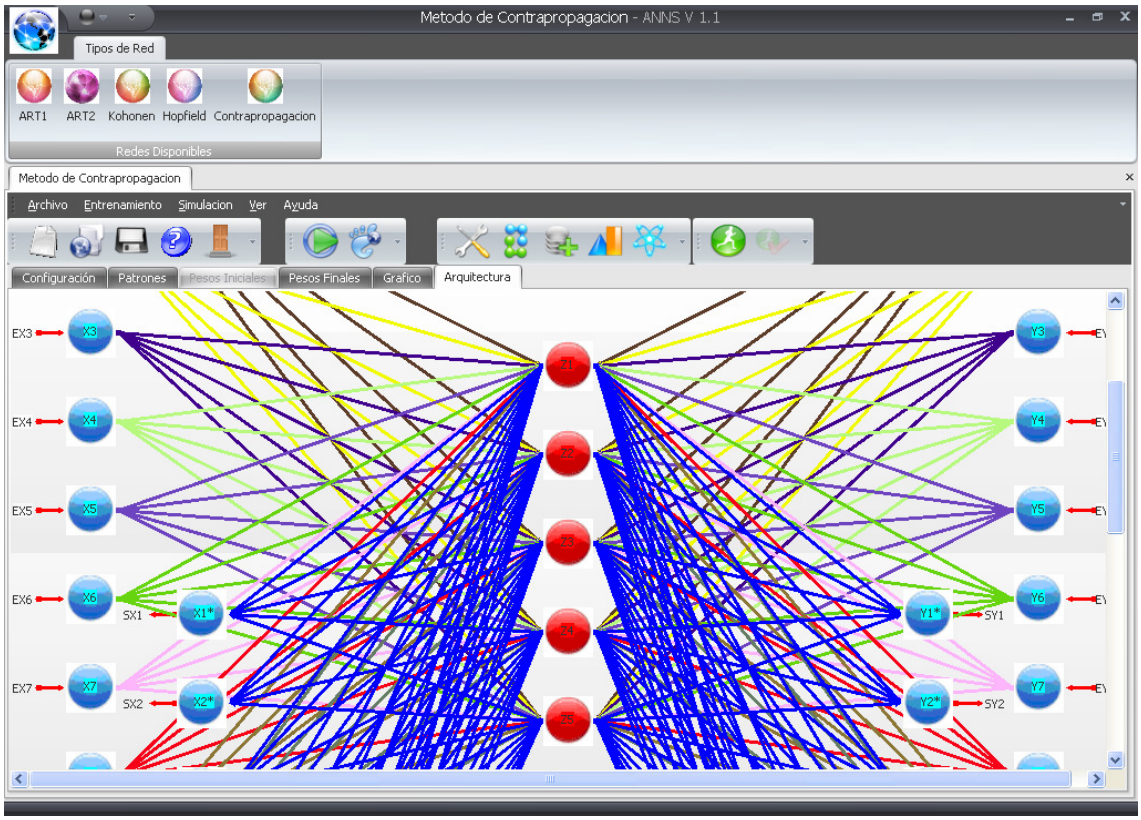


Figura 57. Ventana de arquitectura.

4.3 PASOS PARA CREAR UNA RED NEURONAL ARTIFICIAL CON EL SOFTWARE ANNS V1.1.

En este capitulo se explica como generar, entrenar y simular una red Neuronal Artificial utilizando el simulador ANNS v1.1, para las diferentes topologías de redes que contiene el simulador.

Los temas tratados en este capitulo son:

- ✓ Creación de una Red Neuronal Artificial.
- ✓ Creación de la Red ART1
- ✓ Creación de la Red ART2
- ✓ Creación de la Red Hopfield
- ✓ Creación de la Red Kohonen
- ✓ Creación de la Red Contrapropagación

4.3.1. Creación de una Red Neuronal Artificial.

Una vez iniciado el software ANNS Versión 1.1, el primer paso que se debe seguir es determinar el tipo de Red Neuronal Artificial con la cual se va a trabajar. Esto se realizará desde la "Ventana de Inicio" de la figura 58, dentro de ésta ventana, el usuario deberá ubicarse en la pestaña llamada "Tipos de Red" y hacer un click, sobre el icono correspondiente a la topología elegida.

Las topologías con las que el software cuenta son: ART1, ART2, Hopfield, Kohonen y Contrapropagación, ver figura 59.

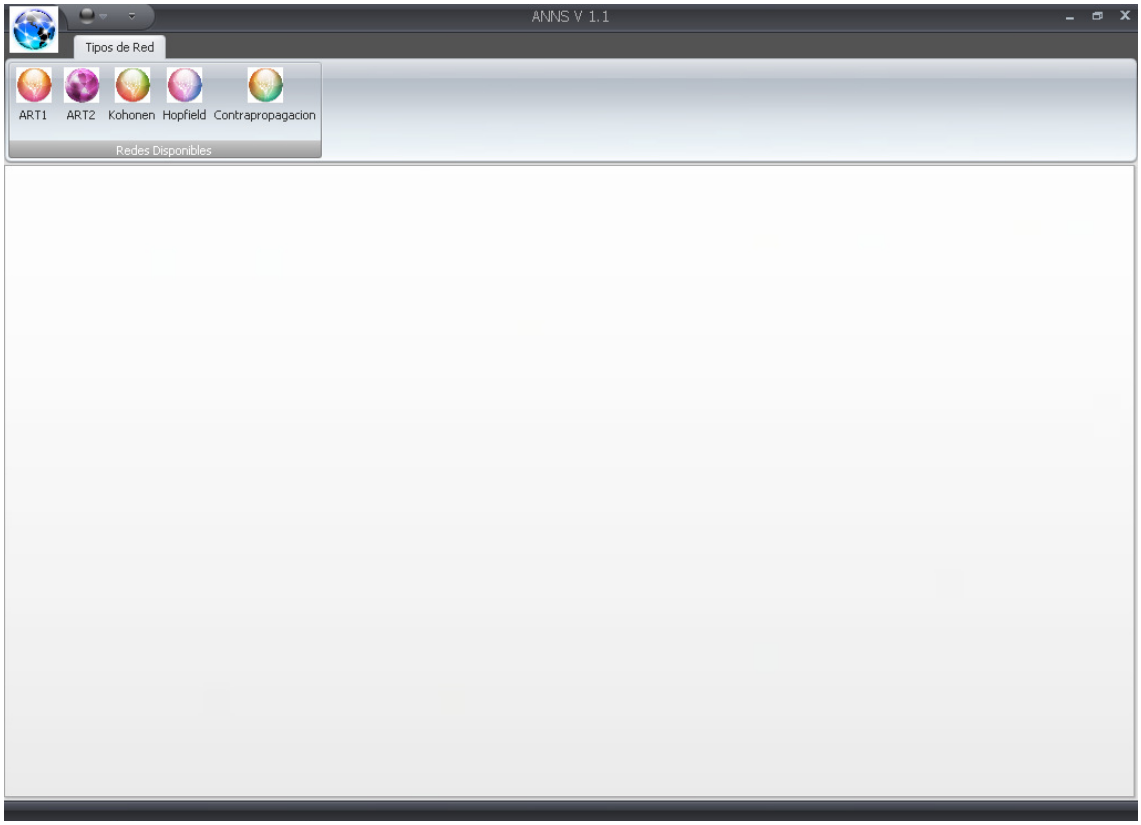


Figura 58. Ventana de Inicio.

Figura 59. Tipo de Red.

Cuando el usuario ha elegido la topología de Red, aparecerá la “Ventana de Configuración”, y dentro de esta ventana, una barra a la que llamaremos “Barra de Accesos Directos”. En esta barra se encuentran los siguientes botones: Nuevo, Abrir, Guardar, Salir; Entrenamiento Completo, Entrenamiento Paso a Paso, Configuración, Patrones, Pesos Finales, Gráficos, Arquitectura de la Red, Galerías, Ayuda, Simular y Reconocer, ver figura 60.

Figura 60. Barra de Accesos Directos.

A continuación se describe la función de cada uno de los botones mencionados en el párrafo anterior:

NUEVO: Al hacer un click sobre este botón, el software habilita la ventana de configuración de parámetros para ingresar los valores necesarios y, así, entrenar la red seleccionada.

ABRIR: Con este botón se habilita la opción de cargar archivos de las redes previamente guardadas.

GUARDAR: Este icono permite guardar la configuración actual de la Red con la que se está trabajando.

GUARDAR COMO: Permite al usuario realizar el guardado en diferentes localidades del sistema.

AYUDA: Cuando se selecciona este botón, se desplegará una ventana donde el usuario accederá al manual de usuario.

SALIR: Por medio de este botón el usuario habilita la opción de salir, levantándose una ventana para guardar la configuración actual de la red.

ENTRENAR: Con este botón se habilita el entrenamiento completo de la red con la que se esta trabajando.

PASO A PASO: Se habilita el entrenamiento paso a paso de la red actual. Es necesario que el usuario haga click las veces que sea necesario sobre este botón, para poder finalizar el entrenamiento.

DETENER: Por medio del botón, se permite al usuario detener en cualquier momento el proceso de entrenamiento. Este botón se habilita cuando se inicia el proceso de entrenamiento y se deshabilita cuando el proceso de entrenamiento ha finalizado por completo

CONFIGURACIÓN: Habilita la ventana de configuración, dando lugar a la opción de poder cambiar la configuración actual de la red con la que se está trabajando.

PATRONES: Permite al usuario acceder a la ventana “Patrones” habilitando la opción de poder realizar cambios en los patrones o vectores de entrenamiento.

PESOS FINALES: Habilita la ventana “Pesos Finales”, para observar la evolución de los mismos durante el entrenamiento o cuando este ha finalizado.

PESOS INICIALES: Permite al usuario habilitar la ventana de “Pesos” para las redes que utilizan este parámetro de inicialización, desde ahí se pueden realizar cambios en los pesos iniciales.

PESOS ASCENDENTES: Habilita la ventana de “Pesos Ascendentes”, brindando la opción de poder observar la evolución de estos pesos desde el principio hasta el final del entrenamiento. Este botón esta disponible solo para las redes ART1 y ART2.

PESOS DESCENDENTES: Habilita la ventana de “Pesos Descendentes”, brindando la opción de poder observar l a evolución de estos pesos desde el principio hasta el final del entrenamiento. Este icono esta disponible solo para las redes ART1 y ART2.

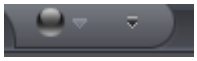
GRAFICOS: Con este botón el usuario puede habilitar la ventana “Grafico”, permitiéndole a este, observar la gráfica de los pesos finales obtenidos después de finalizado el entrenamiento. Este icono está disponible únicamente para la Red Kohonen y Contrapropagación.

GALERIAS: Despliega una lista ejemplos (uno por cada red), ya predeterminados

ARQUITECTURA DE LA RED: Muestra la arquitectura de la red creada por el usuario.

SIMULAR: Habilita la ventana de Simulación de la Red, permitiéndole al usuario, realizar el proceso de simulación de la red actual, que dependerá del tipo de red utilizada.

RECONOCER: Cuando la red actual ha sido entrenada para el reconocimiento de patrones, entonces, este botón permite a l usuario iniciar el proceso de reconocimiento de un patrón de entrada cualquiera y de ese modo comprobar si la red ha aprendido o no.



QUICK ACCES TOOLBAR: Este botón se utiliza para acceder a un menú de funciones que permite realizar cambios de color o del tipo de presentación de las ventanas del ANNS V1.1

Algunos de estos botones inicialmente estarán deshabilitados, sin embargo, a medida se avance en el proceso de creación, entrenamiento y simulación de la red, estos se habilitarán.

Para poder introducir los parámetros iniciales de cada red, es necesario habilitar la "Ventana de Configuración", para ello, el usuario deberá determinar si desea realizar la creación de una red nueva o trabajar con una previamente guardada. Además esta ventana puede habilitarse también a través de la opción llamada "Galerías".

Para poder crear una red nueva, se deberá de hacer un click sobre el botón "Nuevo", ubicado dentro de la "Barra de Accesos Directos", ver figura 61.

Figura 61. Barra accesos directos que muestra botón para generar una nueva Red.

Con la acción anterior se habilitará la "Ventana de Configuración" para poder introducir dentro de la misma, los valores de los parámetros iniciales de la topología de red que se ha elegido.

Si lo que se necesita es abrir una red previamente guardada, entonces se hará un click sobre el botón "Abrir".

Inmediatamente dentro de la "Ventana de Configuración", aparecerá otra ventana llamada "Buscar en", la que por defecto busca los archivos con extensión .rn en Mis Documentos. Sin embargo el usuario puede buscar sus

archivos en la localidad donde están previamente guardados, con solo cambiar la ruta de búsqueda dentro de esta ventana.

Cuando el usuario ha ubicado el archivo que abrirá, lo seleccionará con el Mouse, haciendo doble click sobre él o haciendo un click y luego otro click sobre el botón "Abrir" de la ventana "Buscar en ", ver figura 62.

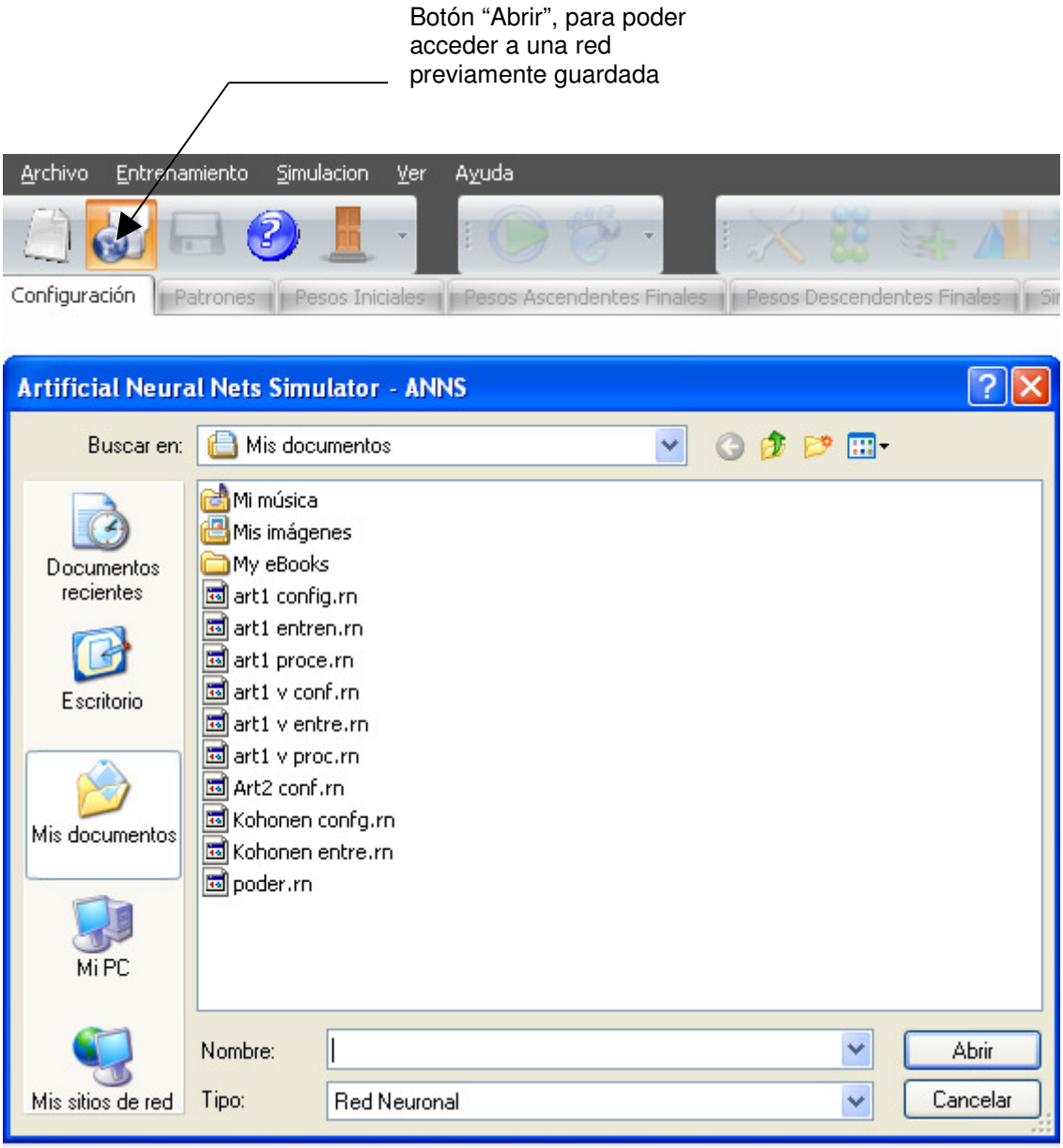


Figura 62. Ventana que muestra la opción "abrir" una Red previamente guardada.

Cuando el usuario abre uno de los archivos previamente guardados, se habilitará la "Ventana de Configuración", se observará que los parámetros iniciales aparecen ya dentro de la ventana. Sin embargo si se realizan cambios en la red, el software lo permite y el usuario puede hacer modificaciones en la

creación de la red previamente guardada. Los cambios podrán ser guardados o no.

Por otro lado, si el usuario desea hacer uso de los recursos propios del Simulador ANNS V1.1, tiene la opción de abrir los ejemplos previamente configurados, entrenados y simulados, que se encuentran guardados en la "Galería" del software.

Dentro de cada topología de red, existe una galería exclusiva inherente a ella, donde los ejemplos guardados son solamente para el tipo de red que se esta trabajando. Para poder utilizar estos ejemplos el usuario deberá de buscar dentro de la barra de menús la opción llamada "Ver" al hacer un click se desplegará un Menú, y se elegirá la opción llamada "Galerías", ver figura 63.

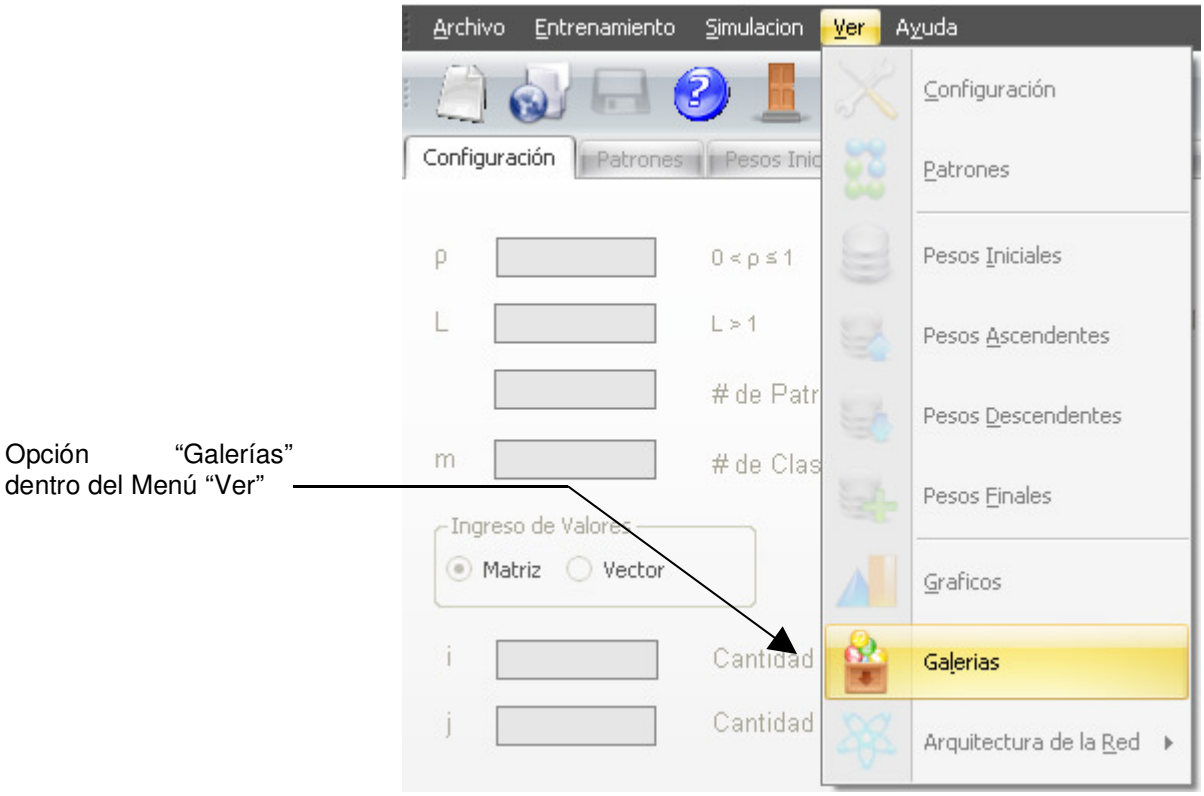


Figura 63. Menú que muestra la opción "Galerías".

Al hacer un click sobre esta opción, se desplegará una lista de los archivos de ejemplo contenidos dentro del simulador para cada red específica. Se observará que aparece una ventana en la cual se seleccionará el archivo que se desea abrir

haciendo doble click sobre este o con un click y luego otro sobre el botón abrir que se encuentra dentro de ésta ventana, ver figura 64.

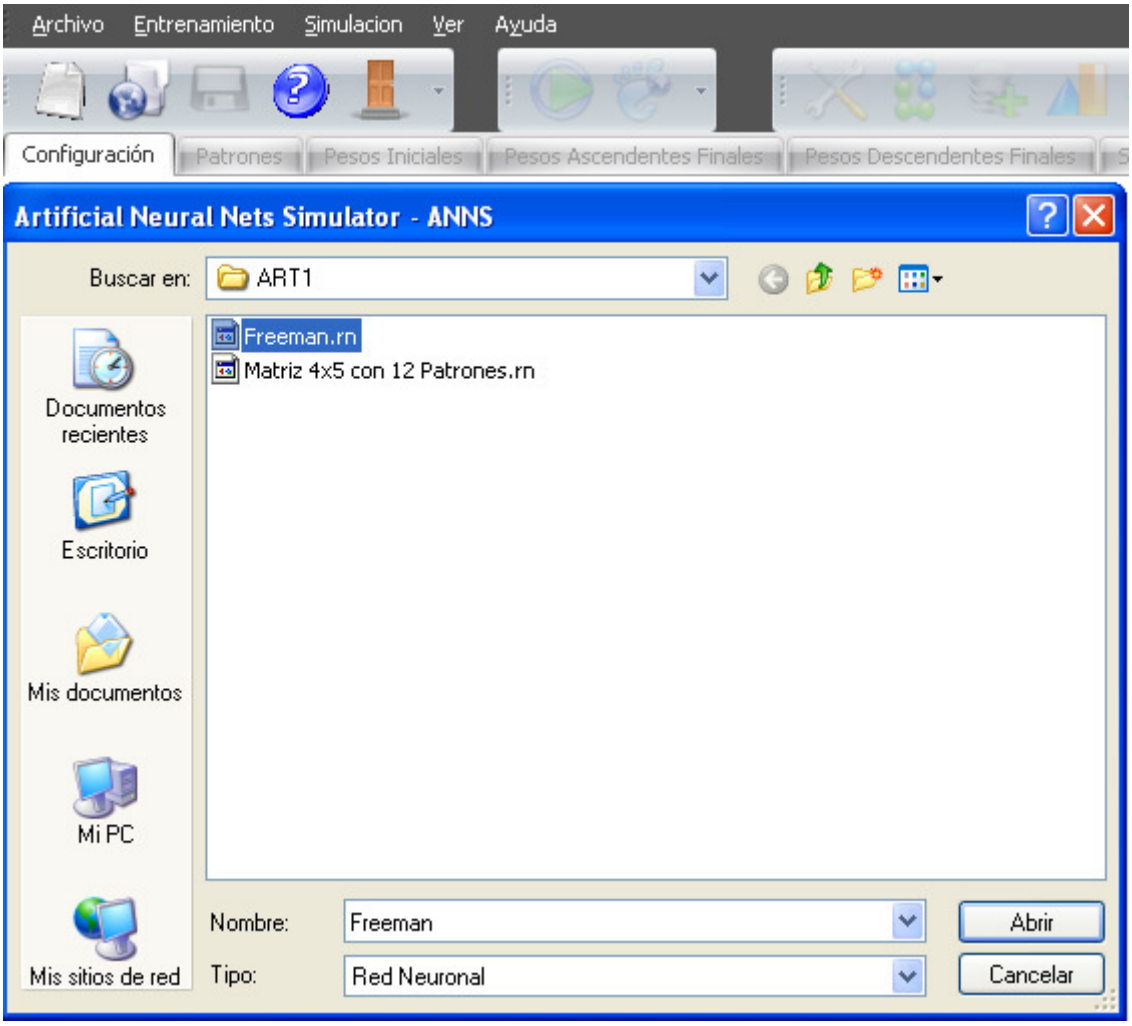


Figura 64. Ventana que muestra los archivos contenidos en la Galería de la Red ART1.

Cuando se ha abierto alguno de los archivos de la galería, también se habilitará la "Ventana de Configuración" observándose que la ventana se encuentra con todos los valores propios del ejemplo que se ha abierto.
El usuario puede realizar cambios en la configuración de la red actual y proceder con el entrenamiento y simulación de la misma, pero, no podrá guardar estos cambios, puesto que son ejemplos propios de la Galería.

Cuando se ha activado la Ventana de Configuración por medio de cualquiera de los tres procesos descritos anteriormente, se puede proceder a digitar dentro de esta, los valores de inicio o realizar cambios en los valores de las redes que se

han abierto por medio del botón “abrir” o por la opción “Galerías” dentro del Menú “Ver”.

Si el usuario comete algún error al introducir o cambiar los valores de inicio de la red, entonces se observará que aparece una ventana que indica el error que se ha cometido. Por ejemplo en la figura 65, se observa la ventana de error que aparece al digitar un valor erróneo del parámetro L.

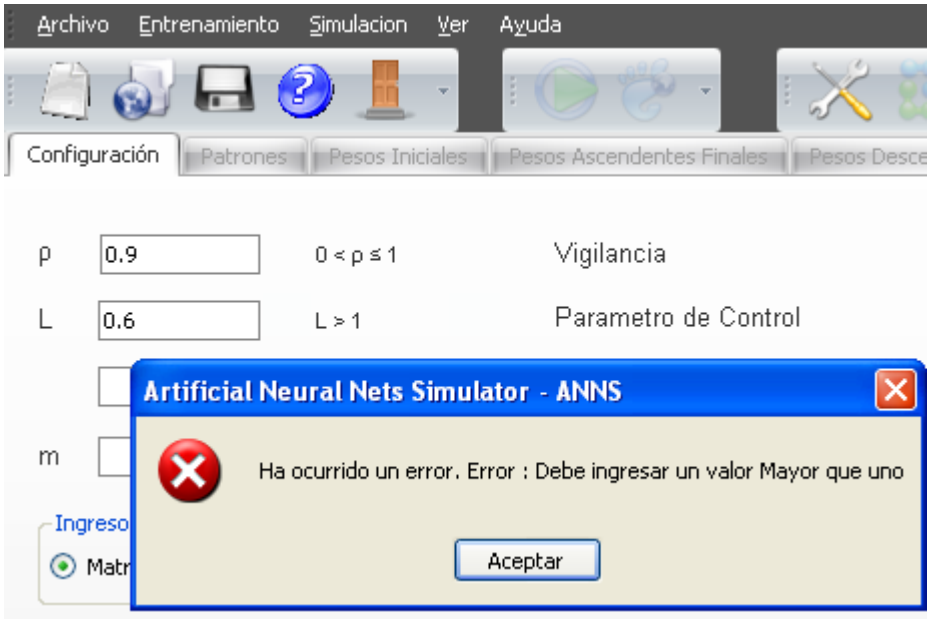


Figura 65. Ventana que muestra el tipo de error cometido al introducir un valor no adecuado en algún parámetro dentro de la ventana de configuración.

Si la Ventana de Configuración esta completa, entonces, se procede a introducir los vectores de Peso Inicial para las redes que así lo requieren. Si la topología de la Red que se esta creando no necesita de los pesos iniciales, entonces se deberá de pasar directamente a la introducción de los patrones o vectores de entrenamiento según sea el caso.

El Software ANNS v1.1, incorpora las opciones de “Guardar” y “Guardar como”. Con la opción Guardar como, el usuario puede guardar la Red actual en cualquiera de las etapas en la que esta se encuentre es decir no importa si la red esta en proceso de configuración de parámetros iniciales o en etapa de introducción de los patrones o en la etapa de entrenamiento o cuando esta ha terminado de entrenarse.

Con hacer un click sobre esta opción que se encuentra dentro del Menú “Archivo”. De la figura 66, se desplegará una ventana donde el software por defecto Buscara en la carpeta Mis Documentos de la figura 67, no obstante, el usuario podrá modificar la ruta de guardado según sus necesidades.

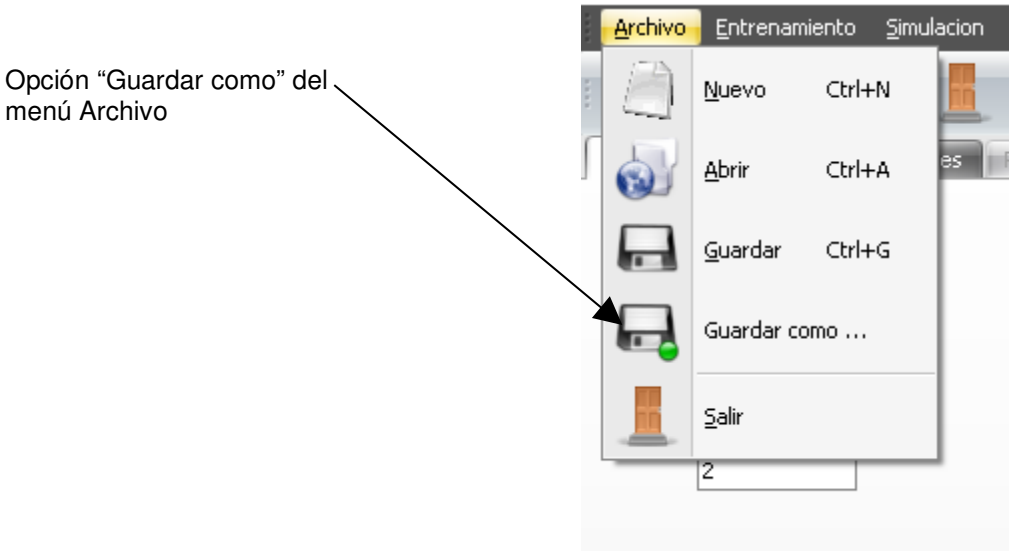


Figura 66. Opción “Guardar como” dentro del menú Archivo.

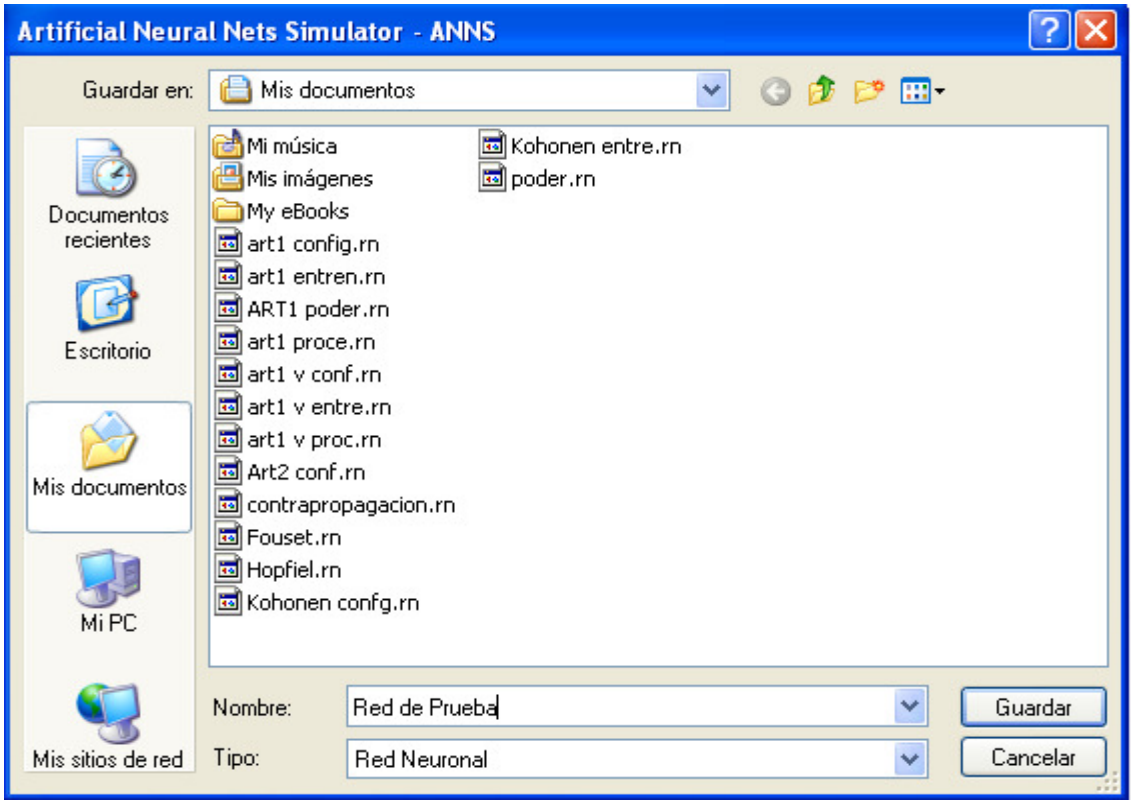


Figura 67. Ventana donde se muestra la ubicación o ruta para guardar el archivo con extensión .rn Generado.

Para guardar la red actual se debe escribir un nombre cualquiera dentro de la ventana que se observa en la figura 67, y luego hacer un click sobre la opción “Guardar” dentro de la misma ventana. El software automáticamente asignará una extensión .rn al archivo guardado.

La opción “Guardar” que aparece tanto en el menú “Archivo” dentro de la Barra de Menú como en la “Barra de Accesos Directos”, se utiliza para guardar los cambios realizados a la red actual, a la que, anteriormente le ha sido asignada una extensión .rn con la opción “Guardar como”.

Cabe recalcar que la red o proyecto actual, puede guardar los datos que se obtienen durante el entrenamiento sin necesidad que este finalice. Si el usuario desea detener en algún punto el entrenamiento y desea guardar los pesos obtenidos hasta ese momento lo puede hacer por medio de la opción detener.

Cuando el entrenamiento inicia se observa que en la posición del botón entrenar, en su lugar, aparece la opción “Detener” de la figura 68. Para detener el entrenamiento se deberá hacer un click sobre dicho botón; se observará la reaparición del botón Entrenar en lugar del botón Detener.
Cuando se ha detenido el proceso se podrán guardar los pesos finales obtenidos hasta el momento en que se detuvo el proceso de entrenamiento.

Figura 68. Botón de detener dentro de la barra de accesos directos.

4.3.2 Creación de una Red ART1.

El proceso para crear una Red ART1, consiste en que el usuario seleccione esta topología, desde la ventana de inicio de la figura 69, haciendo un click sobre el botón llamado ART1. Una vez realizada esta acción aparecerá la ventana de configuración pero no activada para la Red, ver figura 70.

Figura 69. Botón de selección para la red ART1.

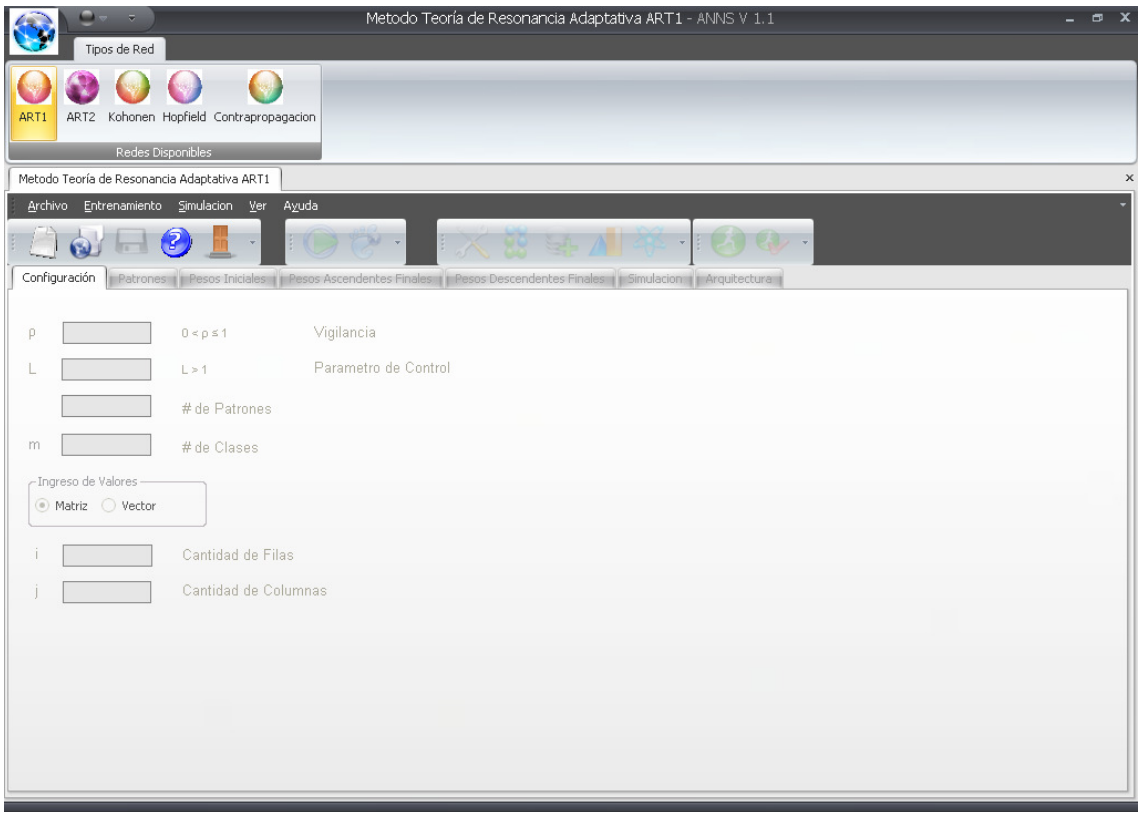


Figura 70. Ventana de Configuración no activada Red ART1.

Para generar un nuevo proyecto se realizara la activación del botón “NUEVO” que se encuentra en la barra de acceso directo figura71. Habilitando la pagina de configuración que se encuentra en la figura 72,

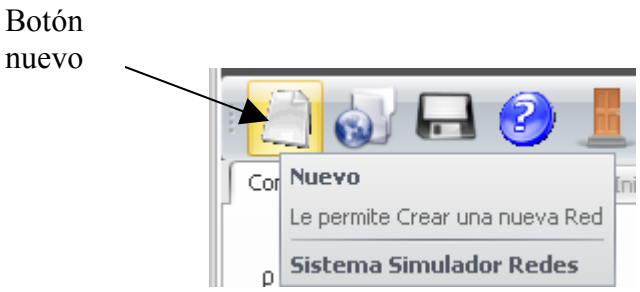


Figura 71. Botón para abrir nuevo proyecto Red ART1.

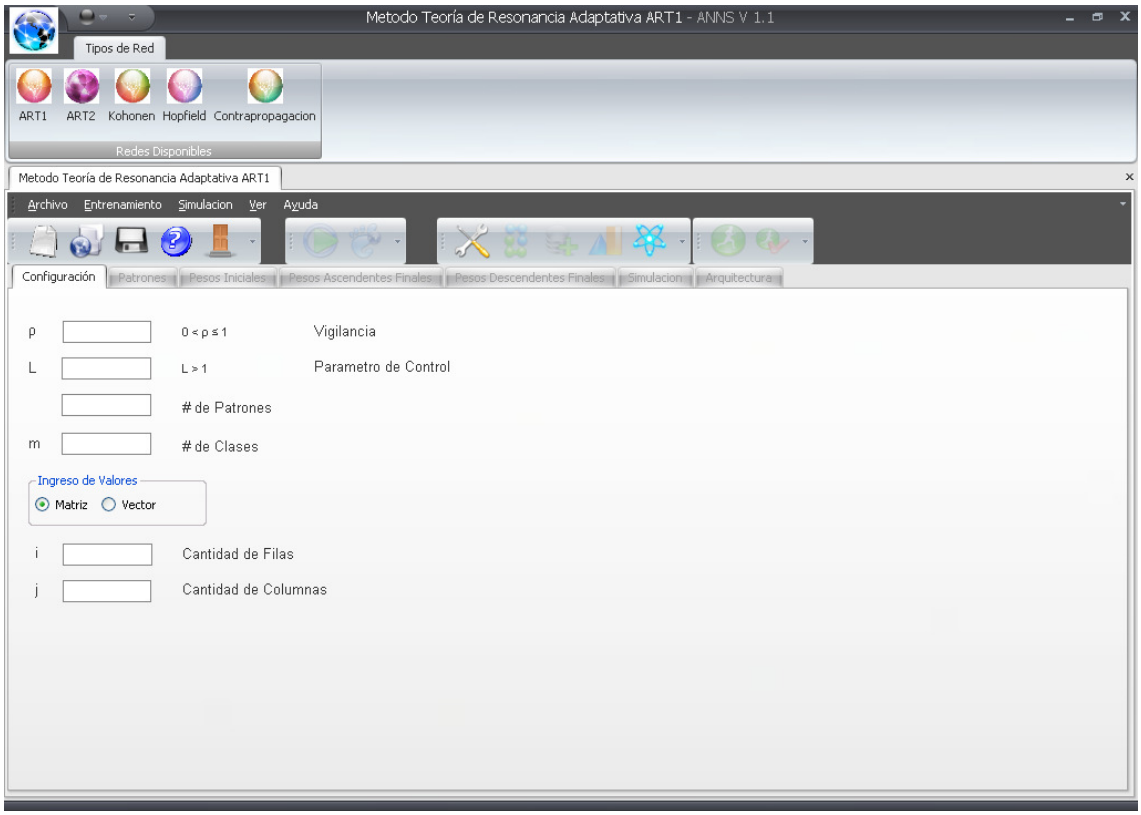


Figura 72. Ventana de configuración activada Red ART1.

Dentro de la ventana de configuración, el usuario deberá digitar los valores correspondientes para los parámetros de inicio. Es necesario que el usuario introduzca los siguientes parámetros iniciales correspondientes a la figura 73, teniendo en cuanta las restricciones en los valores que se usaran:

ρ: Parámetro de vigilancia.
L: Parámetro de Control.
Número de Patrones de entrenamiento.
m: Número de Clases.

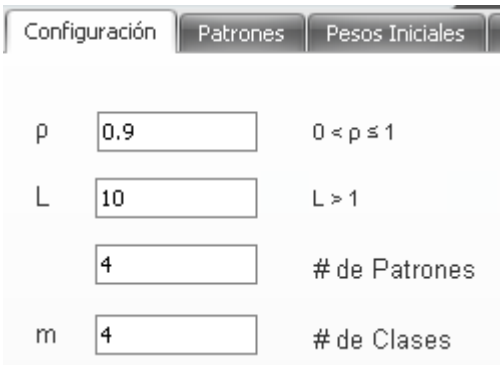


Figura 73. Figura que muestra los parámetros iniciales para la Red Art1.

Además se debe definir si los patrones de entrenamiento serán en formato vector o matriz. Si el formato seleccionado es matriz, entonces se deberá de digitar el número de filas y columnas necesarias para el patrón o patrones que se quieren entrenar, ver figura 74.

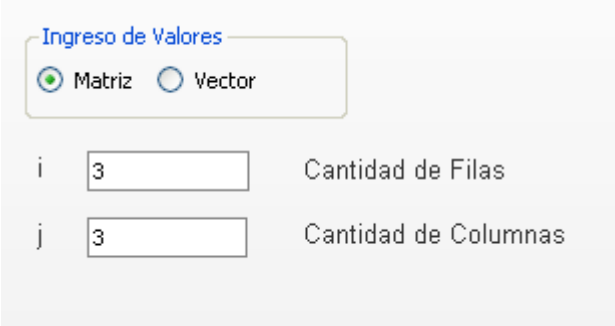


Figura 74. Ingreso de valores en forma de matriz de la Red Art1.

Una vez finalizado el proceso anterior, se selecciona la pestaña “Patrones”, haciendo un click sobre ésta. En la ventana “Patrones” el usuario deberá de introducir los patrones que desea que la red aprenda, si se trata de matrices la ventana habilitada será la que se muestra en la figura 75.

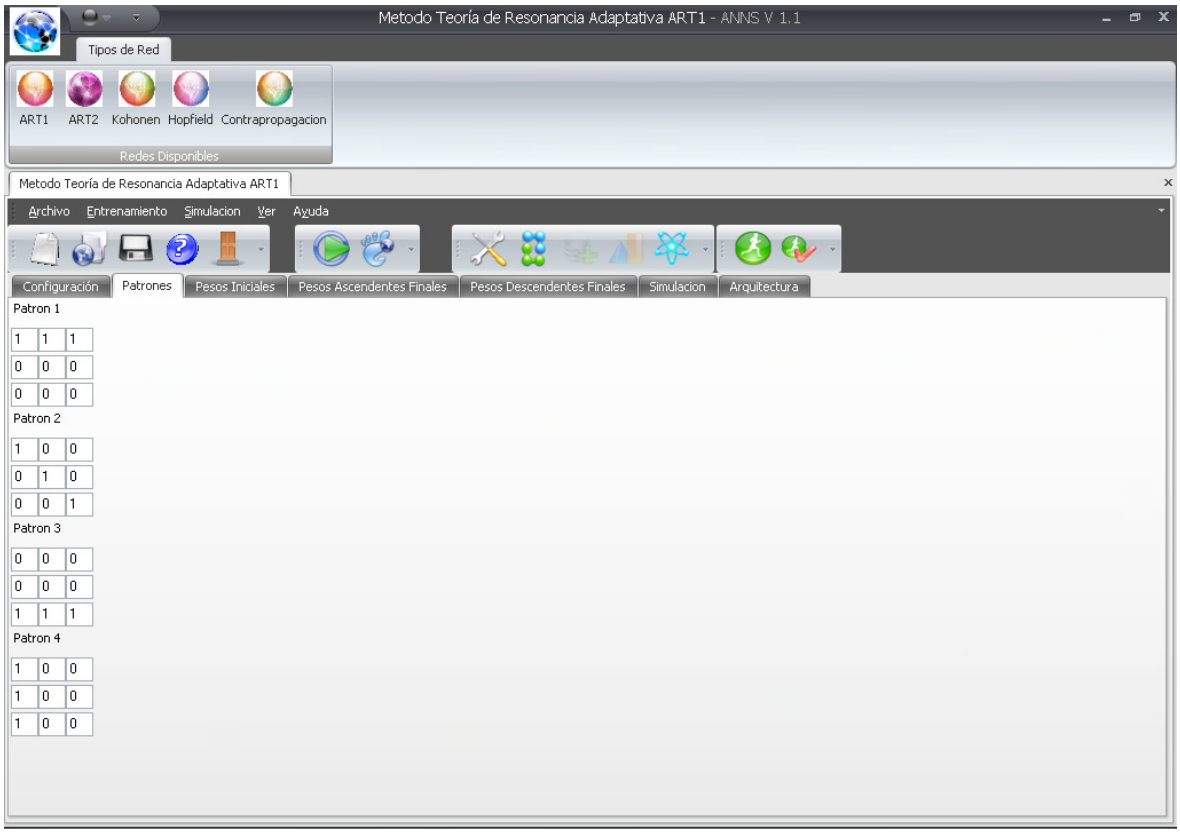


Figura 75. Ventana de patrones en forma de matriz de la Red Art1.

Si la opción seleccionada es en forma de vector, entonces se deberá de digitar el número de elementos que poseerá el vector o vectores de entrenamiento, ver figura 76.

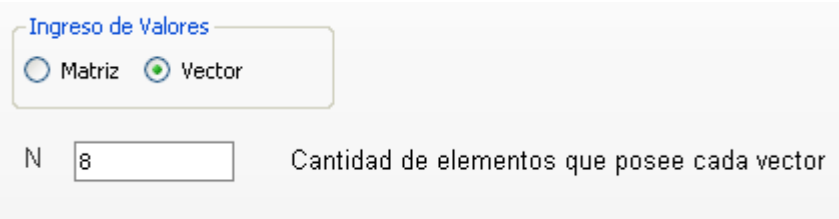


Figura 76. Ingreso de valores en forma de vector de la Red Art1.

Si se trata de Patrones de aprendizaje en forma de vector la ventana tendrá la apariencia mostrada en la figura 77.

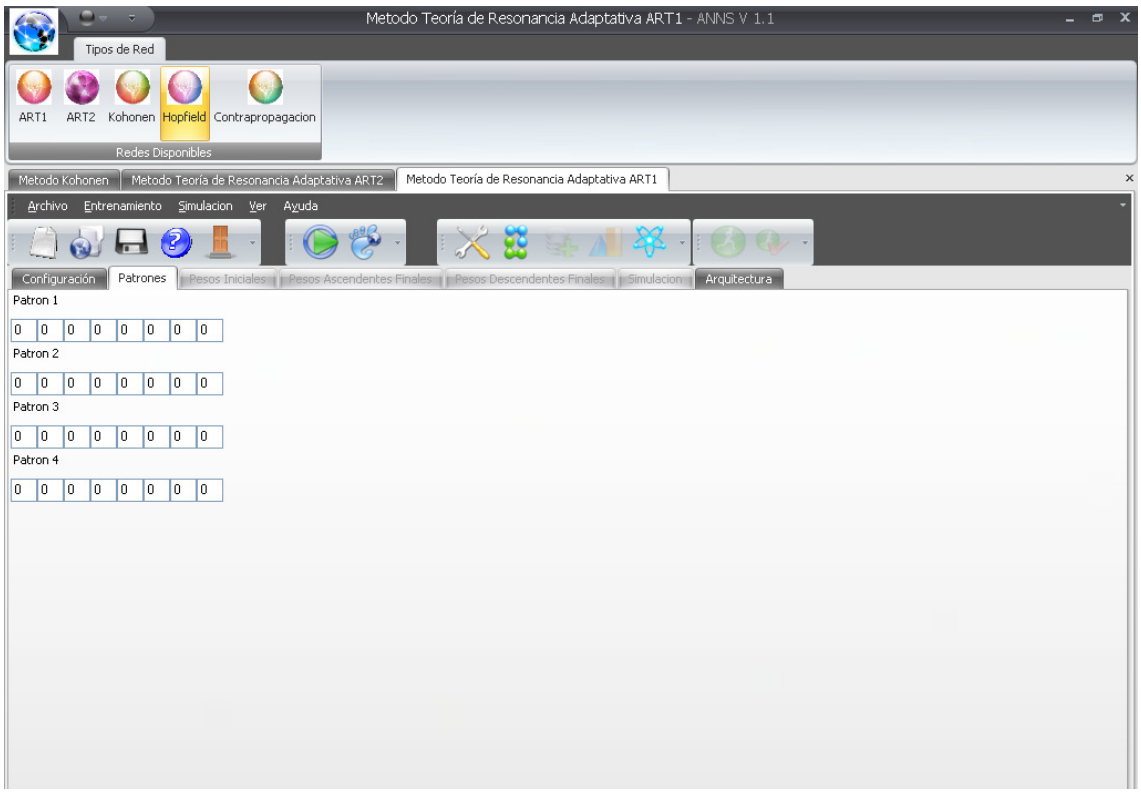


Figura 77. Ventana de patrones en forma de vector de la Red Art1.

Después de introducidos los patrones y los pesos iniciales, quedara a opción del usuario si el entrenamiento lo realiza de una vez o paso a paso, haciendo click sobre uno de los dos botones que habilitan las opciones antes mencionadas, ver figura 77.

Figura 78. Botones de selección de entrenamiento completo o paso a paso de la Red Art1.

Si el usuario quiere detener el entrenamiento deberá de hacer un click sobre el botón “Detener”, ver figura 79.

Figura 79. Botón detener de la Red Art1.

Cuando el proceso de entrenamiento ha finalizada el software mostrara al usuario una ventana que indica que esta acción ha ocurrido, ver figura 80.

Figura 80. Ventana que muestra cuando el proceso de entrenamiento ha finalizado de la Red Art1.

Cuando el entrenamiento ha concluido se habilitan las pestañas que muestran los pesos iniciales de la figura 81, los pesos ascendentes finales de la figura 82 y los pesos descendentes finales, ver figura 83.
Para poder habilitar cualquiera de las ventanas anteriores se deberá de hacer un click sobre la pestaña deseada.

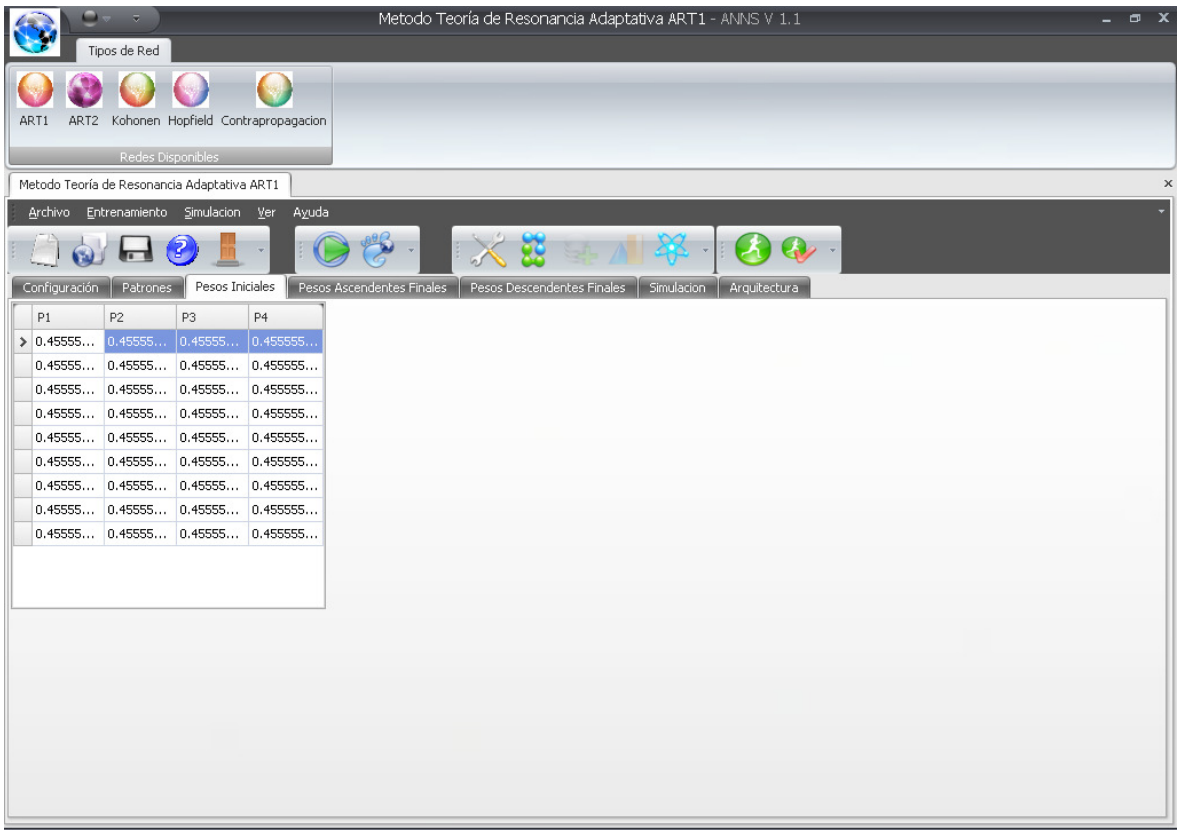


Figura 81. Ventana que muestra los pesos iniciales de la red Art1.

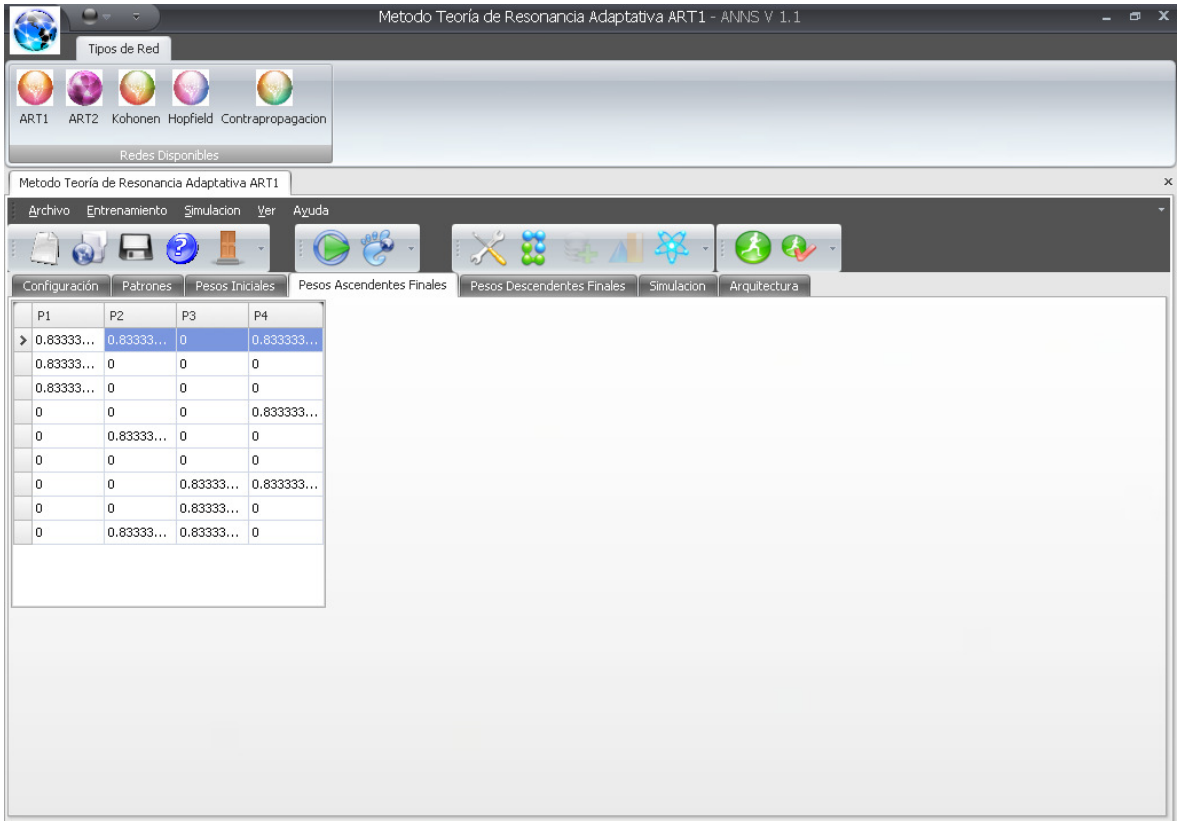


Figura 82. Ventana que muestra los pesos ascendentes finales de la Red Art1.

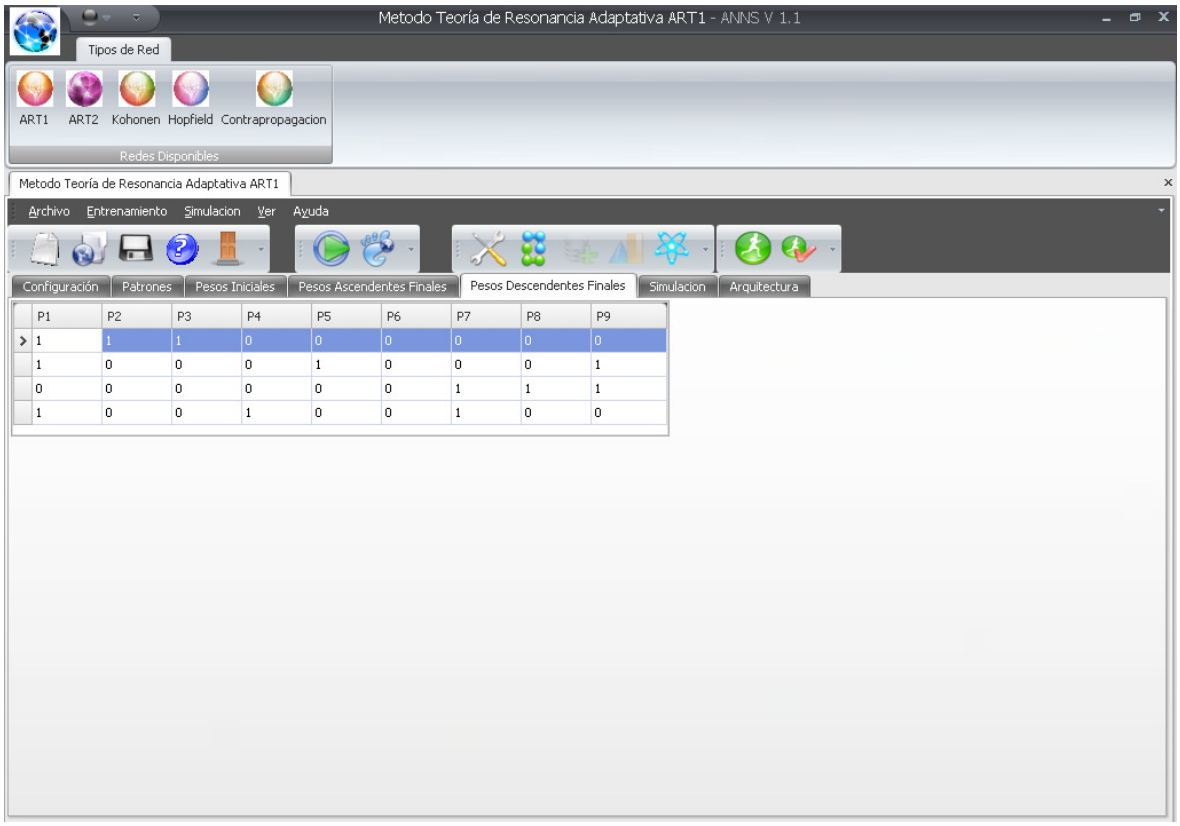


Figura 83. Ventana que muestra los pesos descendentes de la Red Art1.

Para realizar la simulación de la red creada se hará un click, sobre la pestaña llamada “Simulación”, esto habilitara la ventana de simulación, ver figura 84.

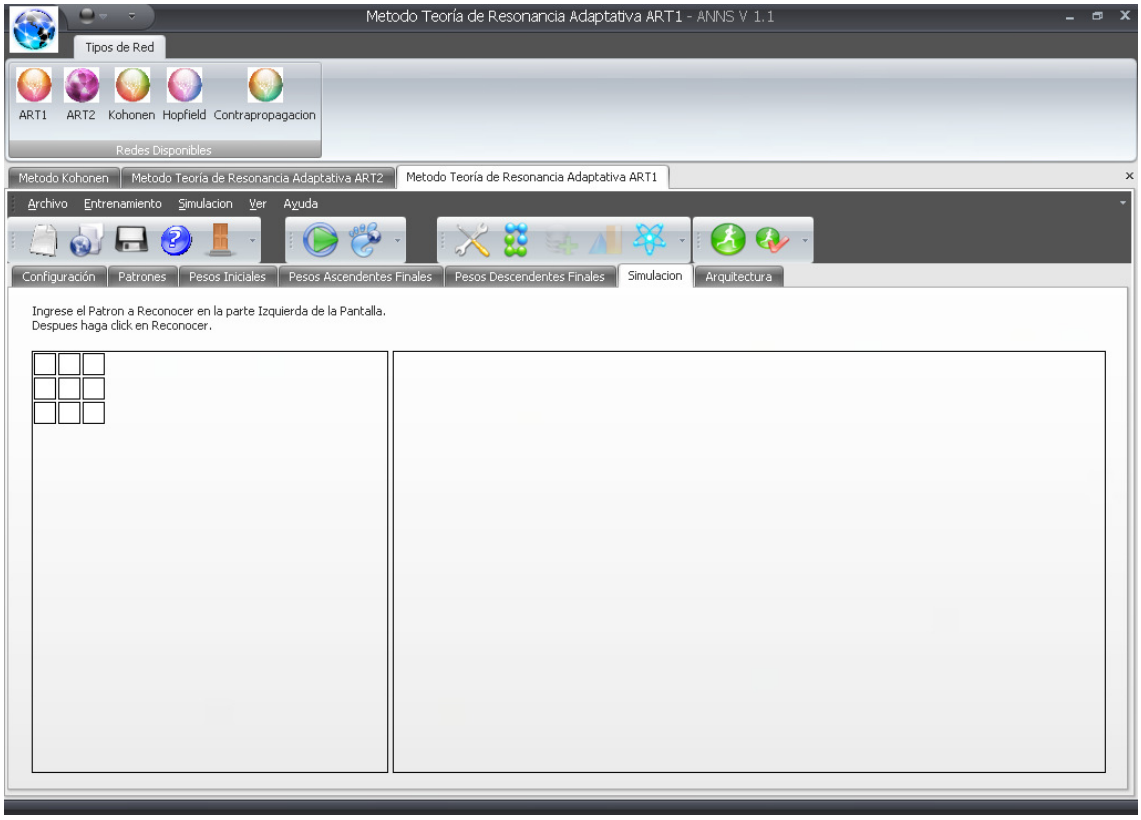


Figura 84. Ventana de simulación de la Red Art1.

En la ventana de simulación aparecerá una matriz o vector, dependiendo del tipo de patrones de entrada entrenados, en la cual, se observa que todas las casillas aparecen en blanco.

Como los valores de los patrones aprendidos son binarios, en la simulación las casillas blancas representan valores iguales a cero y las negras valores iguales a uno. Para poder cambiar el color de la casilla de blanco a negro, se deberá hacer doble click sobre la casilla donde se requiere cambiar el valor de cero a uno, ver figura 85.

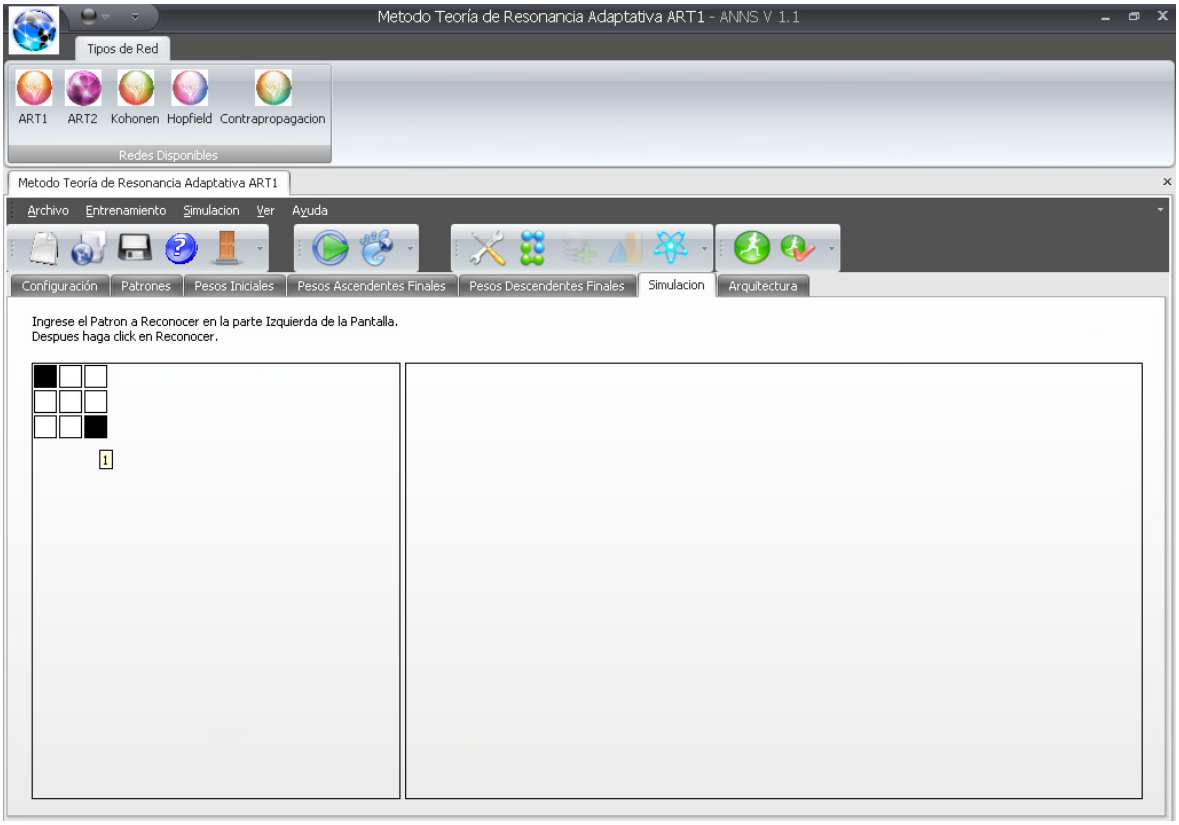


Figura 85. Ventana que muestra una matriz de prueba para realizar la simulación de la Red Art1.

Cuando se ha terminado de colocar los valores deseados en la matriz o vector de prueba se hará un click sobre el botón Reconocer de la figura 86.

Figura 86. Botón reconocer de la Red Art1.

Se desplegará a continuación una ventana mostrando si la matriz o vector de prueba ha sido reconocido o no de la figura 87.a la vez aparecerá a la derecha de la pantalla el patrón que la red debió haber aprendido durante el proceso de entrenamiento, que es con el que la red asocia el patrón de prueba ver figura 88.

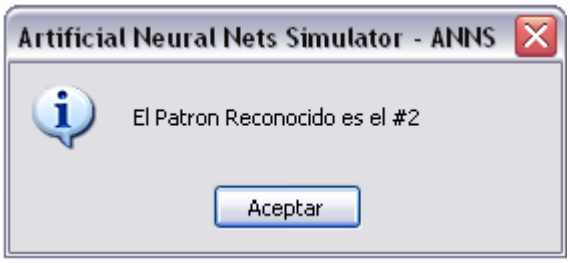


Figura 87. Ventana que indica si el patrón fue o no reconocido de la Red Art1.

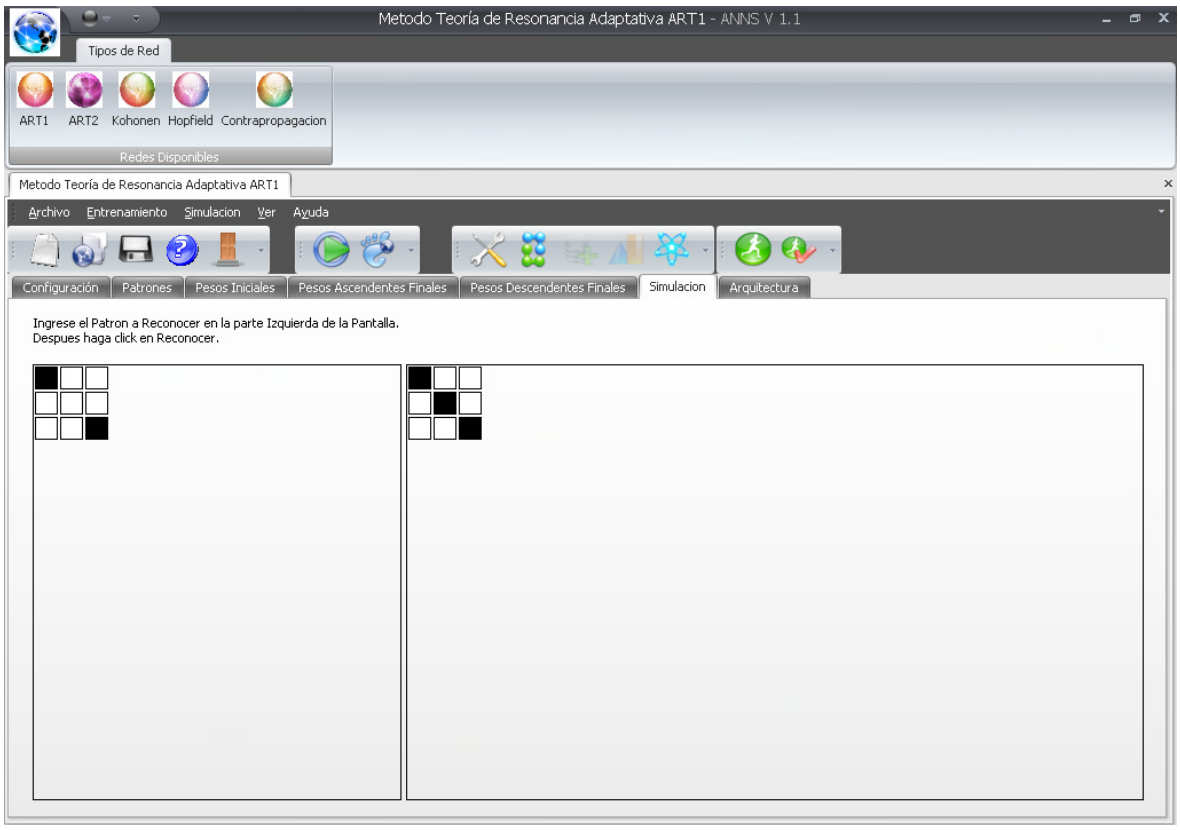


Figura 88. Ventana donde se muestra a la derecha el patrón previamente entrenado asociado con el patrón de prueba (izquierda de la pantalla) de la Red Art1.

Si el usuario quiere observar la arquitectura de la Red con la que esta trabajando, deberá hacer un click sobre la pestaña llamada “Arquitectura”. Se habilitará una ventana que mostrara las conexiones existentes entre las neuronas que contenga la red creada, ver figura 89.

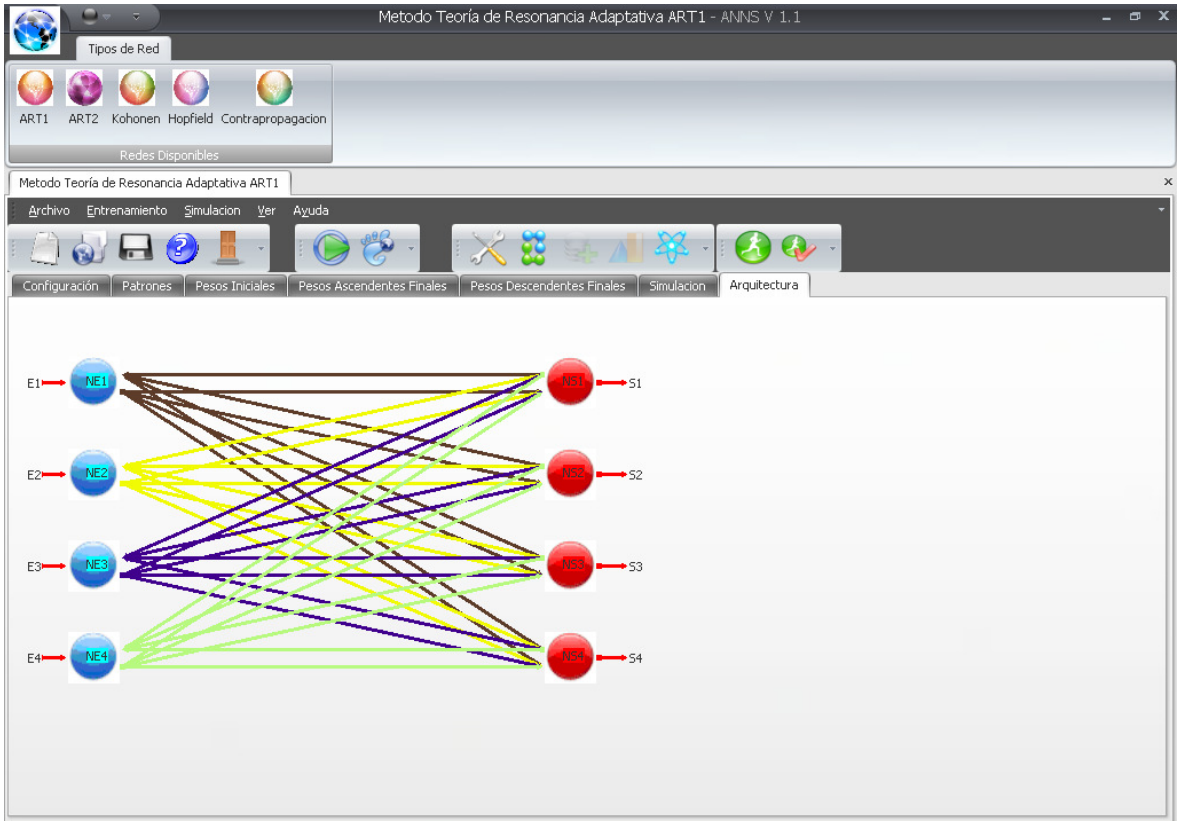


Figura 89. Ventana que muestra la arquitectura de la Red Art1.

4.3.3. Creación de una Red Art2

El proceso para crear una Red ART2, consiste en que el usuario seleccione esta topología, desde la ventana de inicio de la figura 90, haciendo un click sobre el botón llamado ART2. Una vez realizada esta acción aparecerá la ventana de configuración pero no activada para la Red, ver figura 91.

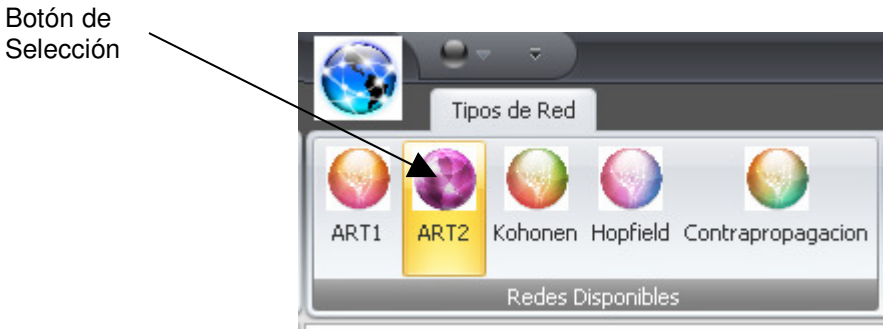


Figura 90. Botón de selección para la red ART2.

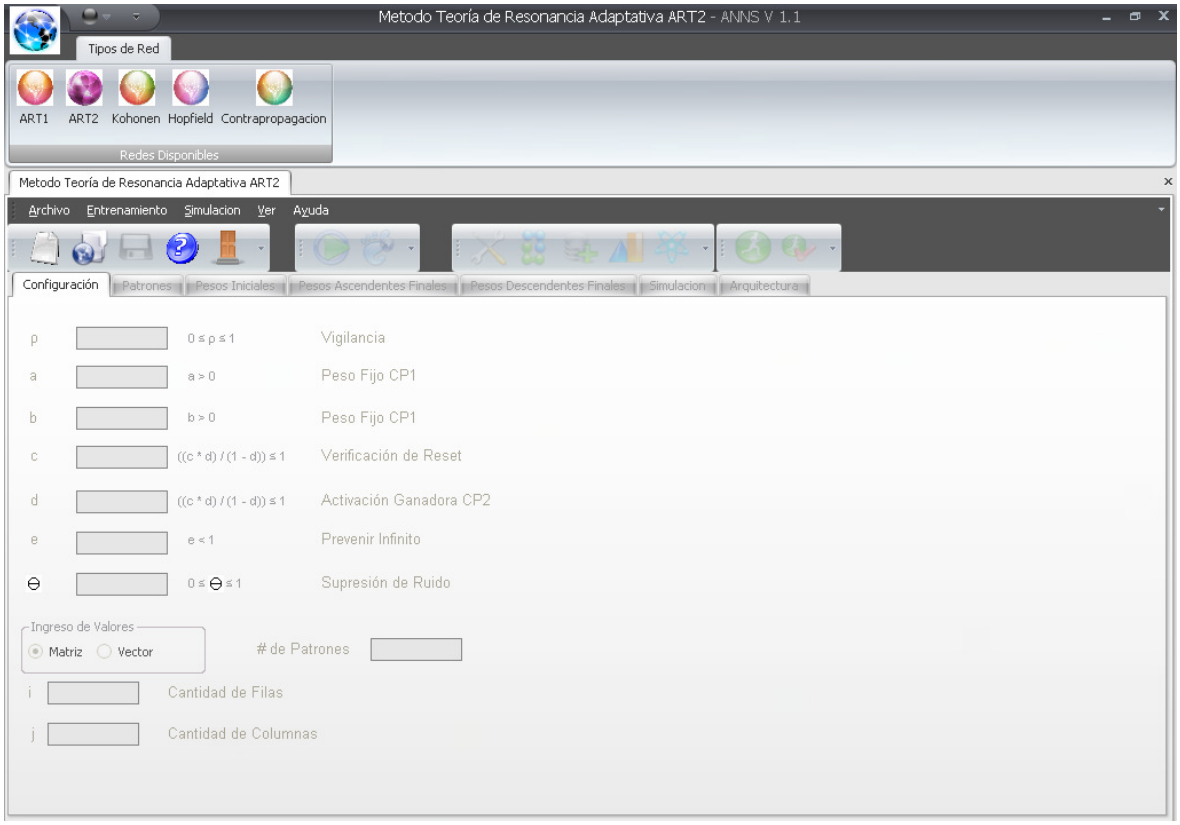


Figura 91. Ventana de configuración no activada Red ART2.

Para generar un nuevo proyecto se realizara la activación del botón “NUEVO” que se encuentra en la barra de acceso directo figura 92. Habilitando la pagina de configuración que se encuentra en la figura 93,

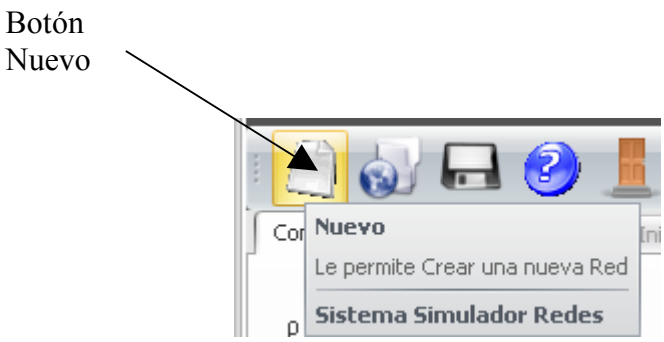


Figura 92. Botón para abrir nuevo proyecto Red ART2.

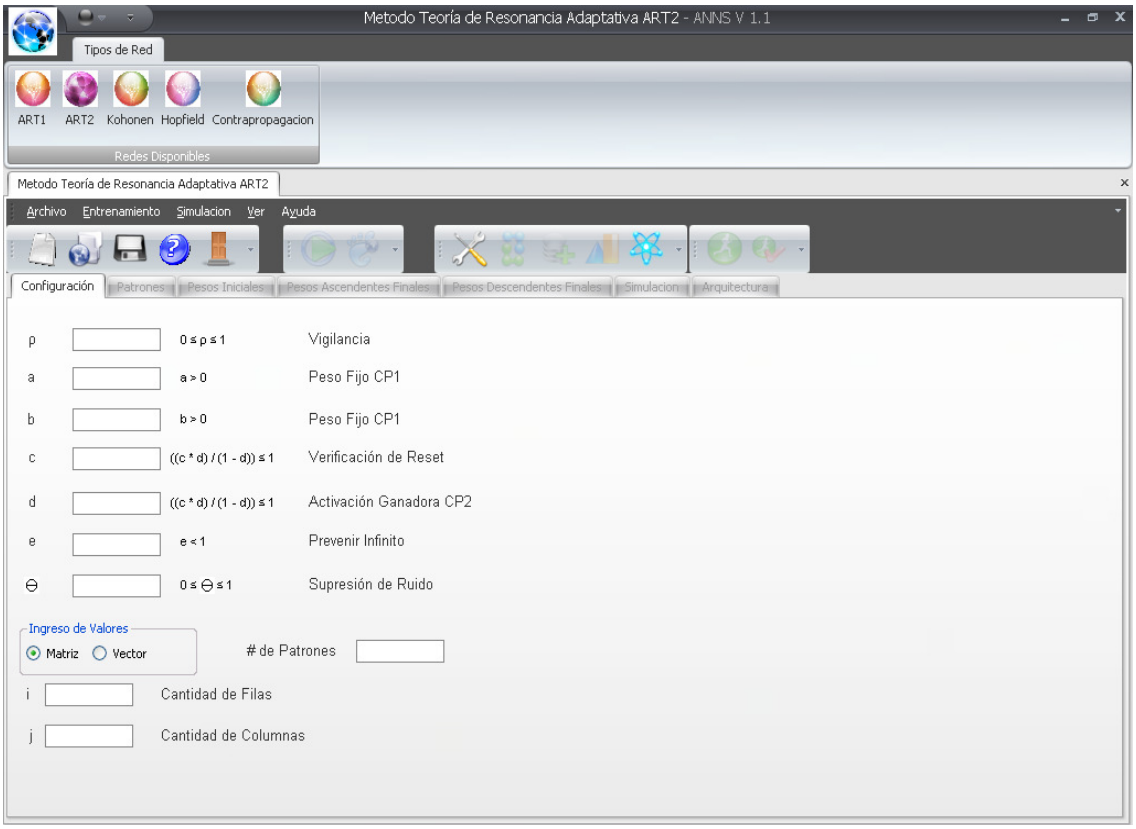


Figura 93. Ventana de Configuración activada Red ART2.

Dentro de la ventana de configuración, el usuario deberá digitar los valores correspondientes para los parámetros de inicio. Es necesario que el usuario introduzca los siguientes parámetros iniciales correspondientes a la figura 94, teniendo en cuanta las restricciones en los valores que se usaran:

ρ: Parámetro de vigilancia.
a : Peso fijo de la capa 1
b : Peso fijo de la capa 1
c : Verificación del Reset
d : Activación de la capa ganadora 2
e : Prevenir la división entre cero
Θ : Supresión de ruido

Figura 94. Ventana que muestra los parámetros iniciales de la Red ART2.

Para este tipo de red es necesario introducir la cantidad de patrones que se encuentra en la figura 95.

Figura 95. Números de patrones de la Red Art2.

Además se debe definir si los patrones de entrenamiento serán en formato vector o matriz. Si el formato seleccionado es matriz, entonces se deberá de digitar el número de filas y columnas necesarias para el patrón o patrones que se quieren entrenar, ver figura 96.

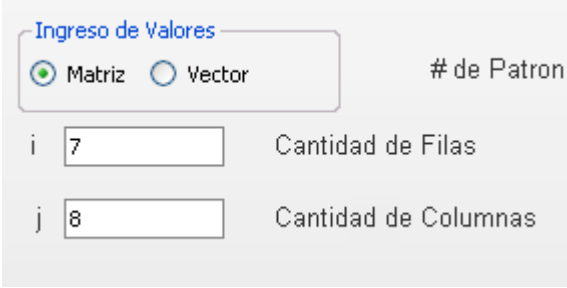


Figura 96. Ingreso de valores en forma de matriz de la Red ART2.

Una vez finalizado el proceso anterior, se selecciona la pestaña "Patrones", haciendo un click sobre ésta. En la ventana "Patrones" el usuario deberá de introducir los patrones que desea que la red aprenda, si se trata de matrices la ventana habilitada será la que se muestra en la figura 97.

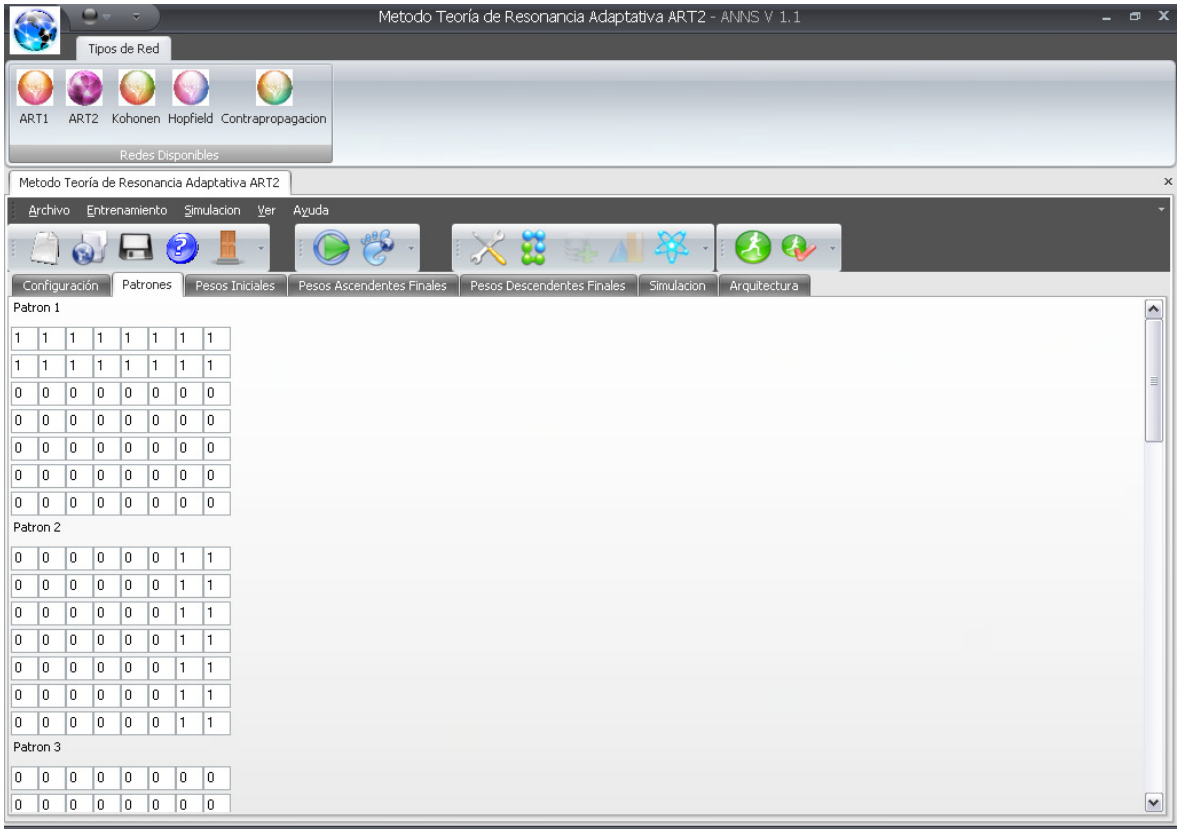


Figura 97. Ventana de patrones en forma de matriz de la Red Art2.

Si la opción seleccionada es en forma de vector, entonces se deberá de digitar el número de elementos que poseerá el vector o vectores de entrenamiento, ver figura 98.

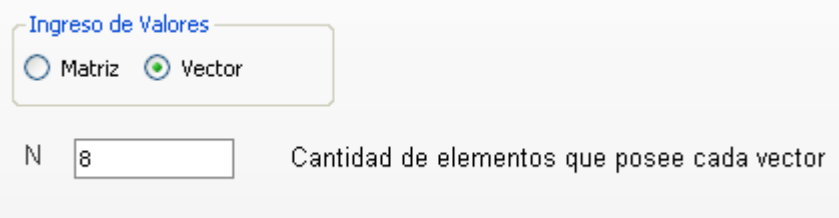


Figura 98. Ingreso de valores en forma de vector de la Red Art2.

Si se trata de Patrones de aprendizaje en forma de vector la ventana tendrá la apariencia mostrada en la figura 99.

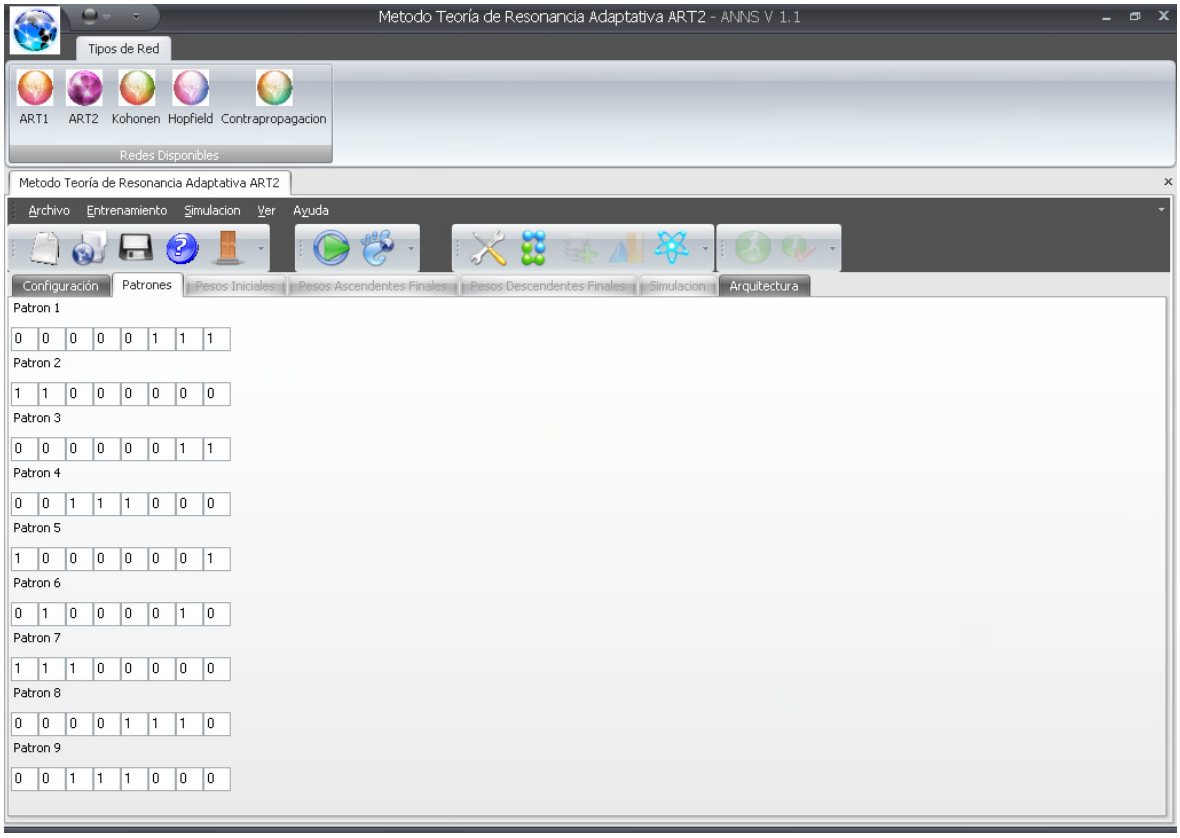


Figura 99. Ventana de patrones en forma de vector de la Red Art2.

Después de introducidos los patrones y los pesos iniciales, quedara a opción del usuario si el entrenamiento lo realiza de una vez o paso a paso, haciendo click sobre uno de los dos botones que habilitan las opciones antes mencionadas, ver figura 100.

Figura 100. Botones de selección de entrenamiento completo o paso a paso de la Red Art2.

Si el usuario quiere detener el entrenamiento deberá de hacer un click sobre el botón “Detener”, ver figura 101.

Figura 101. Botón Detener de la Red Art2.

Cuando el proceso de entrenamiento ha finalizada el software mostrara al usuario una ventana que indica que esta acción ha ocurrido, ver figura 102.

Figura 102. Ventana que muestra cuando el proceso de entrenamiento ha finalizado de la Red Art2.

Cuando el entrenamiento ha concluido se habilitan las pestañas que muestran los pesos iniciales de la figura 103, los pesos ascendentes finales de la figura 104 y los pesos descendentes finales de la figura 105.
Para poder habilitar cualquiera de las ventanas anteriores se deberá de hacer un click sobre la pestaña deseada.

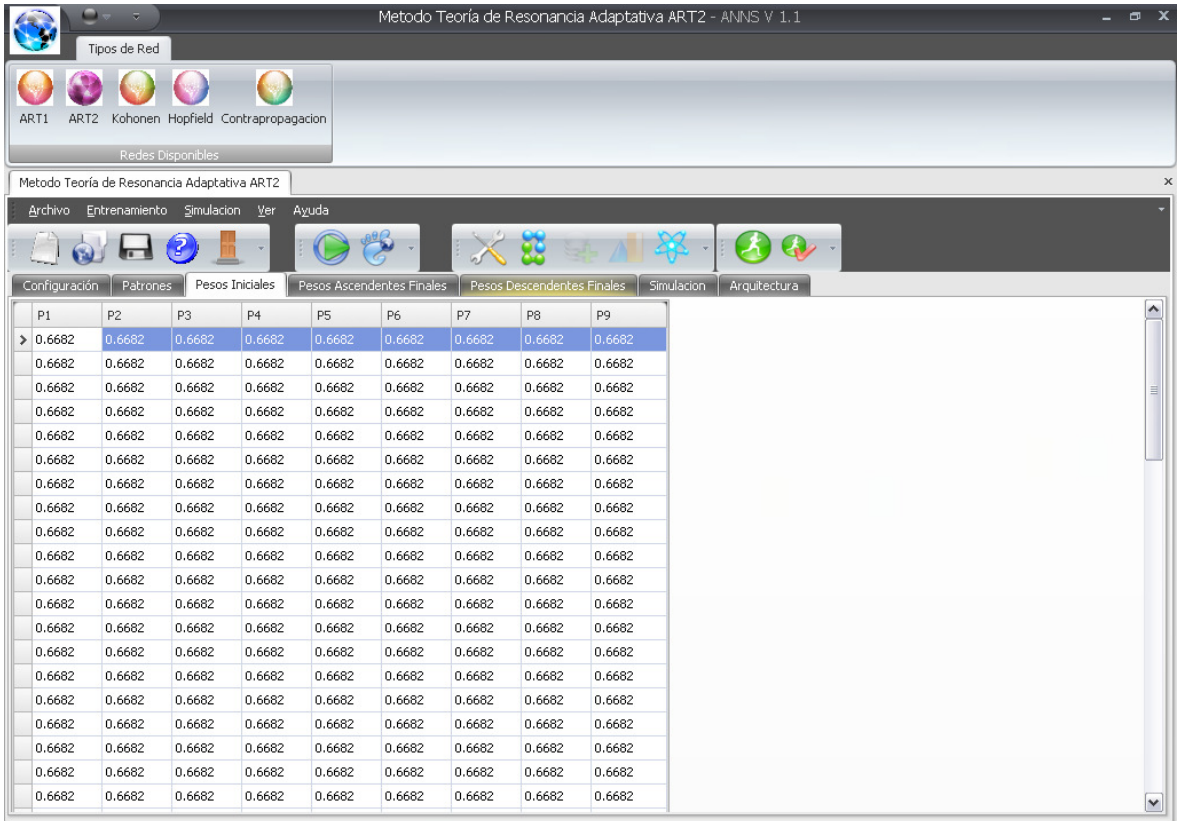


Figura 103. Ventana que muestra los pesos iniciales de la Red Art2.

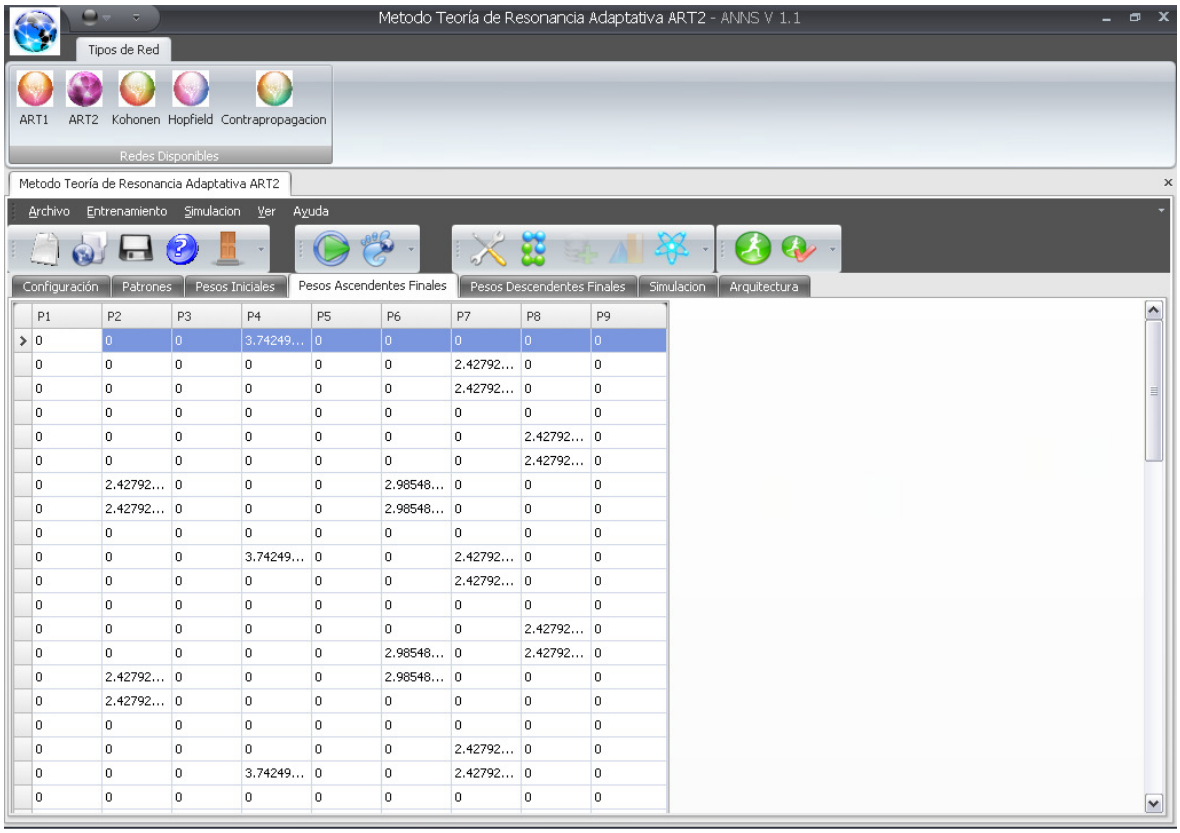


Figura 104. Ventana que muestra los pesos ascendentes finales de la Red Art2.

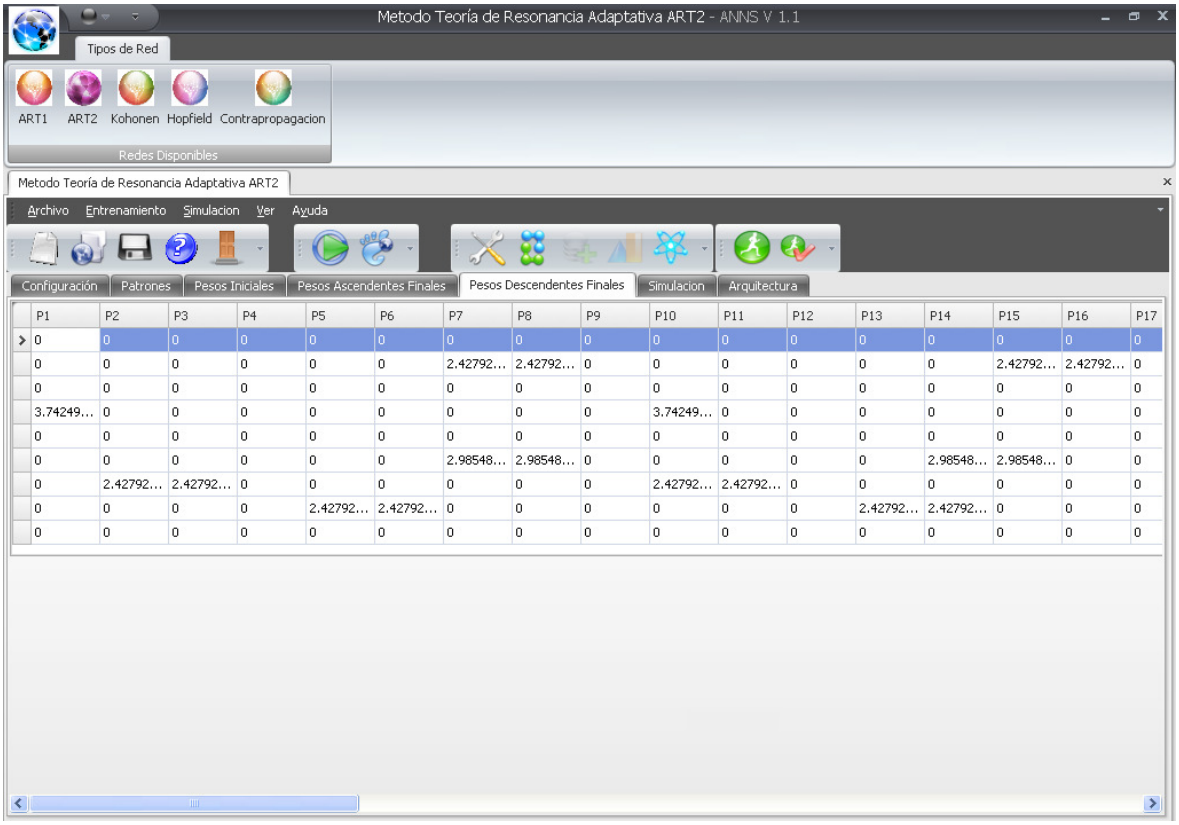


Figura 105. Ventana que muestra los pesos descendentes finales de la Red Art2.

Para realizar la simulación de la red creada se hará un click, sobre la pestaña llamada “Simulación”, esto habilitara la ventana de simulación, ver figura 106.

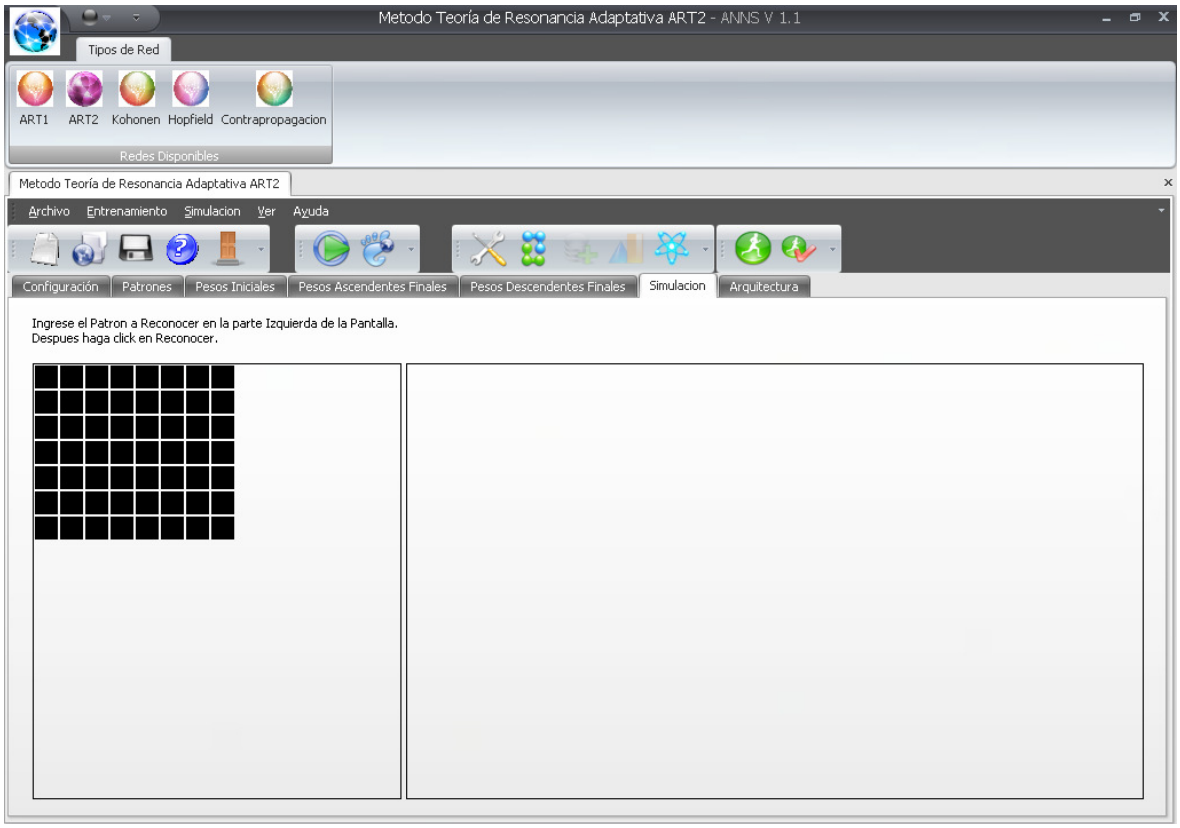


Figura 106. Ventana de Simulación de la Red Art2.

En la ventana de simulación aparecerá una matriz o vector, dependiendo del tipo de patrones de entrada entrenados, en la cual, se observa que todas las casillas aparecen en negro.

Como los valores de los patrones aprendidos pueden ser entero o decimal, en la simulación las casillas negras representan valores por defecto iguales a cero.
Para simular lo aprendido por la red, se introduce el valor correspondiente de dicho patrón en cual aparecerá una ventana que le indicara que valor va introducir y luego se le da aceptar, y así sucesivamente se podrá ser cambios como se muestra la figura 107.

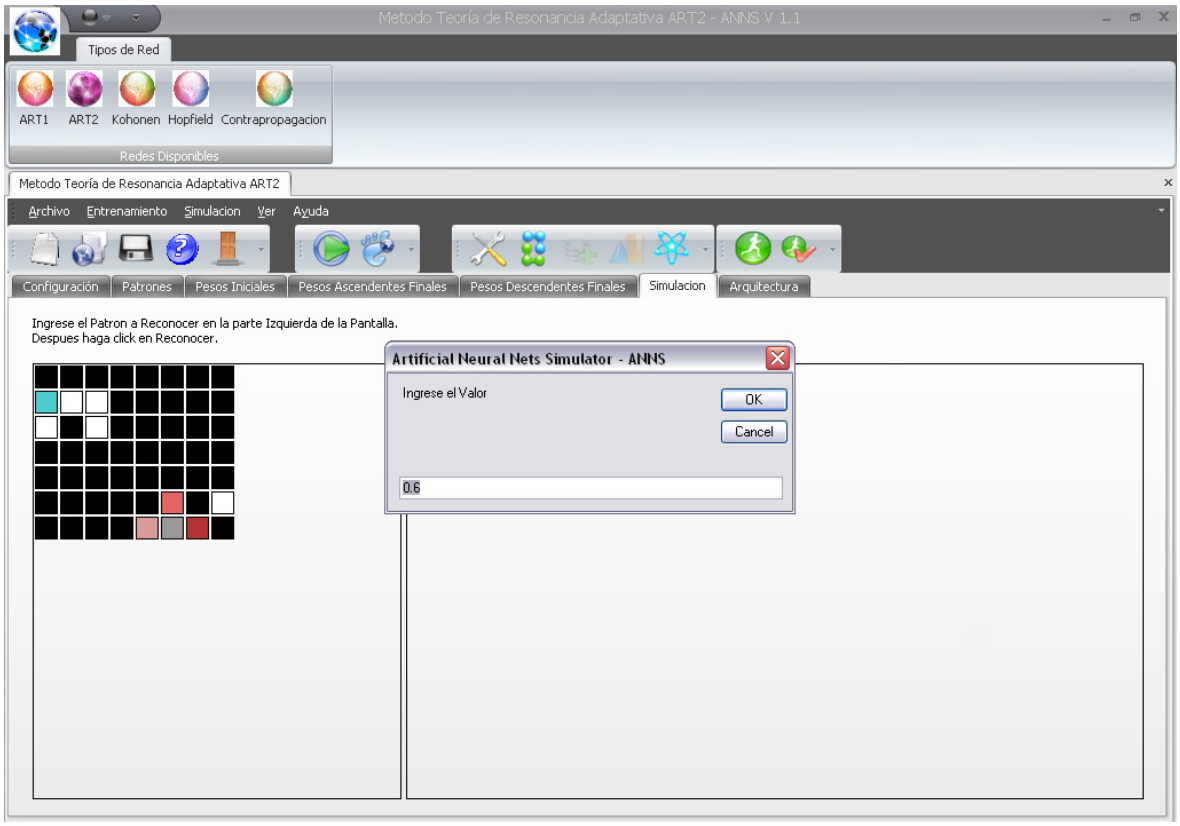


Figura 107. Ventana que muestra una matriz de prueba para realizar la simulación de la Red Art2.

Cuando se ha terminado de colocar los valores deseados en la matriz o vector de prueba se hará un click sobre el botón Reconocer de la figura 108.

Figura 108. Botón reconocer de la Red Art2.

Se desplegará a continuación una ventana mostrando si la matriz o vector de prueba ha sido reconocido o no de la figura 109.

A la vez aparecerá a la derecha de la pantalla el patrón que la red debió haber aprendido durante el proceso de entrenamiento, que es con el que la red asocia el patrón de prueba, ver figura 110.

Figura 109. Ventana que indica si el patrón fue o no reconocido de la Red Art2.

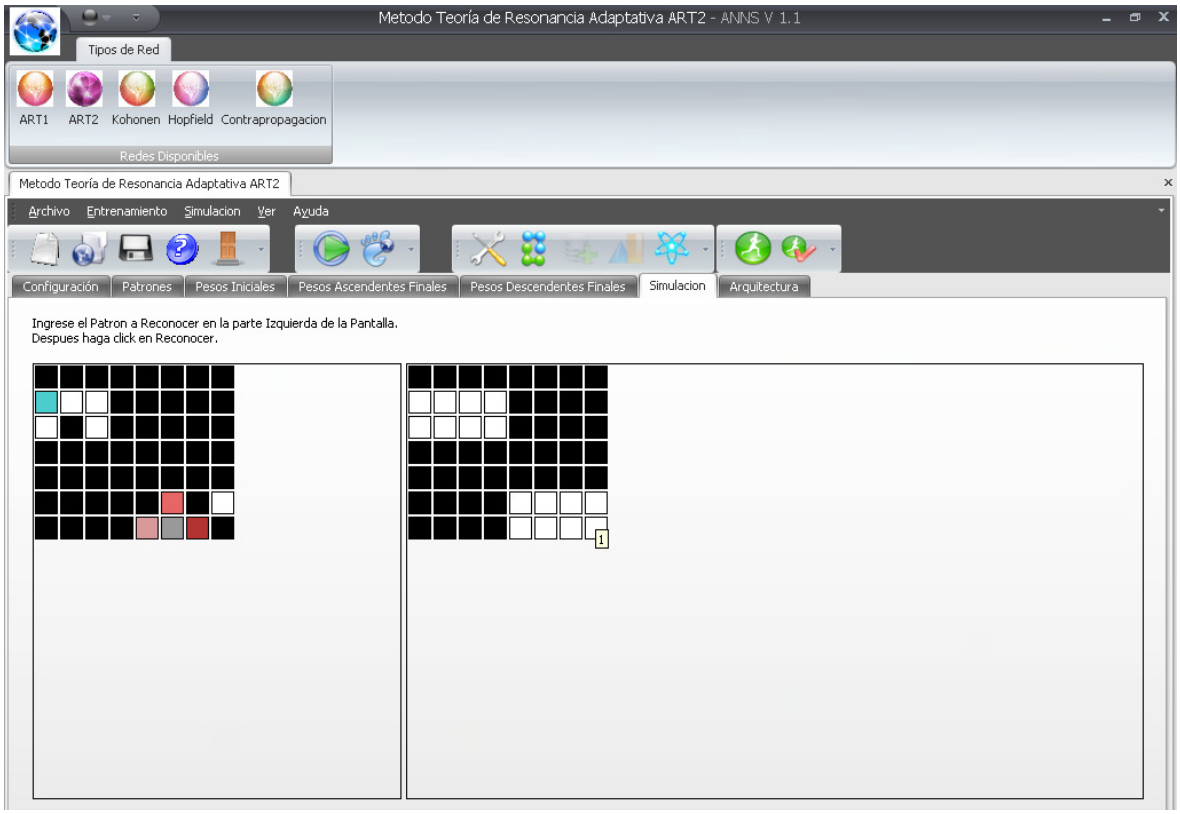


Figura 110. Ventana donde se muestra a la derecha el patrón previamente entrenado asociado con el patrón de prueba (izquierda de la pantalla) de la Red Art2.

Si el usuario quiere observar la arquitectura de la Red con la que esta trabajando, deberá hacer un click sobre la pestaña llamada “Arquitectura”. Se habilitará una ventana que mostrara las conexiones existentes entre las neuronas que contenga la red creada, ver figura 111.

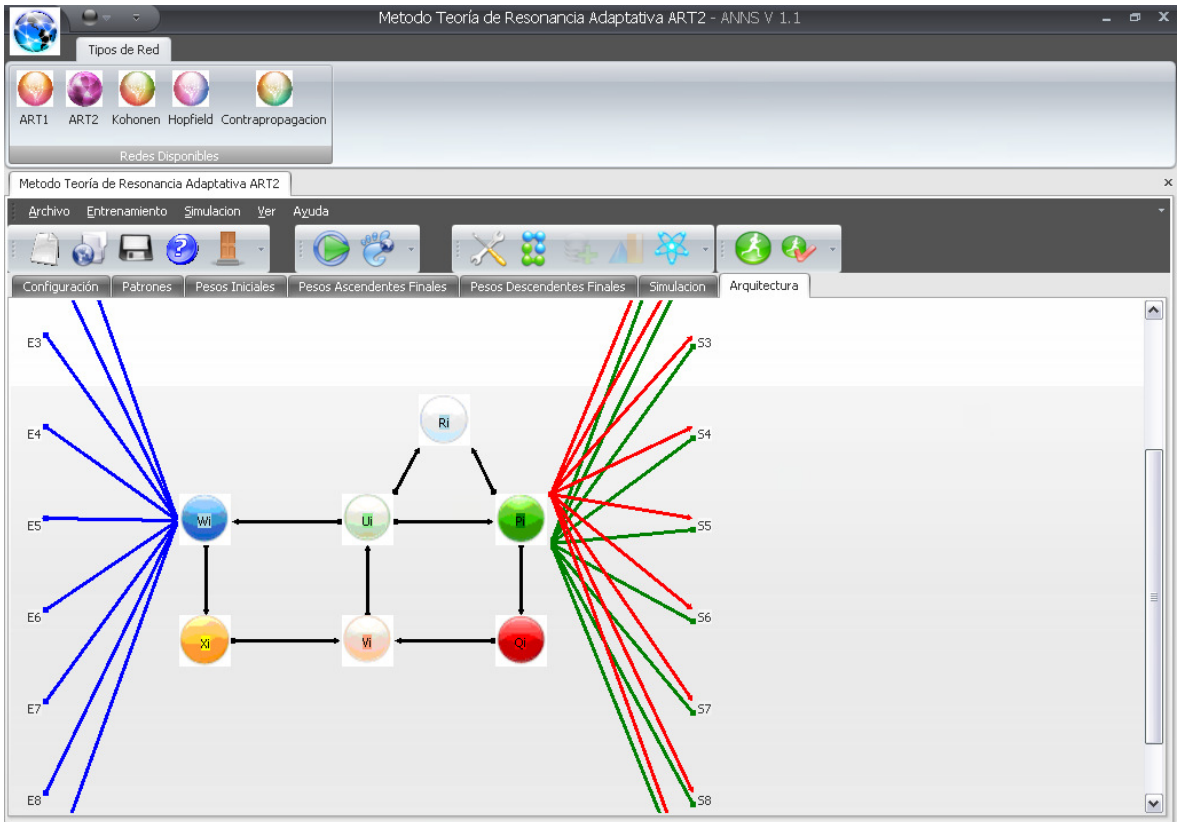


Figura 111. Ventana que muestra la arquitectura de la Red Art2.

4.3.4. Creación de una Red Kohonen

Dentro de la ventana de inicio se deberá seleccionar el botón llamado “Kohonen”, ver figura 112.

Figura 112. Botón para habilitar Red Hopfield.

Haciendo un clic sobre este botón aparecerá la ventana de configuración para este tipo de red. Para habilitar esta ventana es necesario seleccionar el botón “Nuevo “, ubicado dentro de la Barra de Accesos Directos”, ver figura 113.

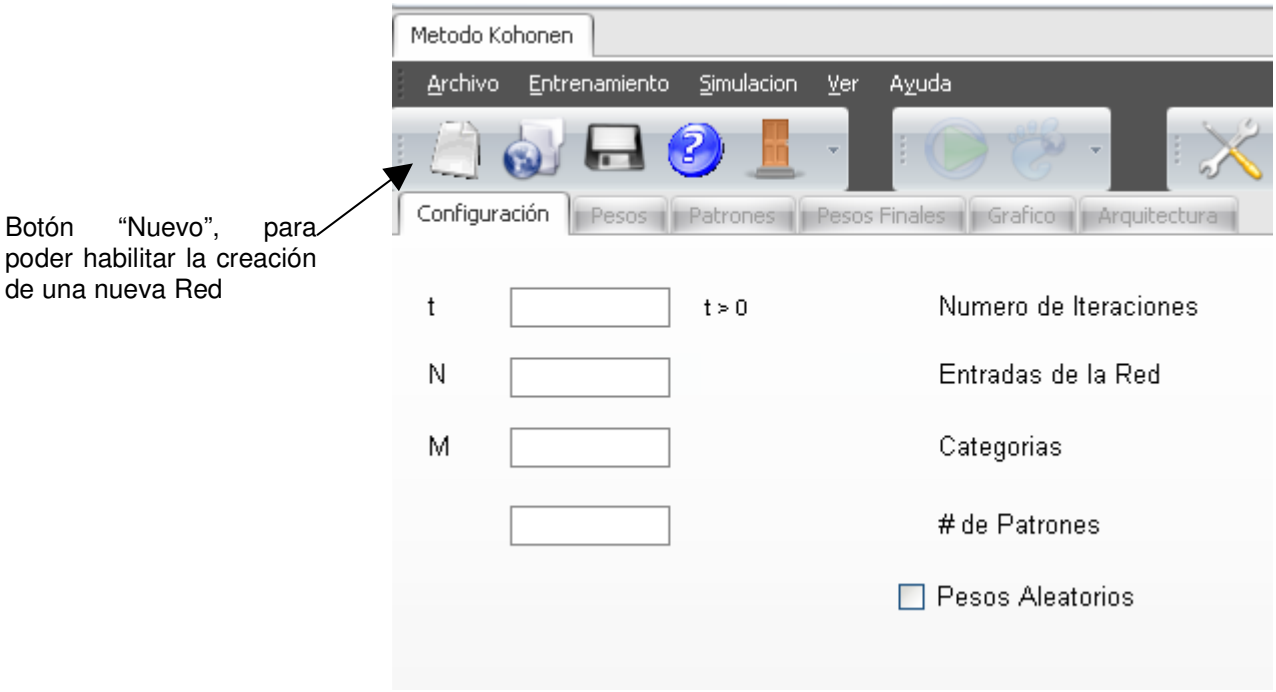


Figura 113. Figura que muestra el botón "Nuevo" dentro de la ventana de configuración para la red Kohonen.

Cuando esta acción es realizada, la ventana permite digitar los valores de los parámetros iniciales, que se presenta en la figura 114, los parámetros que se deben introducir son:

- t = Numero de iteraciones, se sugieren valores mayores de 10,000.
- N = Numero de Neuronas de entrada de la Red, (cantidad de elementos de cada vector de entrenamiento)
- M = Numero de Categorías (número de neuronas de la capa competitiva)
- # Patrones = Cantidad de vectores que se quieren entrenar.

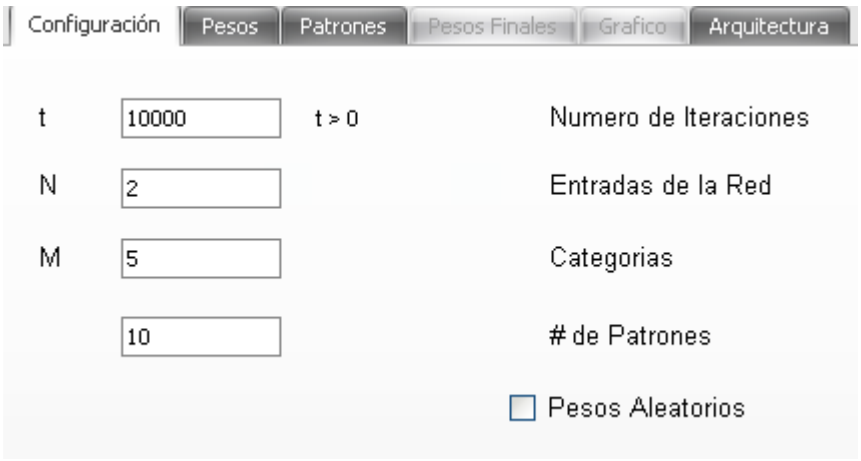


Figura 114. Parámetros de inicio para crear una Red Kohonen.

En la Ventana de Configuración, el usuario deberá elegir si los pesos que utilizará serán los generados por el mismo software (Pesos Aleatorios) o serán valores predefinidos que el usuario deberá de introducir, para entrenar la Red.

Si se utilizan los pesos aleatorios; entonces, en la ventana de Configuración se deberá de hacer un click sobre la opción llamada "Pesos Aleatorios". Se observara que la pestaña llamada pesos quede deshabilitada, ver figura 115.

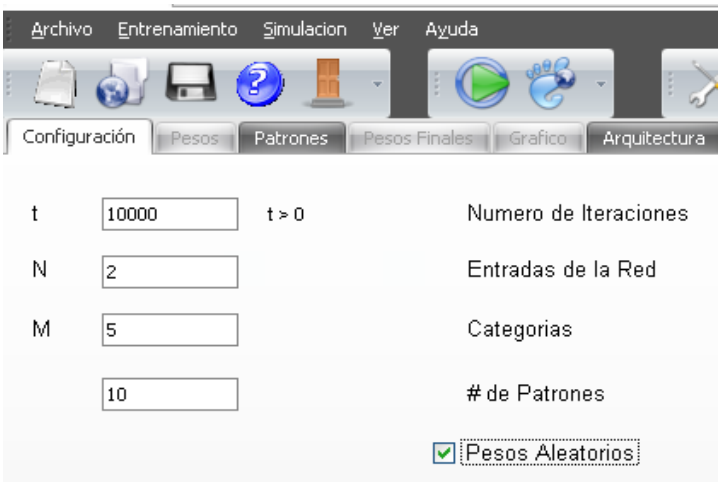


Figura 115. Ventana de configuración que muestra la opción llamada "Pesos Aleatorios" de la Red Kohonen.

Por el contrario, si no se utiliza la opción llamada "Pesos Aleatorios", la pestaña llamada "Pesos", se habilitara, y haciendo un click sobre ella, aparecerá la ventana en la que se introducirán los valores de los pesos predefinidos que el usuario ha seleccionado, ver figura 116.

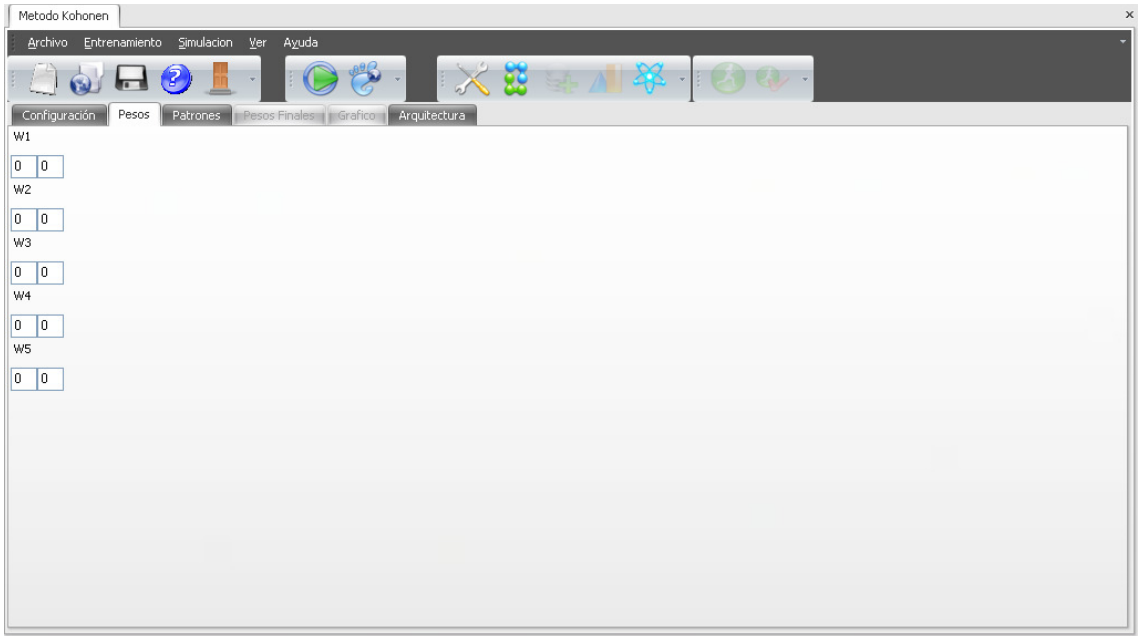


Figura 116. Ventana donde se introducen los pesos iniciales de forma manual de la Red Kohonen.

Una vez definidos los Pesos Iniciales, se procede a introducir los Vectores que la red deberá de entrenar. Para ello se hará un click sobre la pestaña llamada "patrones".

Se desplegará una ventana donde deberán de introducirse los valores de los elementos de cada vector. Estos pueden contener valores mayores a cero, sean estos, en formato decimal o entero, pero no números menores de cero, ver figura 117.

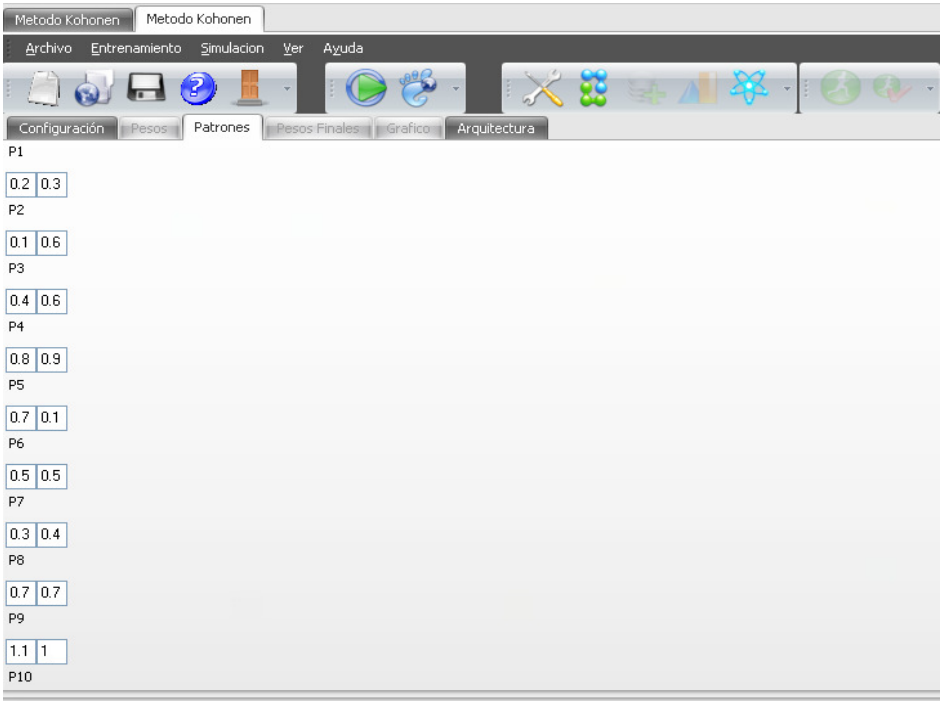


Figura 117. Ventana de patrones de la Red Kohonen.

Una vez se han definido los pesos iniciales sean estos aleatorios o digitados por el usuario. Se procederá ha realizar el entrenamiento. Para ello se deberá de hacer un click sobre el botón entrenar de una vez o paso a paso que se presenta en la figura 118. La selección del tipo de entrenamiento dependerá del usuario.

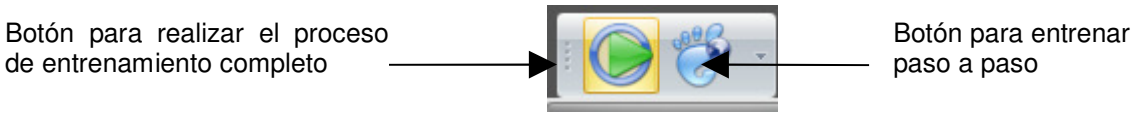


Figura 118. Botones para entrenar y paso a paso de la Red Kohonen.

Si el usuario decide detener el entrenamiento antes de que este finalice, deberá de hacer un clic sobre el botón llamado “Detener”, ver figura 119.

.

Figura 119. Botón detener proceso de entrenamiento.

Cuando el entrenamiento finaliza, el ANNS V1.1, desplegará un mensaje que indica que esto ha ocurrido, ver figura 120.

Figura 120. Mensaje que indica cuando el entrenamiento ha concluido de la Red Kohonen.

Cuando el entrenamiento ha concluido se activaran las pestañas llamadas “Pesos Finales “ y “Gráficos”. Así como los botones que activan estas funciones desde la barra de Accesos directos. ver figura 121.

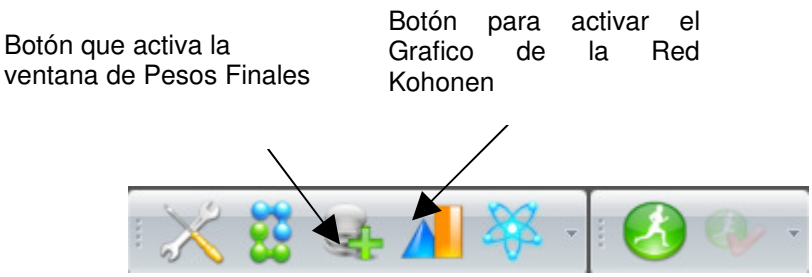
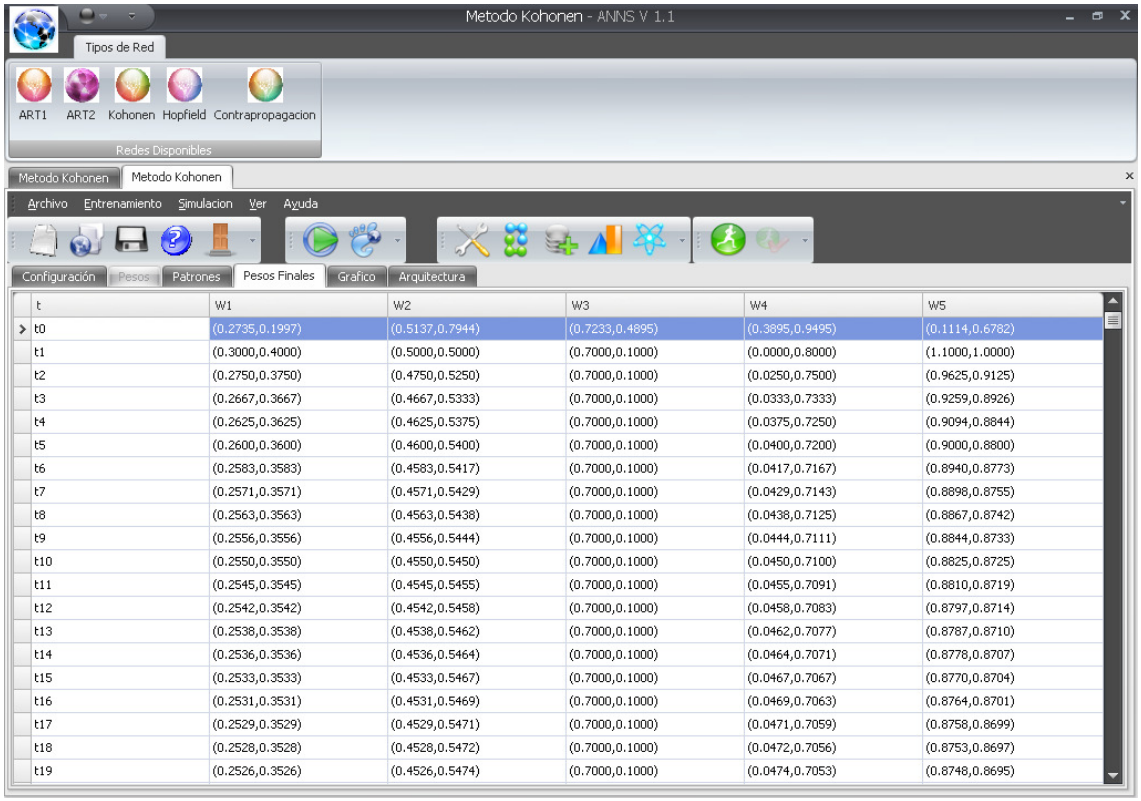


Figura 121. Figura que muestra los botones de pesos finales y gráfico de la Red Kohonen.

Para poder ver los pesos finales, el usuario deberá hacer un click sobre la pestaña llamada “Pesos Finales”. Al realizar lo anterior se observara una ventana donde se muestran los pesos finales obtenidos para cada categoría. Se debe notar que en un costado aparece un Scroll, por medio del cual, se podrá realizar un desplazamiento sobre la ventana para poder visualizar la evolución de los pesos alcanzados, ver figura 122.



t	W1	W2	W3	W4	W5
t0	(0.2735,0.1997)	(0.5137,0.7944)	(0.7233,0.4895)	(0.3895,0.9495)	(0.1114,0.6782)
t1	(0.3000,0.4000)	(0.5000,0.5000)	(0.7000,0.1000)	(0.0000,0.8000)	(1.1000,1.0000)
t2	(0.2750,0.3750)	(0.4750,0.5250)	(0.7000,0.1000)	(0.0250,0.7500)	(0.9625,0.9125)
t3	(0.2667,0.3667)	(0.4667,0.5333)	(0.7000,0.1000)	(0.0333,0.7333)	(0.9259,0.8926)
t4	(0.2625,0.3625)	(0.4625,0.5375)	(0.7000,0.1000)	(0.0375,0.7250)	(0.9094,0.8844)
t5	(0.2600,0.3600)	(0.4600,0.5400)	(0.7000,0.1000)	(0.0400,0.7200)	(0.9000,0.8800)
t6	(0.2583,0.3583)	(0.4583,0.5417)	(0.7000,0.1000)	(0.0417,0.7167)	(0.8940,0.8773)
t7	(0.2571,0.3571)	(0.4571,0.5429)	(0.7000,0.1000)	(0.0429,0.7143)	(0.8898,0.8755)
t8	(0.2563,0.3563)	(0.4563,0.5438)	(0.7000,0.1000)	(0.0438,0.7125)	(0.8867,0.8742)
t9	(0.2556,0.3556)	(0.4556,0.5444)	(0.7000,0.1000)	(0.0444,0.7111)	(0.8844,0.8733)
t10	(0.2550,0.3550)	(0.4550,0.5450)	(0.7000,0.1000)	(0.0450,0.7100)	(0.8825,0.8725)
t11	(0.2545,0.3545)	(0.4545,0.5455)	(0.7000,0.1000)	(0.0455,0.7091)	(0.8810,0.8719)
t12	(0.2542,0.3542)	(0.4542,0.5458)	(0.7000,0.1000)	(0.0458,0.7083)	(0.8797,0.8714)
t13	(0.2538,0.3538)	(0.4538,0.5462)	(0.7000,0.1000)	(0.0462,0.7077)	(0.8787,0.8710)
t14	(0.2536,0.3536)	(0.4536,0.5464)	(0.7000,0.1000)	(0.0464,0.7071)	(0.8778,0.8707)
t15	(0.2533,0.3533)	(0.4533,0.5467)	(0.7000,0.1000)	(0.0467,0.7067)	(0.8770,0.8704)
t16	(0.2531,0.3531)	(0.4531,0.5469)	(0.7000,0.1000)	(0.0469,0.7063)	(0.8764,0.8701)
t17	(0.2529,0.3529)	(0.4529,0.5471)	(0.7000,0.1000)	(0.0471,0.7059)	(0.8758,0.8699)
t18	(0.2528,0.3528)	(0.4528,0.5472)	(0.7000,0.1000)	(0.0472,0.7056)	(0.8753,0.8697)
t19	(0.2526,0.3526)	(0.4526,0.5474)	(0.7000,0.1000)	(0.0474,0.7053)	(0.8748,0.8695)

Figura 122. Ventana que muestra los “Pesos Finales” de la Red Kohonen.

El simulador ANNS V1.1, puede mostrar la gráfica de los vectores de pesos finales, sobre el plano X – Y, para cada una de las categorías creadas por la red actual. Cuando el entrenamiento ha finalizado se activará la pestaña llamada “Grafico”, haciendo un click sobre esta, aparecerá la ventana en la que se podrá visualizar la gráfica de los pesos finales de la red actual. Es necesario que el usuario tenga en cuenta una restricción muy importante: La aplicación “Gráficos” solamente estará activada en el Simulador ANNS V1.12, cuando el entrenamiento se ha realizado para vectores de dos elementos. Si los vectores poseen más de dos elementos la opción de gráfico no estará disponible.

Si el usuario desea visualizar el grafico producto del entrenamiento de la Red actual, deberá hacer un clic sobre la pestaña grafico y se activará la ventana que se muestra en la figura 123.

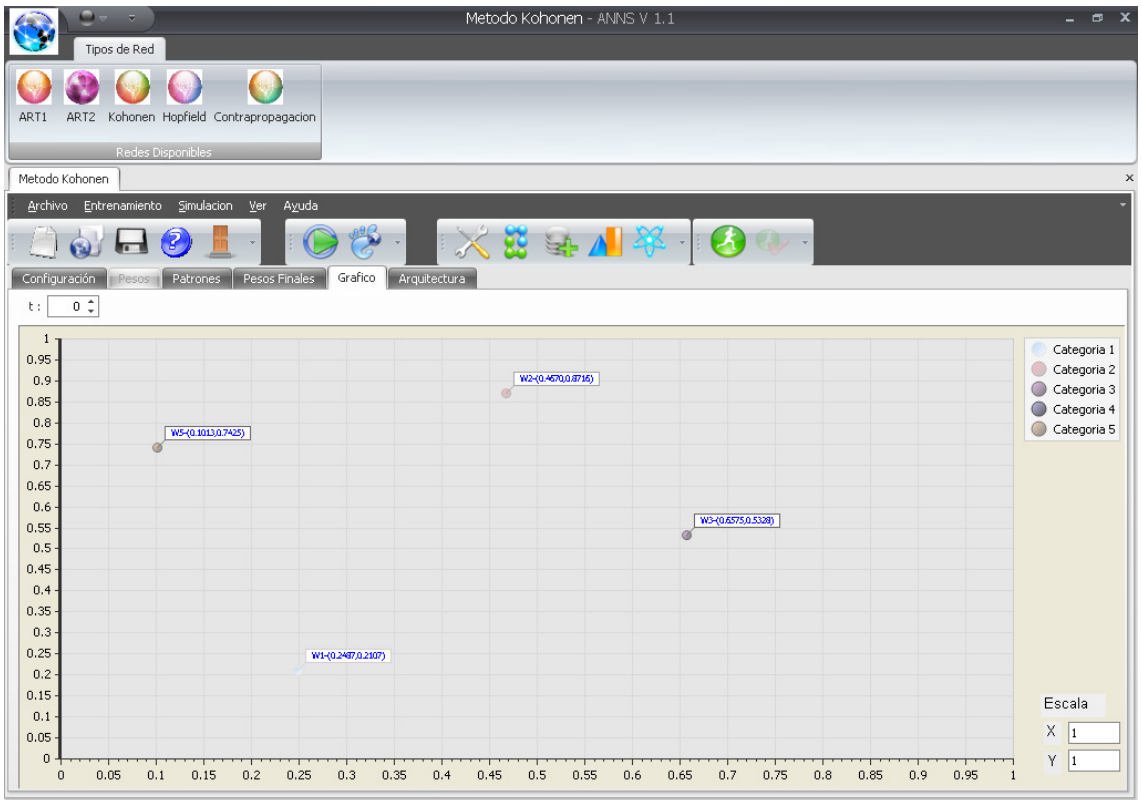


Figura 123. Ventana que muestra la grafica de los pesos finales de la Red Kohonen.

Dentro de esta ventana aparece en la esquina superior izquierda, un control por medio del cual se puede observar la evolución de los pesos de forma grafica, con solo introducir un valor de iteración para la cual se quiere visualizar la grafica de los pesos para esa época especifica, ver figura 124.

Figura 124. Muestra control para variar el número de iteración de la Red Kohonen.

A la derecha de la ventana, se observa un cuadro de dialogo donde se muestra por medio de diferentes colores cada una de las categorías creadas por la red actual, ver figura 125.

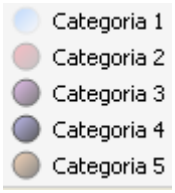


Figura 125. Cuadro que muestra por colores las diferentes categorías de la Red Kohonen.

Para poder realizar la simulación se deberá hacer un click sobre el botón llamado "Simular", que aparece en la barra de accesos directos, ver figura 126.

Figura 126. Botón "Simular" de la Red Kohonen.

Entonces aparecerá una ventana en la cual se deberá de digitar un vector de prueba, el cual deberá ser categorizado por la red actual, ver figura 127.

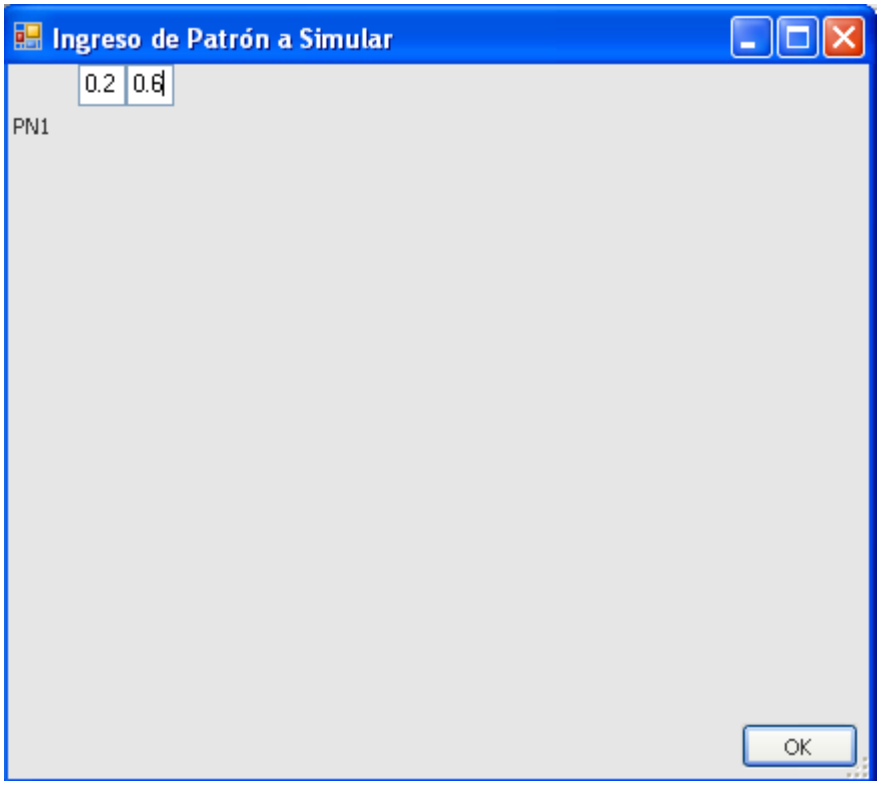


Figura 127. Ventana que muestra la ventana del patrón de prueba para realizar la simulación de la red Kohonen.

Para que la simulación se realice, se deberá hacer un click sobre el botón “OK” de la ventana mostrada en la figura anterior. Entonces aparecerá otra ventana que indica a que categoría pertenece el vector de prueba, ver figura 128.

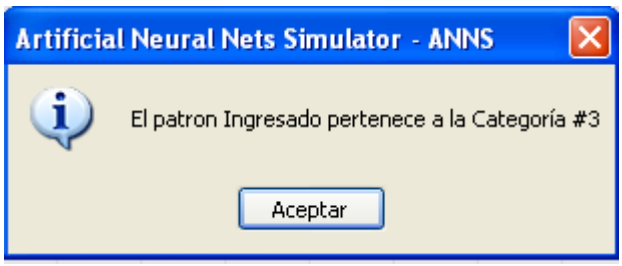


Figura 128. Ventana que muestra a que categoría pertenece el vector de prueba de la Red Kohonen.

Al hacer un click sobre el botón “Aceptar de la ventana mostrada en la figura 128, en la ventana llamada “Grafico” se observará el vector de prueba en color Rojo, mostrando por medio de la distancia, la categoría al cual este pertenece, ver figura 129.

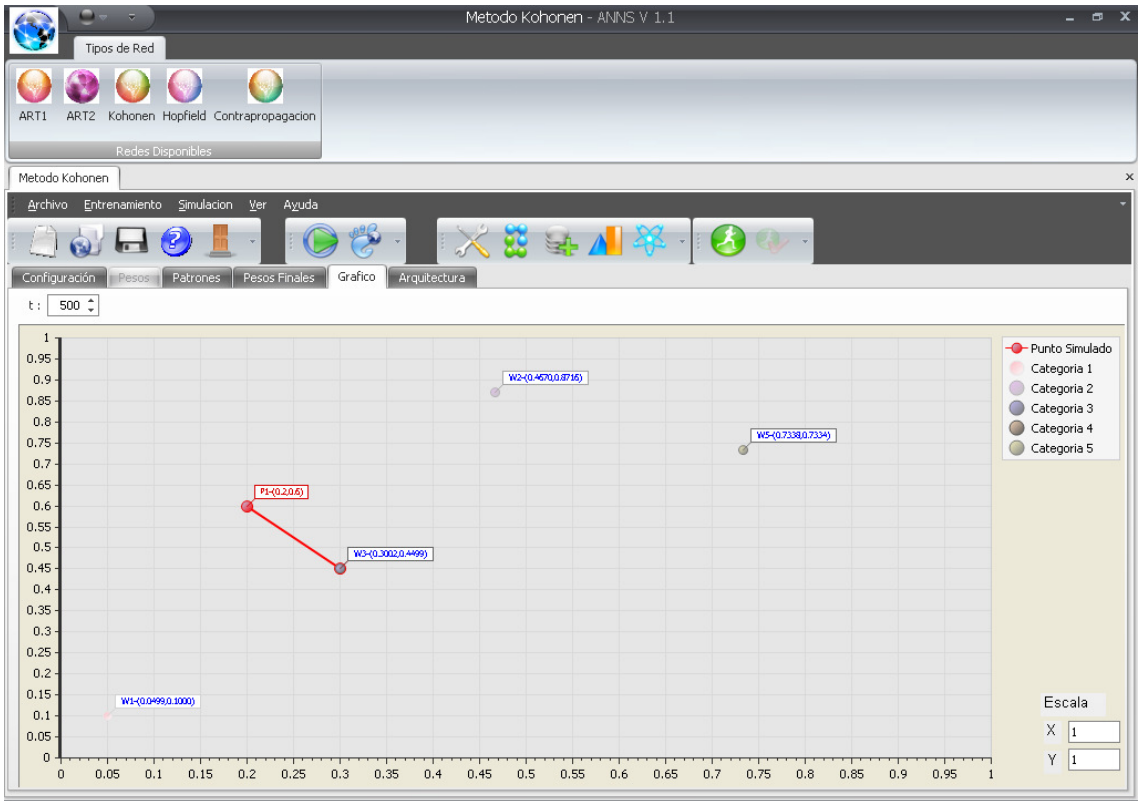


Figura 129. Ventana que muestra la categoría a la que pertenece el vector de prueba de la Red Kohonen.

Si el usuario desea ver la arquitectura de la red deberá hacer un click sobre la pestaña llamada “Arquitectura”, ver figura 130.

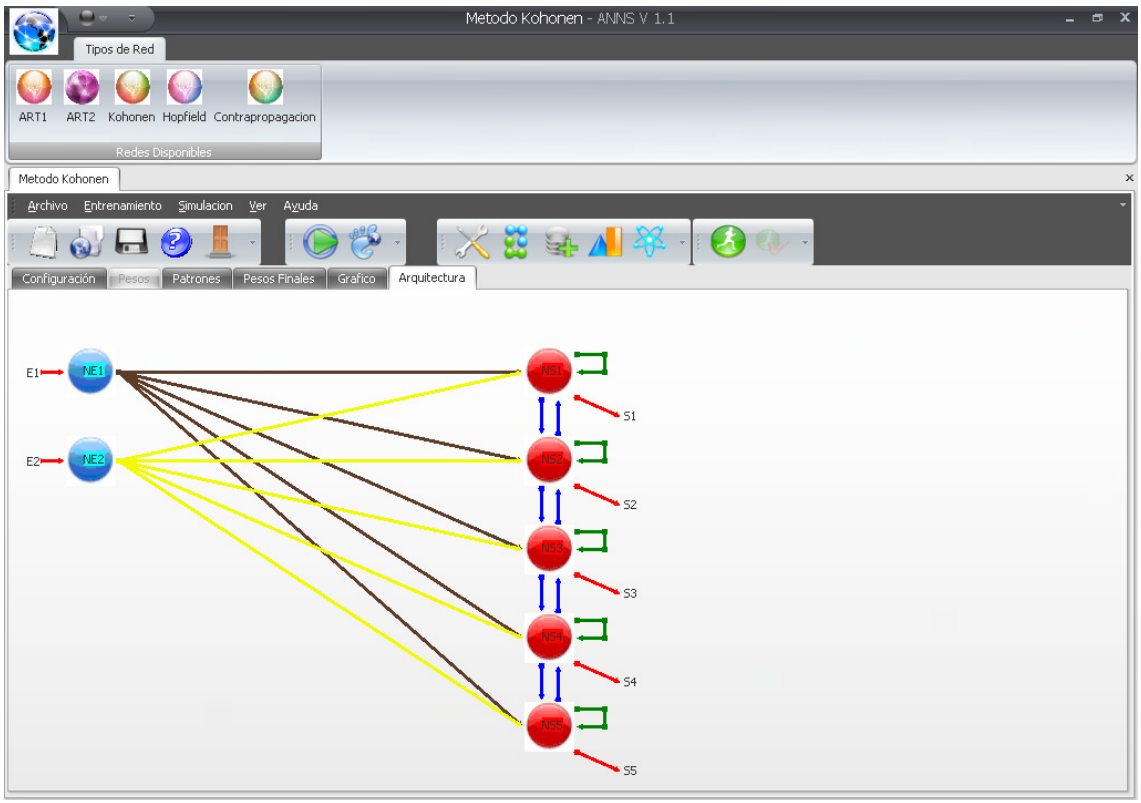


Figura 130. Ventana que muestra la arquitectura de la Red Kohonen.

4.3.5 Creación de una Red Hopfield

Para crear una Red tipo Hopfield, es necesario seleccionar ésta topología desde el menú de tipo de Red, haciendo un click sobre el botón Hopfield, ver figuras 131.

Figura 131. Botón para Red Hopfield.

Cuando se realiza esta acción se despliega la ventana de configuración de parámetros iniciales para la Red Hopfield que se muestra en la figura 132. Para que la ventana permita digitar en ella los parámetros iniciales es necesario que el usuario haga un click sobre el botón “Nuevo”.

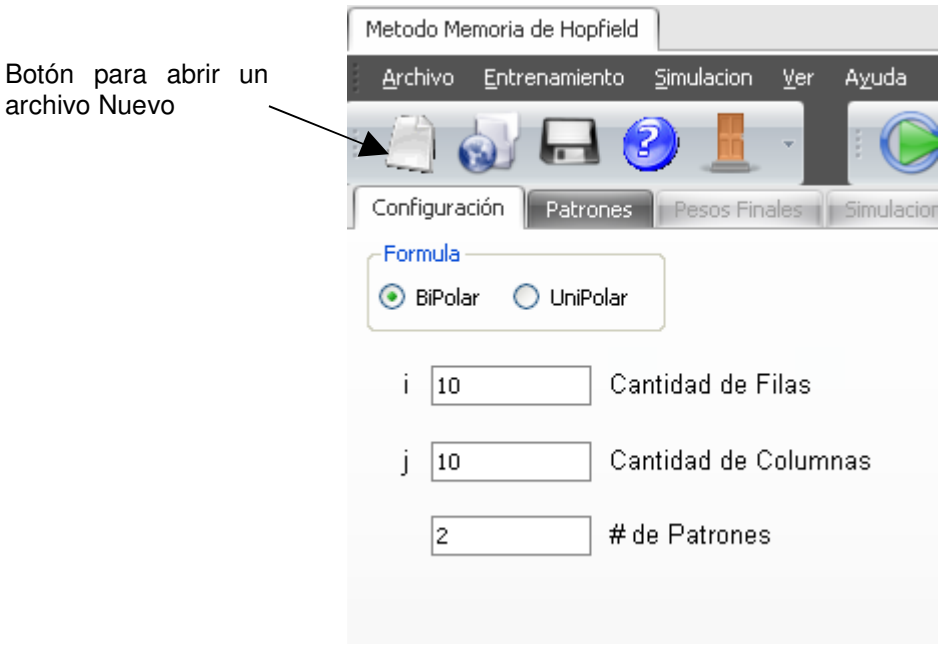


Figura 132. Botón para abrir un archivo nuevo de la Red Hopfield.

Es necesario introducir los parámetros iniciales tomando en cuenta que se deberá de seleccionar el tipo de valores para los que la Red deberá de ser entrenada. En la pantalla de configuración se tiene la opción “Valores” que permite introducir los patrones de entrenamiento en forma bipolar (-1,1) o unipolar (0,1). Además es necesario definir dentro de esta ventana el número de columnas y filas de las matrices de entrenamiento, así como la cantidad de éstas de la figura 131.

Una vez realizado el proceso anterior se deberá buscar la pestaña llamada “Patrones”. Se habilitara la ventana donde se digitaran los valores dentro de las matrices de entrenamiento de la figura 133. Cuando se han completado todas las matrices que la red deberá de aprender durante el proceso de entrenamiento, se hará click sobre el botón de entrenamiento.

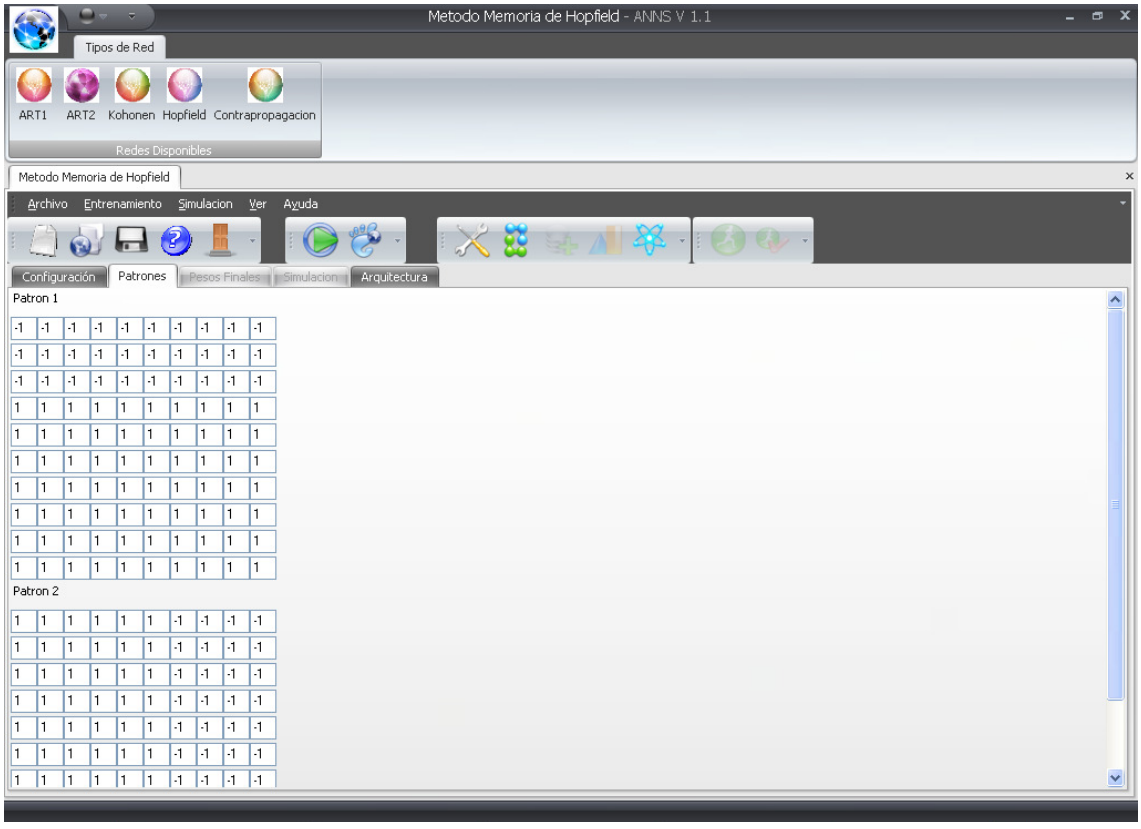


Figura 133. Ventana de patrones de la Red Hopfield.

El usuario podrá definir si el entrenamiento se realizará de una vez o paso a paso, con solo seleccionar uno de los dos botones que se muestran en la figura 134

Figura 134 Botones entrenar de una vez o paso a paso de la Red Hopfield.

Si se desea detener el entrenamiento entonces se deberá hacer un click sobre el botón de paro que se activa cuando el entrenamiento esta en proceso, ver figura 135

Figura 135. Botón Detener de la Red Hopfield.

Se activará una ventana que indicará en que momento el proceso de entrenamiento ha finalizado, desplegando un mensaje que así lo indique, ver figura 136

Figura 136. Ventana que indica cuando el entrenamiento ha finalizado de la Red Hopfield.

Para observar los pesos finales obtenidos del proceso de entrenamiento se deberá hacer un click sobre la pestaña llamada “Pesos”, enseguida se observara la habilitación de la ventana que muestra los pesos finales obtenidos. Si la matriz de pesos obtenida es demasiado grande, en esta ventana se habilitará un Scroll para poder deslizarse a través de la ventana tanto en posición horizontal como en vertical, ver figura 137.

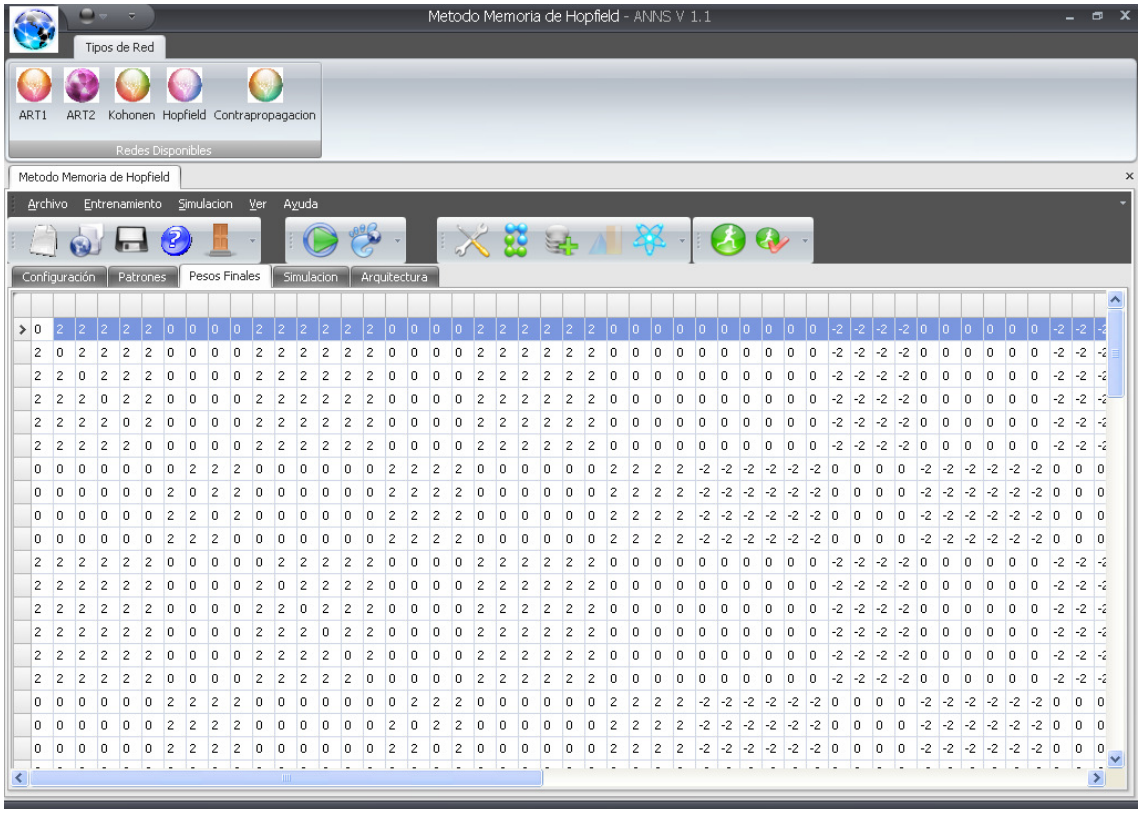


Figura 137. Ventana de pesos finales de la Red Hopfield.

Para poder realizar la simulación de la red entrenada se deberá de hacer click sobre la pestaña llamada “Simulación”, inmediatamente se habilitará la ventana de simulación, en la que se podrán digitar valores dentro de una matriz de prueba. Estos valores dependerán del tipo de red que se configuro al inicio, es decir, si fue de tipo Bipolar o Unipolar, por lo tanto, el patrón de prueba deberá ser del mismo tipo. Por defecto todos los elementos de la matriz de prueba serán 1, para cualquier tipo de configuración. Si el usuario quiere cambiar el valor de algún elemento de la matriz por valores de 0 ó -1, entonces deberá hacer doble click sobre la casilla correspondiente a dicho elemento. Cuando un elemento tiene valores de 0 ó -1, la casilla de éste se tornará de color blanco, ver figura 138

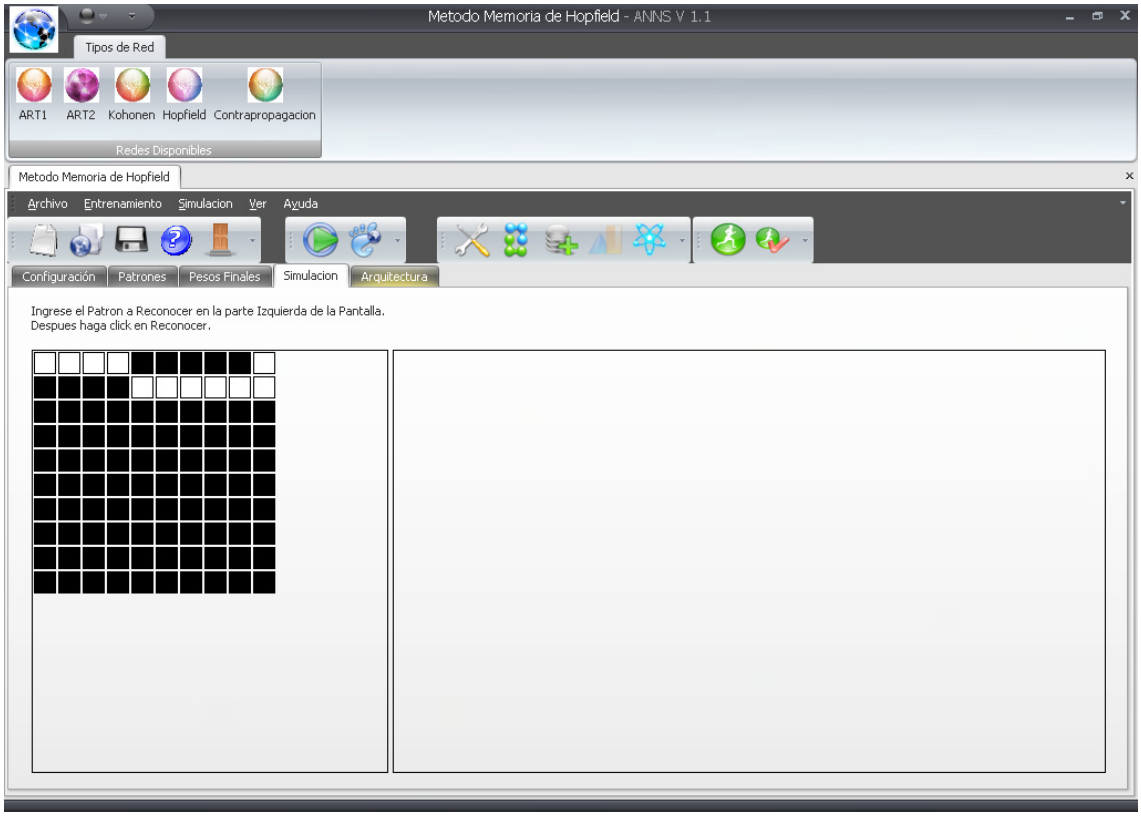


Figura 138. Ventana de simulación que muestra patrón de prueba de la Red Hopfield.

La simulación de la Red Hopfield consiste en verificar si la red aprendió los patrones de entrenamiento realizando una comparación entre el de prueba contra los entrenados. Para poder realizar el reconocimiento del patrón de prueba asociado con alguno de los vectores entrenados, se deberá hacer un click sobre el botón “Reconocer”, Ver figura 139

Figura 139. Botón Reconocer de la Red Hopfield.

Si el patrón de prueba ha sido reconocido, significa que lo ha asociado con alguno de los patrones entrenados, entonces aparecerá a la derecha de la ventana de simulación, una ventana donde se observará un mensaje que indica cual fue el patrón de entrenamiento asociado con el de prueba, ver figura 140

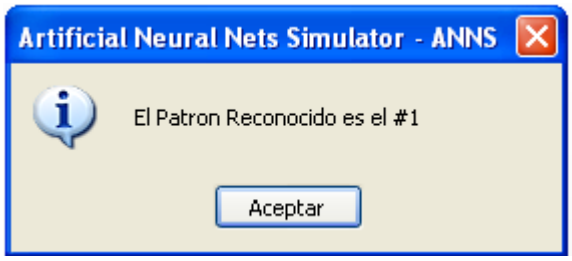


Figura 140. Ventana que indica que patrón de entrenamiento se ha reconocido de la Red Hopfield.

Cuando el usuario haga un click sobre el botón "Aceptar" de la ventana mostrada en la figura 139, entonces aparecerá a la derecha de la ventana de simulación la matriz del patrón de entrada asociado con el de prueba, ver figura 141. Verificando de esta manera si el proceso de entrenamiento ha sido efectivo.

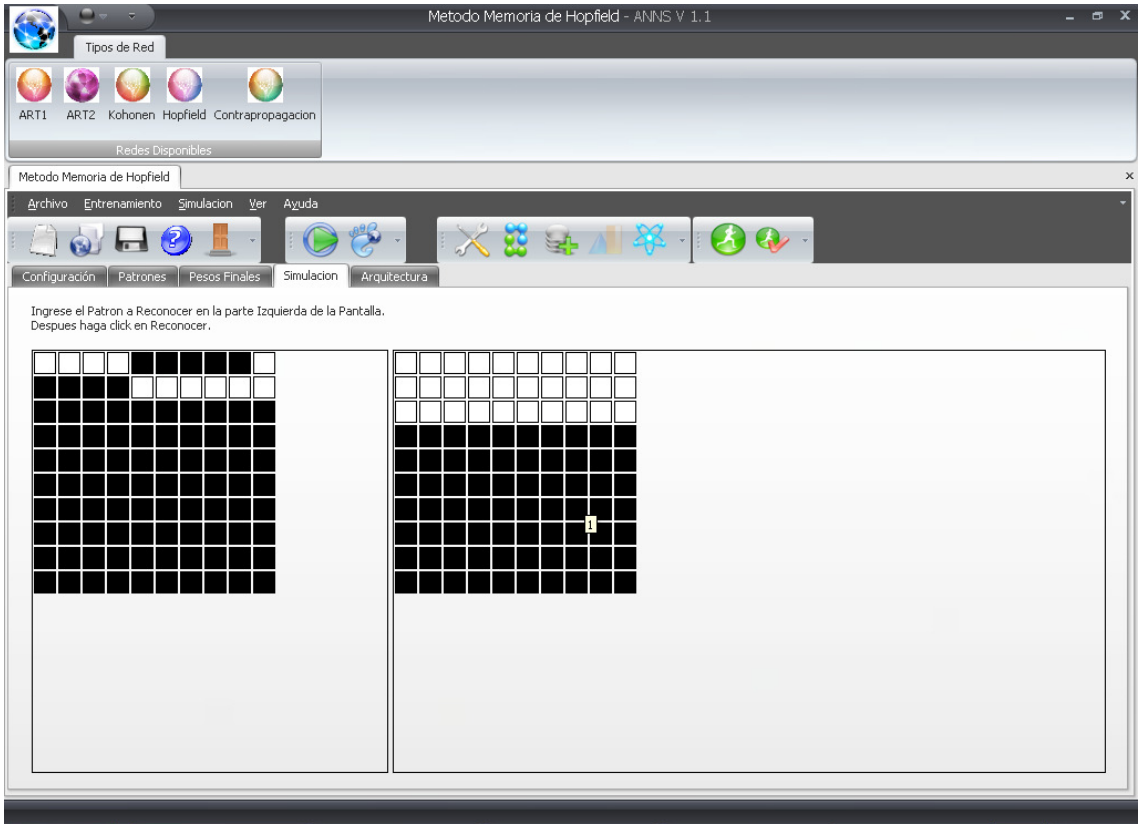


Figura 141. Ventana que muestra el patrón de prueba (derecha) y el patrón previamente entrenado (izquierda).

Si el usuario desea observar la arquitectura de la red actual deberá hacer un click sobre la pestaña llamada Arquitectura, ver figura 142.

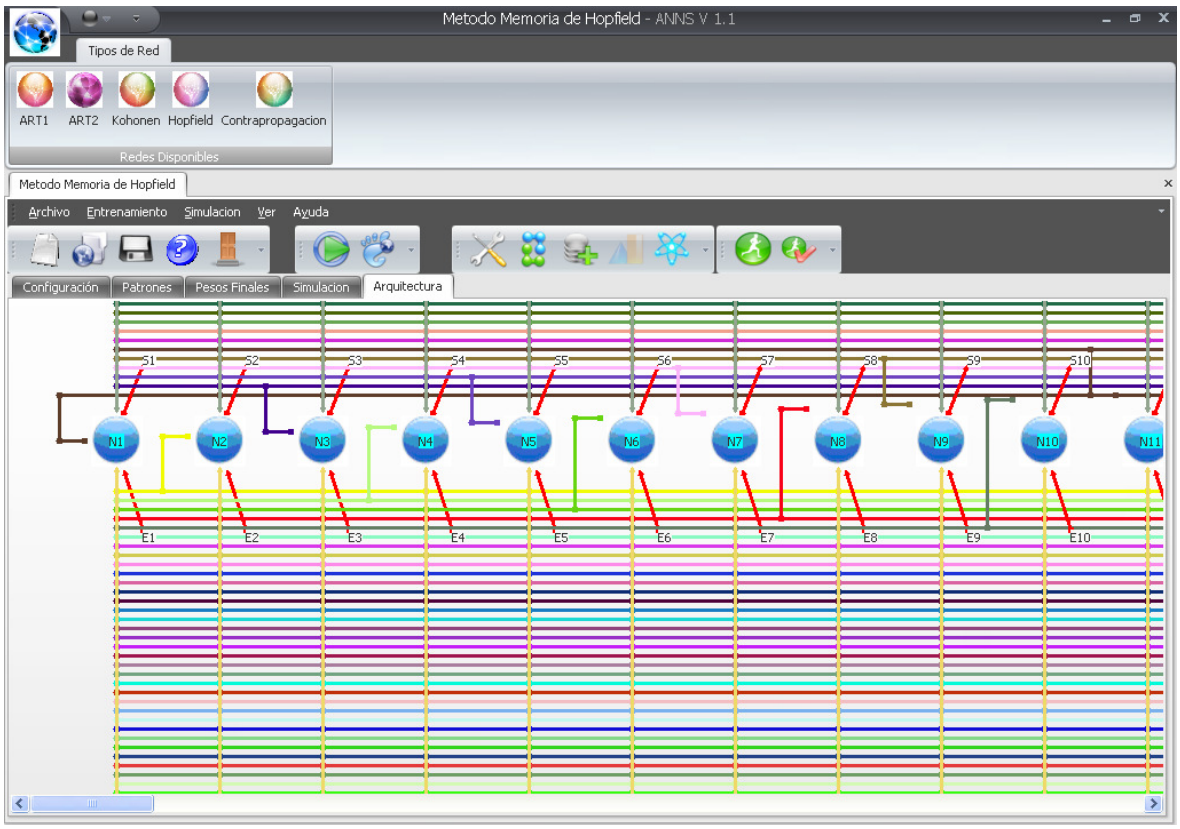


Figura 142. Ventana de arquitectura de la Red Hopfield.

4.3.6 Creación de una Red Contrapropagación.

Para poder generar una red Contrapropagación el usuario deberá de hacer un click sobre el botón llamado “Contrapropagación” ubicado dentro del menú de inicio, ver figura 143.

Figura 143. Botón para crear Red Contrapropagación.

Cuando se realiza esta acción se despliega la Ventana de Configuración de parámetros iniciales para la Red Contrapropagación de la figura 144.

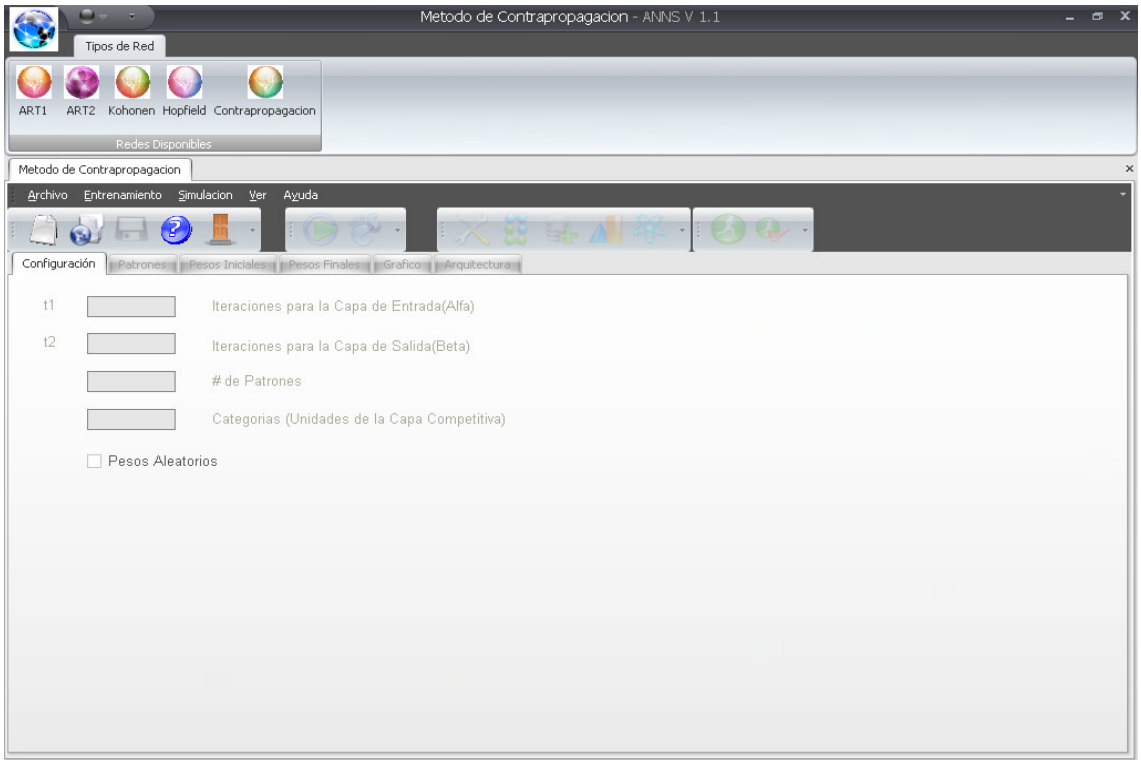


Figura 144. Ventana de Configuración no habilitada para la red Contrapropagación.

En la figura 143, se observa que la ventana de configuración no aparece activada; para poder activarla se hará un click sobre el botón “Nuevo" ubicado en la barra de accesos directos, ver figura 145.

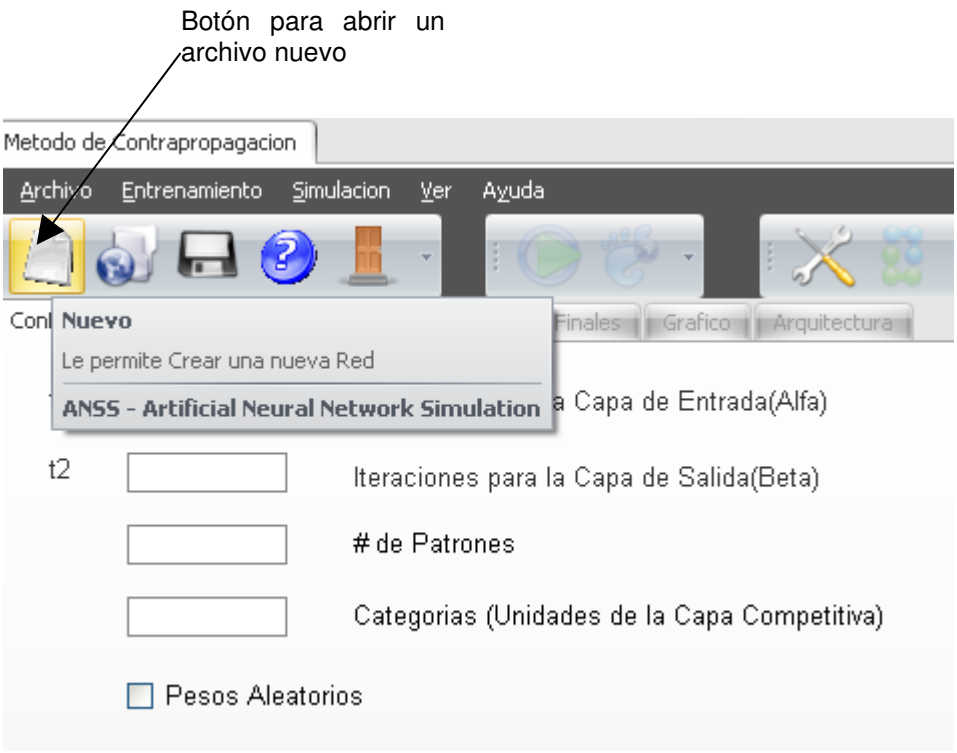


Figura 145. Botón nuevo para habilitar la ventana de Configuración de la Red Contrapropagación.

Con al acción anterior la Ventana estará lista para poder introducir los parámetros iniciales de la Red que se esta creando.

Los parámetros que el usuario deberá digitar en la Ventana de Configuración son los siguientes:

- t1 = Razón de aprendizaje para la capa de entrada (α)
- t2 = Razón de aprendizaje para la capa Competitiva (β)
- # de Patrones = Cantidad de vectores que la red entrenará.
- Categorías = Cantidad de Categorías que la red creara (Cantidad de Neuronas en la capa competitiva)

En la Ventana de Configuración, el usuario deberá elegir si los pesos que utilizará serán los generados por el mismo software (“Pesos Aleatorios”) o serán valores predefinidos que el usuario deberá introducir, para entrenar la Red que se esta creando.

Si se quieren utilizar los pesos que es ANNS genera de forma automática, entonces, dentro de la ventana de Configuración se deberá de hacer un click sobre la opción llamada “Pesos Aleatorios”. Se observará que la pestaña llamada pesos quede deshabilitada, ver figura 146.

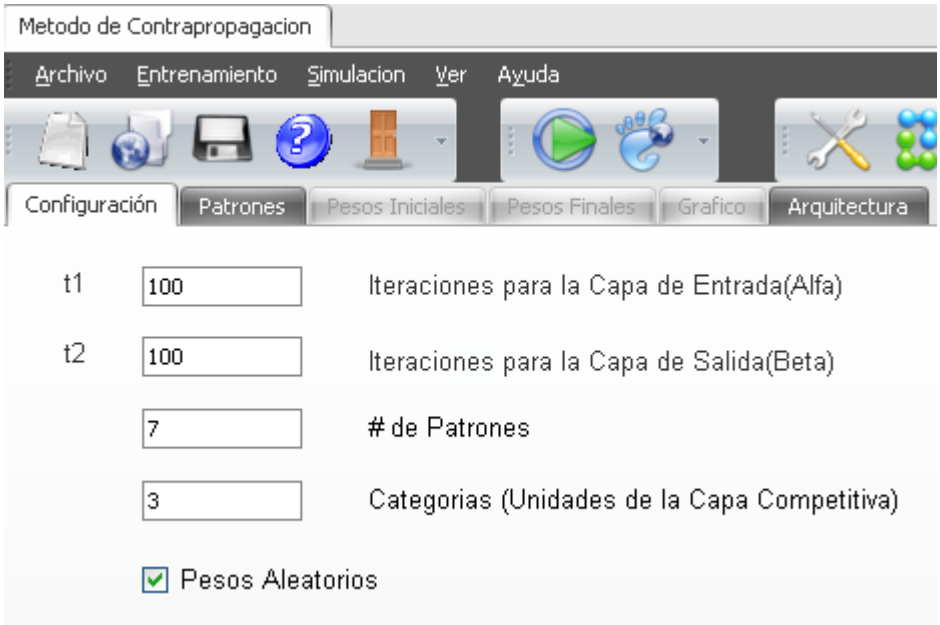


Figura 146. Ventana de configuración con la opción de “Pesos Aleatorios” activada de la Red Contrapropagación.

Por el contrario, si no se utiliza la opción llamada “Pesos Aleatorios”, la pestaña llamada “Pesos Iniciales”, se habilitara, y haciendo un click sobre ella, aparecerá la ventana en la que se introducirán los valores de los pesos que el usuario quiere utilizar para entrenar de la red creada, ver figura 147.

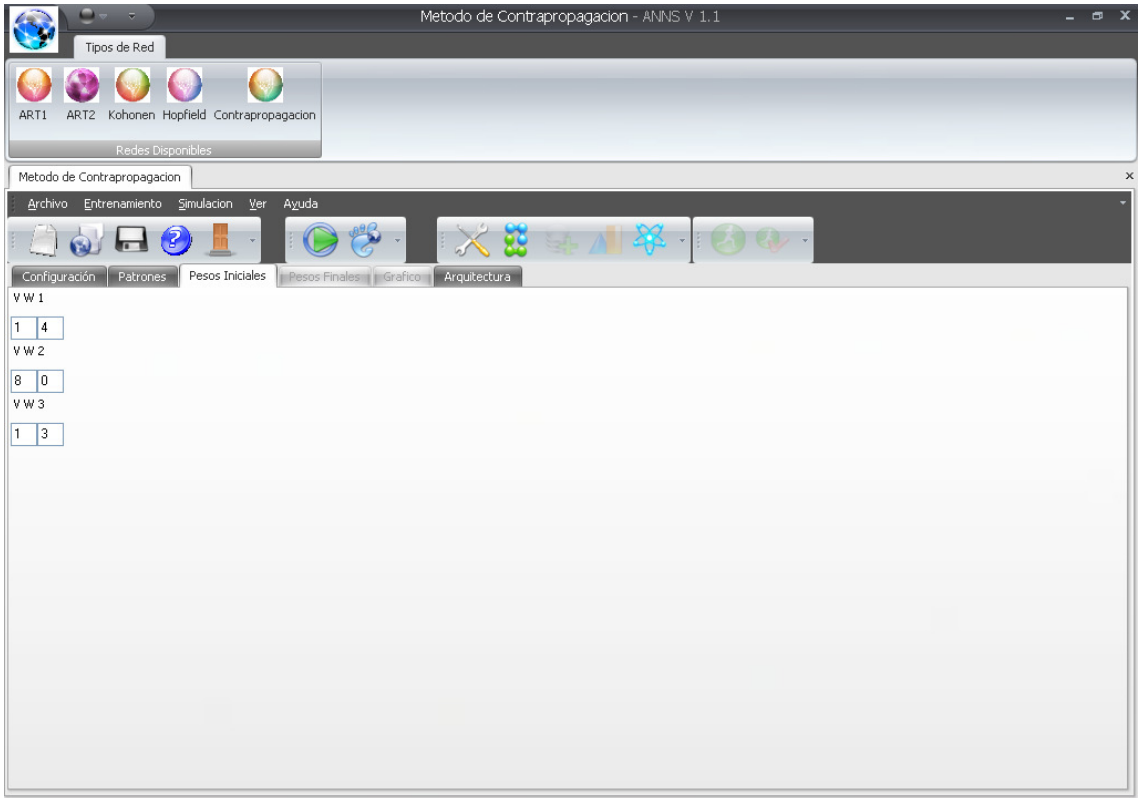


Figura 147. Ventana de Pesos Iniciales de la Red Contrapropagación.

Cuando los pesos han sido definidos, se deberán de introducir los vectores de entrenamiento. Para esta topología el usuario debe tomar en cuanta que solamente se pueden entrenar pares ordenados (X, Y), por lo tanto cada vector de entrenamiento solamente podrá constar de dos elementos.

Para poder digitar los vectores de entrenamiento se deberá de hacer un click, sobre la pestaña llamada “Patrones”. Se observará que se habilita una ventana donde se podrán introducir los valores de los vectores que la red deberá aprender, ver figura 148.

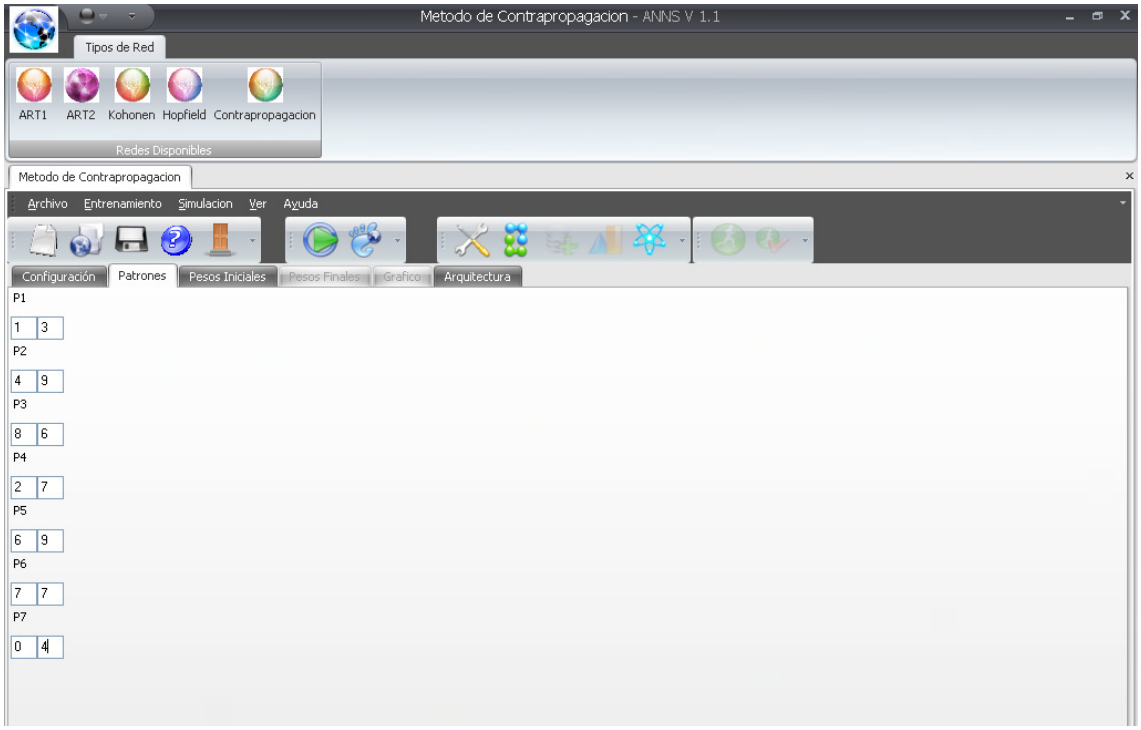


Figura 148. Ventana llamada "Patrones" de la Red Contrapropagación.

Realizado todo lo anterior, la red actual estará lista para ser entrenada. Para entrenar la red el usuario tiene dos opciones disponibles. Una es realizar el entrenamiento de una vez o paso a paso; ambas opciones se encuentran disponibles dentro de la Barra de Accesos Directos. Con hacer un click sobre cualquiera de estos dos botones la red creada será entrenada, ver figura 149.

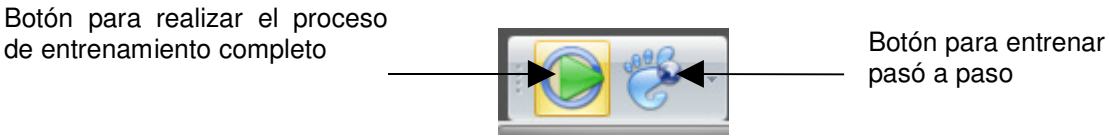


Figura 149. Botones "Entrenar" y "Paso a Paso" de la Red Contrapropagación.

Si el usuario decide detener el entrenamiento antes de que este finalice, deberá de hacer un clic sobre el botón llamado "Detener", ver figura 150.

Figura 150. Botón Detener del proceso de Entrenamiento de la Red Contrapropagación.

Cuando el entrenamiento finaliza, el ANNS V1.1, desplegará un mensaje que indica que esto ha ocurrido, ver figura 151.

Figura 151. Ventana que muestra cuando el entrenamiento de la red Contrapropagación ha finalizado.

Cuando el entrenamiento ha concluido se activaran las pestañas llamadas “Pesos Finales “ y “Gráficos”. Así como los botones que activan estas funciones desde la barra de Accesos directos, ver figura 152.

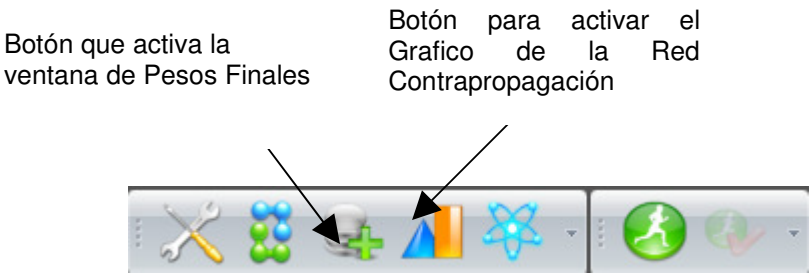


Figura 152. Botones de Pesos Finales y Gráfico de la Red Contrapropagación.

Para poder observar los pesos alcanzados durante el proceso de entrenamiento así como los pesos finales se hará un click sobre la pestaña llamada “Pesos Finales”, ver figura 153.

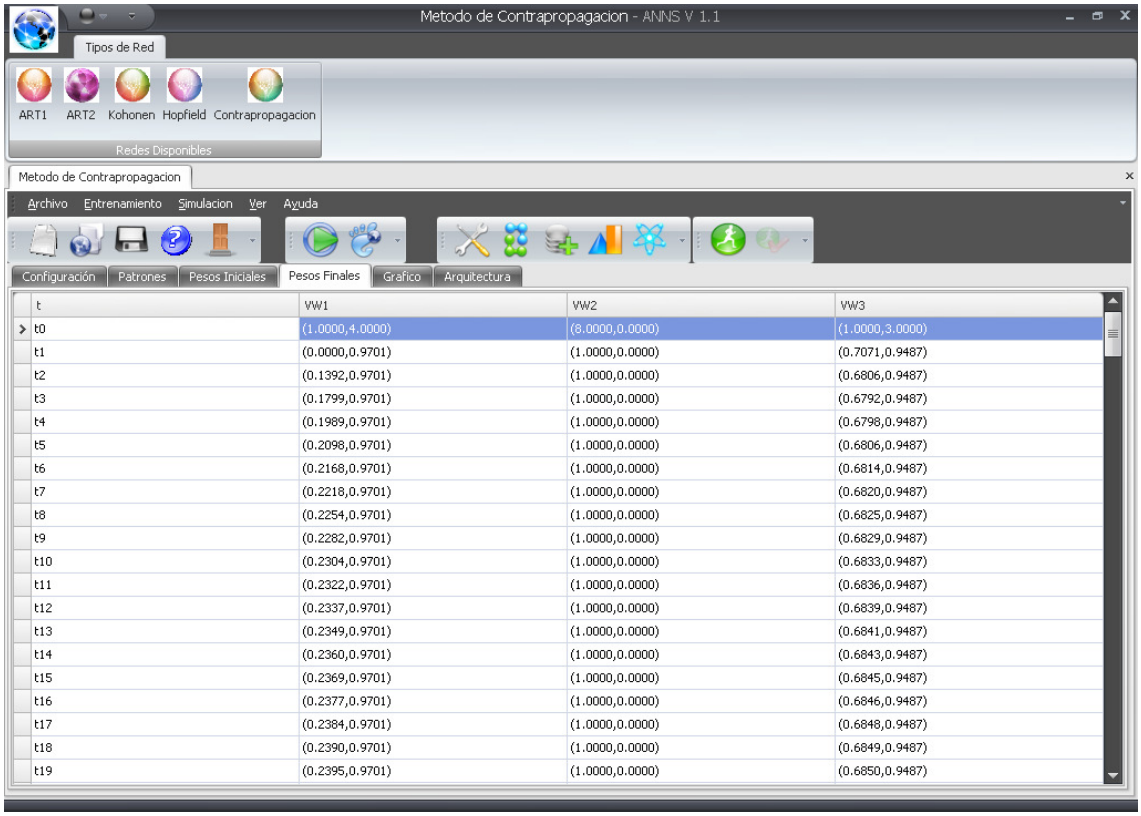


Figura 153. Ventana llamada "Pesos Finales" de la Red contrapropagación.

Para poder desplazarse a lo largo y ancho de la ventana, se utilizarán los Scrolls que se activarán de ser necesario.

La pestaña llamada "Gráfico" se activará y se podrá observar la gráfica de los pesos finales haciendo un click sobre esta, ver figura 154.

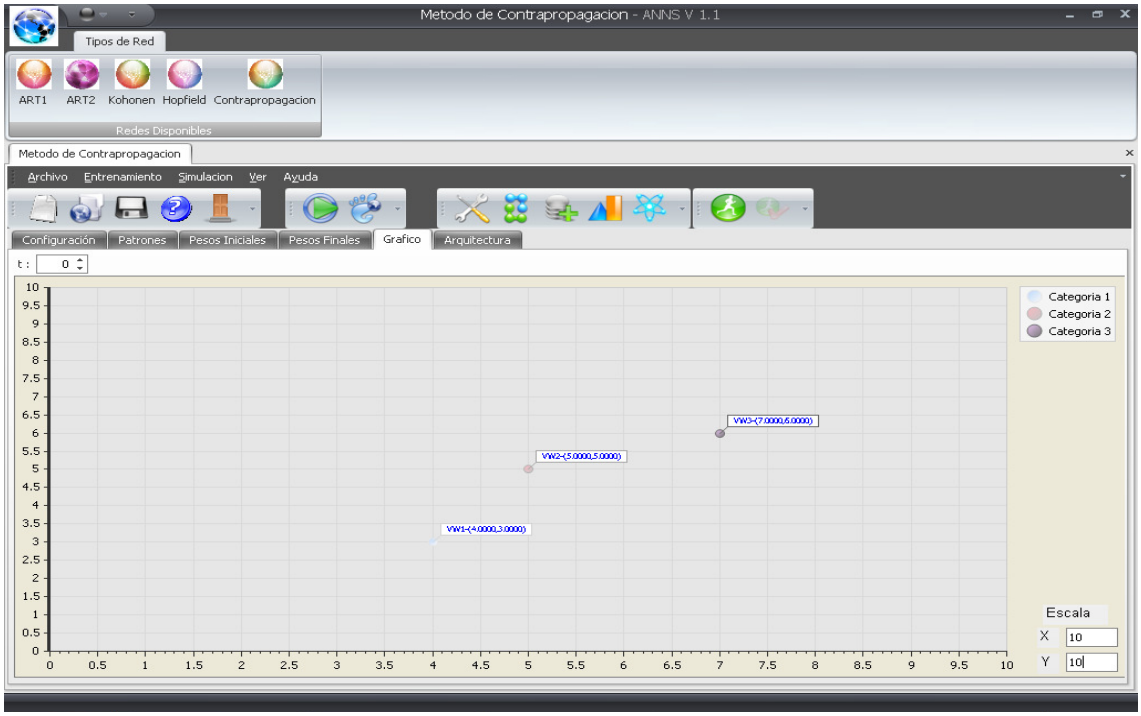


Figura 154. Ventana que muestra la grafica de los pesos generados durante el entrenamiento de la Red Contrapropagación.

Dentro de esta ventana aparece en la esquina superior izquierda, un control por medio del cual se puede observar la evolución de los pesos de forma grafica, con solo introducir un valor de iteración para la cual se quiere visualizar la grafica de los pesos para esa época especifica, ver figura 155.

Figura 155. Muestra control para variar el número de iteración de la Red Contrapropagación.

A la derecha de la ventana, se observa un cuadro de dialogo donde se muestra por medio de diferentes colores cada una de las categorías creadas por la red actual, ver figura 156.

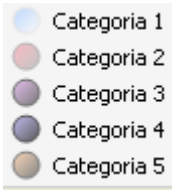


Figura 156. Cuadro que muestra por colores las diferentes categorías de la Red Contrapropagación.

También se observará en la esquina inferior derecha un control, por medio del cual, se podrá cambiar las escalas del grafico con solamente digitar los valores de X y Y en las casillas correspondientes, ver figura 157.

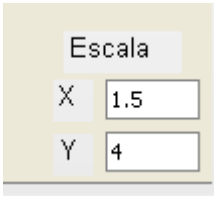


Figura 157. Control para cambiar escala del gráfico de la Red Contrapropagación.

Para poder realizar la simulación se deberá hacer un click sobre el botón llamado “Simular”, que aparece en la barra de accesos directos, ver figura 158.

Figura 158. Botón “Simular” de la Red Contrapropagación.

Entonces aparecerá una ventana en la cual se deberá de digitar un vector de prueba, el cual deberá ser categorizado por la red actual, ver figura 159.

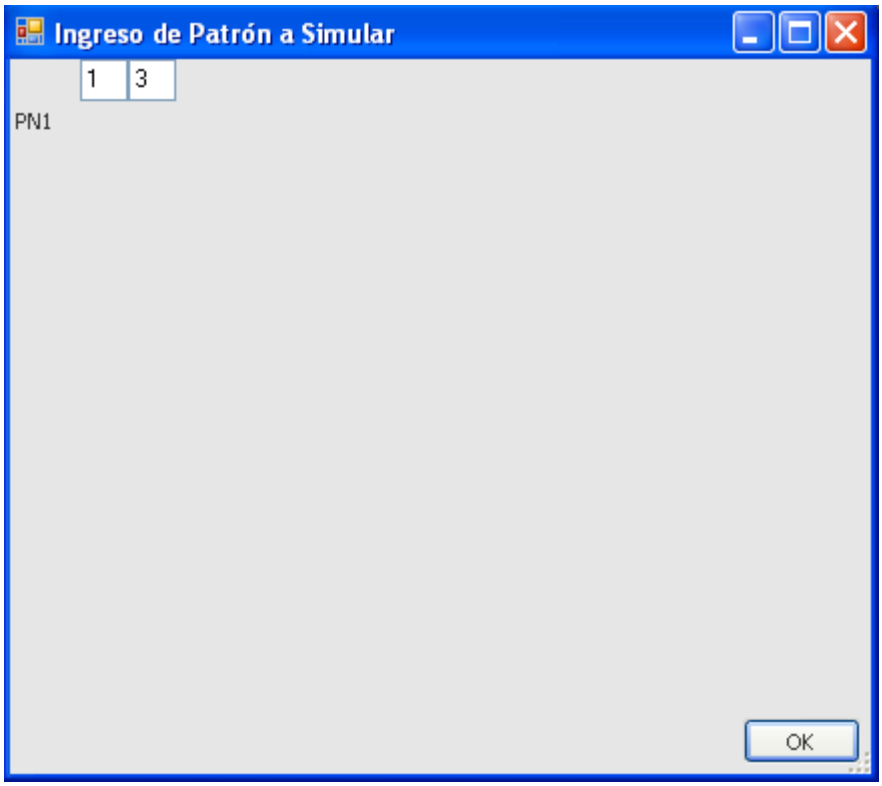


Figura 159. Ventana que muestra la ventana del patrón de prueba para realizar la simulación de la Red Contrapropagación.

Para que la simulación se realice, se deberá hacer un click sobre el botón “OK” de la ventana mostrada en la figura anterior. Entonces aparecerá otra ventana que indica a que categoría pertenece el vector de prueba, ver figura 160.

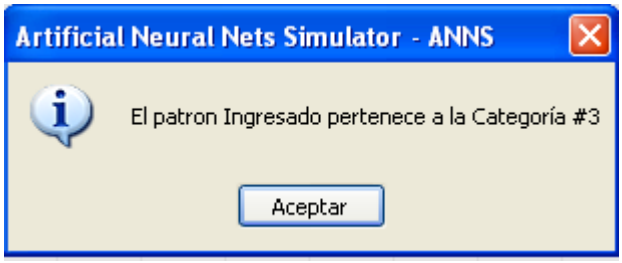


Figura 160. Ventana que muestra a que categoría pertenece el vector de prueba de la Red Contrapropagación.

Al hacer un click sobre el botón “Aceptar de la ventana mostrada en la figura 160. En la ventana llamada “Grafico” se observará el vector de prueba en color Rojo, mostrando por medio de la distancia, la categoría al cual este pertenece, ver figura 161.

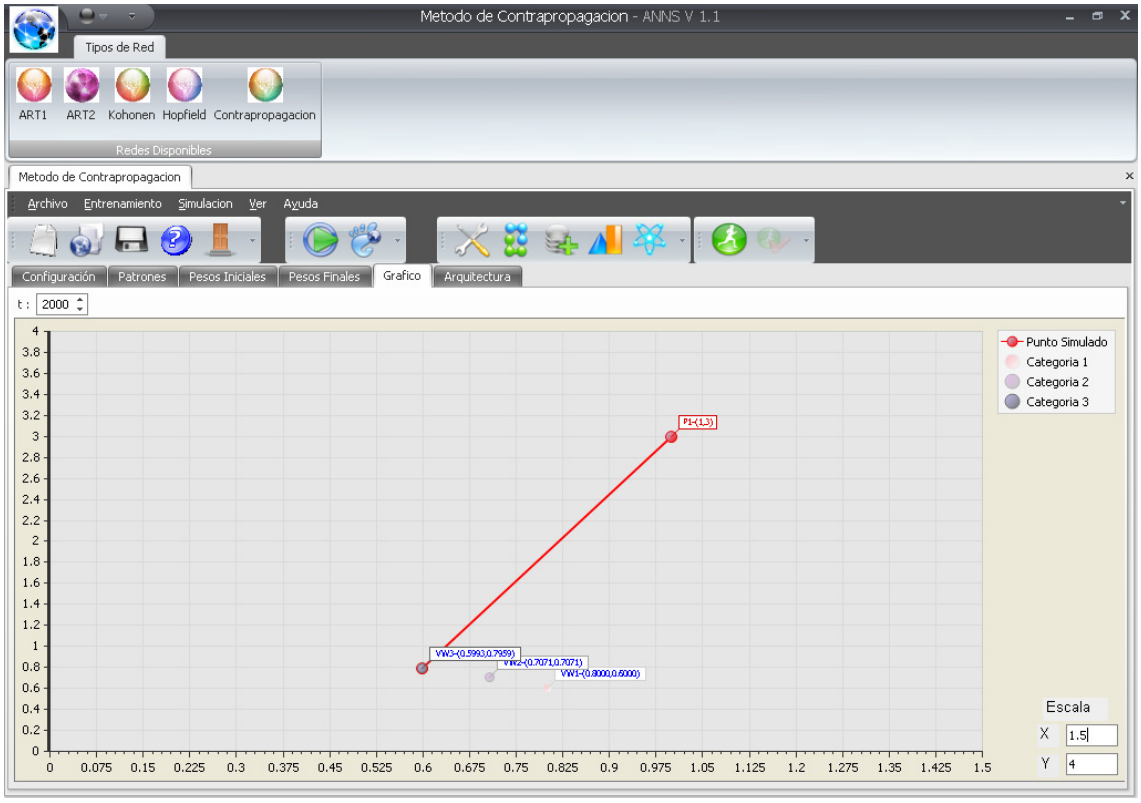


Figura 161. Ventana que muestra la categoría a la que pertenece el vector de prueba de la Red Contrapropagación.

Si el usuario desea graficar la arquitectura de la red deberá hacer un click sobre la pestaña llamada “Arquitectura”, ver figura 162.

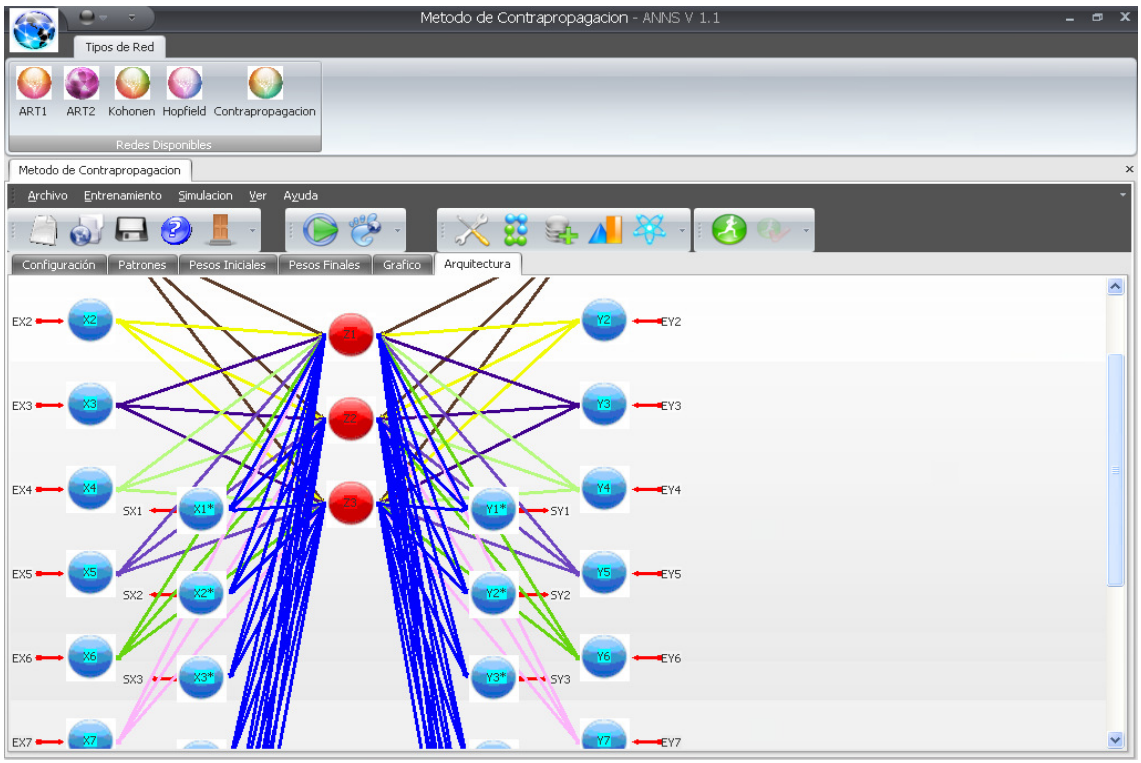


Figura 162. Ventana que muestra la arquitectura de la Red Contrapropagación.

CONCLUSIONES

- El ANNS V 1.1 es un simulador de Redes Neuronales capaz de entrenar y simular redes creadas dentro de la aplicación para cinco topologías distintas. Por medio de este software se brinda la oportunidad al usuario de incorporarse al mundo de las Redes Neuronales Artificiales de manera sencilla sin necesidad de realizar procedimientos engorrosos que entorpezcan el proceso de aprendizaje acerca de las mismas.

- El ANNS V1.1, ha demostrado ser un software con una interfaz gráfica que permite al usuario adaptarse con facilidad a los procesos realizados por el mismo, esto debido, a que el ANNS V1.1, incorpora el uso de barras de menú y botones que generan una mejor interacción con el usuario, dando como resultado una optimización en el proceso de entrenamiento y simulación de las redes creadas dentro de la aplicación.

- Las ventanas del simulador han sido desarrolladas para brindar al usuario la oportunidad de identificar cada uno de los procesos por los que la red creada deberá pasar para llegar al proceso de simulación, lo que permitirá que todos los procesos ejecutados para el desarrollo de la red, puedan ser verificados y estudiados con mayor facilidad.

- Se utilizo la suite de controles DevExpress para .NET 2005 v6.3, con el cual se implemento el interfaz de usuario, mediante el uso del estilo de aplicación Ribbon y además se le incorporo el uso de Skins. Se agrego el uso de XtraBars para la implementación de los menús y barra de herramientas de cada red, además se utilizo el XtraCharts, para el uso de las graficas que se han implementado.

- El ANNS V1.1, posee grandes ventajas con respecto al software utilizado actualmente en el desarrollo de las Prácticas de Laboratorio de la Cátedra de redes Neuronales Artificiales, ya que permite que los proyectos que se están simulando se puedan detener incondicionalmente en cualquier instante, sin perder los datos obtenidos hasta ese momento. La reanudación del proceso no requiere de procedimientos ni prácticas extenuantes para el usuario.

- El Simulador incorpora las opciones de Guardar y Guardar como, las que permiten que el usuario pueda guardar sus proyectos, cuando estos aun se encuentran en fase de desarrollo, sin perder los datos obtenidos hasta ese momento y lo mas importante sin necesidad de terminar obligatoriamente los procesos de entrenamiento y creación de la red para poder guardarlos y abrirlos posteriormente. Estas opciones son de gran ayuda cuando los procesos de entrenamiento son extensos debido a la magnitud de la red creada.

- Para el desarrollo del ANNS V1.1, fue necesario realizar una investigación mas profunda acerca de generalidades de Redes Neuronales Artificiales, además del funcionamiento de las cinco topologías incluidas en el mismo. Siendo entonces, el camino para determinar los procesos de entrenamiento y simulación para cada tipo de red.

- Dentro de las pruebas realizadas al simulador ANNS V1.1, se tomo en cuenta la comparación de los resultados obtenidos por autores de algunos de los libros utilizados como bibliografía, para los cinco tipos de red, además se realizaron comparaciones de datos obtenidos con el software Matlab para la red Kohonen y Hopfield. Los resultados que se obtuvieron producto de estas pruebas generan gran confianza en el ANNS v 1.1, puesto que el porcentaje de certeza fue de un 99%, generándose un error aproximado de un 1%.

- Durante el desarrollo del simulador se fueron realizando pruebas de depuración de procedimientos. La mayoría de estos fueron los relacionados al manejo del número de neuronas utilizados por las redes puesto que uno de los mayores retos fue lograr que estas fueran ilimitadas. Existen restricciones acerca del número de neuronas que se podrán utilizar, pero son propias a la topología de red que se esta creando.

- El ANNS v1.1, permite al usuario utilizar el número de neuronas que estime conveniente, sin existir limitaciones en el manejo de estos datos. Sin embargo, el tiempo de respuesta se verá sacrificado a cambio de esta característica, puesto que los procesos de creación, entrenamiento y simulación de una red muy extensa repercutirá en el tiempo de ejecución de los mismos. Es necesario hacer énfasis que cuando se nos sugirió un manejo ilimitado de neuronas quedo claro el hecho que esto repercutiría en el tiempo que la red tardaría en procesarse.

- Para la red Hopfield se hace necesario recalcar que es una red que posee una limitante muy importante respecto al número de patrones que será capaz de aprender. Esto es, que esta red por su propia topología solamente podrá reconocer o aprender una cantidad menor de informaciones al 13% del total del número de neuronas que ésta contenga. Por ejemplo si la red contiene 100 neuronas, entonces solamente podrá almacenar un valor menor a 13 informaciones diferentes.

- La función de transferencia utilizada para la red ART1, es la función escalón, por tratarse de una red de tipo binaria. Por lo tanto no los autores de varios libros que tratan acerca de esta red (Freeman, Fausett) no recomiendan el uso de otro tipo de función de transferencia. Es por esta razón que el simulador para la topología ART1, no posee una opción para realizar el cambio de la función de transferencia.

- Para la Red ART2, la razón que justifica el uso de la función de transferencia implementada en el simulador, es que, de acuerdo a los creadores de esta red (Carpenter & Grossberg 1987), las unidades de las capas *U* y *P* alcanzan el equilibrio después de tan solo 2 ciclos de actualización de la capa *F1*, y es además la recomendada por los autores que han realizado publicaciones sobre el tema (Fausett 1994, Hilera).

- Tanto para la red Kohonen y Contrapropagación no se mencionan funciones de Transferencia dentro del algoritmo. Por lo tanto en el simulador para estas topologías, no se han incluido opciones que permitan realizar cambios de dichas funciones.

  En el caso de la red Kohonen, se trata de un tipo de aprendizaje competitivo, por lo tanto se trata de encontrar la neurona ganadora de la capa de salida por medio de la distancia Euclidea. Para la red Contrapropagación el algoritmo esta dividido en dos fases, la primera involucra la capa de entrada y una capa competitiva, lo que implica que se trata de un aprendizaje de tipo competitivo para la primera capa. En la segunda fase se entrena la capa outstar de Grossberg.

- El Simulador ANNS V1.1, para la red Hopfield, permite realizar el cambio entre dos posibles funciones de transferencia: La escalón bipolar para valores de (+1, -1) y la escalón unitario para valores (0, 1). Esto se debe a que el simulador se implemento para realizar la creación, entrenamiento y simulación de la versión digital y no se incluyo la versión analógica, debido a falta de información confiable acerca de la misma.

RECOMENDACIONES

- Al programador. El ANNS V1.1, es un simulador creado como la segunda parte del SG-SIM 2006.

  El ANNS V1.1, incorpora topologías que no fueron incluidas dentro del primero pero, que son parte de las Redes estudiadas en la Cátedra de Redes Neuronales Artificiales, impartida en el Universidad Don Bosco. Además este incluye ciertas funciones que mejoran la ejecución de los procesos de creación, entrenamiento y simulación de las Redes. Sin embargo es necesario que se realice la unificación de ambos sistemas permitiendo manejar dentro de un solo software todas las topologías que se tiene disponibles en cada simulador, sin sacrificar las mejoras realizadas en el ANNS V1.1. Una de las funciones que ninguno de los simuladores posee actualmente y que es necesario añadir a un trabajo posterior es la importación y exportación de archivos de texto, para optimizar el tiempo en la introducción de los patrones de entrenamiento, cuando estos son extensos.

- Al programador. Se recomienda incorporar una función que realice el ajuste automático de las graficas a fin de obtener una mejor visualización de los pesos obtenidos sin que el usuario deba recurrir a realizar cambios en las escalas del mismo por medio de botones.

- Al programador. Recomendamos incorporar en la etapa de simulación de las redes Kohonen y Contrapropagación, la posibilidad de poder simular las redes en cualquier iteración, utilizando animaciones para observar la evolución de os pesos de manera gráfica.

- Al Usuario. Antes de comenzar a utilizar el ANNS V1.1, se recomienda tener conocimiento acerca de las topologías que el simulador incorpora. De este modo, el tiempo se reduce el tiempo que le tomara al usuario adaptarse al ambiente del ANNS V 1.1.

- Al usuario. Antes de instalar el ANNS V1.1, se recomienda configurar adecuadamente “La configuración regional y de Idiomas” que se encuentra dentro del panel de control. Cuando hemos abierto esta opción en “opciones regionales” se debe escoger el idioma “Español de El Salvador”. Con lo anterior aseguramos que el uso de los decimales sea del estilo numérico sea el de El Salvador: el uso de las comas para designar miles y el punto para decimales.

- Al Usuario. Al iniciar el simulador ANNS V1.1, y escoger la red que se utilizara, es importante tomar en cuenta la arquitectura de esta, debido a que el simulador dentro de la ventana de configuración solicitara parámetros y numero de neuronas propias de cada topología. Si se intenta construir redes con parámetros que no están bien definidos o con un número de neuronas inadecuado, se incurrirá en una perdida de tiempo innecesaria.

  Por otra parte, puede existir alguna red para la que el tiempo de entrenamiento pueda llevar varias horas, se recomienda detener el entrenamiento y guardar dicha red, para continuar posteriormente. El archivo generado por el simulador es de extensión *.rn, además se recomienda que estos archivos sean abiertos solamente con el simulador ANNS V1.1, para reducir al mínimo la posibilidad de una perdida en alguno de los valores guardados por la red.

BIBLIOGRAFÍA

[1] Fousett Laurene, ”Fundamentals of Neural Networks; Architectures, Algorithms and Aplications”. Prentice Hall, Jersey 1994.

[2] Hilera José y Martínez Víctor, “Redes Neuronales Artificiales; Fundamentos, modelos y aplicaciones”. Addison – Wesley Iberoamericana, S.A. De la
Edición RA-MA 1995.

[3] Freeman, James y Skapura David, “Neural Networks; Algorithms, Aplications and Programming Techniques”. Addison – Wesley Publishing Company,
1991.

[4] Veelenturf, L.P.J., “Analysis And Applications Of Artificial Neural Networks”. Prentice Hall International, 1995.

[5] Blank, Leland y Taquín, Anthony, “Ingeniería Económica”, traducción por Arango Medina, Gladis; Editorial Mc Graw Hill, edición 1999.

[6]Martín del Brío, Bonifacio y Sanz Molina, Alfredo, “Redes Neuronales Y Sistemas Difusos”, Alfaomega Ra-Ma, Segunda Edición Ampliada y Revisada 2005.

[7] Universidad de Antioquia, Facultad de Ingeniería, unidad de investigación. Mecatrónica.
http://ingenieria.udea.edu.co/investigacion/mecatronica/mectronics/redes.htm
(Consulta: Septiembre de 2007).

[8] Universidad de los Andes, Facultad de Ingeniería. Publicaciones del Ing. Lourdes Arroyo.
http://tecnologiaysociedad.uniandes.edu.co/html/memoria/2003/matrix/icaro/noticia-4.html
(Consulta: Febrero de 2008)

[9] Universidad Tecnológica de Pereira, Facultad de Ingeniería Electrónica. *Tutorial*
de Redes Neuronales. http://ohm.utp.edu.co/neuronales/ (Consulta: Octubre de 2007)

[10] Herramientas en GNU/Linux para estudiantes universitarios, http://es.tldp.org/Presentaciones/200304curso-glisa/redes_neuronales/curso-glisaredes_ neuronales-html/x17.html (Consulta: Febrero de 2008)

[11] Artículo publicado por PCWORLD. *Inteligencia Artificial y Redes Neuronales Permiten imitar el cerebro humano*.
http://www.idg.es/pcworld/articulo.asp?id=34270. (Consulta: Febrero de 2008)

[12] Tecnológico de Monterrey, Material de Apoyo de “Lógica computacional”

http://w3.mor.itesm.mx/~logica/log9808/log_pred.html (Consulta: Marzo de 2008)

[13] REYES, Gary "Proyecto de inteligencia artificial" en Universidad de Guayaquil. http://cruzrojaguayas.org/inteligencia/ (Consulta: Marzo de 2008)

[14] PSCOACTIVA, “Las Torres de Hanoi”, Juego interactivo de inteligencia. http://www.psicoactiva.com/juegos/hanoi/jg_hanoi.htm (Consulta: Febrero de 2008)

[15] Programación Básica .Net
http://www.elguille.info/NET/cursoVB.NET/indice.htm (Consulta: Septiembre de 2007)

[16] Microsoft para la tecnología de aplicaciones Windows o exes.
http://www.windowsforms.net (Consulta: Enero de 2008)