UNIVERSIDAD DE GUADALAJARA

CENTRO UNIVERSITARIO DE CIENCIAS BIOLÓGICAS Y AGROPECUARIAS
DEPARTAMENTO DE CIENCIAS AMBIENTALES
CENTRO DE ESTUDIOS E INVESTIGACIONES EN COMPORTAMIENTO

INHIBICIÓN CONDICIONADA EN REDES NEURALES TIPO DONAHOE-BURGOS-PALMER: UN ESTUDIO EXPLORATORIO

TESIS
QUE PARA OBTENER EL GRADO DE

DOCTOR EN CIENCIAS DEL COMPORTAMIENTO
OPCIÓN ANÁLISIS DE LA CONDUCTA

PRESENTA:

ANTONIO PONCE ROJO

DIRECTOR:

DR. JOSÉ E. BURGOS TRIANO

COMITÉ:
DR. JOSÉ E. BURGOS TRIANO
DR. FELIPE CABRERA GONZÁLEZ
DR. OSCAR GARCÍA LEAL

GUADALAJARA, JALISCO, MÉXICO, MAYO DEL 2007

A Don Antonio y Doña Zenaida
Otra vez Gracias por todo

El presente trabajo no habría sido posible sin la ayuda de las siguientes instituciones y personas a quien se expresa infinita gratitud:
Consejo Nacional de Ciencia y Tecnología
Universidad de Guadalajara
Centro Universitario de la Costa de la Universidad de Guadalajara
Centro Universitario de los Lagos de la Universidad de Guadalajara
Centro Universitario de Ciencias Biológicas y Agropecuarias de la Universidad de Guadalajara
Bañuelos Hernández, Martha Cristina
Burgos Triano, José E.
Cabrera González, Felipe
Castelán Rueda, Roberto
Castro Sánchez, Cintia Elizabeth
Cienfuegos Cervantes, Aldo
Cortes, Paola Alejandra
Cortes Velázquez, Consuelo
Díaz Tenorio, José
Estrada Mercado Soriano, Luciano Benedicto de Jesús
Estrada Michel Gerardo Simón
Fernández Rodríguez, Jeffry S.
Figueroa Ypiña, Claudia Patricia
Figueroa Ipiña, Karina
Flores Aguirre, Carlos Javier
Galván Álvarez, Hugo Isaac
García Leal, Oscar
Grajeda Velázquez, Jorge
Hernández Castillo, Blanca Leticia
Hernández Contreras, Jorge
Hernández Islas, Mónica
Hernández Vega, Leticia
Herrera Martín, Angélica María
Lazareno Sotelo, Mirza Liliana
López Barrón, Aurelio Enrique
López López, José Luis
Martínez Borrayo, Juan Gerardo
Martínez Sánchez, Félix Héctor
Orozco Alvarado, Javier
Pérez Hernández, Iris Zuleica
Ponce Alonso, Antonio
Ponce Rojo, Arturo
Ponce Rojo, Juan Carlos
Ponce Rojo, Pedro
Ramírez Veloz, Velvet Leticia
Ribes Iñesta, Emilio
Rojo Vda. de Ponce, Zenaida
Ruiz Aguiar, Adriana Karina
Sánchez Medina, Edmundo
Soltero Robles, Yareli Mailen
Valerio dos Santos, Cristiano
Vazquez Navarro, Javier
Velázquez Gutiérrez, Patricia
Zepeda Peña, Héctor Hugo

RESUMEN

El presente documento pretende presentar al lector el trabajo realizado para explorar la plausibilidad conductual de un modelo específico de redes neurales, el Modelo seleccionista DBP (Donahoe, Burgos y Palmer, sus autores), para la simulación de la inhibición condicionada.

El aprendizaje en organismos vivos, ya sea entendido como cambio conductual o como cambio cognitivo, involucra de manera crítica la estructura y funcionamiento del sistema nervioso. En modelos de aprendizaje basados en cambios neurales, se conceptúa el cambio conductual como el resultado de la interacción entre la estructura y funcionamiento de sistemas nerviosos, por una parte, y por otra, el medio ambiente. Tal conceptualización se concentra en los procesos que tienen lugar en sistemas nerviosos cuando organismos adultos y sanos, se comportan.

Desde las Ciencias Computacionales y las Ciencias del Comportamiento, los circuitos lógicos inspirados en la organización del sistema nervioso o algunas de sus propiedades, son conocidos como *Redes Neurales Artificiales*, o simplemente como *Redes Neurales*. El uso de redes neurales en la simulación de fenómenos conductuales es una tarea interdisciplinaria, por tanto, involucra un tránsito por diversas disciplinas, tales como la neurobiología, neuroanatomía, la inteligencia artificial, la computación, la psicología experimental y la psicología del aprendizaje.

Aunque las redes neurales han sido estudiadas desde mediados de la década de los cuarenta del siglo pasado, y en general se les sitúa en un campo único, se pueden en realidad, identificar dos rutas de desarrollo radicalmente distintas. Por un lado se encuentran las redes neurales que tienen como propósito resolver algún problema complejo con base en algunas de las suposiciones acerca de lo que es el sistema nervioso de organismos superiores. Estas redes neurales son las más conocidas debido a las aplicaciones prácticas que se han derivado a partir de ellas. Bajo este esquema lo importante es la solución del problema, al margen de la plausibilidad biológica del modelo utilizado. Y, por otro lado se tiene a las redes neurales que se usan principalmente en investigación y cuyo propósito es conocer y explicar lo que sucede en organismos vivos. En este segundo grupo, la resolución del problema importa, pero la prioridad es la forma en la que la resolución se da, y cómo ésta puede contribuir al entendimiento de los procesos que tienen lugar en los organismos. Este tipo de redes neurales es menos conocido y se enfoca más hacia la investigación que hacia la obtención de aplicaciones prácticas.

El presente trabajo de investigación se ubica en este segundo grupo al haber implicado la realización de un estudio que abone al conocimiento acerca de las posibilidades de un modelo seleccionista de red neural, el modelo DBP[1], específicamente diseñado para la simulación de fenómenos conductuales. Se simuló con este modelo la *inhibición condicionada*, un fenómeno reportado ya en 1927 por Iván I. Pavlov y acerca del cual existen por lo menos cuatro explicaciones teóricas distintas y un gran número de estudios experimentales. Se trata de un fenómeno que no ha sido simulado aún por el DBP.

[1] Llamado así por las iniciales de los apellidos de sus autores: John Donahoe, José Burgos y David Palmer, o también conocido como *SNNM (Selectionist Neural Network Model,* en inglés).

Realizar este trabajo implicó sortear primero, el problema de la realización de una implementación computacional del modelo; es decir, un programa de computadora que funcionara tal y como el modelo DBP lo establece, para lo cual, se diseñó un simulador que ofrece resultados confiables, comparándolo con el único simulador efectivo que existía a la fecha para el modelo, el SELNET, creado por el Dr. José E. Burgos Triano, uno de los creadores del Modelo. Y segundo, afrontar dos retos más: el estudio de las posibles arquitecturas de red, a fin de poder encontrar alguna que permita simular la *Inhibición condicionada*, y la resolución de algunos aspectos prácticos que se han presentado a lo largo del trabajo con el Modelo DBP, como lo es la imposibilidad actual para que sus implementaciones computacionales manejen más de un estímulo condicionado a la vez.

El fin último del presente trabajo, fue aportar conocimientos nuevos en lo referente a los alcances y limitaciones del Modelo DBP, a partir de la exploración de la posibilidad de simular en una computadora la inhibición condicionada con base en el citado modelo.

CONTENIDO

FIGURAS

TABLAS

ALGORITMOS

1. INTRODUCCIÓN

Este apartado introductorio tiene por objetivo presentar al lector una visión general y breve acerca de los tópicos relacionados con el presente trabajo de investigación; aquello que le sirve de sustento teórico y lo que es necesario definir a propósito del proyecto. Comienza con una muy breve descripción de la estructura del documento completo, se incluye la justificación para el trabajo realizado, la pregunta de investigación a la que se pretendió dar respuesta y los objetivos que guiaron el trabajo realizado, para cerrar con una caracterización básica acerca del fenómeno de la inhibición condicionada.

1.1 ESTRUCTURA DEL DOCUMENTO

El Capítulo 2 presenta una introducción a las redes neurales, sus componentes y su funcionamiento básico, bajo las tres formas principales de trabajo: las redes neurales para *aprendizaje no supervisado* y las redes neurales para *aprendizaje supervisado*, y las redes neurales para *aprendizaje por refuerzo*. Se revisa a mediana profundidad un modelo específico de cada una de estas formas de trabajo, en el entendido de que ello posibilitará al lector comprender el modelo DBP en toda su amplitud. Para el caso del aprendizaje supervisado se revisa la *retropropagación del error* como el caso más simple. Para el caso del Aprendizaje no supervisado se revisa el modelo de Mapas Auto-organizativos, en el entendido de que ello proveerá al lector de los elementos básicos para comprender una de las estrategias de actualización que usa el Modelo DBP que es "*el Ganador se lleva todo*". Finalmente, se revisa brevemente el aprendizaje por refuerzo como origen básico del trabajo que realiza el Modelo DBP.

En el Capítulo 3 se revisan los modelos neurales para la simulación de fenómenos conductuales. Esta revisión se hace con base en el supuesto de que ello permitirá al lector ubicar al Modelo DBP dentro de líneas específicas de trabajo, de acuerdo a ciertos criterios, como la concepción de aprendizaje operante-respondiente, a partir de la cual se estructura, el tipo de aprendizaje de red que se plantea y la forma específica de actualización de pesos a partir de la cual trabaja. Se revisan los siguientes modelos: DYSTAL de Alkon, Vogl y Tam; GMADN de Baxter y colaboradores; SLG de Schmajuck, Lam y Gray; el Modelo de Sutton y Barto; el STM de Grossberg y Schmajuck y, finalmente, el Modelo LNM de Kehoe.

El Capítulo 4, detalla el modelo de Donahoe, Burgos y Palmer (1998), revisa las características de los dos submodelos que lo componen: el Submodelo de Red y el Submodelo Neurocomputacional, para cerrar con una descripción acerca de la manera en la que se actualizan los pesos durante el trabajo del Modelo.

El Capítulo 5 presenta el trabajo realizado en la creación de un simulador propio bajo los lineamientos del Modelo DBP, establece las estructuras de datos empleadas y caracteriza los objetos que lo componen, presentando el algoritmo con base en cual funciona cada uno de ellos. El capítulo cierra con una comparación entre los resultados que arroja el simulador propio y los que arroja el simulador SELNET, ante un problema sencillo de adquisición y extinción.

El Capítulo 6 Presenta el trabajo realizado para comprobar que la arquitectura clásica empleada en el modelo DBO no permite a una red que funciona bajo sus principios, manejar varios EC simultáneamente. A lo largo de la exploración realizada se probaron dos estrategias distintas para lograrlo y se analizan los resultados obtenidos partir de 15 simulaciones realizadas.

El Capítulo 7 presenta el trabajo realizado en la búsqueda de una arquitectura que posibilitara el manejo de varios EC sin modificar las restricciones que impone el Modelo DBP. A partir de 12 simulaciones realizadas se logra conformar la propuesta de un par de arquitecturas que permite resolver el problema, analizándose sus ventajas y desventajas.

El Capítulo 8 presenta los resultados tras el intento por simular la inhibición condicionada, con la arquitectura más idónea, de las reportadas en el Capítulo 7. Para realizar un análisis de la efectividad del inhibidor putativo logrado en la simulación de la inhibición condicionada, se analizan 20 simulaciones más, de prueba, que tuvieron como propósito dar cuenta de la sumación y el retardo, que fueron establecidos como pruebas a las que se sometería el inhibidor putativo, desde el diseño mismo del proyecto.

Finalmente, el Capítulo 9 presenta la discusión de los resultados y aprendizajes obtenidos a partir de la realización del presente trabajo, con las limitaciones y potencialidades identificadas.

1.2 JUSTIFICACIÓN

Desde los trabajos de Pavlov en 1927, los investigadores en el campo del aprendizaje animal han estado interesados en la naturaleza de la inhibición condicionada (Cole y otros, 1997). Este interés se ha visto incrementado a lo largo del tiempo, cuando los hallazgos experimentales han arrojado evidencia de que la inhibición condicionada es un fenómeno más complicado de lo que parecía inicialmente en el procedimiento experimental introducido por Pavlov, y que también es mucho más difícil de observar directamente evidencias de que este fenómeno se está presentando y medirlas. Quizá es por ello, que ha sido tema frecuente de investigación en los casi ochenta años de historia del condicionamiento clásico, al grado que ha llegado a ser considerado como uno de los componentes principales de las teorías contemporáneas del aprendizaje (Savastano y otros, 1998).

Por otra parte, las simulaciones computacionales de fenómenos conductuales, y específicamente las que atienden a modelos conductuales, han adquirido especial importancia en los últimos años, acorde con el desarrollo tecnológico y teórico actual. Existen modelos que neurales que han simulado ya el fenómeno de la inhibición condicionada (Schmajuck, 1997). Sin embargo, no se ha determinado a la fecha si el Modelo DBP puede realizar la simulación de este fenómeno, sobre todo, tomando en consideración que el hacerlo, se reviste de importancia especial ya que implica la resolución de un problema que, aún no queda claro si atañe al modelo, o a las formas específicas de implementación que se han dado del mismo: *la imposibilidad actual para que las implementaciones del Modelo DPB puedan manejar más de un estímulo condicionado a la vez.*

Con este trabajo se intenta abonar al conocimiento del Modelo, a partir de la comprobación del hecho de que este problema tiene relación solamente con la arquitectura de redes usada hasta la fecha, sin que en realidad se tratara de un problema de mayor profundidad que pudiera ser resuelto solamente mediante una revisión profunda del modelo.

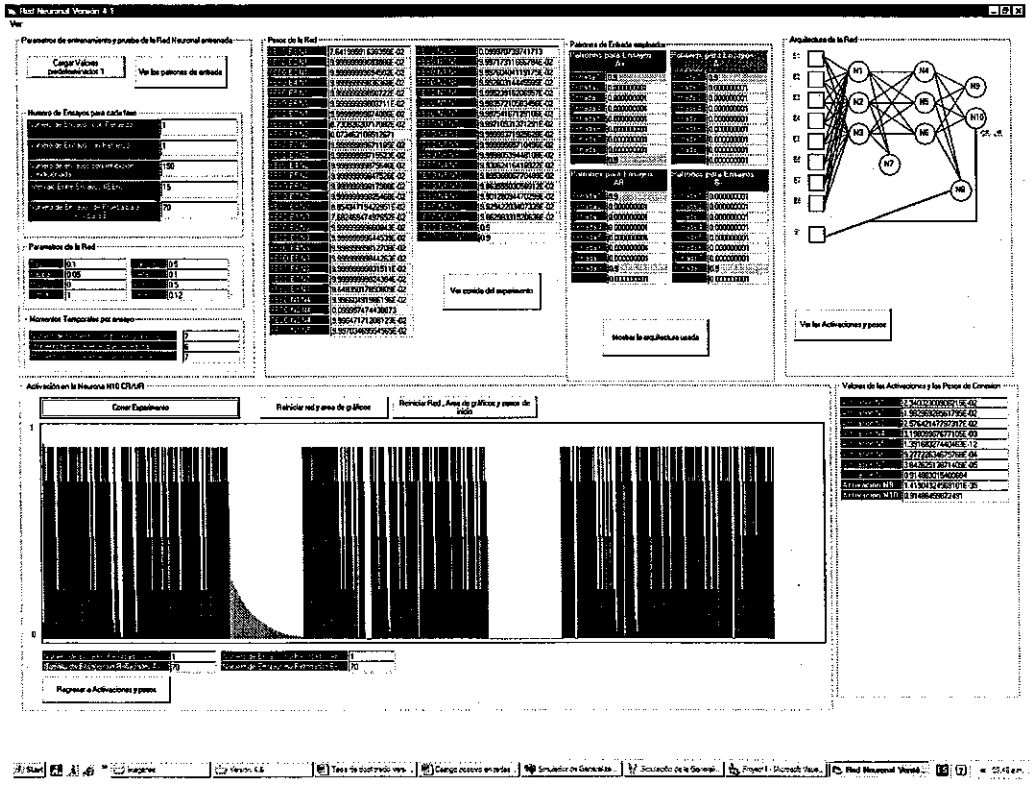
Con este trabajo, se continúa con la exploración de la plausibilidad conductual del Modelo DBP, como parte integral del trabajo del modelado del comportamiento desde una perspectiva bio-conductual. Y asimismo, se pretende abonar también hacia una comprensión más amplia del fenómeno de la inhibición condicionada.

Finalmente, y de forma más general, este trabajo buscó abonar al campo de las simulaciones computacionales con redes neurales y simulaciones computacionales de fenómenos conductuales.

La simulación digital se ha convertido en un recurso cada vez más utilizado en ciencia experimental (Casti, 1997 y Emmeche, 1991), incluyendo por supuesto entre ellas, a las ciencias de la conducta. Estos *experimentos imaginarios*[2], como Dawkins (1982), ha llamado a las simulaciones, sufren constantes ataques por los científicos "clásicos", quienes los acusan de poco realismo, como un argumento en su contra. Queda claro que las simulaciones pueden ser realistas en algunos casos, pero la ventaja precisamente es precisamente, que *pueden no serlo*, sin perder por ello su relevancia. Amén de los casos en los que ser *realista*, independiente de la discusión que la definición del término pueda desatar, es imposible: ¡solo trate de imaginar una simulación realista de la evolución de las especies!

1.2.1 TRABAJOS PREVIOS REALIZADOS POR EL AUTOR

En el 2002, el autor realizó el primer intento para simular la inhibición condicionada, bajo el procedimiento experimental clásico definido por Pavlov (1927). La Figura 1 muestra una imagen de la interfaz de la implementación realizada y la Figura 2 muestra la arquitectura de la red usada.



[2] *Thought Experiments*, en inglés.

Figura 1. Interfaz de la implementación computacional del Modelo DBP para simular inhibición condicionada

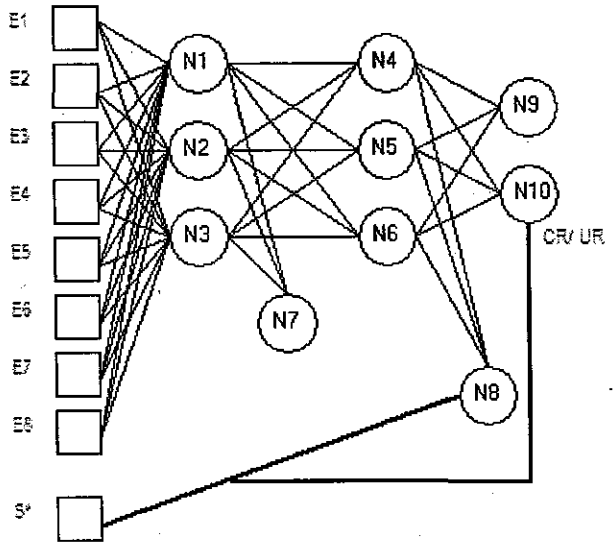


Figura 2. Arquitectura de la Red empleada para simular la inhibición condicionada

Sin embargo, este intento no dio resultado, debido a que no se contempló estrategia alguna para tratar de solucionar la limitación actual de las implementaciones computacionales del Modelo DBP, consistente en la incapacidad para manejar más de un estímulo condicionado al mismo tiempo.

La simulación realizada fue sometida a las pruebas de sumación y retardo (Rescorla, 1969) sin obtener resultados favorables (ver Figuras: 3, 4, 5 y 6).

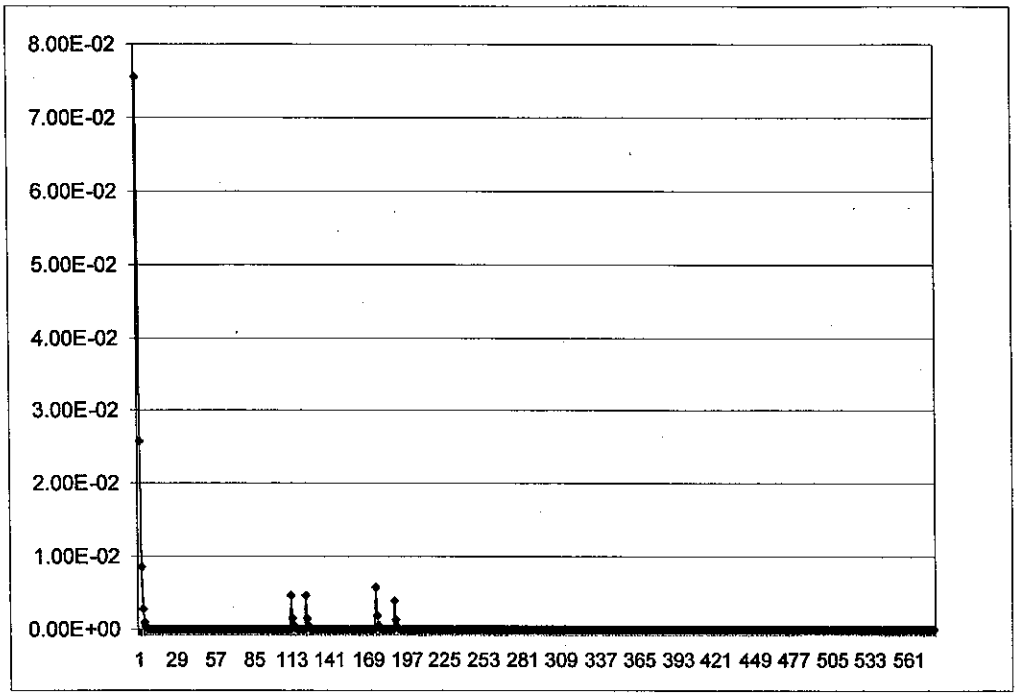


Figura 3. Niveles de activación obtenidos en la neurona OR en el primer intento realizado por el autor para simular la inhibición condicionada

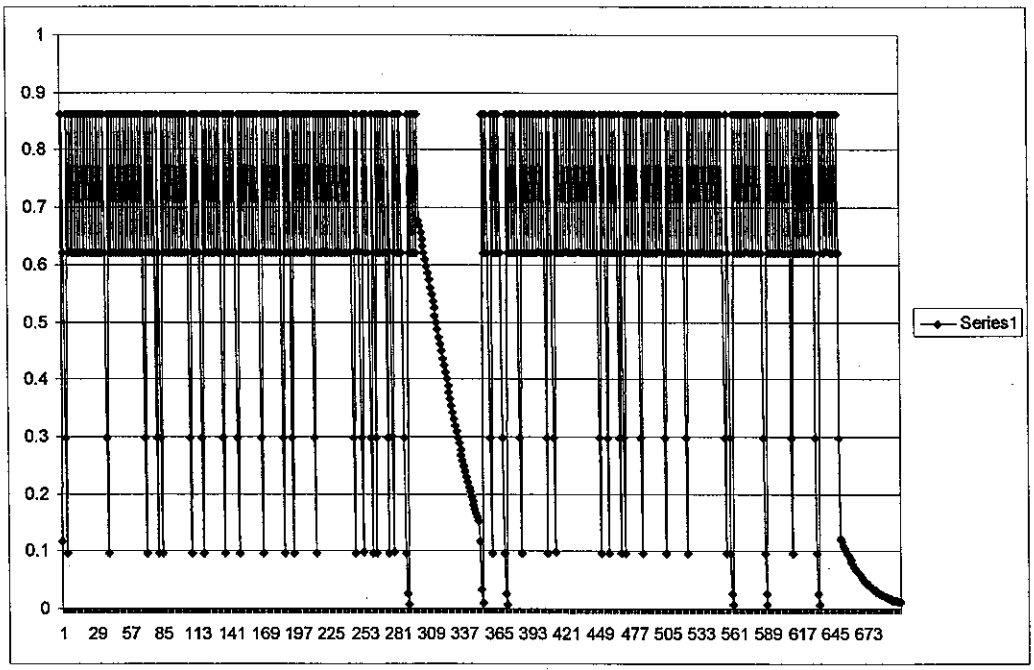


Figura 4. Niveles de activación obtenidos en la Neurona "CR/UR" durante un intento previo por simular la Inhibición condicionada

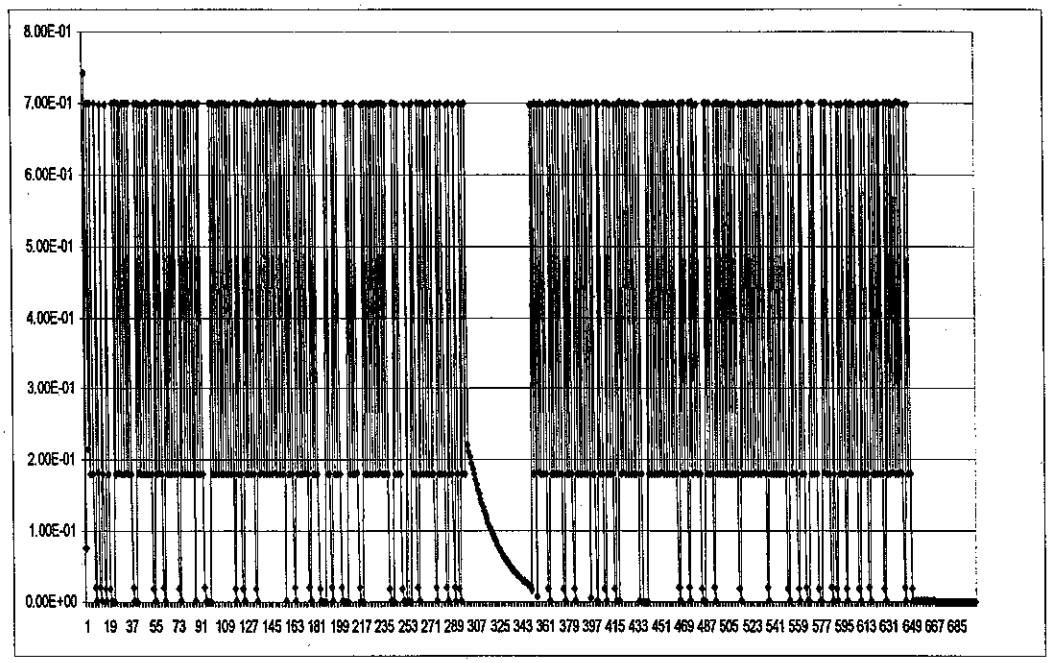


Figura 5. Nivel de Activación en la Neurona CR/UR en el intento previo de simulación de la inhibición condicionada empleando un IEEn = 30.

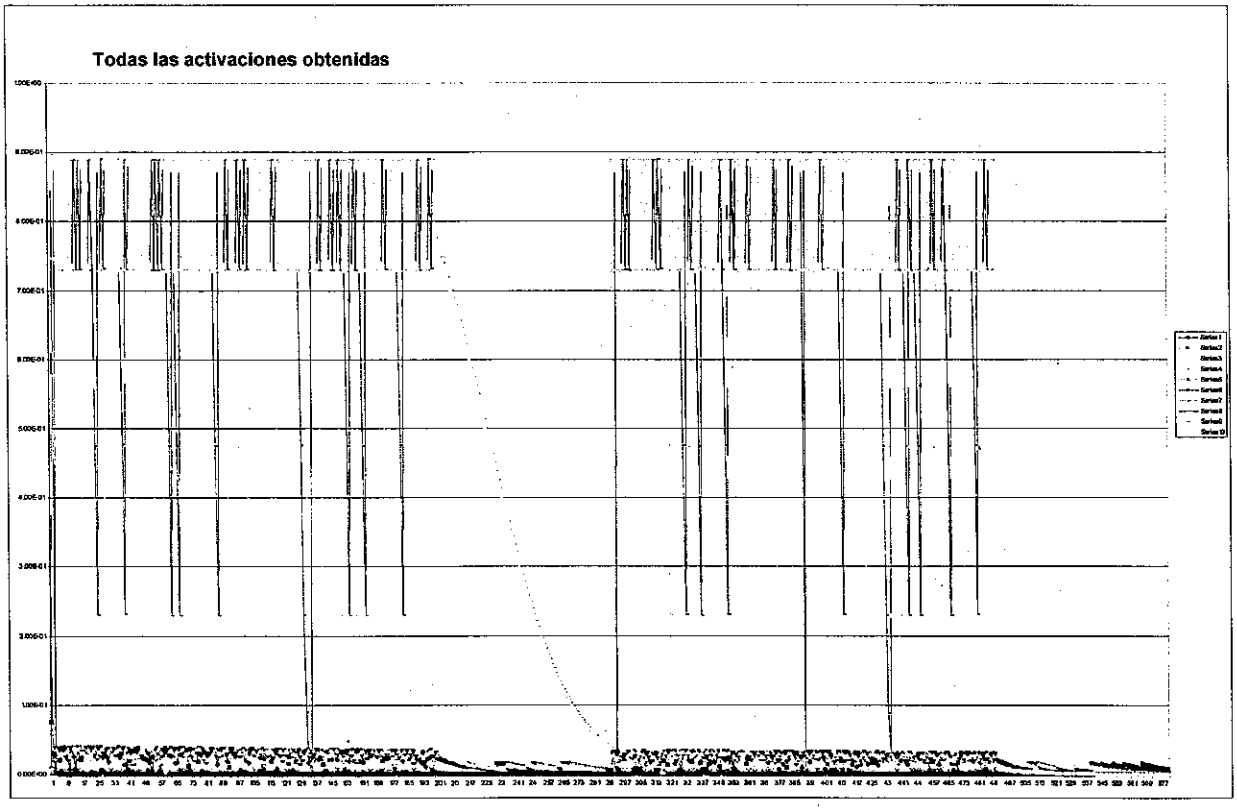


Figura 6. Nivel de activación mostrado por todas las neuronas empleadas en el intento previo de simulación de la inhibición condicionada

El presente trabajo buscó abordar el mismo problema, pero buscando antes, resolver las limitaciones que no fueron contempladas en el intento previo.

1.3. PREGUNTA DE INVESTIGACIÓN

La pregunta básica que guió la realización del presente trabajo es la siguiente:

¿Es posible simular la Inhibición Condicionada con el Modelo Donahoe-Burgos-Palmer?

1.4. OBJETIVOS GENERALES Y ESPECÍFICOS

A continuación se presentan los objetivos que se pretendió lograr con este trabajo de investigación.

1.4.1. OBJETIVOS GENERALES

1.4.1.1. Objetivo general 1

Abonar al conocimiento acerca de los alcances y limitaciones del Modelo DBP explorando la posibilidad de manejar más de un estímulo condicionado simultáneamente.

1.4.1.2. Objetivo general 2

Abonar al conocimiento acerca de los alcances y limitaciones del Modelo DBP explorando la posibilidad de simular la inhibición condicionada mediante el arreglo clásico establecido por Pavlov en 1927: B+\|C+/AC-\|AB+

1.4.2. OBJETIVOS ESPECÍFICOS

A partir de los objetivos generales planteados, se establecieron los siguientes objetivos específicos:

1.4.2.1. Objetivo específico 1

Realizar una implementación computacional del Modelo DBP que corra en una computadora personal, y que sea lo suficientemente flexible como para poder explorar las variables implicadas en la evaluación de una simulación de un fenómeno conductual.

1.4.2.2. Objetivo específico 2

Buscar un tipo de arquitectura que permita a una implementación computacional del Modelo DBP manejar dos o más estímulos incondicionados al mismo tiempo.

1.4.2.3. Objetivo específico 3

Intentar simular efectivamente el fenómeno de inhibición condicionada con el Modelo DBP, a partir de la simulación del procedimiento experimental denominado Inhibición condicionada clásica, establecido por Pavlov (1927), y verificar si el inhibidor condicionado resultante, cumple con las pruebas de sumación y retardo establecidas por Rescorla (1969).

1.5. LA INHIBICIÓN CONDICIONADA

Aunque el Modelo DBP es un modelo que ha demostrado simular tanto condicionamiento respondiente, como condicionamiento operante (Donahoe, Burgos y Palmer, 1993), el presente proyecto de investigación se centra únicamente en la simulación de un fenómeno relacionado con el condicionamiento respondiente, condicionamiento pavloviano o condicionamiento clásico, como también se le conoce, desde que Hilgard, en 1940, acuñó el término para referenciar a aquellos procedimientos de condicionamiento llevados a cabo por Pavlov a mediados de la década de los veinte del siglo pasado (Thompson y otros, 1987).

Las características esenciales de los experimentos de condicionamiento clásico son (Burgos, 1997):

a) Un estímulo incondicionado (EI), que dentro de la situación experimental elicita una respuesta incondicionada (RI) regular y mesurable;

b) Un estímulo condicionado (EC) que es inicialmente neutro, en el sentido de que no evoca la respuesta en observación, pero que como resultado de su apareamiento con el EI llega a evocarla, por lo cual es llamada respuesta condicionada (RC), y

c) Relaciones entre las presentaciones de los EI y los EC, entre las presentaciones de los estímulos condicionados y e incondicionados de una manera específica y controlada, definidas inicialmente en términos de parámetros como el intervalo entre estímulos (IEEs), que es el tiempo transcurrido entre la presentación del estímulo condicionado y el estímulo incondicionado, mismo que representa una medida de contigüidad temporal ambos; y el intervalo entre ensayos (IEEn), que es el tiempo transcurrido entre el final de un ensayo y el inicio del siguiente. Estas dos mediciones dan cuenta de variables temporales, pero también es posible añadir mediciones para dar cuenta de contingencias topográficas, topológicas y contingencias de intensidad, definidas todas ellas de acuerdo a cada caso específico.

El condicionamiento clásico se presenta como un procedimiento completamente distinto del condicionamiento llamado "operante", debido a que en el caso del primero, la ocurrencia de la respuesta condicionada no produce cambio alguno en el procedimiento de condicionamiento. Pero es importante recalcar que, siguiendo el Modelo DBP, la distinción entre "clásico" y "operante" en este documento se hace solamente a nivel de procedimientos, sin asumir en ningún momento que se trata de formas distintas de aprendizaje.

Uno de los conceptos básicos del condicionamiento clásico es la *inhibición.* Pavlov en 1927 usó este término para referenciar a una manifestación de la actividad nerviosa de los organismos (Pavlov, 1927, Curso III, página 44), en la que se incluye también a los reflejos condicionados:

> "... el perro y el experimentador podrían estar aislados en un cuárto experimental, con todas las condiciones permaneciendo constantes. De repente, un factor distractor, podría surgir – un sonido que penetre en el cuarto; algún cambio rápido en la iluminación podría ocurrir a causa del sol cubierto por una nube; o algún líquido podría entrar por debajo de la puerta con un consecuente olor. Si cualquiera de estos estímulos adicionales sucede al tiempo de la aplicación del estímulo condicionado, inevitablemente traerían como consecuencia un debilitamiento o incluso una completa desaparición de la respuesta refleja, dependiendo de la fuerza del estímulo adicional. La interpretación de este caso simple no presenta mucha dificultad; la aparición de cualquier estímulo nuevo inmediatamente evoca el reflejo exploratorio[3] y el animal fija sus órganos receptores apropiados hacia la fuente de disturbio, levantando sus orejas, levantando la mirada hacia la fuente del disturbio y olfateando el aire. El reflejo exploratorio es excitado el reflejo condicionado es, en consecuencia, inhibido..." (Pavlov, 1927, Curso III, página 44).

Pavlov, estableció que la inhibición podía ser originada por dos fuentes distintas, una *directa* o *interna*, y una *indirecta* o *externa*:

- La *Inhibición externa* es un decremento transitorio de una respuesta condicionada por causa de un estímulo extraño, como cuando un sonido fuerte reduce la salivación condicionada a una luz.

- La *Inhibición interna* es un fenómeno que se desarrolla lenta y progresivamente cuando un estímulo condicionado se presenta, repetidas veces, en alguna de las siguientes condiciones: *Extinción experimental*, *Inhibición diferencial*, *inhibición de la demora*, y finalmente, *inhibición condicionada*.

La última de las condiciones enlistada en el segundo punto, la inhibición condicionada, fue llamada de esta manera, a decir de Pavlov, solamente por cuestiones históricas, ya que en realidad parecía más adecuado el término de *Inhibición Diferencial*. Este fenómeno se revestía de un especial interés, para el Fisiólogo Ruso debido a que, a decir de él mismo, demostraba al mismo tiempo la complejidad variada de todos los fenómenos involucrados en la inhibición interna, y demostraba el valor del método experimental como proveedor de medios satisfactorios para el análisis de fenómenos complejos con base en principios generales (Pavlov, 1927).

Pavlov obtenía el fenómeno de la inhibición condicionada en su laboratorio, mediante el siguiente procedimiento:

> "... un estímulo positivo condicionado es establecido firmemente en un perro por medio de las repeticiones usuales con refuerzo. Un nuevo estímulo, entonces es ocasionalmente añadido, y siempre que la combinación es aplicada, hecho que se puede dar en intervalos incluso que se pueden extender entre horas y días, nunca se acompaña del estímulo incondicionado. De esta forma, la combinación se vuelve gradualmente inefectiva, de manera que el estímulo condicionado, cuando se aplica en combinación con el estímulo adicional, pierde su efecto positivo, aún cuando presentado solo, y con constante refuerzo, retiene sus poderes completos..." (Pavlov, 1927, Curso V, página 69 - 70).

---

[3] *Investigatory Reflex*, en inglés.

Formalizando, el procedimiento puede ser descrito de la siguiente manera: Se trata de un esquema de reforzamiento A+/AX-[4], de tres fases, en el que intervienen dos estímulos condicionados: A y X. Inicialmente, en la fase de preparación para el procedimiento, se presentan ensayos repetidos únicamente con el estímulo A, mismos que son reforzados (A+), hasta que la RC esté establecida. En la segunda fase, se da un conjunto de presentaciones ocasionales de un estímulo compuesto por A y por X, estímulo que es nuevo hasta ahora. Estas presentaciones del compuesto no son reforzadas (AX-) y se presentan adicionales a las presentaciones del estímulo A, que si son reforzadas (A+/AX-). Se espera que la RC ante el compuesto AX sufra un decremento casi total, mientras que la de respuestas ante A, se mantiene estable.

Ya en las décadas de los sesenta y setenta del siglo pasado, Rescorla, sugirió que un estímulo debe ser llamado un inhibidor condicionado, si como resultado de la experiencia del organismo con operaciones que implican alguna relación entre este estímulo y un estímulo incondicionado, este estímulo llega a controlar una tendencia opuesta a la que produciría el mismo, como excitador condicionado. Para Rescorla, la inhibición condicionada es un fenómeno relativo al estímulo incondicionado (Rescorla, 1969).

A lo largo de poco más de setenta años de investigación acerca de la Inhibición Condicionada, se ha encontrado que el fenómeno puede ser producido por los siguientes procedimientos experimentales disparadores (Savastano y otros, 1999):

a) El procedimiento de entrenamiento de la inhibición condicionada definido por Pavlov (1927). Bajo este procedimiento, el arreglo es el ya mencionado párrafos antes: A+/AX-.

b) El procedimiento de entrenamiento de la inhibición condicionada diferencial, en el que arreglo es A+/X-.

c) El procedimiento de entrenamiento de retrógrado, cuyo arreglo es +/X-[5].

d) El procedimiento de condicionamiento anterógrado, cuyo arreglo es +X.

e) El procedimiento de condicionamiento de demora larga, cuyo arreglo es X+ en donde X tiene una larga duración. Este es el caso de la inhibición de la demora, que es un procedimiento establecido también por Pavlov en 1927.

f) El procedimiento de condicionamiento huella, cuyo esquema es X----> +.

g) Adicionalmente, se tiene que considerar que la extinción de un excitador le da a ese estímulo cualidades inhibitorias. Siendo el esquema de reforzamiento el siguiente: X+ seguido de X-.

h) La inhibición condicionada de segundo orden, que es producida a través de aparaos con un inhibidor de primer orden y cuyo esquema de reforzamiento es el siguiente: A+/AX-/XB-.

---

[4] Esta forma de representar esquemas de reforzamiento es común en la literatura especializada. Las literales mayúsculas A y X representan estímulos condicionados distintos, que pueden presentarse en forma aislada, como en el caso de A, o en forma compuesta, como en el caso de AX, los símbolos + y – denotan cuando la presentación del EC es reforzada (es decir, seguida por el EI) o no, respectivamente.

[5] El símbolo + en este caso se refiere a la ocurrencia del EI sin el EC, la cual supone que el EI se aparea solo con el contexto

Dada esta variedad de procedimientos experimentales que producen inhibición condicionada, la pregunta lógica es ¿qué es lo que tienen todos estos procedimientos en común que produce el fenómeno? Uno de los grandes problemas en torno a la Inhibición condicionada, ha sido la definición operacional de "inhibidor condicionado".

Savastano y otros (1999), ofrecen una propuesta de definición de la inhibición con base en el supuesto de que existe un elemento subyacente en común a estos procedimientos anteriormente enlistados. En casi todos los casos, el inhibidor candidato se presenta sin refuerzo en presencia de otro estímulo que fue previamente reforzado. Entonces, un inhibidor condicionado puede ser definido operacionalmente como:

> "...un estímulo no reforzado que ocurre en proximidad con otro estímulo que a su vez ha sido apareado con un reforzador..." (Savastano y otros, 1999, página 104)

Y ofrecen asimismo, una explicación acerca de la forma en la cual esta definición operacional se aplica a los fenómenos listados anteriormente, excepto en el caso de la inhibición de segundo orden:

a) Para el caso de la inhibición condicionada pavloviana, (A+/AX-), el excitador A, es aquel estímulo que fue previamente reforzado y que ahora se presenta en un compuesto no reforzado con el estímulo X.

b) Para el caso de la inhibición por contingencias negativas (+/X-), El o los "El" no señalados convierten al contexto en estímulo excitatorio. Entonces, cuando el inhibidor X es presentado, el El no es presentado, a pesar de que se sostiene la presencia de ese contexto excitatorio.

c) Para el caso de la inhibición diferencial (A+/X-), el excitador A no se presenta simultáneamente con X. Sin embargo, se ha demostrado que el excitador crítico en este procedimiento no es A, sino el contexto experimental, el cual ha sido apareado con el estímulo incondicionado en ausencia de X y está presente sin reforzamiento durante las presentaciones de X.

Tanto para el caso de la inhibición diferencial como para el de las contingencias negativas, dicen Savastano y otros (*óp. cit*), el contexto excitatorio putativo típicamente no elicita una respuesta condicionada, a pesar de que las pruebas de sumación y de retardo puedan indicar que el entrenamiento de la inhibición condicionada ha tenido éxito. Sin embargo, esto parece reflejar más bien un problema de medición de estímulos prolongados en el tiempo, como lo son los contextuales. Sin hacer énfasis en ello, la definición actual operacional solamente especifica que el estímulo debe ser apareado con el reforzador, no que necesariamente se demuestre que éste es excitatorio.

d) Para el caso del condicionamiento retrógrado (+X) el contexto es apareado con el estímulo incondicionado inmediatamente antes de la presentación de X. En consecuencia, el contexto puede ser visto como excitatorio, o alternativamente, el estímulo incondicionado, en sí mismo, habiendo sido presentado, puede actuar como señal de un estímulo incondicionado posterior que no es presentado inmediatamente después de la presentación de X. Es notable que la inhibición conseguida con el condicionamiento anterógrado, es mucho más fuerte cuando un segundo estímulo condicionado señala al estímulo incondicionado en el apareo US--->CS de

la inhibición retrograda. En este caso, presumiblemente el segundo CS es el estímulo que es apareado a su vez con el reforzador.

e) Para el caso de la inhibición de la demora (X+ siendo X de larga duración), la presentación de X por sí misma señaliza la ocurrencia del Estímulo Incondicionado, pero entonces, el Estímulo incondicionado no ocurre inmediatamente. Parece, entonces que se trata de un ejemplo de un Estímulo condicionado que sirve tanto como inhibidor y como excitador, en función de la duración del Estímulo Condicionado. Esto es, los segmentos posteriores del Estímulo Condicionado son excitatorios y actúan como el otro estímulo que es apareado con el reforzador durante los segmentos tempranos del Estímulo Condicionado Inhibitorio (Savastano y otros, *óp. cit*).

f) La inhibición producida por el condicionamiento huella (X ----> +) presumiblemente es altamente similar a la inhibición de la demora, con el decaimiento de la huella del Estímulo Condicionado que es más excitatoria que la representación inicial del Estímulo Condicionado que es activada por la presentación del estímulo condicionado.

g) Para el caso de la extinción (X+, seguido de X-), X es en sí mismo, el estímulo apareado previamente con el Estímulo Incondicionado y en la presencia del cual X ahora no es reforzado. La aplicación exacta de esta definición operacional propuesta por Savastano y otros (*óp. cit*), en este caso es menos clara e invita a estudios adicionales acerca de las propiedades inhibitorias de un Estímulo Condicionado extinguido.

El segundo gran problema relacionado con la *inhibición condicionada* es su medición. Aunque ha habido diversos intentos de medir este fenómeno, Savastano y otros (*óp. cit*), refieren que es hasta 1969 cuando Robert Rescorla ofrece una definición conductual de la inhibición condicionada, ampliamente aceptada y que permite, lograr una medición del fenómeno sin que se pueda confundir con otros.

Rescorla, propuso que un estímulo puede ser considerado como inhibitorio, solamente si:

a) Reduce la respuesta ante un estímulo excitatorio cuando el excitador y el inhibidor putativo son presentados en un compuesto. Entonces se dice que pasa la *prueba de sumación*, y

b) Requiere de más apareos con el estímulo incondicionado para convertirse en un excitador condicionado, que los que necesitaría el estímulo si no hubiese sido condicionado como inhibidor. Entonces se dice que pasa la *prueba de retardo* (Rescorla, 1969).

Rescorla (1969) propuso estas pruebas como una estrategia para evitar caer en el error de interpretar como *inhibición condicionada* lo que podría deberse solamente a cambios en el nivel de atención a un estímulo. Se basa en el supuesto de que no se puede dar un incremento en la atención y un decremento al mismo tiempo.

Aunque, no todos los teóricos comparten la idea de la existencia de algo a lo que podamos llamar *atención al estímulo*, con toda la carga que el término pueda implicar, y algunos otros prefieran hablar en términos *conductas de orientación hacia el estímulo*, buscando no hacer alusión a algún tipo de proceso interno, las pruebas de sumación y de retardo propuestas por Rescorla se han aceptado

durante varias décadas como pruebas efectivas para diagnosticar a nivel conductual la inhibición condicionada.

Recientemente se ha cuestionado la suficiencia del paso de estas dos pruebas como evidencia suficiente para la inhibición condicionada. Papini y Bitterman (1993) defienden la idea de que pasar las dos pruebas es usualmente inadecuado para certificar que un estímulo es inhibitorio, debido a la gran cantidad de explicaciones alternativas que existen para el paso de estas dos pruebas. Ellos argumentan que en la literatura podemos encontrar gran cantidad de estudios en los que se pretende demostrar la inhibición condicionada, y que son metodológicamente muy cuestionables, incluso, que hay otros estudios que es imposible replicar con los parámetros de tratamiento que se reportan. Estos autores sostienen que la prueba de retardo, si es correctamente realizada, puede ofrecer evidencia suficiente para hablar de inhibición condicionada.

Sin embargo, Cole y otros (1997) sostienen que sería imprudente aceptar la prueba del retardo solamente, como evidencia, sin estudios con mayor profundidad que confirmen los hallazgos reportados por Papini y Bitterman, por lo cual, es preferible continuar considerando el paso de las dos pruebas como evidencia de la inhibición condicionada.

Asimismo, desde mediados de la década de los setentas, del siglo pasado, ha sido propuesta una nueva manera de establecer evidencias de la inhibición condicionada como válidas, a partir de las conductas de alejamiento y de acercamiento observadas en pichones en automoldeamiento con dos palancas: una para la entrega de alimento y otra para la omisión del mismo (Ver: Wasserman, Franklin y Hearst, 1974). Sin embargo, a decir de Cole y otros, (1997) esta búsqueda de nuevas formas de medir la inhibición condicionada es encomiable, pero inefectiva para explicar muchas de las manifestaciones del fenómeno (Cole y otros, 1997, página 340).

Savastano y otros, (1999) dicen que parece existir un consenso general acerca de que la medición de la inhibición condicionada, parece ser más aplicable para identificarla que para cuantificarla, debido a dos problemas asociados íntimamente con su medición:

Uno es la sumación de los atributos excitatorios de los estímulos a utilizar, que impide conocer el valor puro de la inhibición, es decir, impide conocer cuánto de la inhibición condicionada es realmente debido a la preparación experimental y cuánto a las propiedades mismas de los estímulos a emplear. Savastano y otros, (1999) explican que una posible solución al problema de la excitación concurrente es la extinción de los atributos excitatorios del estímulo a usar, con el fin de revelar sus capacidades inhibitorias puras, ello se propone con base en el hallazgo de que presentaciones solas del estímulo condicionado (X-) extinguen la excitación, pero no la inhibición y que por ello puede esperarse, entonces, que la extinción de la excitación deje intacta la inhibición de manera que pueda ser evaluada sin ningún enmascaramiento de excitación. Sin embargo, también explican estos autores, que un problema al aplicar esta técnica para eliminar el enmascaramiento en las pruebas de sumación y de retardo, es que no existen disponibles medios obvios para determinar si la extinción de la excitación ha sido realizada completamente, o si solamente se ha llevado a un nivel inferior al umbral de respuesta excitatoria y sigue siendo capaz de atenuar el control excitatorio de estímulos (Savastano y otros, 1999). Y el segundo y más problemático, es que la extinción de la excitación es considerada como productora de inhibición y que esta nueva inhibición creada es capaz de ser sumada a cualquier inhibición existente antes de la extinción, con lo que se puede llegar a confundir al intentar medir la inhibición anterior sola.

2. CONEXIONISMO Y REDES NEURALES

Aunque el Conexionismo se puede rastrear hasta la antigua Grecia y un poco más recientemente, en los trabajos de René Descartes o en los de los histologistas de principios del siglo XIX, la primera red neural, fue presentada por Alexander Bain en 1873 en su libro *Mind and Body. The Theories of their relation*, (Bain, 1873, citado por Olmsted, 1999). Su trabajo fue altamente influenciado por los hallazgos en neuroanatomía, que entre 1858 y 1863 permitieron la identificación microscópica de las fibras nerviosas e incluso de algunos tipos de axones, gracias a las nuevas técnicas de tinción como la basada en el uso del azul de metileno y de la hematoxilina.

Con base en estas ideas, Bain ideó un sistema de umbrales lógicos que servían para conectar ideas, bajo el supuesto de que dada la estructura estática de la anatomía cerebral, un mismo nervio debía participar en la estimulación de varios, en momentos distintos. A este tipo de relación Bain le llamó *cooperación* (Olmsted, 1999)

La Figura 7 muestra un circuito nervioso con base en umbrales, en el que las conexiones representan ideas que se combinan y cooperan para producir una respuesta específica ante un estímulo dado:

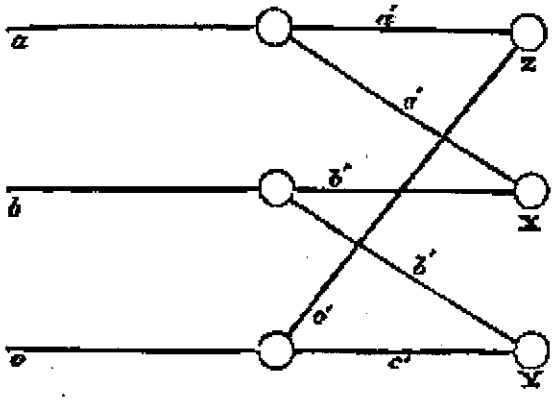


Figura 7. Circuito de ideas cooperantes de Bain (tomado de Olmsted, 1999).

Bain desarrolló más la idea de la resistencia a la estimulación, estableciendo una Teoría de la Adaptación que establecía que la distancia entre nodos en una ruta de estimulación y a la fuerza de la corriente nerviosa eran los dos elementos determinantes del grado de resistencia a la estimulación en partes específica del circuito (ver Figura 8):

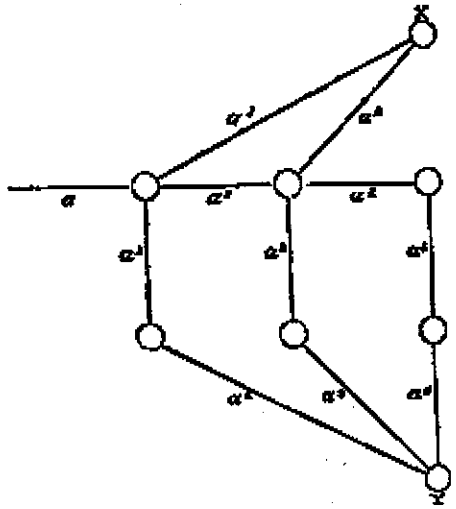


Figura 8. Circuito de ideas cooperantes de Bain con un nodo más resistente ocasionado por una distancia mayor que los del resto del circuito (tomado de Olmsted, 1999).

En 1890 William James rescató la teoría adaptativa de Bain. En esos momentos, el cerebro era concebido como una estructura de redes neurales conectadas al azar, las cuales propagaban o reverberaban corrientes eléctricas en todas las direcciones en una forma análoga a una red de cables metálicos. James propuso que las acciones y los pensamientos se producían como resultado de estas corrientes que fluían en regiones cerebrales (como procesos cerebrales) que tenían un exceso de carga, hacia regiones cerebrales que tenían un déficit de carga eléctrica. La intensidad de los pensamientos y las acciones eran proporcionales a las diferencias de cargas entre regiones. Estas corrientes eléctricas reverberantes fueron llamadas *engramas*. El aprendizaje consistía, según James, en el cambio de las rutas que seguían las corrientes de cargas, hacia nuevas formas, mediante el uso de tres reglas: primero, si los procesos se activaban juntos o en sucesión temporal inmediata, existía la tendencia a la excitación de uno al otro (James, 1890, p. 566); segundo, las corrientes nerviosas se propagan más fácilmente por las rutas de conducción que más transitadas han sido previamente (James, 1890, p. 563) y, tercero, las rutas que no tienden a usarse producen olvido.

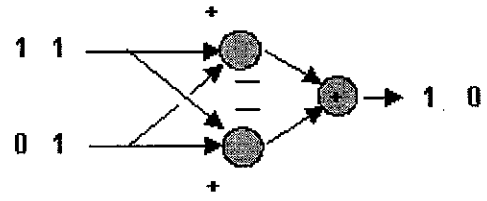


Figura 9. Circuito de Rashevsky para la operación lógica de la disyunción exclusiva (tomado de Olmsted, 1999).

En 1938 Nicolás Rashevsky propuso que el cerebro se organiza con base en operaciones binarias, dado que los potenciales de acción pueden ser vistos como existentes o no existentes en un momento determinado. Rashevsky propuso redes como la que se presenta en la Figura 9, que con base en el uso solamente de operaciones de adición y de substracción, podía llegar a simular el comportamiento de la operación lógica de la disyunción exclusiva. Uno de los objetivos de Rashevsky era llegar a

explicar cómo el comportamiento de los nervios y las redes de nervios podían ser relacionados con procesos psicológicos como el condicionamiento pavloviano (Aizawa, 2002).

En la primera parte de la década de los cuarenta del siglo pasado, Warren McCulloch y Walter Pitts (1943) sentaron las bases para tratar al cerebro como un organismo computacional, basándose en cinco supuestos fundamentales de la neurofisiología de los cuarenta: 1) La transmisión neural es un proceso regido por la *Ley del Todo o nada*. Es decir, las neuronas son binarias en lo que respecta a su actividad: o transmiten, o no lo hacen. No hay puntos intermedios de transmisión; 2) Para la excitación de una neurona se hace necesario un número fijo de sinapsis en un periodo de adición latente. Las neuronas cuentan con un umbral de excitación fijo, que solamente es rebasado cuando todas las sinapsis que en un tiempo determinado entran a la neurona, producen un estímulo total suficiente para sobrepasar este umbral; 3) No hay retardos en la actividad del sistema nervioso ajenos al mismo retardo sináptico; 4) Cuando la sinapsis es inhibitoria, la actividad de la neurona implicada se suspende en un periodo de tiempo; y, 5) Las interconexiones entre neuronas no cambian con el tiempo.

Con base en estos cinco supuestos, McCulloch y Pitts idearon un modelo computacional basado en redes de elementos sencillos que podían emular este supuesto funcionamiento del sistema nervioso. Las redes configuradas eran capaces de realizar operaciones lógicas complejas con base en la combinación de módulos sinápticos que realizaban sólo tres operaciones básicas: la conjunción, la negación y la disyunción.

En 1943 Warren McCulloch y Walter Pitts se dieron cuenta de que la consecuencia natural de que el modelo estándar de umbral de neuronas en combinación con los potenciales de acción binarios, producían otro tipo de lógica, llamada lógica de umbrales.

La lógica de umbrales se expandió al añadírsele más líneas de entrada de manera que la salida del nodo de sumación se convirtió en analógica, debido a que parecía adecuarse más con el modelo estándar de neurona. Aún usando más de dos entradas, las operaciones dejaron de ser lógicas, esto es, dejaron de definir el conjunto básico de clasificadores de patrones usado como bloques de construcción en circuitos más grandes para la clasificación de patrones basados en la distribución de los mismos. En vez de eso, estas operaciones se convirtieron en clasificadores de parámetros que clasificaban los patrones de entrada con base en un parámetro de magnitud de la señal, dado por la sumación de las entradas binarias.

La existencia de valores análogos permitió que las operaciones se volvieran adaptativas usando factores de multiplicación llamados pesos. La manera exacta en la que esta adaptabilidad fue usada, es lo que definió los diferentes tipos de redes neurales de aquí en adelante. Aún, la modificación de un parámetro individual no puede caracterizar completamente la distribución (el patrón) de una entrada de una operación.

El trabajo de McCulloch y Pitts adquirió mayor relevancia cuando a estas redes artificiales de neuronas se les añadió la capacidad de aprender con base en las ideas propuestas por Donald Hebb (1949) de la Universidad McGill en Canadá:

> "...cuando un axón de una neurona A está lo suficientemente próximo para excitar a una neurona B, o cuando ésta toma parte frecuentemente en la excitación para producir su disparo, se da un proceso de crecimiento o cambio de metabolismo en una o ambas neuronas, de modo que la eficiencia de A para producir el disparo de B se ve incrementada" [Hebb, 1949, pág. 50].

Este hecho le sugirió a Hebb la posibilidad de haber encontrado el sustrato neurofisiológico del aprendizaje:

> "... se debe asumir que este cambio estructural a nivel sináptico sucede en situaciones de aprendizaje [...] ya que varias neuronas que mantienen actividad repetidamente terminan asociadas..." [Hebb, 1949, pág. 52].

Aunque en la actualidad se ha descubierto que los supuestos en los que McCulloch y Pitts fundamentaron su trabajo, así como la idea de Hebb acerca del aprendizaje, no son del todo ciertos, sí funcionaron como base moderna de las redes neurales.

En las décadas de los 50´s, 60´s y 70´s los avances en redes neurales se ven frenados casi por completo, debido a dos factores principalmente: primero, al éxito comercial que tuvieron las aplicaciones derivadas de la visión simbólica de la mente, como los sistemas expertos y los solucionadores generales de problemas, y segundo, debido al ya famoso artículo de Seymour Papert y Marvin Minsky de 1969 (Minsky y Papert, 1980) en el que se demostraba que los perceptrones de Rosenblatt (Rosenblatt, 1958) eran incapaces de resolver problemas cuyo espacio de posibles soluciones no era separable linealmente.

En 1984 el Conexionismo resurge al mundo de las ciencias cognoscitivas con una nueva visión, gracias a la comprobación hecha por los mismos Minsky y Papert, de que las redes eran capaces de resolver problemas complejos con algunas modificaciones a las arquitecturas originales descritas y atacadas en su artículo del 69.

Ahora el Conexionismo es definido, como un enfoque computacional para modelar el cerebro, que se basa en la interconexión de un conjunto de unidades simples para producir comportamientos complejos. Bajo este enfoque, los procesos cognoscitivos residen y se implementan en las células del sistema nervioso. Actualmente se ve como una de las opciones más fuertes por parte de los filósofos de la mente al ser propuesto como la mejor manera de modelar la cognición humana, gracias a las propiedades que emergen de esta organización de elementos de bajo nivel (Pagels, 1991)

Acorde con McClelland, Rummelhart y Hinton (1994) los modelos conexionistas pueden ser clasificados por la forma específica en que manejan su representación, en dos categorías: los modelos conexionistas distribuidos y los localistas. Las representaciones distribuidas son vectores en un espacio de estados representacionales que son procesados en muchos nodos en una red. Los modelos localistas por otra parte, usan nodos individuales para representar un concepto completo.

En general, las representaciones distribuidas son en realidad a nivel neurológico más realistas que las localistas, aunque este realismo es sumamente limitado, debido a la abstracción misma que el modelo implica. Muchas de las críticas a los modelos conexionistas que provienen de los neurocientíficos se basan precisamente en la falta de realismo neurológico de las redes

conexionistas ya que tienen poca recursión, demasiada inhibición, algoritmos de aprendizaje poco realistas, funciones de transferencia simplistas, y no hay hasta el momento algún modelo que contenga elementos que permitan establecer alguna analogía entre ellos y el gran numero de neurotransmisores y de hormonas que afectan a la cognición humana.

El Conexionismo ha originado a la *Computación Neural* o *Neurocomputación* (Hecht-Nielsen, 1990) como una alternativa a la computación simbólica (aunque, como ya se ha expuesto en este mismo documento, es históricamente es anterior a ésta) como un medio para obtener aplicaciones prácticas y funcionales, que cuenten con las propiedades y ventajas que el procesamiento paralelo y distribuido ofrece (McClelland, Rummelhart y Hinton, 1994). Los sistemas computacionales realizados con base en estos modelos son conocidos como *Redes Neurales Artificiales* (RNA)[6] o, simplemente *redes neurales* (RN).

En las redes neurales el proceso de la información se aleja completamente de las formas tradicionales de proceso en Inteligencia Artificial (la vieja inteligencia artificial y sus derivados), es por ello que hay autores consideran a la computación neural como una disciplina *aparte* de la Inteligencia Artificial (Turban, 1992). Sin embargo, debido a la coincidencia entre las redes neurales y la inteligencia artificial en lo referente a algunos de los fines últimos y muchos de los elementos de trabajo utilizados, se considera a la Neurocomputación como una *disciplina estrechamente asociada* (Haykin, 1994 y Churchland, 1993), o incluso, como una línea de trabajo más en la inteligencia artificial bajo una concepción extendida de la disciplina (Skapura, 1995; Minsky, 1991; y, Freeman y Skapura, 1991)[7].

Las características que permiten establecer diferencias entre las redes neurales y otras líneas en inteligencia artificial son las siguientes:

a) La representación del conocimiento. En las redes neurales el conocimiento es representado por medio de valores numéricos distribuidos en la estructura interna de la red, sin implicar una estructura sintáctica compleja. La configuración completa de estados en un sistema puede representarse por medio de vectores y el trabajo de la red puede verse como una simple relación entre espacios de vectores (uno de entrada y uno de salida). Este tipo de representación ha llevado la reformulación de los problemas como la identificación de patrones o el aprendizaje a niveles matemáticamente tan sencillos como encontrar el vector más adecuado (Minsky, 1991).

b) El proceso paralelo. El alto grado de interconexión entre las unidades de procesamiento implica un grado elevado de resistencia a errores. Esto se ve reflejado en la robustez del sistema. Igualmente, la posibilidad de contar con muchas unidades de proceso facilita la realización de tareas complejas que mediante un proceso secuencial necesitarían una mayor inversión de recursos.

---

[6] A menudo se usa también el término *Redes neurales* (o *Neural networks,* en inglés) para hacer referencia a la neuro-computación en general.

[7] La concepción original de inteligencia artificial consideraba solamente al proceso simbólico de la información, por lo que, desde la definición misma, las redes neurales y otros enfoques de proceso no simbólico como la computación genética, quedan excluidos. Sin embargo, en los autores más recientes se puede identificar una concepción *ampliada* o *extendida* que incluye ya a otras líneas de trabajo no tradicionales.

c) El proceso distribuido. Cada unidad de proceso realiza tareas muy sencillas, que no van más allá de la suma de entradas y pesos de conexión, y la redefinición del resultado con base en una función de transferencia. La complejidad del proceso radica más en la arquitectura y funcionamiento globales del sistema, que en la estructura y funcionamiento de sus unidades por separado.

Parece no haber una definición universal acerca de lo que son las redes neurales. Sin embargo, con base en su arquitectura y función, se podría decir que las redes neurales artificiales son dispositivos computacionales compuestos a partir de la interconexión de un cierto número de unidades simples, cuya función principal es la transmisión de información numérica. Los canales de transmisión de la información, las reglas para que esta transmisión se lleve a cabo, e incluso la misma arquitectura específica de la red, dependen de la función para la cual la red ha sido diseñada y entrenada; función relacionada, principalmente, con la identificación de patrones.

2.1. ELEMENTOS Y ORGANIZACIÓN DE UNA RED NEURAL

Existen muchos tipos diferentes de redes neurales, y todos ellos implican la integración de diversos elementos y formas de organización. Sin embargo, todos los tipos contemplan unidades de proceso como la que se describe en el apartado siguiente.

2.1.1. UNIDADES DE PROCESO

Las unidades de proceso en una red neural se llaman *neuronas artificiales*, *nodos*, o simplemente *neuronas*.

Cada unidad de proceso en una red neural se caracteriza por:

a) Un nivel de actividad. El nivel de actividad de la unidad de proceso es binario y representa el estado de polarización de la neurona biológica,

b) Un valor de salida. El valor de salida de las unidades de proceso en una red neural representa el disparo de la neurona, que se propagará a otras neuronas a través de las conexiones que ésta mantenga.

c) Un conjunto de conexiones de entrada. Este conjunto de conexiones representa al conjunto de sinapsis entrantes en una neurona biológica y en las unidades de proceso de las redes neurales está compuesto por el total de conexiones que el nodo recibe de otros nodos. Cada conexión tiene asociado un peso o *fuerza de sinapsis* que determina el efecto de la entrada, estos pesos pueden ser positivos (para una sinapsis excitadora) o negativos (para una sinapsis inhibidora).

d) Un umbral de disparo, que representa el grado de resistencia al disparo, que las neuronas biológicas presentan y en el caso de las unidades de proceso se expresa como un valor.

e) Un conjunto de conexiones de salida, que representa a las sinapsis salientes de una neurona biológica, y en redes neurales consiste en las conexiones que de una neurona salen hacia otras. Cada conexión de salida tiene asociado también un peso o *fuerza de sinapsis* que determina el

efecto de la salida. Al igual que en las conexiones de entrada, estos pesos pueden ser positivos (para una sinapsis excitadora) o negativos (para una sinapsis inhibidora).

f) Una función de salida o también conocida como función de activación. La interacción de las características anteriormente descritas permite establecer el funcionamiento de la unidad de proceso con base en una función de salida a partir de la cual se calcula el disparo o no que la unidad realizará. Esta función de salida puede ser de varios tipos según el modelo de la red que se esté implementando. En la Figura 10 se muestran las tres funciones no lineales más usadas, en el entendido de que sólo son las más usadas, pero no las únicas (Lipmann, 1987).

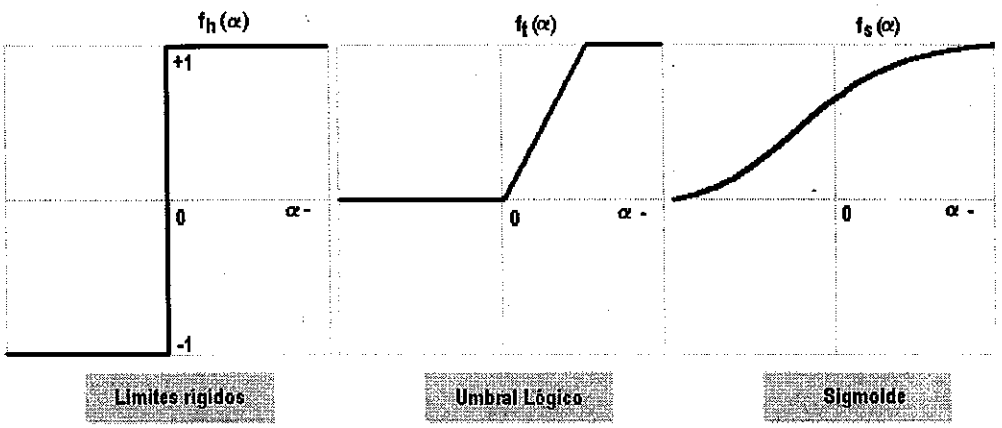


Figura 10. Tres funciones no lineares de las más usadas en redes neurales.

Los elementos que caracterizan a las unidades de procesamiento, listados en la página anterior, se representan matemáticamente por medio de números reales (ver ejemplo en la Figura 11).

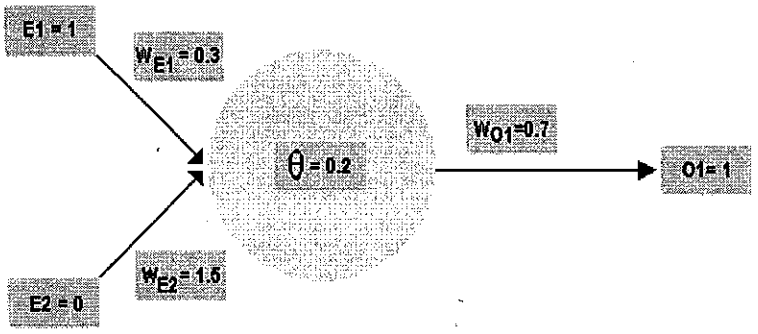


Figura 11. Unidad de proceso con valores de ejemplo.

En la Figura 11, se muestra:

- Una neurona cuyo conjunto de conexiones de entrada está compuesto por: $E1$ y $E2$
- La fuerza de sinapsis o peso de conexión para $E1$ es $W_{E1} = 0.3$ y para $E2$ es $W_{E2} = 1.5$
- En ambos casos ($E1$ y $E2$) se trata de sinapsis excitadoras.

- El conjunto de conexiones de salida está compuesto por una sola salida que en este caso se ha nombrado $O1$.
- La fuerza de sinapsis o peso de conexión para $O1$ es $W_{O1} = 0.7$
- El umbral de disparo está dado por $\theta = 0.2$
- Dado que la entrada $E2$ es la única entrada activa a la neurona, el efecto total de la entrada es igual a $W_{E2} = 1.5$, que al ser mayor que el umbral implica que la neurona dispare. Es por ello que el nivel de actividad de la neurona de ejemplo es $= 1$.
- Al ser el nivel de actividad de la neurona $= 1$, se da un disparo cuya fuerza es $W_{O1} = 0.7$.
- La función de salida para la neurona de este ejemplo es la que se muestra en la ecuación 1 y en la Figura 12:

$$O1 = \begin{cases} 1 & si \quad \sum_{i=1}^{n} E_i \cdot WE_i \geq \theta \\ 0 & de\ cualquier\ otra\ manera \end{cases} \qquad (1)$$

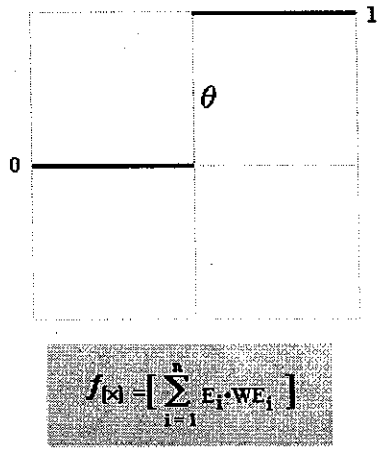


Figura 12. Función de salida tipo *escalón*

2.1.2. CAPAS DE UNIDADES DE PROCESO

Las redes neurales típicamente organizan sus unidades de proceso en capas. Estas capas están compuestas de un número de nodos interconectados que usualmente emplean funciones de activación iguales.

Las entradas se presentan a la red neural a través de la primera capa o capa de entrada[8], la cual comunica con una o más capas ocultas de la red en donde el proceso de la información de entrada se realiza con base en un sistema de pesos de conexión. Las capas ocultas o intermedias, están conectadas a una capa de salida, que es el medio a través del cual la red arroja resultados (ver Figura 13).

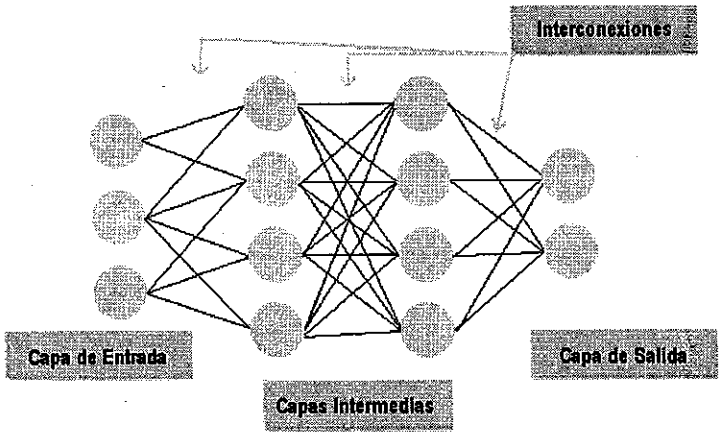


Figura 13. Capas e interconexiones en una red neural típica.

Las redes neurales utilizan una regla de aprendizaje, que modifica los pesos de las conexiones entre las células de acuerdo a los patrones de entrada que le son presentados y sus salidas correspondientes. Así, estos dispositivos aprenden de la manera en que se supone que lo hacen las redes neurales biológicas: reforzando aquellas conexiones que llevan al éxito en el desempeño de una tarea.

2.1.3. TIPOS DE ARQUITECTURA MÁS COMUNES

La manera en la que las neuronas de una red neural se estructuran se relaciona estrechamente con el modelo de aprendizaje a partir del cual la red debe funcionar. En general, se pueden identificar varios tipos de arquitectura de red, de los cuales se presentan a continuación los más comunes.

2.1.4. REDES NEURALES DE UNA SOLA CAPA Y UN SOLO SENTIDO DE PROPAGACIÓN DE SEÑAL[9]

La forma más simple de arquitectura de redes neurales, en cuanto a las capas que las integran, es la red neural de una sola capa (o dos capas, si se considera como capa a las unidades de entrada). Este tipo de redes neurales se conforman a partir de una serie de nodos de entrada que están conectados directamente a los nodos de la capa de salida (ver Figura 14).

---

[8] Muchos autores cuentan como capas de procesamiento solamente los nodos en los que se realiza algún tipo de cálculo. Desde este punto de vista, las neuronas de entrada no son consideradas como una capa.

[9] *Single-layer Feedforward Network*, en inglés

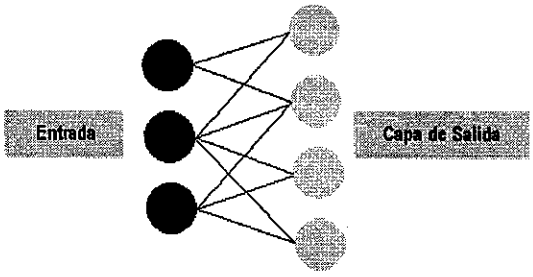


Figura 14. Red neural de una sola capa.

Una de las características principales de este tipo de redes es su imposibilidad para regresar la señal a las unidades de la capa de entrada. Es decir, este tipo de redes propaga la señal en un solo sentido que va desde la entrada hasta éste es de la entrada a la salida (lo que en el ámbito de las redes neurales se conoce como propagación hacia delante o sentido de transmisión delantero).

Las memorias asociativas lineales son un ejemplo típico de esta clase de redes neurales, en las cuales se realiza una asociación entre el patrón de entrada y la información guardada en la estructura de la red. Como en toda red neural, la información se integra en la estructura de la red a partir de las configuraciones de los pesos de interconexión en cada nodo de ésta.

2.1.5. REDES NEURALES MULTICAPA DE UN SOLO SENTIDO DE PROPAGACIÓN DE SEÑAL[10]

Este tipo de redes neurales se distingue por la presencia de una o más capas intermedias entre la entrada y la capa de salida. A estas capas intermedias a menudo se les conoce también como capas escondidas u ocultas[11]. La función de las unidades de las capas escondidas es dotar a la red de capacidades de cálculo más poderosas, necesarias sobre todo cuando se trata de procesar vectores de entrada sumamente complejos o extensos.

En el caso de las redes neurales multicapa, cada una de las salidas de las unidades de procesamiento en cada capa sirve como entrada para los nodos de a siguiente capa.

Se puede hablar de redes neurales multicapa completamente interconectadas, cuando las neuronas de cada una de las capas están conectadas con todas y cada una de las neuronas de la siguiente capa (ver Figura 15). En cambio, cuando algunas de las neuronas de la siguiente capa no son conectadas a un nodo determinado, entonces se habla de redes neurales parcialmente conectadas (ver Figura 16). Este tipo de redes se usan comúnmente en casos en los que se busca algún tipo de proceso especial en cada uno de los nodos que integran la red, o cuando se ha atribuido un significado a los nodos de las capas intermedias. Un ejemplo claro de este caso es el de los sistemas expertos neurales, en los que los nodos de las capas intermedias están relacionados con padecimientos específicos y los nodos de la capa de salida con tratamientos específicos para estos trastornos (Gallant, 1994).

---

[10] Multilayer Feedforward Network, en inglés

[11] Hidden Layers, en inglés

Tanto en el caso de las redes parcialmente interconectadas, como en el de las redes totalmente interconectadas, se habla de que la propagación de la señal se da en un solo sentido. Y al igual que en las redes de una sola capa, el sentido de propagación es *hacia delante*.

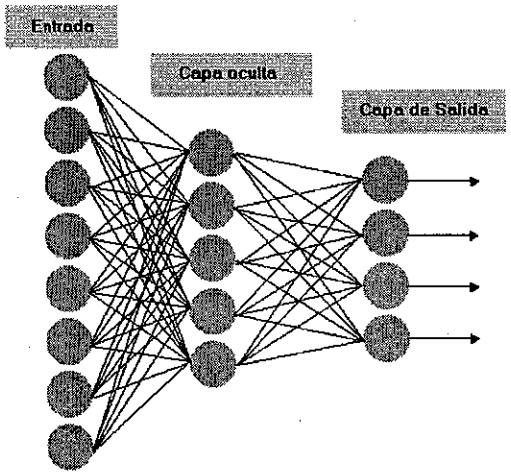


Figura 15. Red neural totalmente conectada.

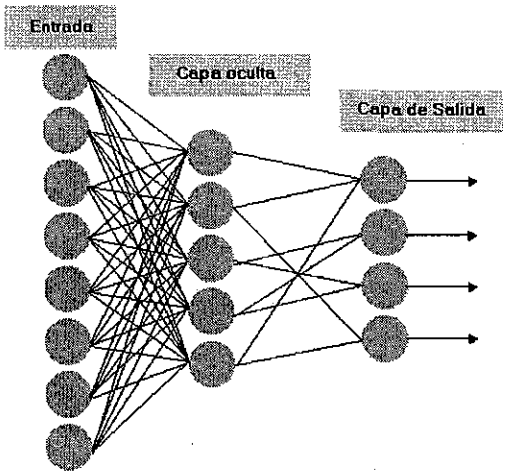


Figura 16. Red neural parcialmente conectada.

2.1.6. REDES NEURALES RECURRENTES[12]

Una red neural recurrente se distingue por tener por lo menos una conexión de retroalimentación. Es decir, integra un nodo en alguna de las capas intermedias o de salida que tiene conexión, no solamente con alguna o algunas de las neuronas de la capa siguiente, sino que también integra conexiones con neuronas de capas anteriores (ver Figura 17). Este tipo de redes se emplea sobre todo, cuando se requiere lograr algún tipo de retardo en el proceso de las entradas de la red, o cuando se quiere mantener algún tipo de información latente que propicie un comportamiento de la red no- lineal.

[12] *Recurrent Network*, en inglés

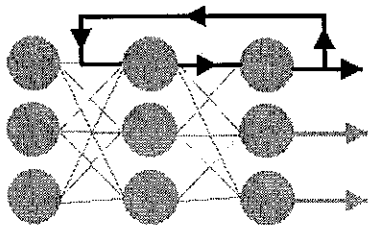


Figura 17. Red neural recurrente.

En la Figura 17, la red neural presentada integra una conexión entre un nodo de la capa de salida y un nodo de la capa intermedia

2.1.7. REDES NEURALES ESTRUCTURADAS EN LÁTICES

Un látice consiste en un arreglo, unidimensional, bidimensional o multidimensional de neuronas con un conjunto correspondiente de nodos de entrada que proporcionan las señales de entrada al arreglo. Las dimensiones del látice hacen referencia al número de dimensiones en el espacio en el que el grafo descansa. La Figura 18 muestra un ejemplo de red neural tipo látice unidimensional, y la Figura 19 muestra uno bidimensional. Nótese que en ambos casos cada nodo de entrada está conectado con cada neurona en el látice. Las estructuras en látices pueden ser vistas como redes neurales de un solo sentido con las neuronas de salida organizadas en filas y columnas (Haykin, 1994)

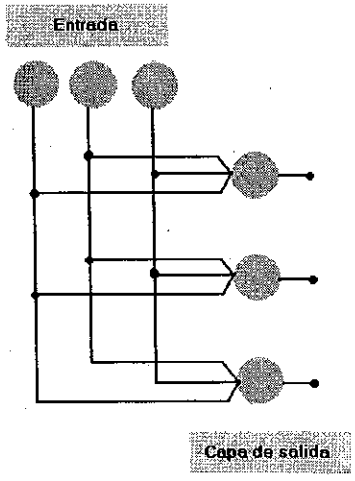


Figura 18. Red neural tipo látice de una dimensión.

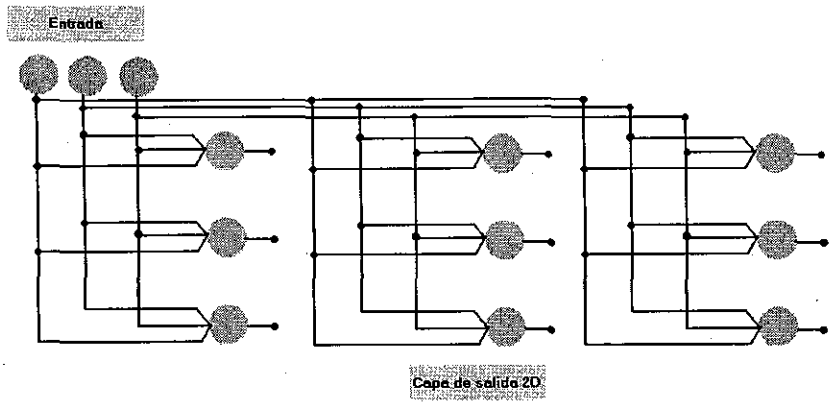


Figura 19. Red neural tipo látice de dos dimensiones.

2.2. FUNCIONAMIENTO DE UNA RED NEURAL

El funcionamiento de una red neural se puede dividir en dos grandes procesos:

- Transmisión de los impulsos a través de la estructura interna de la red, y
- Aprendizaje.

Estos dos grandes procesos están estrechamente asociados, respectivamente, con las dos funciones básicas atribuidas a una red neural: la identificación o clasificación de patrones de entrada (que se realiza mediante la propagación de los impulsos) y la aproximación de funciones, que es la base del aprendizaje de red.

2.2.1. TRANSMISIÓN DE LOS IMPULSOS A TRAVÉS DE LA ESTRUCTURA DE LA RED

Los componentes de una red neural interactúan para transmitir o propagar, datos a través de toda la estructura de la red, de la manera siguiente:

a) Cada una de las unidades de la capa de entrada se asigna a alguna característica o atributo del problema a tratar. Así, se tendrán tantas unidades en la capa de entrada, como características sobresalientes en el problema a tratar. Las neuronas de esta capa de entrada se alimentan con los valores correspondientes de cada característica del problema en un caso específico.

b) Las entradas se procesan en cada nodo de la red acorde con las funciones de salida ya definidas y se transmite el valor resultante a cada una de las unidades con las cuales el nodo actual, tiene conexión. Así, los valores resultantes de cada nodo se alimentan al siguiente nodo como entrada para un nuevo procesamiento. La transmisión de las entradas termina cuando los nodos de la capa de salida son alimentados y éstos han ofrecido su resultado al usuario.

c) La salida de la red representa la solución al problema. En la capa de salida de una red neural se tendrán entonces, tantas unidades como soluciones pueda tener el problema y el nodo activado es el que indica cuál es la solución calculada para el problema procesado.

2.2.2. APRENDIZAJE EN UNA RED NEURAL

Por aprendizaje en una red neural se debe entender al proceso en virtud del cual los pesos de conexión de cada nodo de la red sufren ajustes de manera tal que la capa de salida pueda ofrecer la solución adecuada al problema presentado en la capa de entrada.

Las redes neurales usan formas muy específicas para realizar este ajuste de pesos de conexión, que van de acuerdo al modelo de red bajo el cual están funcionando. Es posible clasificar los métodos de aprendizaje en dos grandes grupos: el aprendizaje no supervisado y el aprendizaje supervisado.

2.2.2.1. Aprendizaje no supervisado

El aprendizaje no supervisado es otra modalidad usada para el ajuste de pesos en las redes neurales. En este tipo de aprendizaje a la red no se le proporcionan modelos de respuestas correctas a obtener a partir de ciertas entradas. Simplemente se introducen los ejemplos del conjunto de casos de entrenamiento en el sistema y se espera que la red neural agrupe los ejemplos con base en la cercanía que entre cada uno de ellos existe. El cálculo de cercanía para el agrupamiento posterior se puede realizar de diversas maneras que van desde procedimientos sencillos como los que se basan en la *distancia de Hamming*[13], hasta otros más complejos (Ver: Skapura, 1995; y Maravall Gómez-Allende, 1994)

Para que el sistema realice el procedimiento de agrupamiento de ejemplos se necesita la constante modificación de los pesos de conexión entre las neuronas que integran la red que permita posteriormente, asignar correctamente cada uno de los ejemplos al grupo adecuado. En este caso también se espera que la red neural ya entrenada sea capaz de trabajar eficientemente con casos similares pero no introducidos previamente.

Entre otros, los modelos que se mencionan a continuación son representativos de la línea de investigación basada en el aprendizaje no supervisado.

- Redes de Hopfield (Hopfield, 1982).
- Mapas auto-asociativos de Kohonen (Kohonen, 1982).
- Los Mapas auto-asociativos bidimensionales de Kohonen (Kaski, 1997).

2.2.2.2. Mapas auto-organizativos de Kohonen

Los Mapas Auto-organizativos, son también conocidos como Mapas de Características, SOM's o Mapas de Kohonen, como también se les conoce por su autor, el Finlandés Teuvo Kohonen, quien diseñó este modelo con la finalidad de modelar la manera en que podría explicarse la formación de mapas topológicos en el cerebro por medio del aprendizaje.

---

[13] La distancia de Hamming (D_H) puede ser definida como el total de elementos distintos en valor y posición, entre dos vectores. Por ejemplo el vector 1={1,0,1,1,0} se encuentra a una $D_H = 1$, del vector 2={1,1,1,1,0}.

Este tipo de redes neurales pueden ser usados como capas de procesamiento para otras redes neurales, o como aplicaciones independientes específicas para clasificación. Ya sea como capa de procesamiento o como aplicación independiente, el tipo de aprendizaje en el cual se basan los Mapas, es el competitivo, que se guía bajo la estrategia de "el ganador se lleva todo".

Existen en la actualidad dos variantes de los Mapas de Kohonen, los mapas para el Aprendizaje con base en la Cuantificación de Vectores y los Mapas con preservación de la Topología (ver: Kohonen, 1982)

La arquitectura de los Mapas Auto-organizativos es sencilla, se integra por una capa de unidades de entrada que se conecta completamente con una capa de unidades de salida de una o dos dimensiones (Ver Figura 20).

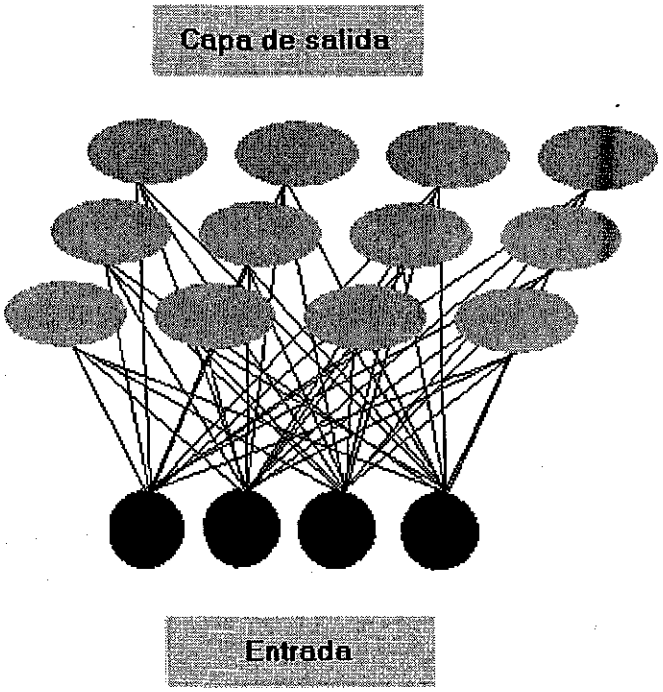


Figura 20. Arquitectura de un Mapa Auto-organizativo.

En este tipo de redes las entradas alimentan a la red directamente a las neuronas de procesamiento dispuestas en una o dos capas de proceso[14], en las que cada salida se calcula de acuerdo a la siguiente ecuación:

$$\text{Salida} = \sum_{i=1}^{n} E_j \cdot W_{ij} \qquad (2)$$

[14] En el modelo original, la arquitectura de los Mapas de Kohonen emplea una sola capa de proceso, que a la vez es la capa de salida de la red. Sin embargo, modificaciones hechas al modelo entre 1996 y el 2003 han llevado a la inclusión de una capa adicional de salida, para problemas cuya solución implique clasificaciones en dos o más dimensiones.

En donde E son las entradas a la red, W_{ij} los pesos que van de la entrada i a la neurona j de la capa de salida, con magnitudes de pesos y de vectores de entrada normalizados.

En este tipo de redes, la neurona con la salida más grande es la neurona ganadora, cuya salida se coloca en 1, y el resto de las salidas de las neuronas de la red se colocan en 0.

El establecimiento de diferencias entre los patrones de entrada se logra con base en las distintas neuronas ganadoras, en donde patrones iguales o similares son relacionados con la misma neurona ganadora.

Para comprender los Mapas auto-organizativos es necesario antes explorar dos procesos que se dan durante el aprendizaje:

a) La Normalización de Vectores, que se da de la siguiente manera:

Si consideramos un vector $A = ax + by + cz$, el vector normalizado A' se obtiene dividiendo cada uno de los componentes de A entre la raíz cuadrada de la suma de cuadrados de todos los componentes. En otras palabras, cada componente es multiplicado por $1/\sqrt{a^2+b^2+c^2}$. Tanto el vector de pesos como el vector de entradas son normalizados durante la operación del Mapa Auto-organizativo de Kohonen, debido a que la ley de entrenamiento usa la resta del vector de pesos al vector de entradas. Usando la normalización de los valores en la sustracción, se reducen ambos vectores a estatus de unidades, y con ello la sustracción se hace posible.

b) La Inhibición Lateral, que es un proceso que se deriva de observaciones realizadas a redes neurales biológicas, sobre todo en circuitos neurales relacionados con la visión humana (Kohonen, 1982). Las conexiones laterales en una capa dada, están determinadas por una fuerza que es inversamente proporcional a la distancia entre las unidades. Las conexiones con valores positivos son consideradas *conexiones excitatorias* y las que tienen conexiones con valores negativos son consideradas *conexiones inhibitorias.*

La relación entre la fuerza de conexión y la distancia de la neurona ganadora, se representa mediante una función de tipo *sombrero mexicano* (ver Figura 21).

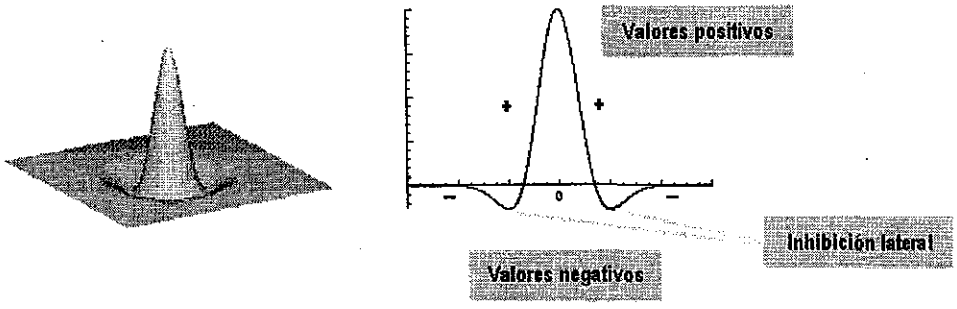


Figura 21. Función de sombrero mexicano, mostrando la inhibición lateral.

El efecto de esta función es la creación de un ambiente competitivo para el aprendizaje (Kohonen, 1982). Solamente las neuronas ganadoras y sus vecinos cercanos participan en el aprendizaje a propósito de un patrón de entrada dado.

La ley de aprendizaje usada en los Mapas Auto-organizativos tiene un solo sentido. Los cambios en los vectores de pesos para una neurona dada sufren ganancias constantes alfa, multiplicados por la diferencia entre el vector de entradas y el vector de pesos en el tiempo $t-1$:

$$w_t = w_{t-1} + a(X_i - w_{t-1}) \qquad (3)$$

Tanto el vector de pesos en t -1 como el vector de entradas son normalizados. Alfa es una constante que se fija al momento de la ejecución del proceso de entrenamiento y sus límites son $[0,1]$.

En los Mapas Auto-organizativos de Kohonen, existe un parámetro además de Alfa, que se conoce como *tamaño del perímetro de vecindad*[15], que se usa para modelar el efecto de la función de sombrero mexicano, ya descrita. Aquellas neuronas que se encuentran dentro del perímetro de vecindad de la neurona ganadora participan en el entrenamiento en el momento t y en la modificación del vector de pesos. Este parámetro de perímetro de vecindad comúnmente se inicia con un valor mayor, dentro de los límites de $[0,1]$ por supuesto, y se va reduciendo conforme avanzan las iteraciones durante el entrenamiento, hasta llegar a señalar solamente a las neuronas ganadoras específicamente.

Actualmente existen muchas versiones del algoritmo para los Mapas Auto-organizativos de Kohonen. A continuación se describe uno de los más usuales y efectivos que implica el uso de una sola capa de unidades de salida, es decir, usa la arquitectura originalmente planteada por Kohonen (ver: Freeman y Skapura, 1991 y Kohonen, 1982):

1. Se asume que los nodos de la capa de salida están conectados en un arreglo o array de una o dos dimensiones
2. Se asume que la red construida está completamente interconectada, es decir, todos los nodos o unidades de la capa de entrada se conectan con cada nodo de la capa de salida
3. Todos los pesos $w_{i,j}$ son inicializados con valores aleatorios comprendidos entre $[0,1]$
4. Se conforma un conjunto de vectores de entrada X
5. Se usa aprendizaje competitivo:

   3a. Se elige aleatoriamente un vector de entrada X_i

   3b. Se determina la neurona ganadora en la capa de salida, definida como aquel nodo pésimo que cumple con la condición $w_i x >= w_k x$ solo si los pesos están normalizados
6. Dada la iésima neurona ganadora, se actualizan los pesos de la siguiente manera:

   $w_k t = w_k t - 1 + \mu \aleph(i,k)(x - w_k)$, en donde $\aleph(i,k)$ es la función de perímetro de vecindad, que tiene un valor de 1 cuando i=k y sale de la distancia comprendida entre $|r_k - r_i|$ entre las unidades i y k del arreglo de unidades de salida. Así, las unidades que se encuentran en vecindad con la neurona ganadora, e incluso la propia neurona ganadora ven modificados sus pesos significativamente. Los pesos asociados con las unidades fuera de la vecindad establecida para la neurona ganadora no cambian significativamente.
7. El proceso se repite hasta que se cumple con un criterio de número de iteraciones alcanzado o hasta que las vecindades entre neuronas no sufren modificaciones
8. La función de establecimiento de perímetro de vecindad más frecuentemente usada es la siguiente:

[15] *Neighborhood size*, en inglés.

$\aleph(i,k) = e^{\left(-|r_k - r_i|^2\right)/(2\delta^2)}$, en donde δ^2 es un parámetro de amplitud [0,1] que sufre un decremento con el tiempo.

9. El proceso se repite presentando cada patrón del conjunto de vectores, cuidando siempre que la selección del patrón a presentar sea aleatoria. Asimismo, el conjunto de vectores se presentará completo varias veces. El número es determinado por el usuario, pero siempre debe ser un número grande (usualmente se usa un t >= 500).

Algoritmo 1. Aprendizaje en un Mapa Auto-organizativo de Kohonen (procedimiento general).

2.2.2.3. Aprendizaje supervisado

En el aprendizaje supervisado la adecuación de los pesos de conexión entre cada nodo se realiza evaluando el desempeño general de la red para lograr una salida deseada ante una configuración de patrones de entrada determinados.

En el proceso de aprendizaje supervisado, el entrenamiento de una red consiste en la evaluación de la salida actual que la red neural presenta ante una entrada, y su comparación con la salida que la red *debería presentar* ante esta entrada. La diferencia entre la salida obtenida y la salida deseada se usa para calcular el ajuste que debe darse al interior de la red en cada uno de los pesos de conexión, a fin de que se logre obtener la salida.

A este tipo de aprendizaje se le conoce como aprendizaje supervisado ya que el uso de salidas deseadas ante entradas determinadas permite supervisar el desempeño de la red en cada momento y con ello, la realización de los ajustes de pesos necesarios en la dirección correcta.

Una de las características de las redes neurales asociadas con este tipo de aprendizaje es la capacidad para generalizar los aprendizajes logrados. Se considera que un proceso de aprendizaje supervisado en una red neural ha tenido éxito cuando el ajuste de pesos, realizado a partir de un conjunto de ejemplos, le permite tener un desempeño efectivo ante problemas similares a los ejemplificados, pero nunca antes introducidos en el sistema.

2.2.2.4. La Retropropagación del error

Quizá el algoritmo de aprendizaje más famoso en redes neurales sea el que se basa en la retropropagación del error[16]. Este algoritmo fue descrito en la compilación de Rummelhart y McClelland (1986) aunque parece haber un reconocimiento generalizado de que los creadores del algoritmo de aprendizaje fue propuesto por Paul Werbos (Werbos, 1974) y por Parker (Parker, 1985).

La arquitectura de una red con aprendizaje basado en la retropropagación el error, tiene un diseño jerárquico consistente en un conjunto de capas de unidades de proceso completamente interconectadas. El proceso de la información bajo la retropropagación tiene por objetivo realizar la aproximación de una función $f : A \subset R^n \rightarrow R^m$ a partir de un subconjunto A de un espacio euclidiano n-dimensional a un subconjunto delimitado $f[A]$ de un espacio euclidiano m-dimensional, por medio de un conjunto de ejemplos de entrenamiento

[16] *Backpropagation*, en inglés.

$(x_1, y_1), ((x_2, y_2), ..., (x_k, y_k)$ de la correspondencia, en donde $y_k = f(x_k)$. Se asume que estos ejemplos de la correspondencia f son generados seleccionando de A, x_k vectores al azar de acuerdo con una función fija de probabilidad $p(x)$.

Actualmente existen muchas variables del algoritmo. A continuación se describe uno de los más usuales y efectivos que implica el uso de una sola capa de unidades ocultas (Freeman y Skapura, 1991), en el entendido de que su extensión a más unidades es sencilla.

1. Se aplica un vector de entrada a la red y se calculan los correspondientes valores de salida.
2. Se comparan las salidas obtenidas con las salidas correctas y se determina una medida del error.
3. Se determina en qué dirección (positiva o negativa) debe cambiar cada peso con objeto de reducir el error.
4. Se determina la cantidad en que es preciso cambiar cada peso
5. Se aplican las conexiones a los pesos
6. Se repiten los pasos del 1 al 5 con todos los patrones de entrenamiento hasta que el error para todos ellos se haya reducido hasta una cantidad aceptable

Algoritmo 2. Aprendizaje por retropropagación del error (procedimiento general).

Como podrá observarse en el algoritmo anterior, el trabajo con la retropropagación del error implica la realización de dos fases o momentos en la red neural:

- Una fase de propagación de la señal, que es precisamente la fase de operación o consulta de la red. En esta fase, gracias a la propagación de la señal desde las unidades de entrada, hasta la capa de salida, la red puede establecer una correspondencia entre vectores de entrada y una salida correspondiente.

- La segunda fase es la de propagación del error hacia atrás o retropropagación. Esta fase que tiene un papel muy importante en el proceso de entrenamiento de la red, ya que gracias a ésta precisamente, es que se pueden ajustar los pesos en la arquitectura interna, para que en la próxima propagación de la señal, la red sea mucho más eficiente. En términos neurocomputacionales, a este ajuste de pesos de le llama también *aproximación de la función.*

Los pasos que tienen lugar en estas dos fases se presentan a continuación.

1. Se localiza la primera unidad de procesamiento de la capa que se encuentre inmediatamente por encima de la capa actual.
2. Se pone a cero el total actual de la entrada.
3. Se calcula el producto del primer peso de conexión de entrada, por la salida de la unidad transmisora.
4. Se añade el producto al total acumulado
5. Se repiten los pasos 3 y 4 para todas las conexiones de entrada
6. Se calcula el valor de salida para esta unidad aplicando la función de salida
7. c en donde x es la entrada total
8. Se repiten los pasos del 2 al 6 para todas las unidades de esta capa
9. Se repiten los pasos del 1 al 7 para todas las capas de la red.

Algoritmo 3. Propagación de la señal hacia delante en el aprendizaje por Retropropagación del error.

Una vez que se ha calculado un valor de salida para todas las unidades de la red, se comparan los valores calculados para las unidades de la capa de salida, con la salida que se ha establecido como deseada, elemento por elemento como se muestra en el algoritmo siguiente.

Se calcula un valor de error en cada unidad de salida. Estos términos de error son realimentados en todas las demás unidades de la estructura de la red mediante la siguiente secuencia de pasos:

1. Se localiza la primera unidad de proceso que se encuentre inmediatamente por debajo de la capa de salida.
2. Se pone a cero el error total actual.
3. Se calcula el producto del peso de conexión de la primera salida por el error proporcionado por la unidad de la capa superior.
4. Se añade ese producto al error acumulado
5. Se repiten los pasos 3 y 4 para todas las unidades de salida
6. Se multiplica el error acumulado por $o(1=o)$ en donde o es el valor de salida de la unidad de la capa oculta que se ha producido durante la operación de la propagación de la señal hacia delante
7. Se repiten los pasos del 2 a 6 para todas las unidades de la capa
8. Se repiten los pasos 1 al 7 para todas las capas
9. Se localiza la primera unidad de proceso que esté en a capa superior a la capa de entrada
10. Se calcula el valor de cambio del peso para la primera conexión de entrada de esta unidad añadiendo una fracción del peso acumulado en esta unidad al valor de entrada de la unidad.
11. Se modifica el término de cambio de peso añadiendo un término de momento, el cual es igual a una fracción del valor del cambio de peso procedente de la iteración anterior.
12. Se guarda el nuevo valor del cambio de peso como valor anterior del cambio de peso para esta conexión.
13. Se modifica el peso de la conexión añadiendo el valor del nuevo cambio de peso de conexión al peso anterior de conexión
14. Se repiten los pasos del 10 al 13 para todas las conexiones de entrada de esta unidad
15. Se repiten los pasos del 10 al 14 para todas las unidades de esta capa
16. Se repiten los pasos del 10 al 15 para todas las capas de la red

Algoritmo 4. Propagación del error hacia atrás en el aprendizaje por Retropropagación

Dado que el algoritmo de retropropagación del error es altamente costoso en términos computacionales, se han realizado numerosos esfuerzos para acelerar su convergencia. Sin embargo, estas modificaciones aún no han sido lo suficientemente buenas como para desplazar del lugar exitoso que ha ocupado el algoritmo original desde su propuesta (Freeman y Skapura, 1991).

Otros ejemplos de redes neurales que funcionan bajo el esquema del aprendizaje supervisado son, entre muchos, los siguientes:

- Perceptrón (Rosenblatt, 1958).
- Contrapropagación (Freeman y Skapura, 1991).
- Máquinas de Boltzmann (Freeman y Skapura, 1991).

2.2.2.5. Aprendizaje competitivo

El propósito principal de los algoritmos para aprendizaje competitivo es la distribución de un cierto número de vectores en un posible espacio altamente dimensional (Fritzke, 1997). La distribución de estos vectores debe reflejar de alguna manera, la distribución probable de las señales de entrada que, en general, no se dan explícitamente, sino a partir de los vectores procesados.

En general, los sistemas que se estructuran con base al aprendizaje competitivo, persiguen alguna de las siguientes metas, que son mutuamente exclusivas:

- Minimización del error,

- Maximización de la entropía
- Mapeo de señales de entrada polidimensionales en estructuras de menores dimensiones, guardando algunas relaciones de similaridad entre ambas estructuras.
- Clasificación y *Clustering*

En el aprendizaje competitivo las neuronas de una red compiten por la activación, es decir, compiten por ser la única que se activa en ese momento temporal. Cuando se trata de elementos de salida, éstos compiten por la activación, o respuesta, ante una configuración de pesos determinada en las unidades internas.

En este tipo de aprendizaje, los pesos presinápticos cambian solamente para aquella neurona que ha ganado la activación y compiten entre sí por una cantidad limitada de peso disponible.

El aprendizaje competitivo puede clasificarse en:

- Aprendizaje competitivo duro. También conocido como aprendizaje del tipo "El ganador se lo lleva todo", implica alguno de dos tipos de ajuste interno de pesos: a) una evaluación de todos los patrones a procesar antes de que se realice cualquier modificación al sistema, ó b) un ajuste denominado en línea, en el que los pesos de conexión son modificados a cada presentación de un patrón de entrada (Gray, 1992).
- Aprendizaje competitivo blando, o también conocido como aprendizaje competitivo de dimensionalidades dinámicas, ya que éstas se ajustan y configuran a partir de las dimensiones encontradas en los patrones de entrada que se habrán de procesar.

2.2.2.6. Aprendizaje pseudo competitivo

Este tipo de aprendizaje comparte muchas de las características del aprendizaje competitivo, solamente que distintos niveles de activación pueden presentarse en las unidades de salida. En este tipo de aprendizaje, las activaciones y por consiguiente, la actualización de pesos presinápticos se da en forma aleatoria, y la verdadera competición se da en la forma en la que los pesos se actualizan al interior de la neurona elegida.

2.2.2.7. Aprendizaje por refuerzo

Muchos de los algoritmos para aprendizaje supervisado y no supervisado en redes neurales, requieren de datos de entrenamiento precisos para establecer los pesos de conexión y la conectividad entre los nodos de la red. Sin embargo, en la vida real, es usualmente difícil obtener datos precisos para poder realizar un proceso de entrenamiento y en muchos de los casos es imposible. Es por ello que existe un creciente interés por los algoritmos de aprendizaje con base en refuerzo.

En este tipo de aprendizaje los datos de entrenamiento tienen un papel de evaluadores para el problema y en nada se parece su uso a la retroalimentación "instructiva" en el caso del aprendizaje supervisado. Bajo este paradigma no se tiene un modelo completo del comportamiento de la red deseado; es decir, no se indica durante la fase de entrenamiento exactamente cuál es la salida que se espera de la red ante una entrada determinada. En este caso, la función del supervisor se reduce a

indicar mediante una señal de refuerzo si la salida obtenida en la red se ajusta en alguna medida a la deseada y en función de ello se ajustan los pesos con base en un mecanismo de probabilidades. Se podría decir que en este tipo de aprendizaje, la función del supervisor se asemeja más a la de un crítico que opina sobre la respuesta de la red, que a la de un maestro que indica la respuesta correcta que se debe generar (Hilera y Martínez, 2000).

El Aprendizaje por refuerzo presenta retos teóricos interesantes. Además de que la señal de refuerzo es solamente aproximada, solamente está disponible una vez en una cadena de eventos o acciones ocurridas. Ello implica que esta señal pudo haber sido causada por una acción realizada muchos pasos atrás o por toda la secuencia completa con múltiples grados de contribución en cada uno de los pasos.

Desde el punto de vista cognitivo y biológico el aprendizaje por refuerzo es más cercano a la forma en la que podemos suponer que se realiza el aprendizaje en organismos superiores, habiendo incluso autores como Cheng-Lian y Chin-Teng Lin (1996) que han inspirado sus algoritmos de aprendizaje por refuerzo en modelos matemáticos del comportamiento que incluso han logrado que se emita respuesta antes de que la señal de refuerzo se proporcione.

El desarrollo del aprendizaje por refuerzo puede ser rastreado y dividido en dos etapas: la primera etapa que comienza en los 50´s cuando psicólogos y matemáticos desarrollaron modelos computacionales para explicar el aprendizaje en animales y en seres humanos, quienes vieron el aprendizaje como un proceso estocástico (Cheng-Lian y Chin-Teng Lin, 1996, citando a Estes, 1950).

Casi al mismo tiempo, los cibernetistas y los estudiosos del control, estaban realizando esfuerzos independientes en el estudio del aprendizaje estocástico. Su trabajo básicamente se basaba en el uso de autómatas deterministas y después autómatas estocásticos, como un modelo para sistemas de aprendizaje operando en sistemas de ambientes aleatorios no dinámicos (Pedrycz y Waletzky, 1997).

En la primer etapa de los algoritmos para aprendizaje por refuerzo, se hablaba de un proceso "no asociativo" debido a que no había entradas al sistema excepto por la señal de refuerzo (Cheng-Lian y Chin-Teng Lin, 1996).

Ejemplos de algoritmos por refuerzo de fines generales, son los siguientes:

- *Linear-reward penalty* o LRP de Narendra y Tathacher en 1974, (Hilera y Martínez, 2000).
- *Associative Reward-Penalty ARP* de Barto y Anandan en 1985 que se aplica en redes de conexiones hacia adelante de dos capas cuyas neuronas de salida presentan una función de activación estocástica, (Hilera y Martínez, 2000).
- *Adaptative Heuristic Critic* presentado por Barton, Sutton y Anderson en 1983, que se utiliza en redes feedforward de tres capas diseñadas especialmente para que una parte de la red sea capaz de generar un valor interno de refuerzo que es aplicado a las neuronas de salida de la red, (Hilera y Martínez, 2000).

En el siguiente apartado se tratarán algunos modelos que funcionan con base en el aprendizaje por refuerzo, pero que fueron realizados con la finalidad específica de simular fenómenos conductuales.

3. MODELOS BASADOS EN REDES NEURALES PARA LA SIMULACIÓN DEL CONDICIONAMIENTO PAVLOVIANO

Es bajo este modelo de aprendizaje, que podemos ubicar a los modelos específicos para la simulación de fenómenos conductuales. Las redes neurales pueden proporcionarnos elementos para conocer cuáles son los procesos que el aprendizaje implica a nivel cerebral y la manera en la que ocurren (Moore, 1991). En el presente apartado se presentan al lector varios modelos de aprendizaje por refuerzo específicamente dedicados para la simulación y estudio de fenómenos conductuales. Aunque todos ellos tienen un objetivo común, existen diferencias teóricas importantes entre ellos, sobresaliendo principalmente la forma en la que conciben al reforzamiento. Existen modelos que asumen una diferencia entre condicionamiento clásico y condicionamiento operante, y con base en ello, se dedican a simular uno u otro aprendizaje. Algunos de los modelos más sobresalientes se presentan a continuación. Y por otro lado podemos encontrar a los modelos que asumen que no existe distinción alguna entre los dos condicionamientos, como el Modelo DBP, que se presenta en el capítulo siguiente en este mismo documento.

Se trató de documentar en la forma más completa, los modelos que aquí se presentan, sin embargo, en muchos de los artículos tomados para la elaboración de este apartado, no se integran algoritmos, o incluso, no se reportan las arquitecturas de red empleadas.

3.1 EL MODELO DYSTAL DE ALKON, VOGL Y TAM.

El modelo DYSTAL (Dynamically Stable Associative Learning) es una red neural artificial modificable basada en las características observadas en sistemas neurales biológicos del molusco *Hermissenda* y del hipocampo del conejo.

En el modelo DYSTAL, la red neural, modifica sus pesos con base en a) la convergencia de entradas de "Flujo" colaterales modificables y no modificables b) apareo temporal de esas entradas, y c) actividades pasadas de esos elementos recibiendo las entradas.

La modificación es independiente de la salida del elemento. Como una consecuencia, DYSTAL, muestra a) escalado linear de los esfuerzos computacionales con el tamaño de la red, b) aprendizaje rápido sin un maestro externo y c) habilidad para completar patrones, asociando independientemente ensambles diferentes de entradas y sirviendo como un clasificador de los patrones de entrada.

Este es un modelo realizado con base en el supuesto de la distinción entre condicionamiento clásico y condicionamiento operante, tratándose en este caso de un modelo para el condicionamiento clásico con base en el aprendizaje supervisado.

En las redes neurales típicas los pesos suelen tratarse uniformemente en toda la arquitectura o por lo menos entra las capas que la componen, en este modelo en cambio, se integran rutas de pesos que pueden correr por toda la arquitectura, específicamente de dos tipos (Ver Figura 22):

a) Rutas de pesos altos, que permiten una rápida y confiable transmisión de señales que produzcan comportamientos estereotípicos o de reflejo, bien definidos, y

b) Rutas de pesos bajos, que son el resto de las conexiones en la red y son llamadas en este modelo "conexiones colaterales", que inicialmente son de pesos bajos y posteriormente se van incrementando de acuerdo a reglas de aprendizaje bien definidas.

Además, el modelo implica interacciones de pesos no hebbianas y sin retroalimentaciones, independientes de la salida de la red, que son conocidas como "interacciones locales". Estas interacciones producen ajustes de pesos derivados del apareamiento de dos estímulos (que en este modelo se presentan como dos presentaciones sucesivas).

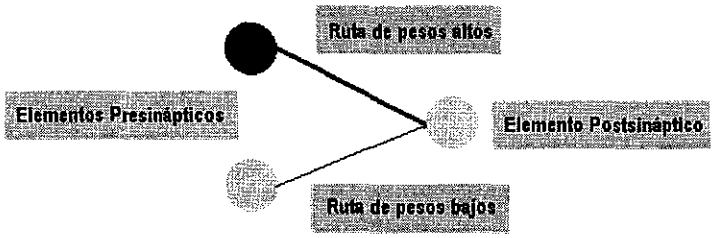


Figura 22. Tipos de rutas de pesos en el modelo DYSTAL.

Entre las ventajas que se reportan para el Modelo DYSTAL se encuentran las siguientes (Alkon, Vogl y Tam, 1991):

1. Escalado lineal de complejidad del algoritmo. En este diseño las salidas no son determinantes de la modificación de los pesos interiores y por ello, su complejidad es de *n*, con lo que el incremento del número de elementos en la red no implica un esfuerzo computacional como el del algoritmo de retropropagación cuya complejidad es *n Log n*. Ello también facilita su implementación en *hardware*.

2. No se requiere de un maestro externo. Lo que permite a DYSTAL clasificar y restablecer patrones sin estándares predeterminados de entradas. Así, a diferencia de las redes que dependen de la retropropagación del error como medio de aprendizaje y la minimización del error, DYSTAL no necesita que los patrones aprendidos sean preprogramados en una función de comparación.

3. El modelo es robusto a las variaciones de los parámetros como el radio de incremento de los pesos o el tamaño de cada paso en el decremento de los pesos. Asimismo, el modelo también es robusto ante cambios estructurales, lo que permite conservar las mismas reglas de aprendizaje aún a pesar de añadir elementos colaterales o conexiones de retroalimentación.

3.2 EL MODELO GMADN DE BAXTER, *ET AL* PARA SIMULAR EL APRENDIZAJE ASOCIATIVO.

Baxter *et al,* presentaron en 1987 el Modelo General de Neuromodulación Independiente de la Actividad, o GMADN por sus siglas en inglés (Baxter *et al,* 1991). Ellos asumen, al igual que en el Modelo DYSTAL se hace, que existen diferencias significativas entre condicionamiento clásico y condicionamiento operante, sin embargo, en este caso no se trata de reglas de aprendizaje distinto, sino arquitecturas distintas para el condicionamiento clásico y el operante. Los autores establecen que buscan con este modelo el entendimiento de los eventos que ocurren entre los elementos individuales neurales y en las redes que contribuyen al aprendizaje y la memoria. El enfoque que utilizan para el

desarrollo del modelo es analizar las propiedades de las neuronas y de los circuitos neurales que median en formas simples de aprendizaje en el molusco marino *Aplysia*, y desarrollar modelos matemáticos de estos modelos neurales y redes. Desde su perspectiva, un elemento individual adaptativo de tipo neural, que refleje las propiedades bioquímicas y biofísicas de las neuronas sensoriales, simula muchas características del aprendizaje no asociativo y del condicionamiento clásico.

Establecen que aún en redes neurales relativamente simples que incorporen una "regla de aprendizaje" asociativa, empíricamente derivada, se exhibirán algunas características de orden superior del condicionamiento clásico y algunas características elementales del condicionamiento operante.

El aprendizaje en este modelo se da con base en la actividad temporal contigua entre dos elementos, en donde un estímulo potente reforzante activa un sistema un sistema neural de respuesta y un sistema modulatorio que regula la eficacia de aferencias difusas al sistema de respuesta (Ver Figura 23).

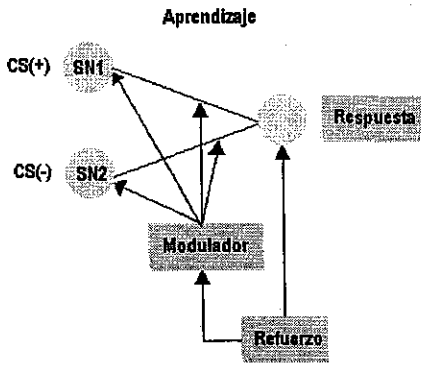


Figura 23. Aprendizaje en GMADN.

En el caso de la memoria, los efectos modulatorios amplificados causan un fortalecimiento en la conexión funcional entre la neurona apareada y el sistema de respuesta (Ver Figura 24).

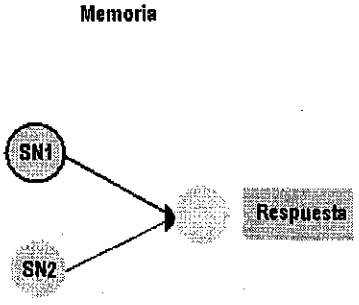


Figura 24. Memoria en GMADN

Para la simulación del Condicionamiento clásico, el GMADN implica el uso de una arquitectura en la que intervienen:

a) Neuronas sensoriales idénticas entre sí, que tienen por objetivo la recepción de los estímulos del ambiente. Pueden ser activadas independientemente por estímulos condicionados separados.

b) Una neurona facilitadora, que puede ser activada por las neuronas sensoriales si su nivel de entrada excede el umbral preestablecido. Se trata de un elemento no plástico que libera transmisores que inciden sobre la actividad de las neuronas sensitivas.

c) Una neurona motora, que es finalmente en donde se observa la respuesta condicionada (Ver Figura 25).

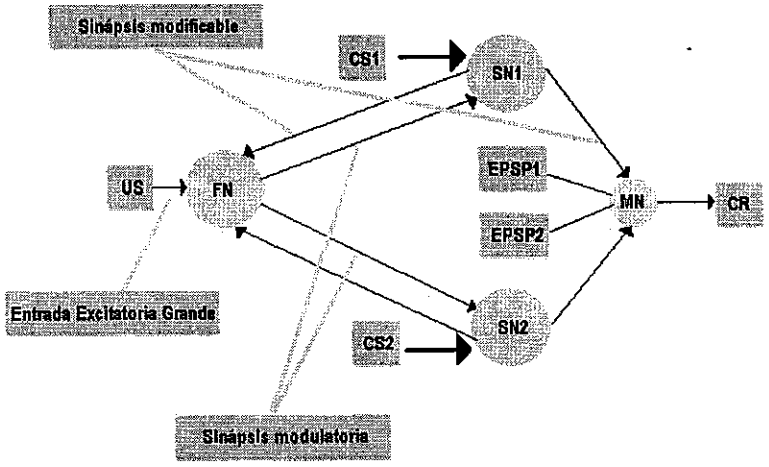


Figura 25. Arquitectura para la simulación del condicionamiento clásico con el GMADN

El condicionamiento operante, por otro lado, requiere que el circuito sea dirigido por dos células que funcionan como generadores de patrones PGA y PGB (en la Figura 26) conectadas por sinopsis inhibitorias. Estas excitan a los elementos asociativos que a su vez estimulan a los elementos motores. Se necesita además una retroalimentación entre los generadores de patrones a fin de incrementar la duración de su actividad.

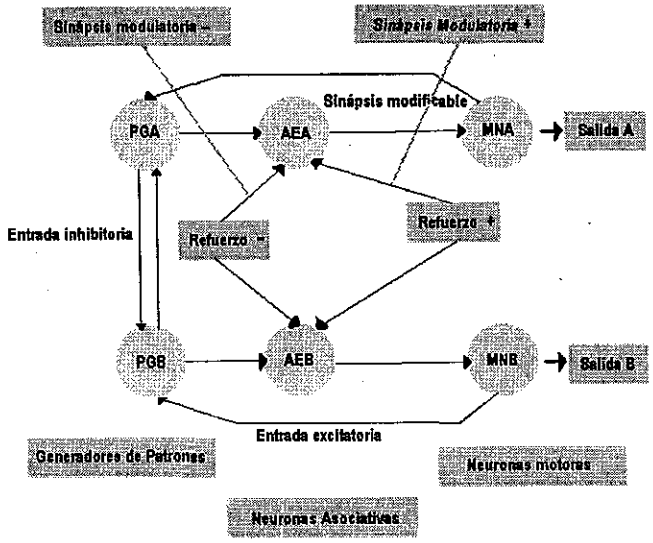


Figura 26. Arquitectura para la simulación del condicionamiento operante con el GMADN

3.3 EL MODELO SCHMAJUCK-LAM-GRAY (SLG) PARA SIMULAR INHIBICIÓN LATENTE

Este modelo formaliza algunas de las ideas de Sokolov y de Gray (Schmajuck, Lam y Gray, 1997). Sokolov establece que los animales construyen una representación interna del ambiente.

Siempre que se detecta alguna novedad (es decir, cuando existe alguna inconsistencia entre lo predicho y los eventos ambientales actuales):

1) Tiene lugar una respuesta de orientación, y

2) El Modelo interno se modifica

Cuando existe coincidencia entre lo observado y el estímulo predicho, el animal puede responder sin cambio en el modelo neural del mundo que ha construido. En 1971 Gray sugirió que un sistema de inhibición del comportamiento responde a señales de castigo, señales de no recompensa, estímulos nuevos o miedos innatos, inhibiendo los comportamientos en curso, incrementando su disponibilidad para la acción (nivel de alerta) e incrementando su atención a los estímulos ambientales.

El modelo de red que se describe a continuación provee un mecanismo que retoma las nociones de Sokolov (Schmajuck, Lam y Gray, 1997) acerca de la existencia de un modelo interno del mundo y un sistema de comparación, así como también considera el concepto de Gray acerca de un sistema inhibitorio. Conforme el modelo del mundo genera predicciones de futuros eventos, el sistema de comparación observa lo previsto y lo observado para computar novedades. La novedad es entonces usada para:

1) Controlar la atención a los eventos ambientales

2) Modificar el modelo del ambiente

3) Inhibir los comportamientos en curso

En la Figura 27 se muestra un diagrama con la arquitectura del modelo SGL:

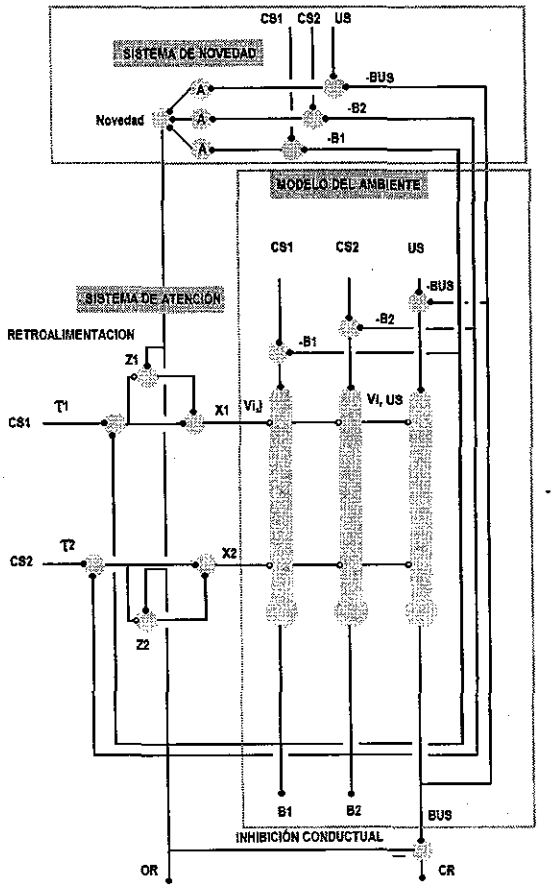


Figura 27. Arquitectura del Modelo SGL de Schmajuck[17]

El modelo SGL asume que un Estímulo Condicionado (EC) activa las representaciones internas. El sistema atencional mejora las representaciones internas de los EC activos al mismo tiempo cuando la novedad total del ambiente es grande (incrementando la atención), y decrementa las representaciones internas de aquellos EC activos al mismo tiempo cuando la novedad total es pequeña (decrementando la atención o quitando la atención por completo). La magnitud de las representaciones internas controla el almacenamiento de información al interior del modelo del ambiente (Asociabilidad) y la recuperación de información del modelo (recuperabilidad). Por lo tanto, eventos con grandes representaciones internas muestran una capacidad incrementada de formar asociaciones nuevas con otros eventos ambientales, así como una eficacia aumentada para recuperar asociaciones viejas. Eventos con representaciones internas pequeñas, muestran una capacidad decrementada para formar nuevas asociaciones con otros eventos ambientales, así como una eficacia decrementada para recuperar viejas asociaciones.

[17] Us, Estímulo incondicionado, Cs, Estímulo condicionado, t Huella, X Representación interna, ViUS novedad de la asociación, B, Predicción agregada del evento, OR, Respuesta de Oridentación, Los puntos negros representan sinápsis fijas y los blancos representan sinápsis variables.

3.4 EL MODELO DE SUTTON Y BARTO

Sutton y Barto (1981) proponen un modelo que ajusta los pesos de acuerdo con la conjunción de una huella de memoria[18] del CS con los componentes de R. Si tenemos que x_0= 1 es el US, el CS sea x_i = 1 para alguna i > 0, y w_i es el iésimo peso que representa la asociación entre x_i y y, entonces las ecuaciones que describen el modelo son:

$$y(t)=\sum_i w_i x_i$$

$$\Delta y(t)=y(t)-y(t-1)$$

$$w_i(t+1)=w_i(t)+c\Delta y(t)\overline{x}_i(t), i>0$$

$$\overline{x}_i(t+1)=\alpha\overline{x}_i(t)+x_i(t), i>0$$

El peso w_i se ajusta de acuerdo con el producto de Δy y $\overline{x_i}$; dado que Δy es un "detector de límites" temporal, siendo positivo cuando y se incrementa, negativo cuando y deçae, y 0 cuando y es constante, y $\overline{x_i}$ es una huella de memoria de CS, w_i se ajusta de acuerdo con la conjunción de la huella de memoria de CS y los incrementos o decrementos de y. Sutton y Barto inicialmente restringieron $y = [0,1]$, pero esta restricción ha sido abandonada posteriormente en un afán de hacer el modelo más simple.

El comportamiento de este modelo se ilustra en la Figura 28.

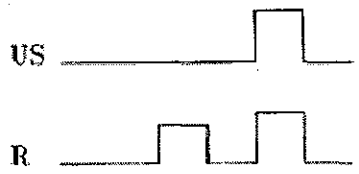


Figura 28. Comportamiento del Modelo de Sutton y Barto

Con los coeficientes α , c y w_0 con valores de 0.9, 0.01 y 1, respectivamente. La salida R se muestra después de que el entrenamiento ha producido un patrón constante de comportamiento. Como se puede ver, se aprende una asociación entre CS y R. Cuando CS precede completamente a US, se construye una asociación como la que se muestra en la Figura 29.

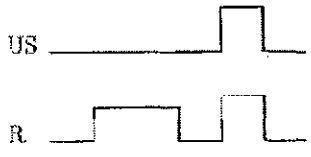


Figura 29. CS precediendo a US en el Modelo de Sutton y Barto

[18] Memory Trace, en inglés

Pero esta asociación es más débil y se aprende más lentamente mientras más tiempo existe entre la presentación del CS y la del US. R cae a 0 durante estos intervalos, como si el sistema hubiese olvidado la presentación del US después de que el CS ha cesado. Si la duración de CS se incrementa, como en la siguiente Figura, sin cambiar la presentación del CS y el US o sin modificar la duración del US, la fuerza de asociación no se ve afectada, pero se aprende más rápidamente. Lo anterior debido a que la huella de memoria de CS lleva más tiempo para construirse a su fuerza máxima, incrementándose el peso de asociación aún más con la presentación del US.

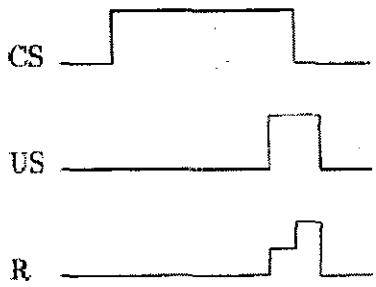


Figura 30. Incremento de la respuesta en el Modelo de Sutton y Barto

Cuando el CS y el US se traslapan como en la Figura 31, si el traslape es pequeño, entonces la asociación aprendida es débil, y cuando el traslapamiento se incrementa, entonces la fuerza de asociación se vuelve negativa, causando la ausencia de respuesta cuando el CS se presenta y solo una débil respuesta cuando se presenta el US solo hasta que cesa el CS.

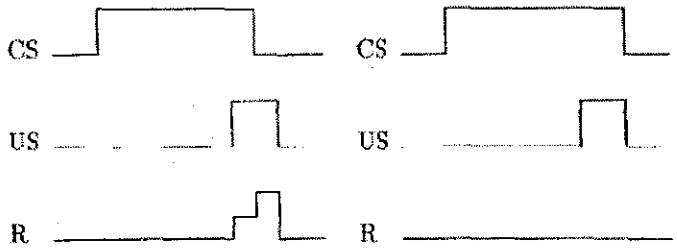


Figura 31. CS y US traslapados en el modelo de Sutton y Barto

Si el US está completamente contenido en el tiempo en el que se presenta el CS, como en la Figura 32, la respuesta del modelo es suprimida por completo.

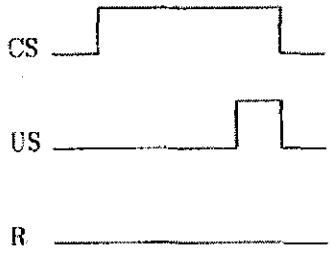


Figura 32. Respuesta suprimida en el Modelo de Sutton y Barto

3.5 EL MODELO DE KLOPF

En el modelo de Klopf (1989) los dos eventos cuya conjunción causa el cambio de pesos de asociación son los límites de R y una huella de memoria del límite inicial del CS. Este modelo emplea dos pesos para cada una de las entradas del modelo, uno positivo para el aprendizaje con una liga excitatoria hacia la salida y uno negativo para el aprendizaje de una liga inhibitoria hacia la salida. A ambos pesos nunca se les permite que alcancen un valor igual a cero, por ello nunca están desactivados y al inicializar la red no es necesario asignarles un valor aleatorio.

En este modelo w_i^+ representa la fuerza de la liga excitatoria entre la iésima entrada x_i y la salida y, y w_i^- representa la fuerza de la correspondiente liga inhibitoria. El modelo se puede describir brevemente con las siguientes ecuaciones:

$$y(t) = \sum_i (w_i^+(t) + w_i^-(t)) x_i(t)$$

$$\Delta y(t) = y(t) - y(t-1)$$

$$\Delta x_i(t) = [x_i(t) - x_i(t-1)]^+$$

$$w_i^+(t+1) = w_i^+(t) + \Delta y(t) \sum_{j=1}^{5} c_j \left| w_i^+(t-j) \right| \Delta x_i(t-j)$$

$$w_i^-(t+1) = w_i^-(t) + \Delta y(t) \sum_{j=1}^{5} c_j \left| w_i^-(t-j) \right| \Delta x_i(t-j)$$

En donde la función $[x]^+$ equivale a x, si $x > 0$, y equivale a 0 de cualquier otra manera. Los coeficientes c_j, con $j = 1,2,3,4,5$, son igual a 5, 3, 1.5, 0.75, y 0.25 respectivamente. Con estos valores, las sumatorias en las dos últimas ecuaciones se aproximan al exponencial suavizado de $\left| w_i^{\pm} \right| \Delta x_i$. Un modelo más simple se logra reemplazando las dos últimas ecuaciones por las siguientes:

$$w_i^+(t+1) = w_i^+(t) + c\Delta y(t) z_i^+(t)$$

$$w_i^-(t+1) = w_i^-(t) + c\Delta y(t) z_i^-(t)$$

$$z_i^+(t+1) = b z_i^+(t) + \left| w_i^+(t) \right| \Delta x_i(t)$$

$$z_i^-(t+1) = b z_i^-(t) + \left| w_i^+(t) \right| \Delta x_i(t)$$

Con los coeficientes b y c con valores de 0.5 y 5.0 respectivamente.

En este modelo el aprendizaje ocurre con la conjunción de un decaimiento de memoria del componente CS y el incremento y caída de R. Cuando R se incrementa ante la presentación del US, los pesos correspondientes al CS se incrementan y cuando R decae debido a la ausencia del CS o del US, los pesos sufren un decremento. La dimensión del cambio es proporcional a la fuerza de memoria del componente CS y a la cantidad de cambio en R. Así, mientras más momentos temporales

transcurran entre la presentación del CS y la del US, más lento será el aprendizaje. Finalmente, la fuerza de asociación depende de la ocurrencia en las fluctuaciones en R.

3.6 EL MODELO SPECTRAL TIME MODEL (STM) DE GROSSBERG-SCHMAJUK

Grossberg y Schmajuck propusieron un modelo de Condicionamiento Clásico que se basa en el uso de clústeres de unidades de proceso (Grossberg y Schmajuck, 1989). Siendo las entradas al clúster x_i y la salida y, se incluyen dos señales más de origen interno, representadas por u_i y v_i. El Modelo STM se compone de 80 canales, cada uno de los cuales es capaz de aprender una asociación entre CS y US para un retardo específico de tiempo entre la presentación de estímulos. Las salidas de estos 80 canales son sumadas juntas para obtener la salida del clúster. En este modelo x_0 representa al US, x_1 representa a US y y representa a R. Las ecuaciones son las siguientes:

$$u_i(t+1) = u_i(t) + \alpha_i(-Au_i(t) + (1 - Bu_i(t))x_1(t))$$
$$v_i(t+1) = v_i(t) + C(1 - v_i(t)) - Df(u_i(t))v_i(t)$$
$$w_i(t+1) = w_i(t) + Ef(u_i(t))v_i(t)(x_0(t) - w_i(t))$$
$$y(t) = \left[\sum_i w_i(t) f(u_i(t)) v_i(t)\right]^+$$

La función $f(x) = x^8/(\beta^8 + x^8)$ donde $\beta = 0.6$. Los coeficientes $\alpha_i = 0.03/i$, y A, B, C, D y E, tienen un valor de 1, 1, 0.0001, 0.125 y 0.01 respectivamente. El modelo asume un mecanismo adicional que proporciona una memoria a corto plazo para el CS. Aún si el CS es de corta duración, la memoria de éste perdura lo suficiente para poder coincidir con el US. En el modelo, se puede simular estableciendo que el valor de x_1 permanezca al menos hasta el final de la presentación del US.

En el iésimo canal del Modelo STM, las variables u_i y v_i actúan para formar una línea directa de retardo, que es disparada por la presentación del CS. Cuando x_i cambia de 0 a un valor fijo positivo, u_i se mueve de 0 a $u_{high} = x_1/(A + Bx_1)$, con una tasa de cambio controlada por el coeficiente α_i, al mismo tiempo, u_i lleva a v_i de un valor de 1 a un valor de $v_{low} = C/(C + Df(u_{high}))$. El resultado es que el producto $f(u_i)v_i$, al cual Grossberg y Schmajuck llaman "señal puenteada", que es una memoria retardada y marcada por la presentación del CS.

El Modelo STM es significantemente diferente del Modelo de Sutton y Barto y del Modelo de Klopf, ya que R no se incrementa inmediatamente ante la presentación del CS, sino que se incrementa en anticipación a la presentación de US y decae después. El STM además, necesita largos periodos de tiempo entre ensayos para regresar a las condiciones iniciales requeridas, y cuando no se espera el tiempo suficiente para la recuperación, el modelo cae en un estado estable de no respuesta.

3.7 EL MODELO DE KEHOE

Kehoe, propuso en 1988 un modelo basado en una red neural compuesta por tres unidades adaptativas, cada una de las cuales opera de acuerdo con una regla de competición asociativa, conocida en el ámbito de la neurocomputación como regla "delta".

En la Figura 33 se muestra la red de Kehoe, diseñada para simular condicionamiento clásico, en la que se encuentran tres tipos de entradas o receptores sensoriales: US, T y L, una unidad oculta, denominada X y una unidad de respuesta tipo CR/UR. Estas dos últimas unidades son de tipo adaptativo y se configuran de acuerdo con la regla de competición asociativa, o regla "delta".

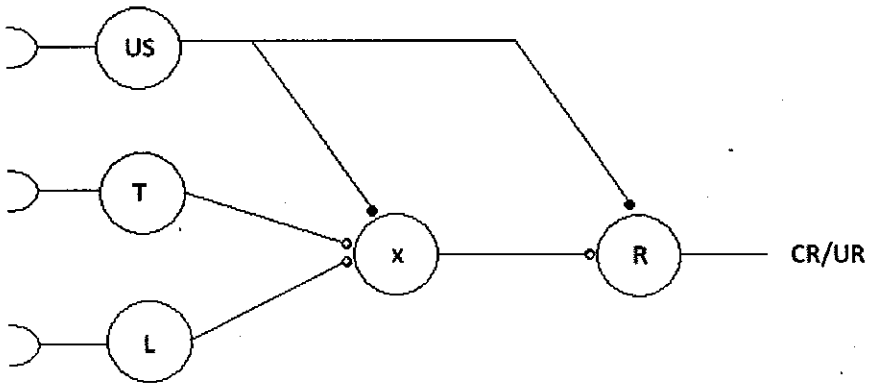


Figura 33. Red mínima de Kehoe, para condicionamiento clásico

En este modelo, las unidades sensoriales T, L y US son binarias, es decir, su activación está restringida a (0,1) y están determinadas por las contingencias ambientales. Asimismo, las unidades X y R son binarias, pero su activación o no, están determinadas por la regla "delta". Cada punto de conexión entre las unidades sensoriales y las unidades adaptativas se designa como $V_{i,j}$ en donde i es la unidad presináptica y j es la unidad postsináptica. Los pesos se inicializan en 0.0 pudiendo ser valores positivos o negativos los resultantes de la corrida de la red. Los pesos entre la unidad US y X, así como los pesos de conexión entre US y R se inicializan en 1.0. La fase de entrenamiento de la Red se subdivide en dos periodos un periodo para el CS y un periodo para el US. El algoritmo para estas dos fases se presenta a continuación.

Para cada la fase CS

1. Se programa la secuencia para las entradas T y L y las conexiones con X se eligen para el cambio
2. Se determina aleatoriamente un umbral para X y para R, independientes uno del otro, el valor se restringe a [0.00, 0.99], con una distribución cuadrática
3. Si la suma de los pesos elegibles de entrada a X o a R es > T entonces FIRE de J = 1, si no, entonces FIRE de J = 0,
4. Una vez que se calcula la salida de X entonces se procede al cálculo de la Salida de R,
5. Si X disparó en el momento anterior, entonces los pesos de X se usan en el cálculo del disparo de R,

Para la fase US

6. Durante esta fase, los pesos de conexión se calculan a partir de la regla "delta" propuesta por el Modelo Rescorla – Wagner:

$$\Delta V_{i,j} = \alpha_j(\lambda_j - \sum V_{i,j})$$

En donde:
α_j= es la tasa de cambio para la unidad actual ($0>\alpha_j<1$). En los ensayos no reforzados, α_j se reemplaza por β_j ($0>\beta_j<1$).
λ_j= El total de la fuerza de conexión que puede ser soportado por el US en un ensayo dado, para las unidades elegibles (λ_j=1.0, en ensayos reforzados y λ_j=0.0 en ensayos no reforzados),
$\sum V_{i,j}$= es la Suma de las fuerzas asociativas de los elementos concurrentes elegibles de entrada para la unidad j-ésima

7. Para el cómputo de CR, no se usa la regla "delta". Hecho que se llama "Periodos fantasmas", en los que no hay cambio de pesos, para evitar el efecto de la extinción del valor.

Algoritmo 5. Fases CS y US en el Modelo de Kehoe.

De acuerdo con Kehoe (1988), este modelo, aunque sencillo, permite explicar fenómenos de aprendizaje como la sumación, el bloqueo, con la inclusión de más unidades.

4. EL MODELO DBP DE DONAHOE, BURGOS Y PALMER

El modelo de Donahoe, Burgos y Palmer (1993) fue diseñado para la simulación del aprendizaje con base en refuerzos, ya sea instrumental o pavloviano. A diferencia de los modelos expuestos en el capítulo anterior, este modelo parte del supuesto de que no hay diferencia entre condicionamiento clásico y condicionamiento operante.

La descripción aquí presentada, del modelo y los submodelos que lo conforman está basada en Donahoe, Burgos y Palmer (1993), Donahoe y Palmer (1994), y Donahoe, Palmer, and Burgos (1997a, 1997b).

El Modelo se compone un Submodelo de Red y un Submodelo Neurocomputacional. El Submodelo de red es una especificación de los diferentes tipos de unidades que conforman la red neural y la forma en la que estas unidades se interconectan. El Submodelo Neurocomputacional, por otra parte, establece la forma en la que se calculan las activaciones y pesos en la red, así como la manera en la que se deben actualizar los valores obtenidos a partir de estos cálculos.

4.1 EL SUBMODELO DE RED

El Submodelo de Red especifica los diferentes tipos de unidades posibles bajo el modelo DBP, clasificadas en cinco tipos distintos: unidades de entrada que pueden ser de dos tipos: a) unidades de entrada regular y unidades de entrada para estímulos incondicionados, o unidades "S*"; b) unidades sensoriales asociativas o "sa"; c) unidades "ca1"; d) unidades asociativas motoras, o "ma"; d) unidades "vta", y e) unidades de salida, que pueden ser unidades respondientes, o unidades "CR/UR" y unidades operantes, o simplemente unidades "R". Las unidades descritas atienden a una representación simplificada y no a principios teóricos.

4.1.1 CLASIFICACIÓN DE LOS ELEMENTOS DE PROCESAMIENTO

En la literatura especializada en redes neurales se emplean tres grandes tipos de unidades de procesamiento, las unidades de *entrada*, las unidades *ocultas*, y las unidades de *salida*. Sin embargo, en el modelo DBP estos tres tipos de unidades se ubican dentro del grupo de las unidades corticales, y se incluye un tipo más de unidades, las subcorticales, entre las que se encuentran las unidades "sa" y las unidades "vta". La Figura 34 muestra la clasificación general de los elementos bajo el Submodelo de Red del Modelo DBP.

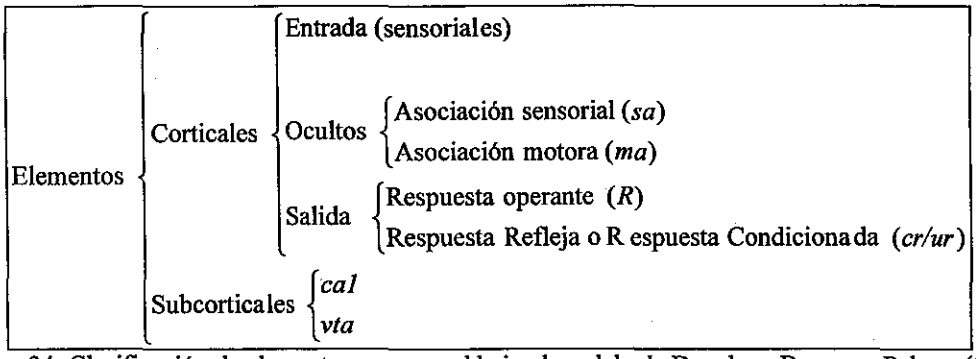


Figura 34. Clasificación de elementos en una red bajo el modelo de Donahoe, Burgos y Palmer (1993)

En la Figura 35 se muestra una red típica conformada bajo el modelo DBP, en donde las unidades 1,2,3,4,5,6,8,9,10,12 y 13 son unidades corticales y las unidades 7 y 11 son unidades subcorticales.

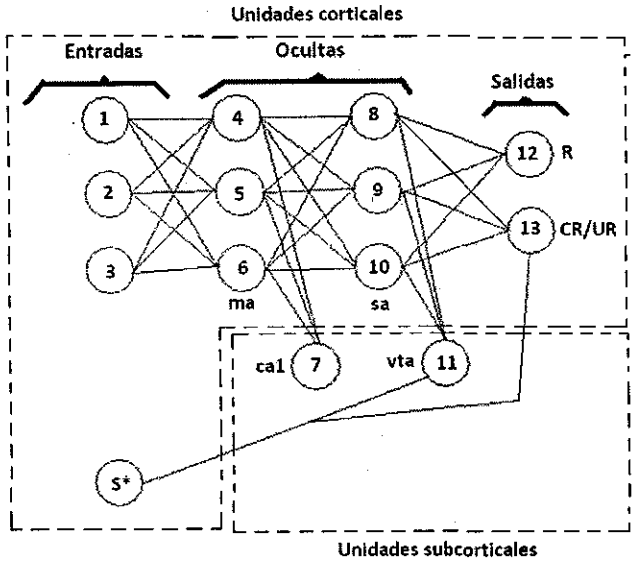


Figura 35. Tipos de unidades en un ejemplo típico de red bajo el modelo DBP

El número de unidades de entrada, ocultas o de salida, así como la cantidad de unidades corticales y subcorticales, puede ser modificado de acuerdo con las necesidades específicas del problema que se requiere resolver.

Se detalla a continuación cada uno de los tipos específicos de unidad que se pueden encontrar en las unidades corticales: de entrada, ocultas y de salida, y en las unidades subcorticales: de tipo "sa" y de tipo "vta".

4.1.2 UNIDADES CORTICALES

4.1.2.1 Unidades de entrada

Las unidades de entrada son los sensores o receptores de la red, por lo que no reciben entonces conexiones presinápticas. Las unidades de entrada pueden subdividirse en:

a) Entradas para estímulos condicionados, que son las unidades receptoras de la señal codificada que se asume como estímulo de entrada a la red, y

b) Las entradas para estímulos incondicionados, simbolizada como "S*" y que constituye el conjunto de unidades a partir de las cuales se subministra a la red el estímulo incondicionado. Estas unidades de entrada, de acuerdo con el Submodelo de Red del Modelo DBP, tienen conexión directa con un tipo específico de unidad subcortical de las unidades ocultas, las unidades "vta" y con la, o las unidades de salida del tipo "CR/UR", es decir, las unidades de salida para respuestas condicionadas e incondicionadas, bajo aprendizaje pavloviano.

En la Figura 36 se muestran la parte de una red con 3 unidades de entrada para estímulos condicionados y 1 unidad para estímulo incondicionado.

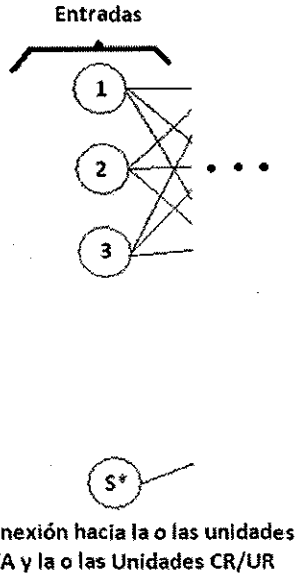


Figura 36. Unidades de entrada en un ejemplo típico de red bajo el modelo DBP

Generalmente, en el Modelo DBP se usa una sola unidad S*, pudiéndose usar más, si es necesario, bajo las mismas reglas de conexión descritas. Asimismo, se usan tantas unidades de entrada como componentes del estímulo incondicionado se hayan definido. Las conexiones de las unidades de entrada con las unidades ocultas se da a través de las unidades de tipo sensorial asociativo, o "sa", y no necesariamente se tiene que realizar una conexión entre cada unidad de entrada y cada unidad del tipo "sa", es decir que, de acuerdo con el problema específico que se trata de resolver, el diseñador de la red puede elegir entre interconectar completamente todas las entradas con las unidades "sa", o realizar solamente algunas conexiones entre ambos tipos.

En la Figura 37 se muestra parte de dos redes, una completamente interconectada entre las unidades de entrada y las "sa" (red 1) y otra parcialmente interconectada entre estos dos tipos de unidades (red 2).

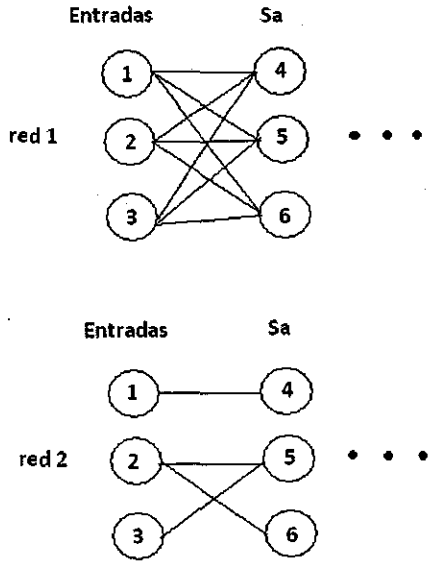


Figura 37. Ejemplo de entradas completamente conectadas y de entradas parcialmente conectadas

Así, en la Figura 37, la parte mostrada de la red 1, las entradas 1, 2 y 3 tienen conexión con las unidades "sa" 4, 5, y 6; mientras que en la parte mostrada de la red 2, la entrada 1 tiene conexión solamente con la unidad "sa" número 4, la entrada 2 tiene conexión con las unidades "sa" número 5 y 6, y la entrada 3 tiene solamente conexión con la unidad "sa" número 5.

La activación de las unidades de entrada se establece de acuerdo con arreglos de contingencias preespecificados, que representan los estímulos o los componentes de un estímulo que habrá de procesar la red, como señal de entrada en cada momento temporal. El arreglo de contingencias, entonces, deberá contener tantos componentes como número de unidades de entrada tenga la red, más uno o varios componentes correspondientes a la activación de la o las unidades de tipo "S*". Asimismo, deberá especificar la activación para cada unidad de entrada (incluyendo las "S*") en cada momento temporal que componga un "ensayo" completo.

En la tabla siguiente se muestra, de acuerdo con lo anterior, el valor de las activaciones que debe tener una red con 3 unidades de entrada y 1 unidad "S*" durante cada momento temporal en un arreglo compuesto por 8 de ellos. Nótese, que en este ejemplo, la unidad "S*" es activada solamente en el momento temporal número 8.

Ent	Ts1	Ts2	Ts3	Ts4	Ts5	Ts6	Ts7	Ts8
1	1	1	1	1	1	1	1	1
2	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0
S*	0	0	0	0	0	0	0	1

Tabla 1. Ejemplo de patrones de entrada por momento temporal

Formalmente, podemos describir las entradas en cada momento temporal como vectores, en donde, cada vector tendría dos componentes: el valor de activación de las unidades de entrada y el valor de activación de la unidad "S*".

Generalmente, las activaciones de entrada en el Modelo DBP se acotan a [0,1] con valores continuos, pero solamente es un convencionalismo, no una restricción del modelo.

4.1.2.2 Unidades ocultas

Las unidades ocultas pueden ser de dos tipos distintos:

a) Unidades tipo "sa", que son las que reciben la señal directamente de las unidades de entrada. Estas unidades se conectan a su vez, completa o parcialmente, con las unidades de tipo "ma". Asimismo, estas unidades tienen conexión completa o parcial, con las unidades tipo "ca1".

b) Unidades tipo "ma", que reciben la señal directamente de las unidades "sa" y se conectan a su vez, parcial o completamente con las unidades "vta" y con las unidades de salida.

4.1.2.3 Unidades de salida

Las unidades de salida pueden ser de dos tipos:

a) Unidades "CR/UR", que reciben conexiones directas de las unidades "S*" y de las unidades "ma", pudiendo haber conexiones completas entre este conjunto de unidades o pudiendo también estar parcialmente interconectadas. A este tipo de unidades se les conoce como unidades "respondientes" debido al uso específico en simulación de fenómenos del condicionamiento en el modelo DBP.

b) Unidades "R", que reciben conexiones directas de las unidades tipo "ma", ya sea de todas (interconexión completa), o solamente de algunas (interconexión parcial). A este tipo de unidades se les conoce como "operantes" debido al uso específico que se les da en la simulación de fenómenos de condicionamiento en el modelo DBP.

4.1.3 UNIDADES SUBCORTICALES

Las unidades subcorticales pueden ser de dos tipos:

a) Unidades tipo "ca1" o "hipocámpicas". Estas unidades reciben señal directamente de las unidades de tipo "sa", con las que pueden estar completamente interconectadas (es decir reciben una conexión por cada unidad de este tipo) o parcialmente interconectadas (reciben solo conexión de algunas de las unidades "sa").

b) Unidades tipo "vta", o "dopaminérgicas" que reciben conexión directamente de las unidades de entrada del tipo S* (estímulo incondicionado) y también de las unidades "ma", pudiendo estar interconectadas con éstas, completa o parcialmente.

Las unidades "ca1" y "vta" no envían sus activaciones directamente a alguna otra unidad, sino que tienen la función de modular los pesos de las conexiones de las unidades de la red, a través de las discrepancias identificadas entre sus activaciones en un momento dado y las del momento anterior. La forma específica en la que esta modulación se realiza, es descrita más adelante en el apartado que describe el Submodelo Neurocomputacional.

El Modelo DBP basa su desempeño en dos funciones que ocurren durante el paso de un estímulo por toda la arquitectura de la red hasta la producción de una respuesta: una función de activación, que es la que determina el valor que tendrá el impulso enviado a la siguiente capa de la red, y una función de aprendizaje, que es la que determina el grado de cambio que tendrá cada una de las conexiones entre las unidades de la red para adecuarse mejor a la respuesta requerida.

4.2 EL SUBMODELO NEUROCOMPUTACIONAL

El Submodelo Neurocomputacional establece el funcionamiento de una red bajo el Modelo DBP, conformada a partir de la arquitectura establecida en el Submodelo de Red descrito anteriormente. Este Submodelo tiene dos grandes fases: activación y aprendizaje.

4.2.1 ACTIVACIÓN

En Donahoe, Burgos y Palmer (1993), la activación se determina a partir de una función, conocida como "función de activación", que procesa dos tipos de entradas: excitatorias e inhibitorias, que pueden presentarse juntas o no, en la unidad, pero son procesadas por separado en su fase inicial. En todo momento, el valor de las activaciones es acotado a [0,1] con valores continuos.

Esta función es definida de la siguiente manera:

Siendo $N=\{x\in N|1\le x\le n\}$ y $P=\{j\in N|m\le j\le n\}$ los conjuntos de unidades neurales de procesamiento, en una red neural artificial, siendo N el conjunto de enteros positivos, n el número de unidades en la red y m el número de unidades de salida; y siendo $R=\{x\in\Re^{+}|0.0\le X\le 1.0\}$ el conjunto de posibles activaciones y valores de los pesos de conexión, donde $\Re^{+}$ es el conjunto de número reales positivos. $a: P\times T\rightarrow R$ es la función de activación, en donde $T\subset N$, representando los elementos de T eventos en el tiempo.

La regla para implementar la función a en las simulaciones de la red neural, se define como sigue:

Siendo $e(j,t)$ el vector de activaciones de las entradas excitatorias al elemento postsináptico j en el tiempo t; $i(j,t)$ el vector de activaciones de las entradas inhibitorias de j en t y $w(j,t)$ el vector de los pesos excitatorios asociados con j en t, y $w'(j,t)$ el vector de los pesos inhibitorios asociados con j en t, donde $j\in P$ y $t\in T$, asumiendo que $e(j,t), i(j,t), w(j,t)$ y $w'(j,t)\in R^{n}$ la cantidad de excitación (exc) e inhibición (inh) producida en j durante t, dada respectivamente por $exc(j,t)=e(j,t)\cdot w(j,t)$ e $inh(j,t)=i(j,t)\cdot w'(j,t)$.

La activación a en t, vista como la probabilidad de disparo, se define como sigue:

$$a_{j,t}=\begin{cases} L(exc_{j,t})+\tau_j L(exc_{j,t-1})[1-L(exc_{j,t})]-L(inh_{j,t}) \text{ si } exc_{j,t}>inh_{j,t} \text{ y } exc_{j,t}\ge\theta_{j,t} \\ a_{j,t}-k_j a_{j,t-1}(1-a_{j,t-1})-L(inh_{j,t-1}) \text{ si } exc_{j,t}>inh_{j,t} \text{ y } exc_{j,t}<\theta_{j,t} \\ 0 \text{ si } exc_{j,t}\le inh_{j,t} \end{cases} \quad (1)$$

En donde:

j es la unidad actual,
t es el momento temporal actual,
exc es la cantidad de excitación, definida por el producto interno de las activaciones excitatorias de las unidades presinápticas y los pesos de sus correspondientes conexiones,
inh es la cantidad de inhibición, definida por el producto interno de las activaciones inhibitorias de las unidades presinápticas y los pesos de sus correspondientes conexiones,
τ que denota la sumación temporal (parámetro libre que determina la proporción del valor de exc en el momento temporal anterior, que se usa en el momento temporal actual (este valor se puede fijar por el usuario, de manera independiente para cada unidad en la red),
θ que denota el umbral de excitación, valor que es calculado aleatoriamente entre 0 y 1 de acuerdo con una distribución Gaussiana, con una media y una desviación estándar predefinidas por el usuario
κ que denota la proporción de la activación de la unidad en el momento temporal anterior, con la que habrá de decrementarse el valor de la activación en el momento temporal actual. Este también es un parámetro libre predefinido por el usuario para cada unidad.

L es la función probabilística logística, con los parámetros $\gamma = .5$ y $\delta = .1$ [19], es definida como sigue:

$$L(x) = \frac{1}{1 + e^{\frac{-(x-\gamma)}{\delta}}} \qquad (2)$$

4.2.2 APRENDIZAJE (CÁLCULO DE LOS PESOS DE CONEXIÓN ENTRE LAS UNIDADES)

El aprendizaje se realiza bajo el modelo DBP a partir del ajuste de pesos de conexión entre las unidades. Los pesos se restringen a valores continuos entre [0,1]. El aprendizaje se basa en un mecanismo de competición entre las conexiones presinápticas de cada unidad, por una cantidad disponible de peso. Este ajuste se realiza por medio de una función llamada, función de aprendizaje, que se describe a continuación:

$$w_{i,j,t} = \begin{cases} w_{i,j,t-1} - \alpha_j a_{j,t} d_t p_{i,t} r_{j,t} & si\ d_t > 0 \\ w_{i,j,t-1} - \beta_j\, w_{i,j,t-1}\, a_{i,t}\, a_{j,t} & si\ d_t \leq 0 \end{cases} \qquad (3)$$

En donde:

i denota el elemento presináptico,
j el elemento postsináptico,
t es el momento temporal actual,
α es la tasa de adquisición (parámetro libre, predeterminado por el usuario),

[19] Valores usados por los autores en las simulaciones reportadas en Donahoe, Burgos y Palmer (1993).

β es la tasa de extinción (parámetro libre, predeterminado por el usuario),
p es la proporción con la que contribuye i ya sea en el valor de exc o el de inh ,
r denota la cantidad de peso disponible en j, y
d es la señal de discrepancia, que se calcula de acuerdo con lo siguiente:

$$d_t = \phi_t + v_t[1 - v_t]$$

En donde:

ϕ_t es el valor medio M de la discrepancia en las unidades "ca1", dada por: $\phi_t = |M[h_t - h_{t-1}]|$, en donde: h_t es el vector de activaciones de las unidades "ca1", y

v_t es el valor medio M de la discrepancia de las "vtas", dada por $v_t = M[v_t - v_{t-1}]$.

4.3 MÉTODO DE ACTUALIZACIÓN DE VALORES CALCULADOS

El método de actualización de pesos indica la manera en la que los pesos nuevos substituyen a los pesos anteriores en la red, de un momento temporal a otro, se basa en una selección aleatoria de las unidades a las cuales se les tiene que ajustar los pesos en un momento determinado. Este aspecto es muy importante para una adecuada implementación del modelo, ya que presenta una forma de realizar cálculos en una computadora secuencial, cercanos al efecto que obtendría en una computadora con capacidades de procesamiento en paralelo (Donahoe, Burgos y Palmer, 1993).

El método se describe en el siguiente algoritmo:

Para cada Momento Temporal de la red

1. Se conforma un conjunto de Unidades de procesamiento de la red, descartándose las unidades de entrada y las unidades S*,
2. Se selecciona aleatoriamente una unidad de este conjunto,
3. Se actualiza el valor de activación, calculando el nuevo valor de activación y reemplazándolo por el actual. El valor anterior de la activación se almacena en una variable aparte
4. Se repiten los pasos 2 y 3 hasta que se han actualizado las activaciones de todas las unidades en el conjunto
5. Se calculan las señales de discrepancia para las unidades "ca1" y "vta",
6. Se conforma un nuevo conjunto de unidades de procesamiento de la red, descartándose las unidades de entrada y las unidades S*,
7. Se selecciona aleatoriamente una unidad, sin modificar sus valores
8. Se genera aleatoriamente una lista con los pesos presinápticos de la unidad seleccionada,
9. Se actualizan los pesos de cada conexión en la lista, en el orden establecido en el paso 8,
10. Se repiten los pasos 7 al 9, hasta que se han agotado todas las unidades del conjunto conformado en el paso 6,

Algoritmo 6. Actualización de valores calculados en el Modelo DBP.

Previo a la realización del presente trabajo, se realizaron simulaciones con procedimientos de actualización de valores distintos al presentado en el algoritmo anterior, sin que se hayan obtenido resultados satisfactorios, por ello se enfatiza la importancia que tiene la realización de acuerdo con este procedimiento.

5. EL SIMULADOR CREADO

5.1 GENERALIDADES

El simulador fue programado en lenguaje C#, plataforma .NET, creado por Microsoft para computadoras personales que trabajan con el Sistema Operativo Windows XP. La versión empleada del lenguaje es la del 2007.

5.2 ESTRUCTURA DEL PROGRAMA

En el presente apartado se describen las funciones más importantes del simulador realizado. El código completo de la aplicación realizada, se encuentra incluido en los anexos 1 y 2 del presente documento.

5.2.1 ESTRUCTURAS DE DATOS

Las unidades, en la aplicación realizada, conformadas con base en el uso de una estructura que incluye los siguientes datos: Numero de unidad, Tipo de unidad, Tipo de activación, Valor de la activación en el tiempo actual y valor de la activación en el tiempo t-1, valor obtenido del paso de la función logística del valor obtenido tras la multiplicación de las activaciones presinápticas por los pesos de conexión de dichas unidades, los valores específicos de los parámetros libres de la unidad y que son independientes para cada una de ellas, el número de conexiones entrantes y finalmente, dos arreglos, uno para números enteros, en el que se almacenan las unidades presinápticas con las que la unidad tiene conexión y otro, para números de punto flotante, en el que se almacenan los pesos correspondientes a cada conexión correspondiente a las unidades almacenadas en el primer arreglo.

Asimismo, se declaró un arreglo de unidades, para almacenar los datos de la estructura anterior, correspondientes a cada una de las unidades que integrarán la red.

Una vez definidas estas estructuras de datos, fue posible cargar los valores de los parámetros correspondientes a cada unidad en la red.

El uso de estructuras permitió poder establecer el número de unidades en la red en forma dinámica, de acuerdo con el problema a resolver. Asimismo, permitió lograr el proceso con rapidez, ya que las estructuras se guardan en la memoria de acceso aleatorio de la computadora, sin retardo por lectura de información en disco duro.

5.2.2 FUNCIONES PROGRAMADAS

Las funciones programadas para el proceso de los datos almacenados en las estructuras de datos definidas, se describen a continuación.

5.2.2.1 Funciones para la suma de pesos excitatorios e inhibitorios presinápticos

Para sumar los pesos excitatorios y los pesos inhibitorios presinápticos de una unidad determinada, se definieron dos funciones por separado. Aunque el algoritmo es muy similar para ambas, la diferencia

radica en el uso del tipo de conexiones para cada una: conexiones excitatorias y conexiones inhibitorias.

La estrategia seguida para la suma de pesos excitatorios es la siguiente:

1. Se recibe como parámetro de la función un número entero que representa la unidad actual
2. Se declara la variable tot de tipo doble y se inicializa en 0.0
3. Se define un contador que va de 1 hasta el total de número deconexiones entrantes a la unidad actual
4. Para cada iteración del contador se realiza una comparación, enla que si la unidad presináptica es de tipo excitatorio entonces se suma a tot, el valor del peso correspondiente de esa conexión. En caso contrario el valor detot, se coloca en 0.0
5. La función arroja el valor de la variable tot, como resultado

Algoritmo 7. Suma de pesos excitatorios entrantes a la unidad actual.

La estrategia seguida para la suma de pesos inhibitorios es muy similar a la seguida para la suma de los pesos excitatorios. Se presenta a continuación:

1. Se recibe como parámetro de la función un úmero entero que representa la unidad actual
2. Se declara la variable tot de tipo doble y se inicializa en 0.0
3. Se define un contador que va de 1 hasta el total de número de conexiones entrantes a la unidad actual
4. Para cada iteración del contador se realiza ua comparación, en la que si la unidad presináptica es de tipo inhibitorio entonces se suma a tot, el valor del peso correspondiente de esa conexión. En caso contrario el valor de tot, se coloca en 0.0
5. La función arroja el valor de la variable tot, como resultado

Algoritmo 8. Suma de pesos inhibitorios entrantes a la unidad actual.

5.2.2.2 Funciones para la suma de la multiplicación de los pesos excitatorios e inhibitorios por la activación de la unidad presináptica correspondiente

Se establecieron funciones separadas para el cálculo del valor de las activaciones excitatorias presinápticas a la unidad seleccionada, por el valor del peso de conexión correspondiente.

La estrategia seguida para la suma de la multiplicación de los pesos excitatorios se presenta a continuación:

1. Se recibe como parámetro de la función un úmero entero que representa la unidad actual
2. Se declara la variable tot de tipo doble y se inicializa en 0.0
3. Se define un contador que va de 1 hasta el total de número de conexiones entrantes a la unidad actual
4. Para cada iteración del contador se realiza ua comparación, en la que si la unidad presináptica es de tipo excitatorio entonces se suma a tot, el valor del peso correspondiente de esa conexión multiplicado por el valor de la activación de la unidad presináptica correspondiente. En caso contrario el valor de tot, se coloca en 0.0
5. La función arroja el valor de la variable tot, como resultado

Algoritmo 9. Suma de la multiplicación de los pesos excitatorios entrantes a la unidad actual por la activación de la unidad presináptica correspondiente.

La estrategia seguida para la suma de la multiplicación de los pesos inhibitorios por la activación de la unidad presináptica correspondiente es muy similar a la seguida para la suma de los pesos excitatorios por su correspondiente activación presináptica. Se presenta a continuación:

1. Se recibe como parámetro de la función un ńmero entero que representa la unidad actual
2. Se declara la variable tot de tipo doble y se inicializa en 0.0
3. Se define un contador que va de 1 hasta el total de número de conexiones entrantes a la unidad actual
4. Para cada iteración del contador se realiza ua comparación, en la que si la unidad presináptica es de tipo inhibitorio entonces se suma a tot, el valor del peso correspondiente de esa conexión multiplicado por el valor de la activación de la unidad presináptica correspondiente. En caso contrario el valor de tot, se coloca en 0.0
5. La función arroja el valor de la variable tot, como resultado

Algoritmo 10. Suma de la multiplicación de los pesos inhibitorios entrantes a la unidad actual por la activación de la unidad presináptica correspondiente.

5.2.2.3 Función para el cálculo de un valor al pasarlo por la función logística

Se requirió de una función para poder calcular el valor de x al pasarlo por la función logística.

La estrategia seguida es la siguiente:

1. Se reciben como parámetros de la función un número enteroque representa la unidad actual y un número de tipo doble, que representa el valor a pasar por la función logística
2. Se declara la variable producto de tipo doble
3. Producto = 1/(1+Math.Exp((-x + Unidad[Cur_NPE].LogisMu)/Unidad[Cur_NPE].LogisSigma))
4. Si ((-x + Unidad[Cur_NPE].LogisMu) / Unidad[Cur_NPE].LogisSigma < -(11000)), entonces la función arroja 1.0 como resultado
5. Si ((-x + Unidad[Cur_NPE].LogisMu) / Unidad[CurNPE].LogisSigma > 11000), entonces la función, arroja 0.0 como resultado
6. Si no se dan las condiciones del paso 4 ó 5, entonces la función arroja el valor de la variable producto como resultado

Algoritmo 11. Cálculo del paso de un valor por la función logística.

5.2.2.4 Función para el cálculo de inh pasado por la función logística

Se requirió de una función para poder calcular el valor de inh al pasarlo por la función logística, distinguiéndolo del cálculo de los valores excitatorios, debido a que se requería de un paso especial, en el que si las inhibiciones multiplicadas por la activación correspondiente no es mayor a 0, se produjera como valor de salida un 0.0. En esencia, esta función hace uso de la función definida en el apartado anterior, para calcular el paso de éste por la función logística, si es mayor a cero.

La estrategia seguida es la siguiente:

1. Se reciben como parámetros de la función un número entero que representa la unidad actúal y un número de tipo doble, que representa el valor de inh a pasar por la función logística
2. Si la suma de los pesos de conexión de las unidades inhibitorias presinápticas multiplicado por la activación presináptica correspondiente,para la unidad actual, es mayor que 0, entonces se calcula el valor de esta suma, pasado por la función logística. Para ello este valor de la suma se pasa como parámetro, junto con el número de unidad, a la función definida en el apartado 5.2.2.3 de este mismo capítulo.
3. Si la suma no es mayor que 0.0, entoncesla función devuelve 0.0 como valor resultante

Algoritmo 12. Cálculo del paso de inh por la función logística.

5.2.2.5 Función para el cálculo del valor de la activación de una unidad determinada

La estrategia seguida para el cálculo de una activación es la siguiente:

1. Se definen las variables thr, p_epsp_t_1, exc, inh de tipo doble
2. Se obtiene el valor actual de p_epsp_t para la unidad y se pasa a la variable local que almacena este valor en t-1 : p_epsp_t_1
3. exc = a la suma de los pesos excitatorios por las activacionesexcitatorias correspondientes, que entran a la unidad actual
4. Se pasa exc por la función logística y se almacena en la variable global p_epsp_t en la estructura Unidad[], correspondiente a la unidad actual
5. inh = al Cálculo de p_ipsp para la unidad actual
6. Se calcula thr = Que es un número aleatorio gaussiano con Thr_mu como media y Thr_sigma como desviación estándar, estos parámetros son tomados de los valores correspondientes en las variables definidas en la estructura Unidad[]
7. Si (Unidad[Cur_NPE].p_epsp_t > inh) y si (Unidad[Cur_NPE].p_epsp_t >= thr) entonces: la función devuelve (Unidad[Cur_NPE].p_epsp_t + Unidad[Cur_NPE].Tau * p_epsp_t_1 * (1 - Unidad[Cur_NPE].p_epsp_t)) - inh;
8. Si (Unidad[Cur_NPE].p_epsp_t > inh) pero no (Unidad[Cur_NPE].p_epsp_t >= thr), entonces se realiza la siguiente comparación: if ((Unidad[Cur_NPE].Kappa * Unidad[Cur_NPE].Valor_Activacion < Unidad[Cur_NPE].Valor_Activacion)&& (Unidad[Cur_NPE].Valor_Activacion- Unidad[Cur_NPE].Kappa * Unidad[Cur_NPE].Valor_Activacion > inh)), entonces: la función devuelve (Unidad[Cur_NPE].Valor_Activacion- Unidad[Cur_NPE].Kappa * Unidad[Cur_NPE].Valor_Activacion) - inh; si no, entonces devuelve 0.0 como valor de salida.

Algoritmo 13. Cálculo de la activación para la unidad determinada.

5.2.2.6 Función para el cálculo del valor de la discrepancia de las unidades "ca1"

La estrategia seguida para el cálculo de la discrepancia de las unidades "ca1" es la siguiente:

1. Se define una variable local llamada Neurona de tipo int;
2. Se coloca la variable global Tot_CA1_d_t = 0.0;
3. Se hace un recorrido de todas las neuronas en la red, y en cada iteración se comprueba si la unidad current es de tipo "ca1", en cuyo caso se calcula Tot_CA1_d_t = Tot_CA1_d_t + Math.Abs(Unidad[Neurona].Valor_Activacion- Unidad[Neurona].Valor_Activacion_Tmenos1);
4. Una vez realizado el barrido, se realiza el cálculo del valor de la variable global Avg_CA1_d_t mediante el llamado de la función para el cálculo del promedio de las discrepancias de las unidades ca1;

Algoritmo 14. Cálculo de la discrepancia de las unidades ca1.

5.2.2.7 Función para el cálculo del valor de la discrepancia de las unidades "vta"

La estrategia seguida para el cálculo del valor de la discrepancia para las unidades "vta" se presenta a continuación.

```
1. Se define una variable local llamada Neurona de tipo int;
2. Se coloca la variable global Tot_CA1_d_t = 0.0;
3. Se hace un recorrido de todas las neuronas en la red, y en cada iteración se
   comprueba si la unidad current es de tipo "vta", en cuyo caso se calcula
   Tot_VTA_d_t = Tot_VTA_d_t + (Unidad[Neurona].Valor_Activacion
   Unidad[Neurona].Valor_Activacion_Tmenos1);
4. Una vez realizado el barrido, se realiza el cálculo del valor de la variable
   global  Avg_VTA_d_t mediante el llamado de la función para el cálculo del
   promedio de las discrepancias de las unidades vta;
```

Algoritmo 15. Cálculo de la discrepancia de las unidades vta.

5.2.2.8 Función para el cálculo del valor promedio de las discrepancias de las unidades "ca1"

La estrategia seguida para el cálculo del valor promedio de las discrepancias para las unidades "ca1" se presenta a continuación.

```
1. Si el número total de unidades ca1 es mayor a 0, entonces se divide
   Tot_CA1_d_t entre el número de unidades ca1 y el resultado se pasa como valor
   resultante de la función.
2. Si el número total de unidades ca1 no es mayor a 0, entonces el valor
   resultante de la función es 0.0
```

Algoritmo 16. Cálculo del valor promedio de la discrepancia de las unidades ca1.

5.2.2.9 Función para el cálculo del valor promedio de las discrepancias de las unidades "vta"

La estrategia seguida para el cálculo del valor promedio de las discrepancias para las unidades "vta" se presenta a continuación.

```
1. Si el número total de unidades vta es mayor a 0, entonces se divide
   Tot_vta_d_t entre el número de unidades vta y el resultado se pasa como valor
   resultante de la función.
2. Si el número total de unidades ca1 no es mayor a0, entonces el valor
   resultante de la función es 0.0
```

Algoritmo 17. Cálculo del valor promedio de la discrepancia de las unidades vta.

5.2.2.10 Función para el cálculo del valor de la discrepancia de las unidades "ca1" amplificado

La estrategia seguida para el cálculo del valor de las discrepancias para las unidades "vta" amplificado se presenta a continuación.

```
1. Si ((Avg_CA1_d_t > 0.0) && (Avg_VTA_d_t > 0.0)), entonces Avg_CA1_d_t =
   Avg_CA1_d_t + (Avg_VTA_d_t * (1 - Avg_CA1_d_t));
2. Si no, entonces la discrepancia de las ca1 no se amplifica
```

Algoritmo 18. Cálculo del valor amplificado de la discrepancia de las unidades ca1.

5.2.2.11 Función para la actualización de los valores de activación

La estrategia seguida para la actualización de los valores de activación se presenta a continuación.

1. Se definen dos variables locales de tipo entero: TimeStep y Neurona;
2. Para cada TimeStep definido:
   a. Se meten los valores del array de entradas como activaciones para las unidades de entrada.
   b. Se crea una lista con los números de las unidades en forma aleatoria
   c. Se elige una unidad de esta lista
   d. Si la unidad no es unidad de entrada, entonces se pasa su valor actual de activación a la variable para almacenar la activación de esa neurona en T menos1, y a continuación de calcula la activación, llamando a la función para cálculo de la activación, pasando como parámetro el número de la unidad.
   e. Si la unidad es VTA o CR/UR y el Momento temporal es reforzado, entonces se pasa el valor de la activación de la unidad S* como activación de las unidades de estos dos tipos
   f. Se repiten los pasos desde c. hasta que no hay más unidades.

Algoritmo 19. Actualización de los valores de activación.

5.2.2.12 Función para el cálculo del fortalecimiento de pesos

La estrategia seguida para el cálculo del fortalecimiento de pesos se presenta a continuación.

1. Se definen las siguientes variables como tipo doble Tot_Wgt, N_t, pi_t, dWij_t, y como entero la variable dd;
2. Se coloca dWij_t = 0.0;
3. Si la suma de pesos entrantes por las activaciones presinápticas de la unidad actual es mayor a 0, entonces Tot_Wgt = Sum_Exc_Wgts(Cur_NPE) y N_t = Nr- Tot_Wgt;
4. Se crea una lista aleatoria con las conexiones entrantes a la unidad current y para cada conexión excitatoria, se calcula pi_t = (Unidad[Unidad[Cur_NPE].Unidad_Presinaptica[ArrayConexionesElegidas[dd]]].Valor_Activacion * Unidad[Cur_NPE].Peso_Presinaptico[ArrayConexionesElegidas[dd]]) / Sum_Exc_ActWgt(Cur_NPE); dWij_t = Unidad[Cur_NPE].Alpha * Unidad[Cur_NPE].Valor_Activacion * d * pi_t * N_t;
5. Se coloca el nuevo peso realizando la siguiente suma: Unidad[Cur_NPE].Peso_Presinaptico[ArrayConexionesElegidas[dd]] = Unidad[Cur_NPE].Peso_Presinaptico[ArrayConexionesElegidas[dd]] + dWij_t;
6. Se regresa el valor de dWij_t a 0.0;
7. Si la suma de los pesos por las activaciones entrantes inhibitorias es mayor que 0, entonces Tot_Wgt = Sum_Inh_Wgts(Cur_NPE) y N_t = Nr- Tot_Wgt;
8. Se crea una lista aleatoria con las conexiones inhibitorias y se barren todas calculando pi_t = (Unidad[Unidad[Cur_NPE].Unidad_Presinaptica[ArrayConexionesElegidas[dd]]].Valor_Activacion * Unidad[Cur_NPE].Peso_Presinaptico[ArrayConexionesElegidas[dd]]) / Sum_Inh_ActWgt(Cur_NPE); y dWij_t = Unidad[Cur_NPE].Alpha * Unidad[Cur_NPE].Valor_Activacion * d * pi_t * N_t;
9. Se coloca el nuevo peso, realizando la siguiente suma: Unidad[Cur_NPE].Peso_Presinaptico[ArrayConexionesElegidas[dd]] = Unidad[Cur_NPE].Peso_Presinaptico[ArrayConexionesElegidas[dd]] + dWij_t;

Algoritmo 20. Cálculo del fortalecimiento de pesos.

5.2.2.13 Función para el cálculo del debilitamiento de pesos

La estrategia seguida para el cálculo del debilitamiento de pesos se presenta a continuación.

```
1. Se definen las siguientes variables localesdouble dWij_t;int dd; y se coloca
   el valor de dWij_t = 0.0;
2. Se genera una lista aleatoria de las conexiones exitatorias y se barre toda,
   realizando lo siguiente para cadauna: dWij_t = Unidad[Cur_NPE].Beta *
   Unidad[Unidad[Cur_NPE].Unidad_Presinaptica[ArrayConexionesElegidas[dd]]].Valor
   _Activacion * Unidad[Cur_NPE].Valor_Activacion *
   Unidad[Cur_NPE].Peso_Presinaptico[ArrayConexionesElegidas[dd]];
3. Se establece el nuevo peso de acuerdo con la siguiente suma:
   Unidad[Cur_NPE].Peso_Presinaptico[ArrayConexionesElegidas[dd]] =
   Unidad[Cur_NPE].Peso_Presinaptico[ArrayConexionesElegidas[dd]]- dWij_t;
```

Algoritmo 21. Actualización de los valores de activación.

5.2.2.14 Función para la actualización de los pesos de conexión

La estrategia seguida para la actualización de los de pesos de conexión se presenta a continuación.

```
1. Se crea una lista aleatoria de las unidades de la red, y
2. Para cada unidad de tipo sa o ca1, Si (Avg_CA1_d_t >= d_thr) entonces se pasa
   a modo de fortalecimiento de los pesos, si no, se pasa a modo de
   debilitamiento de pesos, en ambos casos se pasa como parámetro el promedio de
   la discrepancia CA1
3. Para cada unidad de tipo ma, vta, R, o CR/UR, si (Avg_VTA_d_t >= d_thr)
   entonces se pasa a modo de fortalecimiento de pesos, si no, se pasa a modo de
   debilitamiento de pesos, en ambos casos se pasa como parámetro el promerio de
   la discrepancia VTA
```

Algoritmo 22. Actualización de los pesos de conexión.

5.2.2.15 Función para la re-inicialización de activaciones

La estrategia seguida para la re-inicialización de las activaciones, se presenta a continuación.

```
1. Para cada unidad diferente de las unidades de entrada, se coloca el valor de
   la función logística con argumento 0, como valor de activación actual
```

Algoritmo 23. Re-inicialización de las activaciones.

5.2.2.16 Función para la actualización de la red

La estrategia seguida para la actualización de la red se presenta a continuación.

```
1. Para cada ensayo definido:
   a. Se actualizan activaciones
   b. Se calculan las discrepancias ca1
   c. Se calculan las discrepancias vta
   d. Se amplifica la señal de la discrepancia de ca1
   e. Si la opción de aprendizaje de la red está activado, entonces se actualizan
      los pesos
   f. Se reinicializan las activaciones
```

Algoritmo 24. Actualización de la red en cada ensayo

5.3 PRUEBA DEL SIMULADOR CREADO Y RESULTADOS COMPARADOS CON SELNET

Se empleó una red de tres entradas considerando que es la arquitectura y configuración más usada en los trabajos previos. Todas las corridas se hicieron considerando 300 ensayos compuestos cada uno de 8 momentos temporales. Se registra la actividad de la unidad CR/UR en el momento temporal 7 y cuando el refuerzo es administrado se hace en el momento temporal 8 a través de la unidad S*

Se realizaron 10 replicaciones con el simulador creado y 10 replicaciones usando el simulador SELNET.

5.3.1 ESTRUCTURA DE LA RED

La red empleada se compuso de 12 unidades, de las cuales 3 son unidades de entrada,3 unidades de tipo "sa" completamente interconectadas a las 3 unidades de la capa de entrada; 1 unidad de tipo "ca1", que recibe conexiones de todas las unidades de tipo "sa", 3 unidades de tipo "ma" completamente interconectadas con las unidades "sa", 1 unidad de tipo "vta" que recibe conexiones de todas las unidades de tipo "ma" y 1 unidad de salida, tipo "CR/UR", que recibe conexiones de todas las unidades de tipo "ma". Esta red incluyó 27 conexiones en total. Los valores para todas las unidades fueron los que típicamente se han usado en las simulaciones previas con el DBP:

- Tasa de Incremento de pesos (α) = 0.5;
- Tasa de decremento de pesos (β) = 0.1;
- Desviación estándar de la función logística (δ)= 0.1;
- Parámetro de sumación temporal (τ) = 0.1;
- Parámetro de decaimiento ($\aleph$) de 0.01.

La arquitectura empleada se muestra en la Figura 38:

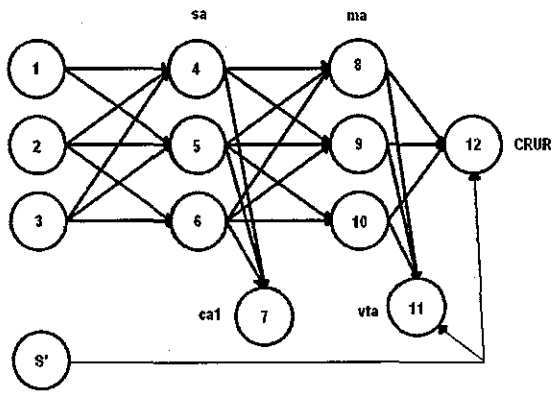


Figura 38. Arquitectura empleada para la prueba del simulador creado

5.3.2 PATRONES DE ENTRADA USADOS

La simulación se integró por dos condiciones: la primera, en la que se presentó el estímulo "A" reforzado, y la segunda, en la que se presentó el mismo estímulo "A" sin refuerzo.

Las condiciones establecidas para esta simulación se pueden esquematizar empleando barras verticales que denotan el fin de una condición y el inicio de la siguiente, las letras mayúsculas representan el estímulo presentado y los signos positivo (+) o negativo (-) indican si el estímulo fue reforzado o no, respectivamente. El esquema de condiciones para esta simulación fue: A+ | A-.

Los patrones de entrada usados para la presentación del estímulo "A" reforzado, en cada momento temporal de cada ensayo, se presentan en la siguiente tabla:

Ent	Ts1	Ts2	Ts3	Ts4	Ts5	Ts6	Ts7	Ts8
1	1	1	1	1	1	1	1	1
2	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0
S*	0	0	0	0	0	0	0	1

Tabla 2. Patrones de entrada por momento temporal correspondientes a la primera condición (A+) presentados en la simulación de prueba

Los patrones de entrada usados para la presentación del estímulo "A" sin refuerzo, en cada momento temporal de cada ensayo, se presentan en la siguiente tabla:

Ent	Ts1	Ts2	Ts3	Ts4	Ts5	Ts6	Ts7	Ts8
1	1	1	1	1	1	1	1	1
2	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0
S*	0	0	0	0	0	0	0	0

Tabla 3. Patrones de entrada por momento temporal correspondientes a la segunda condición (A-) presentados en la simulación de prueba

5.3.3 RESULTADOS OBTENIDOS

En la Figura 39 se muestran los valores de activación promedio, en rangos de 50 ensayos, para la unidad "CR/UR" (unidad número 12), obtenidos con el Simulador SELNET:

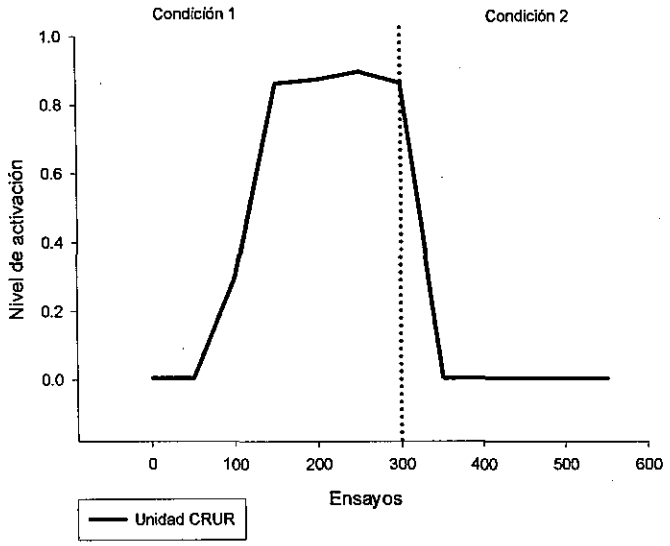


Figura 39. Niveles de activación promedio de la unidad 12 (CR/UR) después de la simulación de prueba, empleando el Simulador SELNET

En la Figura 40 se muestran los valores de activación promedio, en rangos de 50 ensayos, de la unidad CR/UR (unidad número 12), obtenidos con el simulador creado en este trabajo:

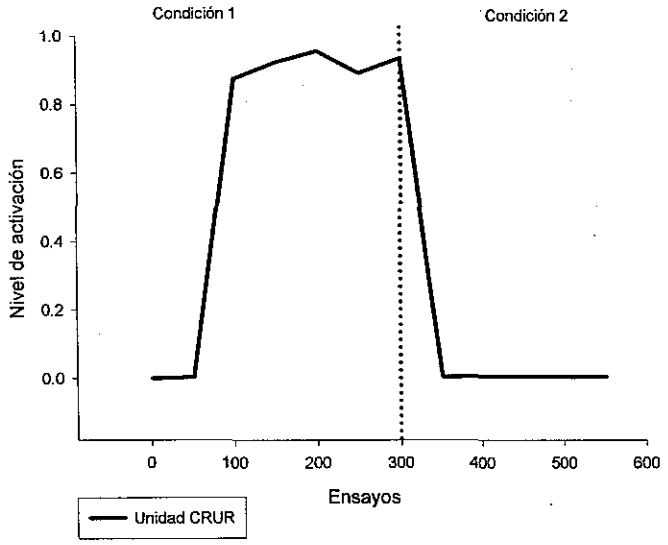


Figura 40. Niveles de activación promedio de la unidad 12 (CR/UR) después de la simulación de prueba empleando el simulador propio.

Para efectos de la simulación no se observan diferencias significativas entre ambos simuladores, en ambos casos el nivel de activación se incrementa entre el ensayo 50 y 100; se sostiene por arriba del 0.85 entre los ensayos 100 y 300; y sufre un decremento entre el ensayo 300 al 350, para mantener valores cercanos al 0.0 entre el ensayo 350 y el 600. Este es el comportamiento esperado para este tipo de condiciones presentadas a una red que funcione bajo el modelo DBP.

6. LA BÚSQUEDA DEL MANEJO DE DOS EC CON LA ARQUITECTURA CLÁSICA EN EL MODELO DBP.

Un problema que presenta actualmente el Modelo DBP es la imposibilidad, de que mediante las arquitecturas usadas hasta la fecha, se puedan manejar dos o más estímulos condicionados a la vez (Ponce, 2003).

La arquitectura clásica consiste en una red completamente interconectada, con la siguiente estructura:

$Red\ Clásica = \{E, sa, ca1, ma, vta, CR/UR\}$,

En donde:

E es el conjunto de entradas tal que $E = \{E_1 \dots E_n\}$,
sa es el conjunto de unidades "sa" tal que $sa = \{sa_1 \dots sa_m\}$, en donde cada unidad en el conjunto recibe una conexión de todas y cada una de las entradas,
$ca1$ es el conjunto de entradas tal que $ca1 = \{ca1_1 \dots ca1_o\}$, en donde cada unidad en el conjunto recibe una conexión de todas y cada una de las unidades "sa",
ma es el conjunto de entradas tal que $ma = \{ma_1 \dots ma_p\}$, en donde cada unidad en el conjunto recibe una conexión de todas y cada una de las unidades "sa",
vta es el conjunto de entradas tal que $vta = \{vta_1 \dots vta_q\}$, en donde cada unidad en el conjunto recibe una conexión de todas y cada una de las unidades "ma",
CR/UR es el conjunto de entradas tal que $CR/UR = \{CR/UR_1 \dots CR/UR_r\}$, en donde cada unidad en el conjunto recibe conexión de todas y cada una de las unidades "ma".

En este apartado se presenta un conjunto de simulaciones realizadas para demostrar esta imposibilidad. Estas simulaciones fueron realizadas con el simulador propio y verificadas con el Simulador SELNET, sin encontrar diferencias significativas.

Para poder explorar el problema se definió la siguiente estrategia:

- Las simulaciones se realizaron empleando 4 tamaños distintos de red: una red "pequeña" compuesta por 12 unidades; una red "mediana" compuesta por 24 unidades; una red "grande" compuesta por 36 unidades; y finalmente, una red "pequeña con 8 entradas", en la que se explora el comportamiento de una red bajo la arquitectura clásica, de tamaño "pequeño", pero con un mayor número de entradas (8).

- Con estas redes se corrieron simulaciones bajo 3 diferentes esquemas de entrenamiento, dos con dos condiciones y uno con tres condiciones: A+|B+; A+/B+; y A+|A-|B+.

- Finalmente, se corrieron simulaciones en la red "pequeña con 8 entradas" empleando diferentes valores de entrada en los patrones a procesar.

Todas las corridas se hicieron considerando 300 ensayos compuestos cada uno, de 8 momentos temporales. Se registró la actividad de la unidad "CR/UR" en el momento temporal 7 y cuando el

refuerzo fue administrado, se hizo en el momento temporal 8, a través de la unidad "S*", la que se activó con un valor de 1.0

Los resultados de la simulación manipulando estas variables se muestran a continuación

6.1 ESTRUCTURA DE LAS REDES EMPLEADAS

Se conformaron cuatro tipos de redes distintas en cuanto a su tamaño, las cuales fueron etiquetadas como "Pequeña", "Mediana", "Grande", y "Pequeña con 8 entradas", con la finalidad de verificar que los resultados no dependen del tamaño de la red. Todas las redes empleadas en este apartado fueron estructuradas con base en la arquitectura "clásica" usada en la literatura relacionada con el Modelo DBP.

Los valores para todas las unidades, en los tres tamaños de red, fueron los que típicamente se han usado en las simulaciones previas con el DBP:

- Tasa de Incremento de pesos (α) = 0.5;
- Tasa de decremento de pesos (β) = 0.1;
- Desviación estándar de la función logística (δ)= 0.1;
- Parámetro de sumación temporal (τ) = 0.1;
- Parámetro de decaimiento ($\aleph$) de 0.01.

6.1.1 RED "PEQUEÑA"

La red para esta simulación se estructuró en la forma clásica reportada en la literatura generada con el uso del modelo. Se compuso de 12 unidades, de las cuales 3 son unidades de entrada,3 unidades de tipo "sa" completamente interconectadas a las 3 unidades de la capa de entrada; 1 unidad de tipo "ca1", que recibe conexiones de todas las unidades de tipo "sa", 3 unidades de tipo "ma" completamente interconectadas con las unidades "sa", 1 unidad de tipo "vta" que recibe conexiones de todas las unidades de tipo "ma" y 1 unidad de salida, tipo "CR/UR", que recibe conexiones de todas las unidades de tipo "ma". Esta red incluyó 27 conexiones en total, tratándose de una red completamente interconectada.

La Arquitectura empleada se muestra en la Figura 41:

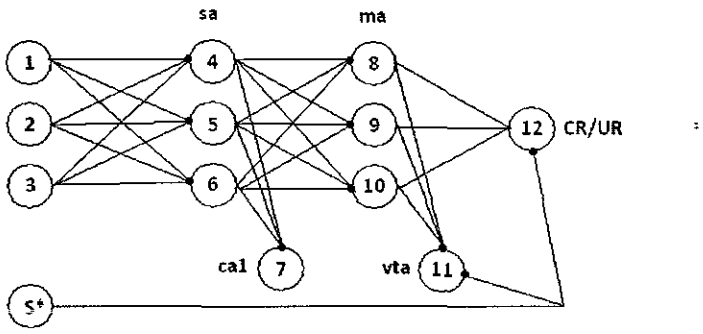


Figura 41. Arquitectura de una red "pequeña" completamente interconectada

6.1.2 RED "MEDIANA"

La red para esta simulación se estructuró en la forma clásica reportada en la literatura generada con el uso del modelo. Se compuso de 24 unidades, de las cuales 7 son unidades de entrada,7 unidades de tipo "sa" completamente interconectadas a las 7 unidades de la capa de entrada; 1 unidad de tipo "ca1", que recibe conexiones de todas las unidades de tipo "sa", 7 unidades de tipo "ma" completamente interconectadas con las unidades "sa", 1 unidad de tipo "vta" que recibe conexiones de todas las unidades de tipo "ma" y 1 unidad de salida, tipo "CR/UR", que recibe conexiones de todas las unidades de tipo "ma". Esta red incluyó 119 conexiones en total, tratándose de una red completamente interconectada.

La Arquitectura empleada se muestra en la Figura 42:

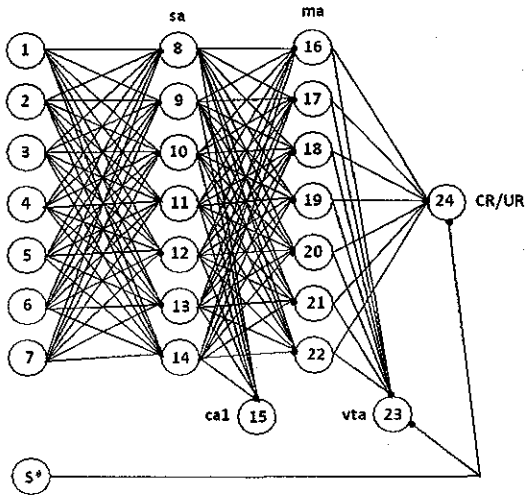


Figura 42. Arquitectura de una red "mediana" completamente interconectada

6.1.3 RED "GRANDE"

La red para esta simulación se estructuró en la forma clásica reportada en la literatura generada con el uso del modelo. Se compuso de 36 unidades, de las cuales 11 son unidades de entrada, 11 unidades de tipo "sa" completamente interconectadas a las 11 unidades de la capa de entrada; 1 unidad de tipo "ca1", que recibe conexiones de todas las unidades de tipo "sa", 11 unidades de tipo "ma" completamente interconectadas con las unidades "sa", 1 unidad de tipo "vta" que recibe conexiones de todas las unidades de tipo "ma" y 1 unidad de salida, tipo "CR/UR", que recibe conexiones de todas las unidades de tipo "ma". Esta red incluyó 275 conexiones en total, tratándose de una red completamente interconectada.

La Arquitectura empleada se muestra en la Figura 43:

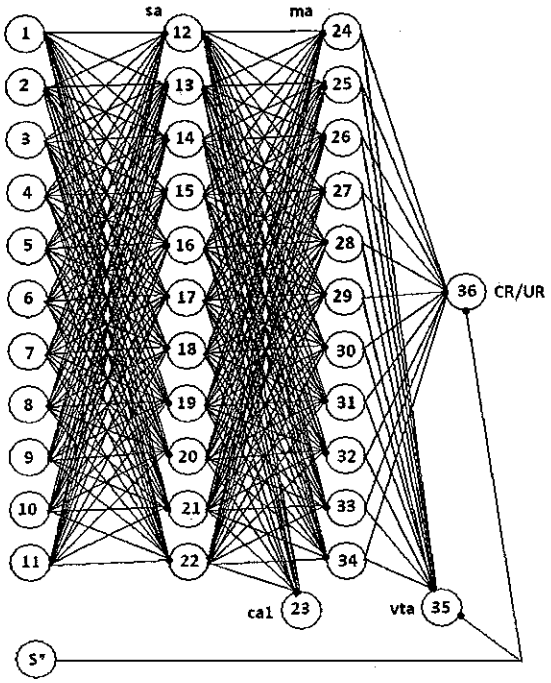


Figura 43. Arquitectura de una red "grande" completamente interconectada

6.1.4 RED "PEQUEÑA CON 8 ENTRADAS"

La red para esta simulación se compuso de 17 unidades, de las cuales 8 son unidades de entrada, 3 unidades de tipo "sa" completamente interconectadas a las 8 unidades de la capa de entrada, 1 unidad de tipo "cal", que recibe conexiones de todas las unidades de tipo "sa", 3 unidades de tipo "ma" completamente interconectadas con las unidades tipo "sa", 1 unidad de tipo "vta" que recibe conexiones de todas las unidades de tipo "ma" y 1 unidad de salida, tipo "CR/UR", que recibe conexiones de todas las unidades de tipo "ma". Esta red incluyó 42 conexiones en total.

La Arquitectura empleada se muestra en la Figura 44:

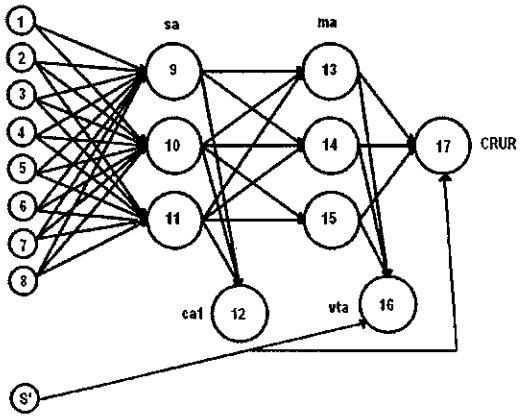


Figura 44. Arquitectura de una red "pequeña con 8 entradas".

6.2 EL USO DE DIFERENTES ESQUEMAS DE ENTRENAMIENTO.

En las siguientes 12 simulaciones se probaron tres esquemas de entrenamiento distintos: A+|B+, en las simulaciones 1, 2, 3 y 4; A+/B+ en las simulaciones 5, 6, 7 y 8; y A+|A-|B+ en las simulaciones 9, 10, 11 y 12.

6.2.1 SIMULACIONES 1, 2, 3 Y 4.

En estas primeras cuatro simulaciones se sometió a dos condiciones distintas a las redes "Pequeña", "Mediana", "Grande" y "Pequeña con 8 entradas". En la primera condición se reforzó el estímulo "A", en la segunda condición se presentó un estímulo diferente, denominado "B", el cual recibió refuerzo. El esquema de condiciones para estas tres simulaciones fue A+|B+. Se realizaron 10 replicaciones de cada simulación, y cada condición se compuso de 300 ensayos. No se empleó re-inicialización de pesos entre la condición 1 y la condición 2.

Los patrones específicos de entrada empleados en cada red para cada condición se muestran en las siguientes tablas.

6.2.1.1 Patrones de entrada para la red "Pequeña".

En las tablas 4 y 5 se presentan los patrones de entrada por cada momento temporal, presentados en la simulación número 1.

Ent	Ts1	Ts2	Ts3	Ts4	Ts5	Ts6	Ts7	Ts8
1	1	1	1	1	1	1	1	1
2	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0
S*	0	0	0	0	0	0	0	1

Tabla 4. Patrones de entrada por momento temporal, presentados en la condición A+ en la simulación 1

Ent	Ts1	Ts2	Ts3	Ts4	Ts5	Ts6	Ts7	Ts8
1	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0
3	1	1	1	1	1	1	1	1
S*	0	0	0	0	0	0	0	1

Tabla 5. Patrones de entrada por momento temporal, presentados en la condición B+ en la simulación 1

6.2.1.2 Patrones de entrada para la red "Mediana".

En las tablas 6 y 7 se presentan los patrones de entrada por cada momento temporal, presentados en la simulación número 2.

Ent	Ts1	Ts2	Ts3	Ts4	Ts5	Ts6	Ts7	Ts8
1	1	1	1	1	1	1	1	1
2	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0
S*	0	0	0	0	0	0	0	1

Tabla 6. Patrones de entrada por momento temporal, presentados en la condición A+ en la simulación 2

Ent	Ts1	Ts2	Ts3	Ts4	Ts5	Ts6	Ts7	Ts8
1	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0
7	1	1	1	1	1	1	1	1
S*	0	0	0	0	0	0	0	1

Tabla 7. Patrones de entrada por momento temporal, presentados en la condición B+ en la simulación 2

6.2.1.3 Patrones de entrada para la red "Grande".

En las tablas 8 y 9 se presentan los patrones de entrada por cada momento temporal, presentados en la simulación número 3.

Ent	Ts1	Ts2	Ts3	Ts4	Ts5	Ts6	Ts7	Ts8
1	1	1	1	1	1	1	1	1
2	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0
11	0	0	0	0	0	0	0	0
S*	0	0	0	0	0	0	0	1

Tabla 8. Patrones de entrada por momento temporal, presentados en la condición A+ en la simulación 3

Ent	Ts1	Ts2	Ts3	Ts4	Ts5	Ts6	Ts7	Ts8
1	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0
11	1	1	1	1	1	1	1	1
S*	0	0	0	0	0	0	0	1

Tabla 9. Patrones de entrada por momento temporal, presentados en la condición B+ en la simulación 3

6.2.1.4 Patrones de entrada para la red "Pequeña con 8 entradas".

En las tablas 10 y 11 se presentan los patrones de entrada por cada momento temporal, presentados en la simulación número 4.

Ent	Ts1	Ts2	Ts3	Ts4	Ts5	Ts6	Ts7	Ts8
1	1	1	1	1	1	1	1	1
2	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	0
S*	0	0	0	0	0	0	0	1

Tabla 10. Patrones de entrada por momento temporal, presentados en la condición A+ en la simulación 4

Ent	Ts1	Ts2	Ts3	Ts4	Ts5	Ts6	Ts7	Ts8
1	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0
8	1	1	1	1	1	1	1	1
S*	0	0	0	0	0	0	0	1

Tabla 11. Patrones de entrada por momento temporal, presentados en la condición B+ en la simulación 4

6.2.1.5 Resultado de las simulaciones

En la Figura 45 se muestran los valores de activación registrados en la unidad "CR/UR" en las cuatro simulaciones realizadas bajo este esquema:

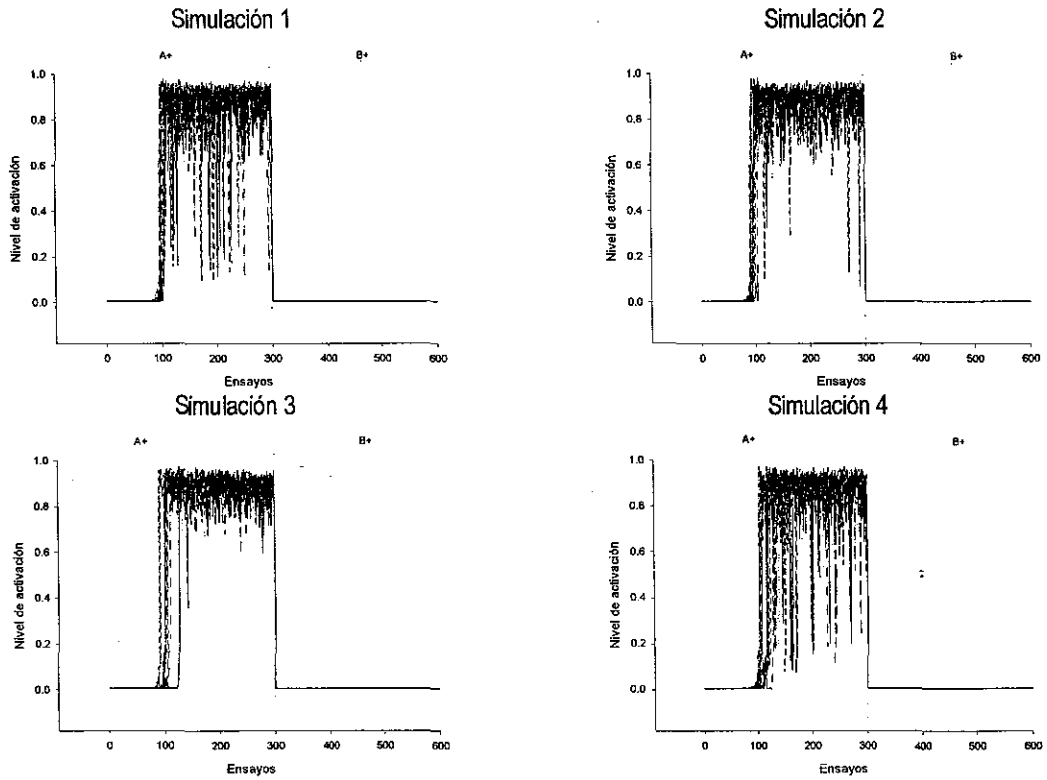


Figura 45. Niveles de activación registrados en las unidades "CR/UR" en las simulaciones 1, 2, 3 y 4, usando el esquema A+|B+.

Se puede observar en la Figura 45, que la arquitectura clásica no permite que se presenten niveles elevados de activación en la unidad "CR/UR", ante un segundo estímulo condicionado, una vez que se ha configurado su estructura para presentar niveles elevados de activación en esta unidad de salida. Asimismo, la Figura muestra que esta imposibilidad para responder ante un segundo estímulo condicionado, no cambia con el tamaño de la red, ni con el tamaño de las entradas que se presentan a una misma red.

Debido a la naturaleza de las entradas presentadas en la red y la estructura de las unidades de entrada, en la arquitectura usada, no existe diferencia alguna si se presentan los estímulos en el orden inverso, es decir, bajo un esquema B+|A+.

6.2.2 SIMULACIONES 5, 6, 7 Y 8.

En las simulaciones 5, 6, 7 y 8, se sometió a las redes "Pequeña", "Mediana", "Grande" y "Pequeña con 8 entradas", a una sola condición, en la que se presentaron alternadamente, los estímulos A y B reforzados.

El esquema de condiciones para estas tres simulaciones fue A+/B+. Se corrieron 10 replicaciones de cada una de estas simulaciones.

Se realizaron 600 ensayos para esta condición, en la que se presentaron en forma alternada los estímulos reforzados. Los patrones específicos de entrada empleados en cada red se presentan a continuación.

6.2.2.1 Patrones de entrada para la red "Pequeña"

En la tabla 12 se presentan los patrones de entrada correspondiente a cada estímulo, presentados alternadamente en la simulación número 5.

Ent	A+	B+
1	1	0
2	0	0
3	0	1
S*	1	1

Tabla 12. Patrones de entrada presentados alternadamente en la única condición de la simulación 5

6.2.2.2 Patrones de entrada para la red "Mediana"

En la tabla 13 se presentan los patrones de entrada correspondiente a cada estímulo, presentados alternadamente en la simulación número 6.

Ent	A+	B+
1	1	0
2	0	0
3	0	0
4	0	0
5	0	0
6	0	0
7	0	1
S*	1	1

Tabla 13. Patrones de entrada presentados alternadamente en la única condición de la simulación 6

6.2.2.3 Patrones de entrada para la red "Grande"

En la tabla 14 se presentan los patrones de entrada correspondiente a cada estímulo, presentados alternadamente en la simulación número 7.

Ent	A+	B+
1	1	0
2	0	0
3	0	0
4	0	0
5	0	0
6	0	0
7	0	0
8	0	0
9	0	0
10	0	0
11	0	1
S*	1	1

Tabla 14. Patrones de entrada presentados alternadamente en la única condición de la simulación 7

6.2.2.4 Patrones de entrada para la red "Pequeña con 8 entradas"

En la tabla 15 se presentan los patrones de entrada correspondiente a cada estímulo, presentados alternadamente en la simulación número 8.

Ent	A+	B+
1	1	0
2	0	0
3	0	0
4	0	0
5	0	0
6	0	0
7	0	0
8	0	1
S*	1	1

Tabla 15. Patrones de entrada presentados alternadamente en la única condición de la simulación 8

6.2.2.5 Resultado de las simulaciones

En la Figura 46 se muestran los valores de activación registrados en la unidad "CR/UR" en las cuatro simulaciones, tanto para los casos en los que el estímulo "A" era aprendido (gráficos de los incisos a) en la Figura 46), como para los casos en los que el estímulo "B" es el que fue aprendido (gráficos de los incisos b)).

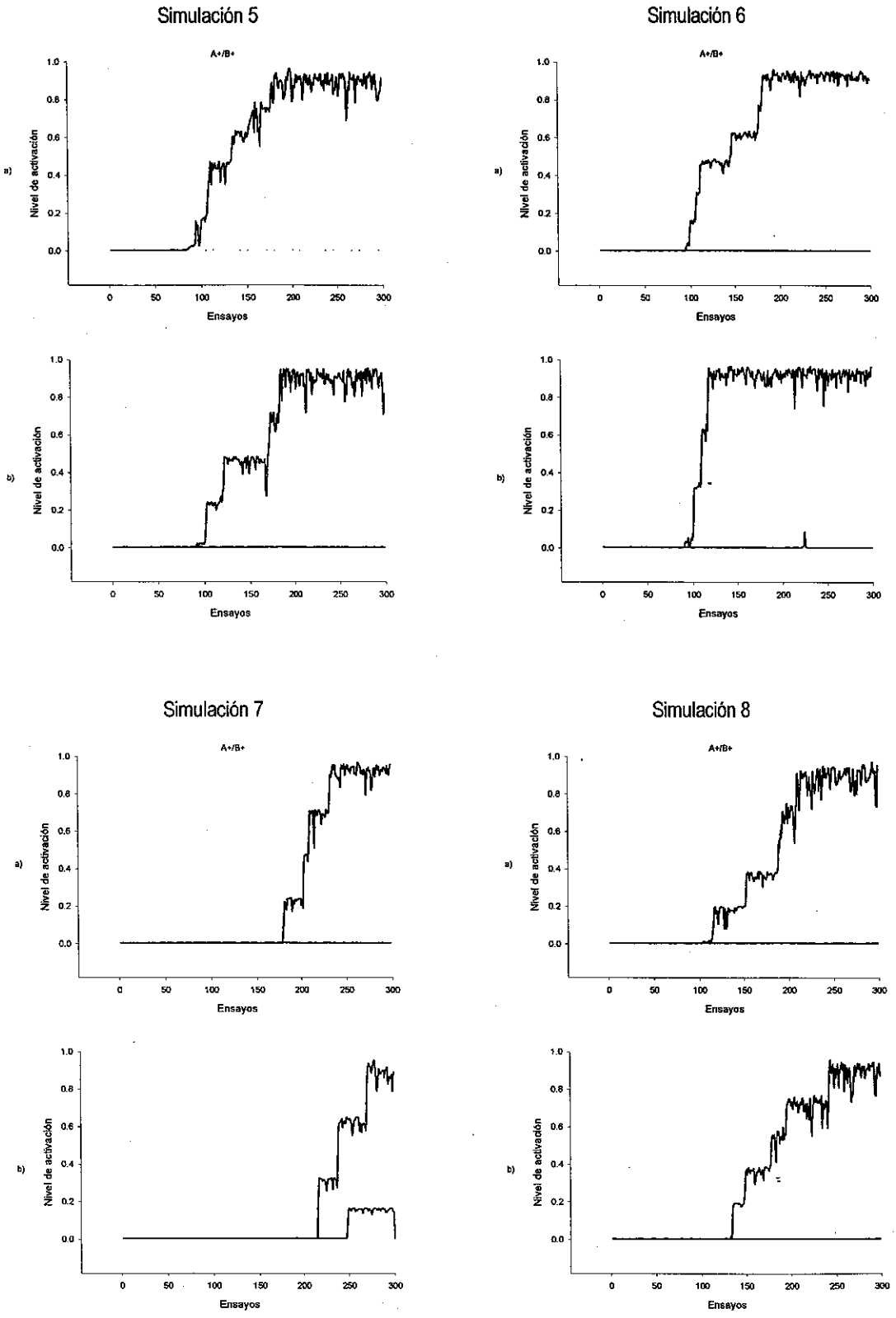


Figura 46. Niveles de activación registrados en la unidad "CR/UR" en las simulaciones 5, 6, 7 y 8, usando el esquema A+/B+.

En la Figura 46, se observa que la arquitectura clásica no permite que se presenten niveles elevados de activación en la unidad "CR/UR", ante un segundo estímulo condicionado, una vez que se ha configurado su estructura para presentar niveles elevados de activación en esta unidad de salida, aún cuando en este caso se presentaron alternadamente ambos estímulos. Asimismo, se observó durante las 10 replicaciones de cada simulación, que ambos estímulos tenían una probabilidad muy cercana a ½ de ser aprendidos, pero una vez que la red se configuraba para responder ante uno de ellos, el segundo no logró elevar los niveles de activación. También hubo casos en los que bajo este esquema de entrenamiento, ninguno de los dos estímulos pudo ser aprendido, observándose niveles bajos de activación ante la presencia de ambos. El tamaño de la red, bajo este esquema, tampoco tiene efecto alguno en el comportamiento observado.

Debido a la naturaleza de las entradas presentadas en la red y la estructura de las unidades de entrada, en la arquitectura usada, no existe diferencia alguna si se presentan los estímulos en el orden inverso, es decir, bajo un esquema B+/A+.

6.2.3 SIMULACIONES 9, 10, 11 Y 12.

En las simulaciones 9, 10, 11 y 12, se sometió a las redes "Pequeña", "Mediana", "Grande" y "Pequeña con 8 entradas", a un esquema de entrenamiento de tres condiciones: en la primera condición se presentó el estímulo "A" reforzado, en la segunda condición se presentó el estímulo A sin refuerzo y en la tercera condición se presentó el estímulo "B" reforzado. Todas las condiciones se conformaron por 300 ensayos cada una.

El esquema de condiciones para estas tres simulaciones fue A+|A-|B+.

Los patrones específicos de entrada empleados en cada red se presentan a continuación:

6.2.3.1 Patrones de entrada para la red "Pequeña"

En la tablas 16, 17 y 18 se presentan los patrones de entrada correspondiente a cada estímulo, presentados alternadamente en la simulación número 9.

Ent	Ts1	Ts2	Ts3	Ts4	Ts5	Ts6	Ts7	Ts8
1	1	1	1	1	1	1	1	1
2	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0
S*	0	0	0	0	0	0	0	1

Tabla 16. Patrones de entrada por momento temporal, presentados en la condición A+ en la simulación 9

Ent	Ts1	Ts2	Ts3	Ts4	Ts5	Ts6	Ts7	Ts8
1	1	1	1	1	1	1	1	1
2	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0
S*	0	0	0	0	0	0	0	0

Tabla 17. Patrones de entrada por momento temporal, presentados en la condición A- en la simulación 9

Ent	Ts1	Ts2	Ts3	Ts4	Ts5	Ts6	Ts7	Ts8
1	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0
3	1	1	1	1	1	1	1	1
S*	0	0	0	0	0	0	0	1

Tabla 18. Patrones de entrada por momento temporal, presentados en la condición B+ en la simulación 9

6.2.3.2 Patrones de entrada para la red "Mediana".

En las tablas 19, 20 y 21 se presentan los patrones de entrada por cada momento temporal, presentados en la simulación número 10.

Ent	Ts1	Ts2	Ts3	Ts4	Ts5	Ts6	Ts7	Ts8
1	1	1	1	1	1	1	1	1
2	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0
S*	0	0	0	0	0	0	0	1

Tabla 19. Patrones de entrada por momento temporal, presentados en la condición A+ en la simulación 10

Ent	Ts1	Ts2	Ts3	Ts4	Ts5	Ts6	Ts7	Ts8
1	1	1	1	1	1	1	1	1
2	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0
S*	0	0	0	0	0	0	0	0

Tabla 20. Patrones de entrada por momento temporal, presentados en la condición A- en la simulación 10

Ent	Ts1	Ts2	Ts3	Ts4	Ts5	Ts6	Ts7	Ts8
1	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0
7	1	1	1	1	1	1	1	1
S*	0	0	0	0	0	0	0	1

Tabla 21. Patrones de entrada por momento temporal, presentados en la condición B+ en la simulación 10

6.2.3.3 Patrones de entrada para la red "Grande".

En las tablas 22, 23 y 24 se presentan los patrones de entrada por cada momento temporal, presentados en la simulación número 11.

Ent	Ts1	Ts2	Ts3	Ts4	Ts5	Ts6	Ts7	Ts8
1	1	1	1	1	1	1	1	1
2	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0
11	0	0	0	0	0	0	0	0
S*	0	0	0	0	0	0	0	1

Tabla 22. Patrones de entrada por momento temporal, presentados en la condición A+ en la simulación 11

Ent	Ts1	Ts2	Ts3	Ts4	Ts5	Ts6	Ts7	Ts8
1	1	1	1	1	1	1	1	1
2	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0
11	0	0	0	0	0	0	0	0
S*	0	0	0	0	0	0	0	0

Tabla 23. Patrones de entrada por momento temporal, presentados en la condición A- en la simulación 11

Ent	Ts1	Ts2	Ts3	Ts4	Ts5	Ts6	Ts7	Ts8
1	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0
11	1	1	1	1	1	1	1	1
S*	0	0	0	0	0	0	0	1

Tabla 24. Patrones de entrada por momento temporal, presentados en la condición B+ en la simulación 11

6.2.3.4 Patrones de entrada para la red "Pequeña con 8 entradas".

En las tablas 25, 26 y 27 se presentan los patrones de entrada por cada momento temporal, presentados en la simulación número 12.

Ent	Ts1	Ts2	Ts3	Ts4	Ts5	Ts6	Ts7	Ts8
1	1	1	1	1	1	1	1	1
2	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	0
S*	0	0	0	0	0	0	0	1

Tabla 25. Patrones de entrada por momento temporal, presentados en la condición A+ en la simulación 12

Ent	Ts1	Ts2	Ts3	Ts4	Ts5	Ts6	Ts7	Ts8
1	1	1	1	1	1	1	1	1
2	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	0
S*	0	0	0	0	0	0	0	0

Tabla 26. Patrones de entrada por momento temporal, presentados en la condición A- en la simulación 12

Ent	Ts1	Ts2	Ts3	Ts4	Ts5	Ts6	Ts7	Ts8
1	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0
8	1	1	1	1	1	1	1	1
S*	0	0	0	0	0	0	0	1

Tabla 27. Patrones de entrada por momento temporal, presentados en la condición B+ en la simulación 12

6.2.3.5 Resultado de las simulaciones

En la Figura 47 se muestran los valores de activación registrados en la unidad "CR/UR" en las cuatro simulaciones:

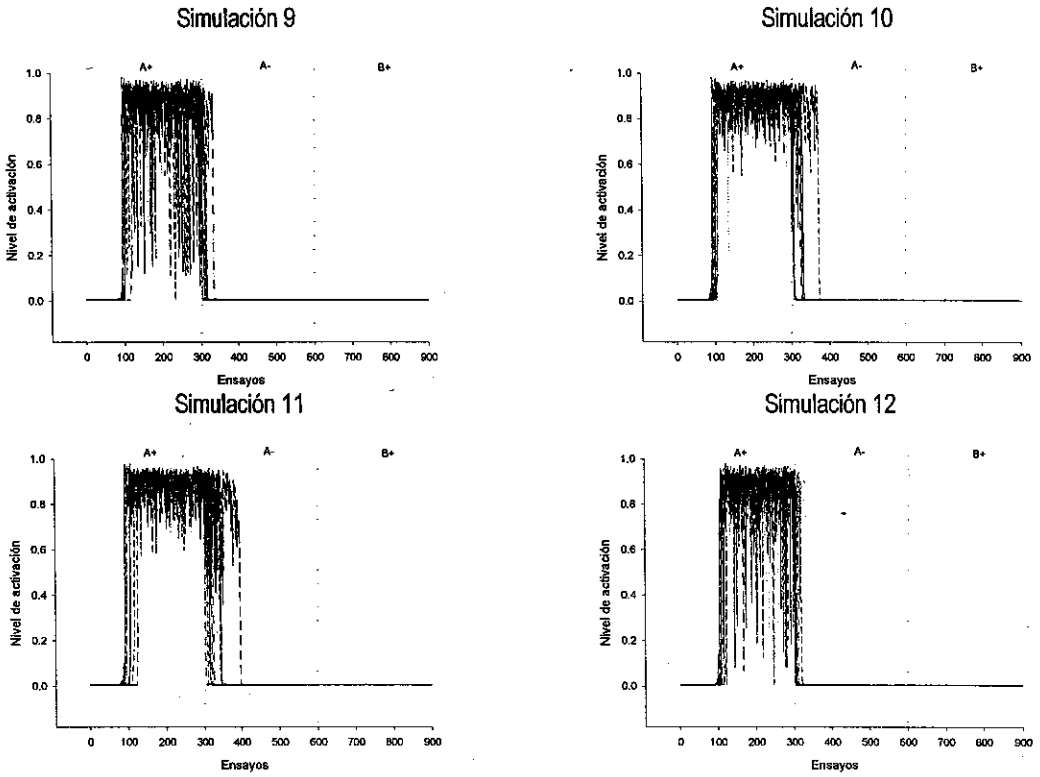


Figura 47. Niveles de activación registrados en las unidades "CR/UR" en las simulaciones 9, 10, 11 y 12.

Se puede observar en la Figura 47, que la arquitectura clásica no permite que se presenten niveles elevados de activación en la unidad "CR/UR", ante un segundo estímulo condicionado, una vez que se ha configurado su estructura para presentar niveles elevados de activación en esta unidad de salida, aún cuando en este caso se presentaron alternadamente ambos estímulos. Asimismo, se observó durante las 10 replicaciones de cada simulación, que ambos estímulos tenían una probabilidad de ½ de ser aprendidos, pero una vez que la red se configuraba para responder ante uno de ellos, el segundo no logró elevar los niveles de activación. El tamaño de la red, bajo este esquema, tampoco tiene efecto alguno en el comportamiento observado.

Debido a la naturaleza de las entradas presentadas en la red y la estructura de las unidades de entrada, en la arquitectura usada, no existe diferencia alguna si se presentan los estímulos en el orden inverso, es decir, bajo un esquema B+/A+.

6.3 EL USO DE DIFERENTES VALORES DE ENTRADA EN LA RED

En esta simulación se exploró la posibilidad de que el problema se pudiese resolver por medio de una activación parcial de las unidades de entrada que no participaran con una activación de 1.0 en el estímulo. Los valores empleados para esta activación parcial fueron fijados en 0.65, después de una serie de pruebas reportadas en (Ponce, 2004).

6.3.1 SIMULACIÓN 13

Para esta simulación se empleó una red "pequeña con 8 entradas". Todas las corridas se hicieron considerando 300 ensayos compuestos cada uno, de 8 momentos temporales. Se registró la actividad de la unidad "CR/UR" en el momento temporal 7 y cuando el refuerzo fue administrado, se hizo en el momento temporal 8, a través de la unidad S*, la que se activó con un valor de 1.0

Se realizaron 10 replicaciones de cada simulación.

6.3.1.1 Patrones de entrada usados

En esta simulación se sometió la red a tres condiciones distintas. En la primera condición se reforzó el estímulo A, en la segunda condición se presentó el mismo estímulo A, pero sin refuerzo, y en la tercera condición se presentó un estímulo diferente, denominado B, el cual recibió refuerzo. Se realizaron 300 ensayos para cada condición y no se empleó re-inicialización de pesos entre la condición A- y la condición B+. Las unidades de entrada que no recibían activación con un valor de 1.0 fueron activadas parcialmente con un valor de 0.65, excepto la unidad que se emplea para el segundo estímulo, cuya activación permaneció con valor de 0.0. El esquema de condiciones para esta simulación fue A+|A-|B+.

Los patrones específicos de entrada para cada condición se muestran en las siguientes tablas.

Ent	Ts1	Ts2	Ts3	Ts4	Ts5	Ts6	Ts7	Ts8
1	1	1	1	1	1	1	1	1
2	0.65	0.65	0.65	0.65	0.65	0.65	0.65	0.65
3	0.65	0.65	0.65	0.65	0.65	0.65	0.65	0.65
4	0.65	0.65	0.65	0.65	0.65	0.65	0.65	0.65
5	0.65	0.65	0.65	0.65	0.65	0.65	0.65	0.65
6	0.65	0.65	0.65	0.65	0.65	0.65	0.65	0.65
7	0.65	0.65	0.65	0.65	0.65	0.65	0.65	0.65
8	0	0	0	0	0	0	0	0
S*	0	0	0	0	0	0	0	1

Tabla 28. Patrones de entrada por momento temporal correspondientes a la condición A+ presentados en la simulación 13

Ent	Ts1	Ts2	Ts3	Ts4	Ts5	Ts6	Ts7	Ts8
1	1	1	1	1	1	1	1	1
2	0.65	0.65	0.65	0.65	0.65	0.65	0.65	0.65
3	0.65	0.65	0.65	0.65	0.65	0.65	0.65	0.65
4	0.65	0.65	0.65	0.65	0.65	0.65	0.65	0.65
5	0.65	0.65	0.65	0.65	0.65	0.65	0.65	0.65
6	0.65	0.65	0.65	0.65	0.65	0.65	0.65	0.65
7	0.65	0.65	0.65	0.65	0.65	0.65	0.65	0.65
8	0	0	0	0	0	0	0	0
S*	0	0	0	0	0	0	0	0

Tabla 29. Patrones de entrada por momento temporal correspondientes a la condición A- presentados en la simulación 13

Ent	Ts1	Ts2	Ts3	Ts4	Ts5	Ts6	Ts7	Ts8
1	0	0	0	0	0	0	0	0
2	0.65	0.65	0.65	0.65	0.65	0.65	0.65	0.65
3	0.65	0.65	0.65	0.65	0.65	0.65	0.65	0.65
4	0.65	0.65	0.65	0.65	0.65	0.65	0.65	0.65
5	0.65	0.65	0.65	0.65	0.65	0.65	0.65	0.65
6	0.65	0.65	0.65	0.65	0.65	0.65	0.65	0.65
7	0.65	0.65	0.65	0.65	0.65	0.65	0.65	0.65
8	1	1	1	1	1	1	1	1
S*	0	0	0	0	0	0	0	1

Tabla 30. Patrones de entrada por momento temporal correspondientes a la condición A- presentados en la simulación 13

6.3.1.2 Resultados obtenidos

En la Figura 48 se muestran los valores de activación promedio en rangos de 50 ensayos, de la unidad "CR/UR" (unidad número 17):

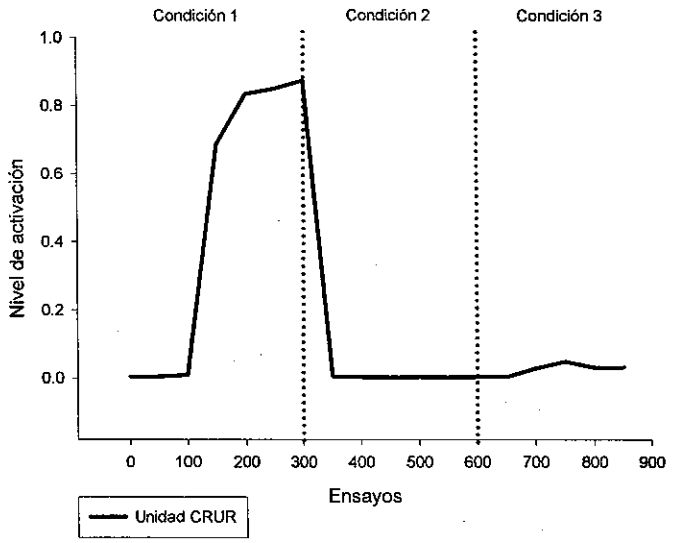


Figura 48. Niveles de activación promedio de la unidad 17 (CR/UR) después de la simulación 13

Se puede observar en la Figura 48, que la arquitectura empleada, empleando unidades de entrada parcialmente activadas con un valor fijo de 0.65, tampoco permite que se presenten niveles elevados de activación en la unidad 17 ("CR/UR"), ante un segundo estímulo condicionado una vez que se ha configurado su estructura para presentar niveles elevados de activación en esta unidad de salida. Aunque si se observa una elevación en el nivel de activación de la unidad "CR/UR", entre el ensayo 700 y 800, sin que éste haya sido significativo.

Esta imposibilidad para responder ante el estímulo B reforzado, se presenta aún a pesar de que se sometió la red a una condición de "extinción" de la activación ante el estímulo A.

6.3.2 SIMULACIÓN 14

En esta simulación se exploró la posibilidad de que el problema se pudiese resolver por medio de una activación parcial de las unidades de entrada que no participaran con una activación de 1.0 en el

estímulo. Los valores empleados para esta activación parcial fueron fijados en 0.65, y las unidades de entrada que sí participan en el segundo estímulo fueron activadas con un valor de 0.30, después de una serie de pruebas reportadas en (Ponce, 2004).

Todas las corridas se hicieron considerando 300 ensayos compuestos cada uno, de 8 momentos temporales. Se registró la actividad de la unidad CR/UR en el momento temporal 7 y cuando el refuerzo fue administrado, se hizo en el momento temporal 8, a través de la unidad S*, la que se activó con un valor de 1.0.

Se realizaron 10 replicaciones de cada simulación, empleando una red "pequeña con 8 entradas".

6.3.2.1 Patrones de entrada usados

En esta simulación se sometió la red a tres condiciones distintas. En la primera condición se reforzó el estímulo A, en la segunda condición se presentó el mismo estímulo A, pero sin refuerzo, y en la tercera condición se presentó un estímulo diferente, denominado B, el cual recibió refuerzo. Se realizaron 300 ensayos para cada condición y no se empleó re-inicialización de pesos entre la condición A- y la condición B+. Las unidades de entrada que no recibían activación con un valor de 1.0 fueron activadas parcialmente con un valor fijo de 0.65, excepto aquellas unidades que se emplean para el segundo estímulo, mismas que fueron activadas con valores fijos de 0.30. El esquema de condiciones para esta simulación fue A+|A-|B+.

Los patrones específicos de entrada para cada condición se muestran en las siguientes tablas.

Ent	Ts1	Ts2	Ts3	Ts4	Ts5	Ts6	Ts7	Ts8
1	1	1	1	1	1	1	1	1
2	1	1	1	1	1	1	1	1
3	0.65	0.65	0.65	0.65	0.65	0.65	0.65	0.65
4	0.65	0.65	0.65	0.65	0.65	0.65	0.65	0.65
5	0.65	0.65	0.65	0.65	0.65	0.65	0.65	0.65
6	0.65	0.65	0.65	0.65	0.65	0.65	0.65	0.65
7	0.30	0.30	0.30	0.30	0.30	0.30	0.30	0.30
8	0.30	0.30	0.30	0.30	0.30	0.30	0.30	0.30
S*	0	0	0	0	0	0	0	1

Tabla 31. Patrones de entrada por momento temporal correspondientes a la condición A+ presentados en la simulación 14

Ent	Ts1	Ts2	Ts3	Ts4	Ts5	Ts6	Ts7	Ts8
1	1	1	1	1	1	1	1	1
2	1	1	1	1	1	1	1	1
3	0.65	0.65	0.65	0.65	0.65	0.65	0.65	0.65
4	0.65	0.65	0.65	0.65	0.65	0.65	0.65	0.65
5	0.65	0.65	0.65	0.65	0.65	0.65	0.65	0.65
6	0.65	0.65	0.65	0.65	0.65	0.65	0.65	0.65
7	0.30	0.30	0.30	0.30	0.30	0.30	0.30	0.30
8	0.30	0.30	0.30	0.30	0.30	0.30	0.30	0.30
S*	0	0	0	0	0	0	0	0

Tabla 32. Patrones de entrada por momento temporal correspondientes a la condición A- presentados en la simulación 14

Ent	Ts1	Ts2	Ts3	Ts4	Ts5	Ts6	Ts7	Ts8
1	0.30	0.30	0.30	0.30	0.30	0.30	0.30	0.30
2	0.30	0.30	0.30	0.30	0.30	0.30	0.30	0.30
3	0.65	0.65	0.65	0.65	0.65	0.65	0.65	0.65
4	0.65	0.65	0.65	0.65	0.65	0.65	0.65	0.65
5	0.65	0.65	0.65	0.65	0.65	0.65	0.65	0.65
6	0.65	0.65	0.65	0.65	0.65	0.65	0.65	0.65
7	1	1	1	1	1	1	1	1
8	1	1	1	1	1	1	1	1
S*	0	0	0	0	0	0	0	1

Tabla 33. Patrones de entrada por momento temporal correspondientes a la condición A- presentados en la simulación 14

6.3.2.2 Resultados obtenidos

En la Figura 49 se muestran los valores de activación promedio en rangos de 50 ensayos, de la unidad "CR/UR" (unidad número 17):

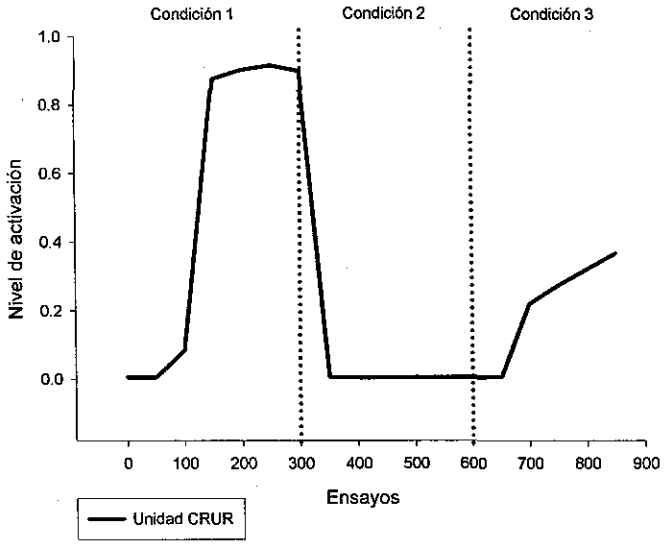


Figura 49. Niveles de activación promedio de la unidad 17 (CRUR) después de la simulación 14

Se puede observar en la Figura 49, que la arquitectura empleada, empleando unidades de entrada parcialmente activadas con valores fijos de 0.65 y de 0.30, tampoco permite que se presenten niveles elevados de activación en la unidad 17 ("CR/UR"), ante un segundo estímulo condicionado una vez que se ha configurado su estructura para presentar niveles elevados de activación en esta unidad de salida. Aunque si se observa una elevación en el nivel de activación de la unidad "CR/UR", entre el ensayo 700 y 900, que llega hasta niveles cercanos al 0.4.

Esta imposibilidad para responder ante el estímulo B reforzado, se presenta aún a pesar de que se sometió la red a una condición de "extinción" de la activación ante el estímulo A.

6.3.3 SIMULACIÓN 15

En esta simulación se exploró, al igual que en la simulación anterior, la posibilidad de que el problema se pudiese resolver por medio de una activación parcial de las unidades de entrada que no participaran con una activación de 1.0 en el estímulo. Los valores empleados para esta activación parcial fueron asignados aleatoriamente dentro de un rango de [0.6, 0.7] en las unidades de entrada que no participaban en los estímulos y de 0.30 en aquellas que sí participaban en el segundo estímulo, lo anterior después de una serie de pruebas reportadas en (Ponce, 2004).

Se empleó el mismo tipo de red que en la simulación anterior, una red "pequeña con 8 entradas". Todas las corridas se hicieron considerando 300 ensayos compuestos cada uno, de 8 momentos temporales. Se registró la actividad de la unidad CR/UR en el momento temporal 7 y cuando el refuerzo fue administrado, se hizo en el momento temporal 8, a través de la unidad S*, la que se activó con un valor de 1.0

Se realizaron 10 replicaciones de cada simulación.

6.3.3.1 Patrones de entrada usados

En esta simulación se sometió la red a tres condiciones distintas. En la primera condición se reforzó el estímulo A, en la segunda condición se presentó el mismo estímulo A, pero sin refuerzo, y en la tercera condición se presentó un estímulo diferente, denominado B, el cual recibió refuerzo. Se realizaron 300 ensayos para cada condición y no se empleó re-inicialización de pesos entre la condición A- y la condición B+. Las unidades de entrada que no recibían activación con un valor de 1.0 fueron activadas parcialmente con un valor aleatorio comprendido entre [0.6, 0.7], excepto la unidad que se emplea para el segundo estímulo, misma que fue activada con valores fijos de 0.30. El esquema de condiciones para esta simulación fue A+|A-|B+.

Los patrones específicos de entrada para cada condición se muestran en las siguientes tablas.

Ent	Ts1	Ts2	Ts3	Ts4	Ts5	Ts6	Ts7	Ts8
1	1	1	1	1	1	1	1	1
2	RND[0.6,07]	RND[0.6,07]	RND[0.6,07]	RND[0.6,07]	RND[0.6,07]	RND[0.6,07]	RND[0.6,07]	RND[0.6,07]
3	RND[0.6,0.7]	RND[0.6,0.7]	RND[0.6,0.7]	RND[0.6,0.7]	RND[0.6,0.7]	RND[0.6,0.7]	RND[0.6,0.7]	RND[0.6,0.7]
4	RND[0.6,0.7]	RND[0.6,0.7]	RND[0.6,0.7]	RND[0.6,0.7]	RND[0.6,0.7]	RND[0.6,0.7]	RND[0.6,0.7]	RND[0.6,0.7]
5	RND[0.6,0.7]	RND[0.6,0.7]	RND[0.6,0.7]	RND[0.6,0.7]	RND[0.6,0.7]	RND[0.6,0.7]	RND[0.6,0.7]	RND[0.6,0.7]
6	RND[0.6,0.7]	RND[0.6,0.7]	RND[0.6,0.7]	RND[0.6,0.7]	RND[0.6,0.7]	RND[0.6,0.7]	RND[0.6,0.7]	RND[0.6,0.7]
7	RND[0.6,0.7]	RND[0.6,0.7]	RND[0.6,0.7]	RND[0.6,0.7]	RND[0.6,0.7]	RND[0.6,0.7]	RND[0.6,0.7]	RND[0.6,0.7]
8	0.30	0.30	0.30	0.30	0.30	0.30	0.30	0.30
S*	0	0	0	0	0	0	0	1

Tabla 34. Patrones de entrada por momento temporal correspondientes a la condición A+ presentados en la simulación 15

Ent	Ts1	Ts2	Ts3	Ts4	Ts5	Ts6	Ts7	Ts8
1	1	1	1	1	1	1	1	1
2	RND[0.6,07]	RND[0.6,07]	RND[0.6,07]	RND[0.6,07]	RND[0.6,07]	RND[0.6,07]	RND[0.6,07]	RND[0.6,07]
3	RND[0.6,0.7]	RND[0.6,0.7]	RND[0.6,0.7]	RND[0.6,0.7]	RND[0.6,0.7]	RND[0.6,0.7]	RND[0.6,0.7]	RND[0.6,0.7]
4	RND[0.6,0.7]	RND[0.6,0.7]	RND[0.6,0.7]	RND[0.6,0.7]	RND[0.6,0.7]	RND[0.6,0.7]	RND[0.6,0.7]	RND[0.6,0.7]
5	RND[0.6,0.7]	RND[0.6,0.7]	RND[0.6,0.7]	RND[0.6,0.7]	RND[0.6,0.7]	RND[0.6,0.7]	RND[0.6,0.7]	RND[0.6,0.7]
6	RND[0.6,0.7]	RND[0.6,0.7]	RND[0.6,0.7]	RND[0.6,0.7]	RND[0.6,0.7]	RND[0.6,0.7]	RND[0.6,0.7]	RND[0.6,0.7]
7	RND[0.6,0.7]	RND[0.6,0.7]	RND[0.6,0.7]	RND[0.6,0.7]	RND[0.6,0.7]	RND[0.6,0.7]	RND[0.6,0.7]	RND[0.6,0.7]
8	0.30	0.30	0.30	0.30	0.30	0.30	0.30	0.30
S*	0	0	0	0	0	0	0	0

Tabla 35. Patrones de entrada por momento temporal correspondientes a la condición A- presentados en la simulación 15

Ent	Ts1	Ts2	Ts3	Ts4	Ts5	Ts6	Ts7	Ts8
1	0.30	0.30	0.30	0.30	0.30	0.30	0.30	0.30
2	RND[0.6,07]	RND[0.6,07]	RND[0.6,07]	RND[0.6,07]	RND[0.6,07]	RND[0.6,07]	RND[0.6,07]	RND[0.6,07]
3	RND[0.6,0.7]	RND[0.6,0.7]	RND[0.6,0.7]	RND[0.6,0.7]	RND[0.6,0.7]	RND[0.6,0.7]	RND[0.6,0.7]	RND[0.6,0.7]
4	RND[0.6,0.7]	RND[0.6,0.7]	RND[0.6,0.7]	RND[0.6,0.7]	RND[0.6,0.7]	RND[0.6,0.7]	RND[0.6,0.7]	RND[0.6,0.7]
5	RND[0.6,0.7]	RND[0.6,0.7]	RND[0.6,0.7]	RND[0.6,0.7]	RND[0.6,0.7]	RND[0.6,0.7]	RND[0.6,0.7]	RND[0.6,0.7]
6	RND[0.6,0.7]	RND[0.6,0.7]	RND[0.6,0.7]	RND[0.6,0.7]	RND[0.6,0.7]	RND[0.6,0.7]	RND[0.6,0.7]	RND[0.6,0.7]
7	RND[0.6,0.7]	RND[0.6,0.7]	RND[0.6,0.7]	RND[0.6,0.7]	RND[0.6,0.7]	RND[0.6,0.7]	RND[0.6,0.7]	RND[0.6,0.7]
8	1	1	1	1	1	1	1	1
S*	0	0	0	0	0	0	0	1

Tabla 36. Patrones de entrada por momento temporal correspondientes a la condición A- presentados en la simulación 15

6.3.3.2 Resultados obtenidos

En la Figura 50 se muestran los valores de activación promedio en rangos de 50 ensayos, de la unidad "CR/UR" (unidad número 17):

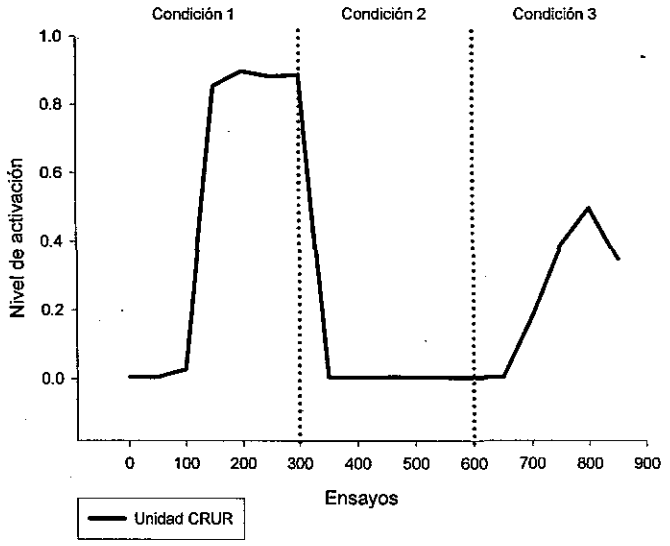


Figura 50. Niveles de activación promedio de la unidad 17 (CR/UR) obtenidos en la simulación 15

Se puede observar en la Figura 50, que la arquitectura empleada, empleando unidades de entrada parcialmente activadas con valores aleatorios entre 0.6 y 0.7, tampoco permite que se presenten niveles elevados de activación en la unidad 17 ("CR/UR"), ante un segundo estímulo condicionado una vez que se ha configurado su estructura para presentar niveles elevados de activación en esta unidad de salida. Aunque si se observa una elevación en el nivel de activación de la unidad "CR/UR", entre el ensayo 700 y 900, que llega hasta niveles cercanos al 0.6 hacia el ensayo 800.

Esta imposibilidad para responder ante el estímulo B reforzado, se presenta aún a=pesar de que se sometió la red a una condición de "extinción" de la activación ante el estímulo "A".

7. LA BÚSQUEDA DE UNA ARQUITECTURA QUE PERMITA EL MANEJO DE DOS EC BAJO EL MODELO DBP

Como se ha visto ya en el capítulo anterior, para la arquitectura clásica empleada por el modelo DBP, es imposible aprender a responder ante un segundo estímulo, una vez que se ha logrado que la red responda ante un primer estímulo. En este capítulo se presenta un conjunto de simulaciones que buscaron explorar la posibilidad de nuevas arquitecturas que pudiesen vencer esta limitación.

En esta fase se partió intuitivamente de los siguientes hechos:

- Una red neural S1, bajo el Modelo DBP es capaz de aprender a responder efectivamente ante un estímulo condicionado CS1.
- Una segunda red neural S2, bajo el Modelo DBP, es capaz de aprender a responder efectivamente ante un estímulo condicionado CS2.
- Las dos redes neurales S1 y S2, si fuesen consideradas como parte de un mismo sistema, darían a éste la capacidad de manejar dos estímulos condicionados CS1 y CS2 al mismo tiempo (ver Figura 51)

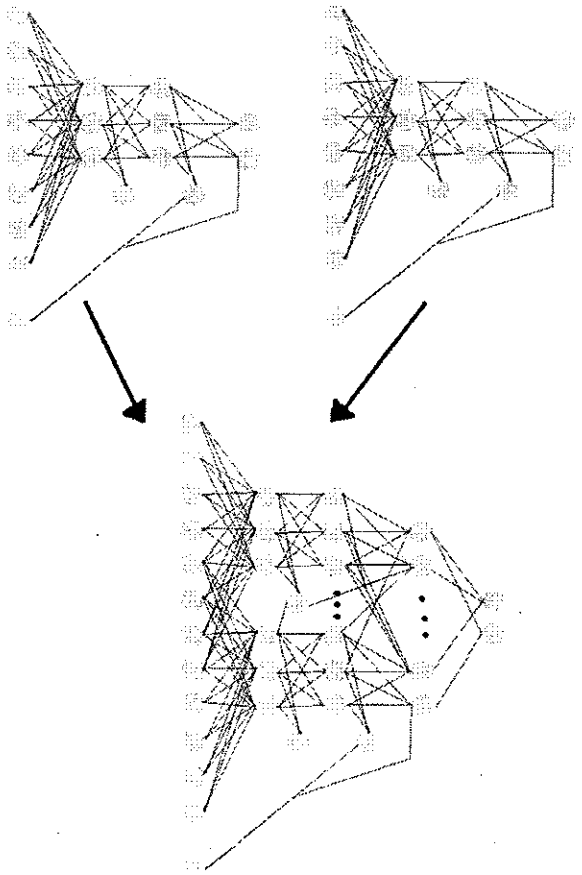


Figura 51. Esquema de la estrategia empleada en la búsqueda de una arquitectura adecuada para que una Red DBP pueda manejar 2 CS's.

Sin embargo, se buscaba una solución que no implicara considerar como un mismo sistema a dos redes neurales completamente separadas, por ello, se experimentó con arquitecturas que podrían ubicarse dentro de este rango de arquitecturas posibles que van desde los dos sistemas separados, que si pueden manejar a CS1 y CS2 simultáneamente y el sistema completamente integrado, que está impedido para manejarlos.

El trabajo reportado en este capítulo consistió en buscar sistemáticamente una arquitectura que permitiera el manejo de dos estímulos, dentro de este rango de posibilidades. Cada nueva versión de la arquitectura de red propuesta se basó en los resultados de las versiones previas.

7.1 SIMULACIÓN 16

En esta simulación se emplea una estrategia distinta, apartándose de las anteriores simulaciones. Se comienza a explorar la posibilidad de otras arquitecturas, parcialmente interconectadas, que pudiesen permitir la resolución del problema. Se partió de la idea básica de unir dos redes en una sola, de manera tal que se pudiera conservar el manejo de un estímulo por cada una. Lo anterior de acuerdo con lo reportado en (Ponce, 2003 y 2005).

Todas las corridas se hicieron considerando 300 ensayos compuestos cada uno, de 8 momentos temporales. Se registró la actividad de las unidades CR/UR en el momento temporal 7 y cuando el refuerzo fue administrado, se hizo en el momento temporal 8, a través de la unidad S*, la que se activó con un valor de 1.0.

Se realizaron 10 replicaciones de cada simulación.

7.1.1 ESTRUCTURA DE LA RED

La red empleada en esta simulación se compone de 25 elementos en total, de los cuales 7 son elementos de entrada y 18 son unidades de procesamiento. El número total de conexiones en la red es de 60. La red está estructurada de la siguiente manera: las unidades de entrada 1,2,3 y 4 tienen conexiones completas con las unidades 8, 9 y 10 (unidades de tipo "sa"), y las unidades de entrada 4, 5, 6 y 7, tienen conexiones completas con las unidades 11, 12 y 13 (también unidades del tipo "sa"), a partir de aquí se pueden identificar dos subredes, ya que las unidades "sa" 8, 9 y 10, mantienen conexión con la unidad "ca1" número 14 y con las unidades de tipo "ma" 16, 17 y 18. Asimismo, las unidades de tipo "sa" 11, 12 y 13, mantienen conexiones solamente con las unidades de tipo "ma" 19, 20 y 21, así como con la unidad de tipo "ca1" número 15. Siguiendo con la primera subred, las unidades "ma" 16, 17 y 18, tienen conexión solamente con la unidad "vta" número 22 y con una unidad de salida "CR/UR" de las dos que integra la red, la número 24. La segunda subred tiene a sus unidades "ma" 19, 20 y 21 conectadas solamente con la unidad "vta" 23 y con la segunda de las unidades de salida "CR/UR", la unidad número 25. Se registró la salida en ambas unidades "CR/UR".

En esta arquitectura la unión entre las dos subredes se da de dos maneras, la primera, a través de las unidades de entrada, en donde la unidad de entrada número 4 tiene conexión con las unidades "ma" de ambas subredes, y segundo, a partir de los cálculos de las discrepancias tanto en las unidades "ca1" como "vta" que sirven para retroalimentar los pesos de toda la red.

La arquitectura empleada se presenta en la Figura 52:

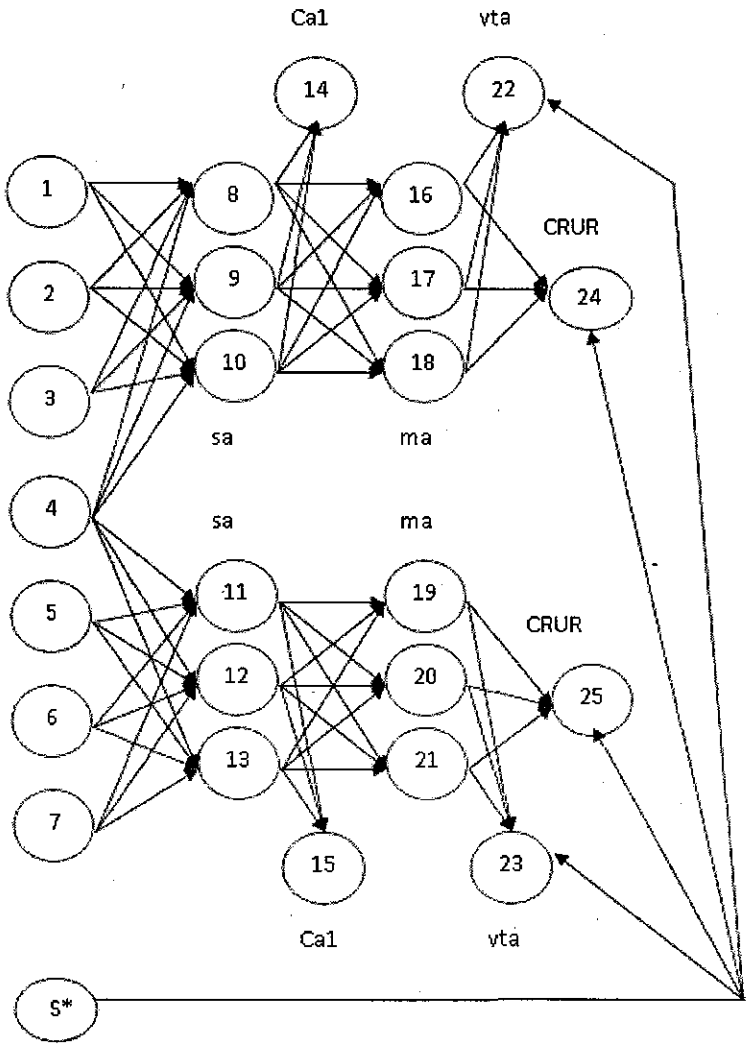


Figura 52. Arquitectura empleada en la simulación 16

7.1.2 PATRONES DE ENTRADA USADOS

En esta simulación se sometió la red a tres condiciones distintas. En la primera condición se reforzó el estímulo A, en la segunda condición se presentó el mismo estímulo A, pero sin refuerzo, y en la tercera condición se presentó un estímulo diferente, denominado B, el cual recibió refuerzo. El esquema de condiciones para esta simulación fue A+|A-|B+.

Se realizaron 300 ensayos para cada condición y no se empleó re-inicialización de pesos entre la condición A- y la condición B+.

A las unidades de entrada que no recibían activación con un valor de 1.0, les fue asignado un valor de 0.0.

A lo largo de los momentos temporales, los patrones empleados en esta simulación fueron los siguientes:

Ent	Ts1	Ts2	Ts3	Ts4	Ts5	Ts6	Ts7	Ts8
1	1	1	1	1	1	1	1	1
2	1	1	1	1	1	1	1	1
3	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0
S*	0	0	0	0	0	0	0	1

Tabla 37. Patrones de entrada por momento temporal correspondientes al estímulo A+ presentados en la simulación 16

Ent	Ts1	Ts2	Ts3	Ts4	Ts5	Ts6	Ts7	Ts8
1	1	1	1	1	1	1	1	1
2	1	1	1	1	1	1	1	1
3	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0
S*	0	0	0	0	0	0	0	0

Tabla 38. Patrones de entrada por momento temporal correspondientes al estímulo A- presentados en la simulación 16

Ent	Ts1	Ts2	Ts3	Ts4	Ts5	Ts6	Ts7	Ts8
1	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0
6	1	1	1	1	1	1	1	1
7	1	1	1	1	1	1	1	1
S*	0	0	0	0	0	0	0	1

Tabla 39. Patrones de entrada por momento temporal correspondientes al estímulo B+ presentados en la simulación 16

7.1.3 RESULTADOS OBTENIDOS

En la Figura 53 se muestran los valores de activación promedio en rangos de 50 ensayos, de las unidades "CR/UR" (unidades número 24 y 25):

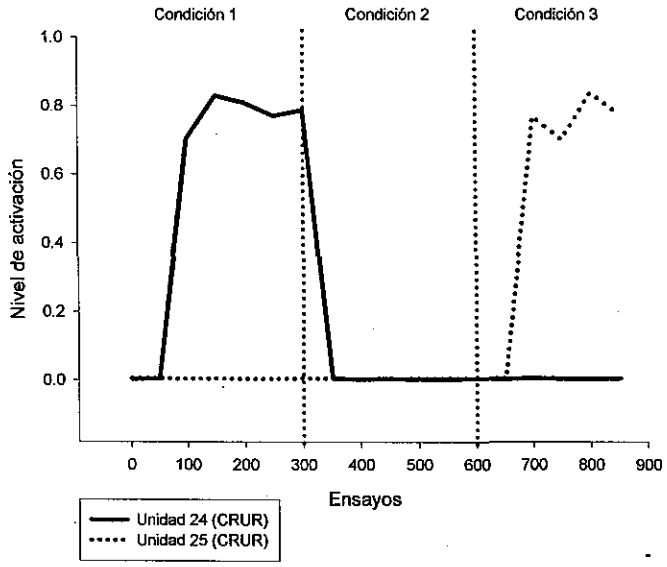


Figura 53. Niveles de activación promedio de las unidades 24 y 25 después de la simulación 16

Se puede observar en la Figura 53, que la arquitectura empleada, empleando dos subredes, si permite que la red configurada responda con niveles altos de activación, alcanzándose en ambos casos, valores entre el 0.7 y el 0.9. La respuesta ante el estímulo "A", se observa en la unidad "CR/UR" número 24, que alcanza niveles altos de activación cuando "A" es reforzado y sus niveles de activación decaen hasta un valor cercano a 0.0 cuando se retira el refuerzo. Asimismo, la respuesta ante el estímulo "B" se observa en la unidad "CR/UR" número 25, que presenta niveles de activación cercanos a cero durante las condiciones 1 y 2 (cuando el estímulo "B" no es presentado), elevándose esta activación ante la presentación del estímulo "B" reforzado durante la condición número 3.

7.2 SIMULACIÓN 17

En esta simulación se explora aún más la posibilidad de que la arquitectura encontrada, con base en subredes parcialmente interconectadas, pudiesen permitir la resolución del problema. Para buscar encontrar un mayor nivel de activación se incrementó el tamaño de la red, con base en un hallazgo reportado en redes neurales, reportado por Grossberg y Schmajuck (1989), relacionado con la elevación de los niveles de activación obtenidos a partir del incremento en el número de unidades de la red.

Todas las corridas se hicieron considerando 300 ensayos compuestos cada uno, de 8 momentos temporales. Se registró la actividad de las unidades CR/UR en el momento temporal 7 y cuando el refuerzo fue administrado, se hizo en el momento temporal 8, a través de la unidad S*, la que se activó con un valor de 1.0

Se realizaron 10 replicaciones de cada simulación.

7.2.1 ESTRUCTURA DE LA RED

La red empleada en esta simulación se compone de 43 elementos en total, de los cuales 25 son elementos de entrada y 18 son unidades de procesamiento. El número total de conexiones en la red es de 114. La red está estructurada de la siguiente manera: las unidades de entrada 1 a la 13 tienen conexiones completas con las unidades 26, 27 y 28 (unidades de tipo "sa"), y las unidades de entrada 13 a la 25 tienen conexiones completas con las unidades 29, 30 y 31 (también unidades del tipo "sa"), a partir de aquí se pueden identificar dos subredes, ya que las unidades "sa" 26, 27 y 28, mantienen conexión con la unidad "ca1" número 32 y con las unidades de tipo "ma" 34, 36 y 36. Asimismo, las unidades de tipo "sa" 29, 30 y 31, mantienen conexiones solamente con las unidades de tipo "ma" 37, 38 y 39, así como con la unidad de tipo "ca1" número 33. Siguiendo con la primera subred, las unidades "ma" 34, 35 y 36, tienen conexión solamente con la unidad "vta" número 40 y con una unidad de salida "CR/UR" de las dos que integra la red, la número 42. La segunda subred tiene a sus unidades "ma" 37, 38 y 39 conectadas solamente con la unidad "vta" 41 y con la segunda de las unidades de salida "CR/UR", la unidad número 43. Se registró la salida en ambas unidades "CR/UR".

En esta arquitectura la unión entre las dos subredes se da de dos maneras, la primera, a través de las unidades de entrada, en donde la unidad de entrada número 13 tiene conexión con las unidades "ma" de ambas subredes, y segundo, a partir de los cálculos de las discrepancias tanto en las unidades "ca1" como "vta" que sirven para retroalimentar los pesos de toda la red.

La arquitectura empleada se presenta en la Figura 54:

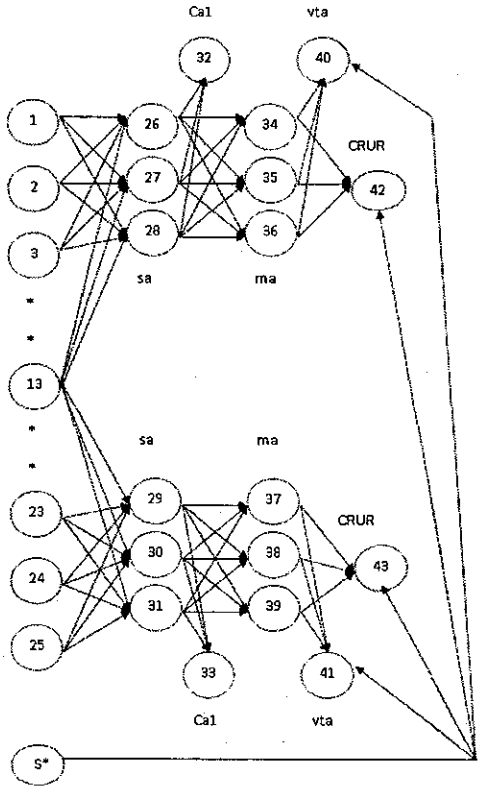


Figura 54. Arquitectura empleada en la simulación 17

7.2.2 PATRONES DE ENTRADA USADOS

En esta simulación se sometió la red a tres condiciones distintas. En la primera condición se reforzó el estímulo A, en la segunda condición se presentó el mismo estímulo A, pero sin refuerzo, y en la tercera condición se presentó un estímulo diferente, denominado B, el cual recibió refuerzo. El esquema de condiciones para esta simulación fue A+|A-|B+.

Se realizaron 300 ensayos para cada condición y no se empleó re-inicialización de pesos entre la condición A- y la condición B+.

A las unidades de entrada que no recibían activación con un valor de 1.0, les fue asignado un valor de 0.0.

A lo largo de los momentos temporales, los patrones empleados en esta simulación fueron los siguientes:

Ent	Ts1	Ts2	Ts3	Ts4	Ts5	Ts6	Ts7	Ts8
1	1	1	1	1	1	1	1	1
2	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0
11	0	0	0	0	0	0	0	0
12	0	0	0	0	0	0	0	0
13	0	0	0	0	0	0	0	0
14	0	0	0	0	0	0	0	0
15	0	0	0	0	0	0	0	0
16	0	0	0	0	0	0	0	0
17	0	0	0	0	0	0	0	0
18	0	0	0	0	0	0	0	0
19	0	0	0	0	0	0	0	0
20	0	0	0	0	0	0	0	0
21	0	0	0	0	0	0	0	0
22	0	0	0	0	0	0	0	0
23	0	0	0	0	0	0	0	0
24	0	0	0	0	0	0	0	0
25	0	0	0	0	0	0	0	0
S*	0	0	0	0	0	0	0	1

Tabla 40. Patrones de entrada por momento temporal correspondientes al estímulo A+ presentados en la simulación 17

Ent	Ts1	Ts2	Ts3	Ts4	Ts5	Ts6	Ts7	Ts8
1	1	1	1	1	1	1	1	1
2	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0
11	0	0	0	0	0	0	0	0
12	0	0	0	0	0	0	0	0
13	0	0	0	0	0	0	0	0
14	0	0	0	0	0	0	0	0
15	0	0	0	0	0	0	0	0
16	0	0	0	0	0	0	0	0
17	0	0	0	0	0	0	0	0
18	0	0	0	0	0	0	0	0
19	0	0	0	0	0	0	0	0
20	0	0	0	0	0	0	0	0
21	0	0	0	0	0	0	0	0
22	0	0	0	0	0	0	0	0
23	0	0	0	0	0	0	0	0
24	0	0	0	0	0	0	0	0
25	0	0	0	0	0	0	0	0
S*	0	0	0	0	0	0	0	0

Tabla 41. Patrones de entrada por momento temporal correspondientes al estímulo A- presentados en la simulación 17

Ent	Ts1	Ts2	Ts3	Ts4	Ts5	Ts6	Ts7	Ts8
1	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0
11	0	0	0	0	0	0	0	0
12	0	0	0	0	0	0	0	0
13	0	0	0	0	0	0	0	0
14	0	0	0	0	0	0	0	0
15	0	0	0	0	0	0	0	0
16	0	0	0	0	0	0	0	0
17	0	0	0	0	0	0	0	0
18	0	0	0	0	0	0	0	0
19	0	0	0	0	0	0	0	0
20	0	0	0	0	0	0	0	0
21	0	0	0	0	0	0	0	0
22	0	0	0	0	0	0	0	0
23	0	0	0	0	0	0	0	0
24	0	0	0	0	0	0	0	0
25	1	1	1	1	1	1	1	1
S*	0	0	0	0	0	0	0	1

Tabla 42. Patrones de entrada por momento temporal correspondientes al estímulo B+ presentados en la simulación 17

7.2.3 RESULTADOS OBTENIDOS

En la Figura 55 se muestran los valores de activación promedio en rangos de 50 ensayos, de las unidades "CR/UR" (unidades número 42 y 43):

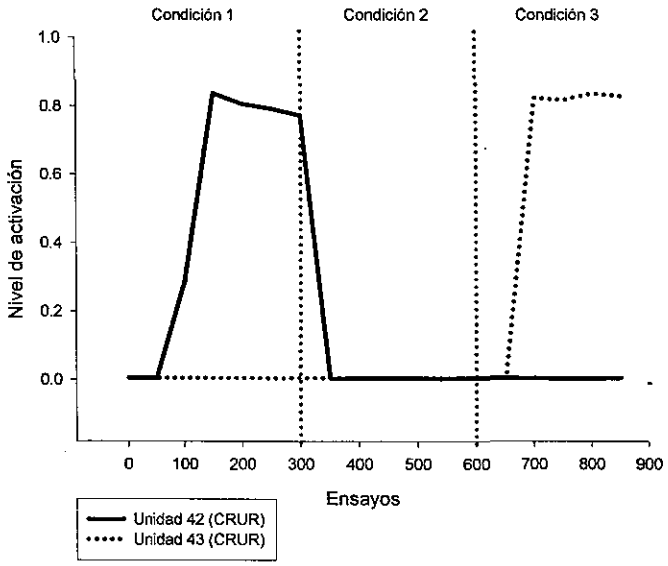


Figura 55. Niveles de activación promedio de las unidades 42 y 43 después de la simulación 17

Se puede observar en la Figura 55, que la arquitectura empleada, empleando dos subredes, si permite que la red configurada responda con niveles altos de activación. No se observa un nivel mayor de activación ampliando el número de unidades en la arquitectura de la red. En ambos casos, los valores entre el 0.78 y el 0.84. La respuesta ante el estímulo "A", se muestra en la unidad "CR/UR" número 42, que alcanza niveles altos de activación cuando "A" es presentado con refuerzo y niveles de activación que decaen hasta un valor cercano a 0.0, cuando "A" es presentado sin refuerzo. Asimismo, la activación de la unidad "CR/UR" número 43 presenta la respuesta ante el estímulo B, que durante las condiciones 1 y 2, no es presentado y por ello se observan niveles de activación cercanos al cero en esta unidad, para elevarse durante la condición número 3, una vez que el estímulo "B" es presentado con refuerzo.

7.3 SIMULACIÓN 18

En esta simulación se exploró la posibilidad de que las subredes que componen la red anterior se integraran a partir de la inclusión de una sola unidad "CR/UR" a la que convergieran las dos unidades que anteriormente estaban declaradas como "CR/UR" y en esta nueva exploración fueron declaradas como unidades del tipo "ma", pero sin que estas unidades tuvieran conexión directa con la unidad "vta" de la subred correspondiente.

Todas las corridas se hicieron considerando 300 ensayos compuestos cada uno, de 8 momentos temporales. Se registró la actividad de la unidad CR/UR en el momento temporal 7 y cuando el

refuerzo fue administrado, se hizo en el momento temporal 8, a través de la unidad S*, la que se activó con un valor de 1.0

Se realizaron 10 replicaciones de cada simulación.

7.3.1 ESTRUCTURA DE LA RED

En esta simulación, como ya se comentó, se buscó incorporar un elemento de procesamiento más, de manera que se pudiera tener una arquitectura más "cerrada". La red empleada se compone de 44 elementos en total, de los cuales 25 son elementos de entrada y 19 son unidades de procesamiento. El número total de conexiones en la red es de 116. Los cambios en relación con la red usada en la simulación anterior, consisten en integrar una unidad más de tipo "CR/UR", la número 44, que recibe conexión de las unidades 42 y 43 que en esta nueva red son de tipo "ma".

En esta arquitectura la unión entre las dos subredes se da de tres maneras, la primera, a través de las unidades de entrada, en donde la unidad de entrada número 13 tiene conexión con las unidades "ma" de ambas subredes, la segunda, a partir de la unidad integradora "CR/UR" número 44 que recibe la conexión del final de ambas subredes, y tercero, a partir de los cálculos de las discrepancias tanto en las unidades "ca1" como "vta" que sirven para retroalimentar los pesos de toda la red.

La arquitectura empleada se presenta en la Figura 56:

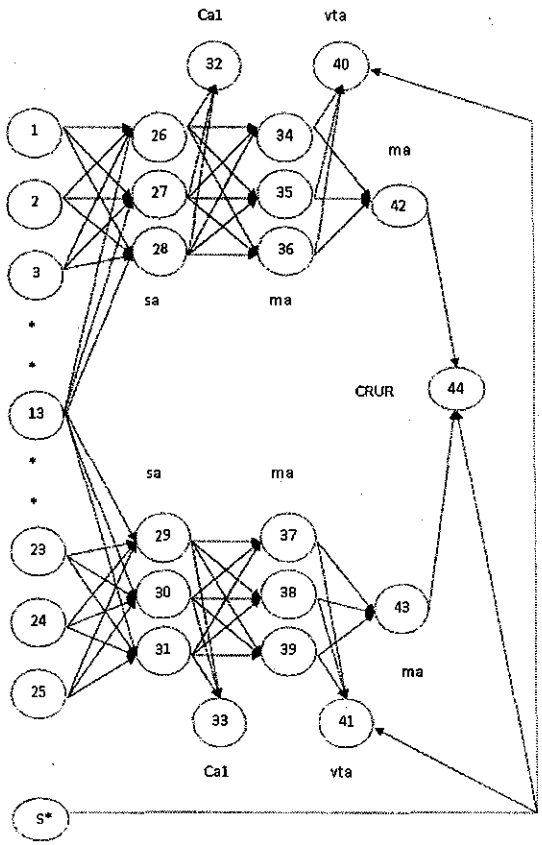


Figura 56. Arquitectura de la Red empleada en la simulación 18

7.3.2 PATRONES DE ENTRADA USADOS

En esta simulación se sometió la red a tres condiciones distintas. En la primera condición se reforzó el estímulo A, en la segunda condición se presentó el mismo estímulo A, pero sin refuerzo, y en la tercera condición se presentó un estímulo diferente, denominado B, el cual recibió refuerzo. El esquema de condiciones para esta simulación fue A+|A-|B+.

Se realizaron 300 ensayos para cada condición y no se empleó re-inicialización de pesos entre la condición A- y la condición B+.

A las unidades de entrada que no recibían activación con un valor de 1.0, les fue asignado un valor de 0.0.

A lo largo de los momentos temporales, los patrones empleados en esta simulación fueron los siguientes:

Ent	Ts1	Ts2	Ts3	Ts4	Ts5	Ts6	Ts7	Ts8
1	1	1	1	1	1	1	1	1
2	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0
11	0	0	0	0	0	0	0	0
12	0	0	0	0	0	0	0	0
13	0	0	0	0	0	0	0	0
14	0	0	0	0	0	0	0	0
15	0	0	0	0	0	0	0	0
16	0	0	0	0	0	0	0	0
17	0	0	0	0	0	0	0	0
18	0	0	0	0	0	0	0	0
19	0	0	0	0	0	0	0	0
20	0	0	0	0	0	0	0	0
21	0	0	0	0	0	0	0	0
22	0	0	0	0	0	0	0	0
23	0	0	0	0	0	0	0	0
24	0	0	0	0	0	0	0	0
25	0	0	0	0	0	0	0	0
S*	0	0	0	0	0	0	0	1

Tabla 43. Patrones de entrada por momento temporal correspondientes al estímulo A+ presentados en la simulación 18

Ent	Ts1	Ts2	Ts3	Ts4	Ts5	Ts6	Ts7	Ts8
1	1	1	1	1	1	1	1	1
2	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0
11	0	0	0	0	0	0	0	0
12	0	0	0	0	0	0	0	0
13	0	0	0	0	0	0	0	0
14	0	0	0	0	0	0	0	0
15	0	0	0	0	0	0	0	0
16	0	0	0	0	0	0	0	0
17	0	0	0	0	0	0	0	0
18	0	0	0	0	0	0	0	0
19	0	0	0	0	0	0	0	0
20	0	0	0	0	0	0	0	0
21	0	0	0	0	0	0	0	0
22	0	0	0	0	0	0	0	0
23	0	0	0	0	0	0	0	0
24	0	0	0	0	0	0	0	0
25	0	0	0	0	0	0	0	0
S*	0	0	0	0	0	0	0	0

Tabla 44. Patrones de entrada por momento temporal correspondientes al estímulo A- presentados en la simulación 18

Ent	Ts1	Ts2	Ts3	Ts4	Ts5	Ts6	Ts7	Ts8
1	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0
11	0	0	0	0	0	0	0	0
12	0	0	0	0	0	0	0	0
13	0	0	0	0	0	0	0	0
14	0	0	0	0	0	0	0	0
15	0	0	0	0	0	0	0	0
16	0	0	0	0	0	0	0	0
17	0	0	0	0	0	0	0	0
18	0	0	0	0	0	0	0	0
19	0	0	0	0	0	0	0	0
20	0	0	0	0	0	0	0	0
21	0	0	0	0	0	0	0	0
22	0	0	0	0	0	0	0	0
23	0	0	0	0	0	0	0	0
24	0	0	0	0	0	0	0	0
25	1	1	1	1	1	1	1	1
S*	0	0	0	0	0	0	0	1

Tabla 45. Patrones de entrada por momento temporal correspondientes al estímulo B+ presentados en la simulación 18

7.3.3 RESULTADOS OBTENIDOS

En la Figura 57 se muestran los valores de activación promedio en rangos de 50 ensayos, de la unidad "CR/UR" (unidad número 44):

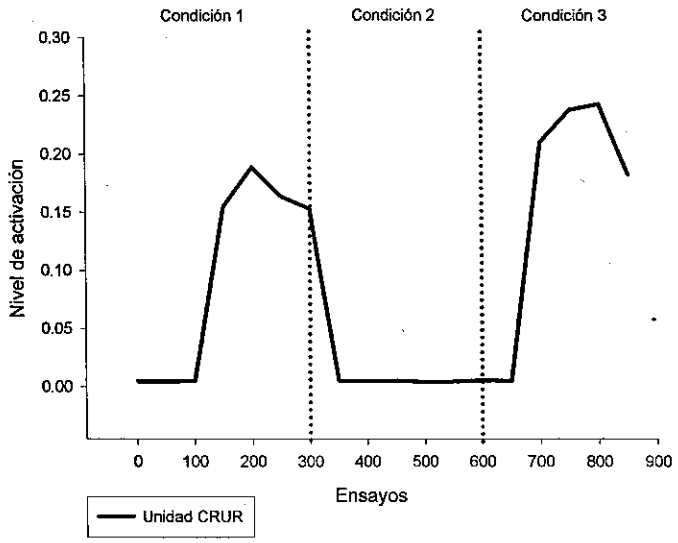


Figura 57. Niveles de activación promedio de la unidad 44 ("CR/UR") después de la simulación 18

Se puede observar en la Figura 57, que la arquitectura empleada, empleando dos subredes, unidas por la unidad "CR/UR" integradora no permite que la red configurada responda con niveles altos de activación. En ambos casos, los valores no sobrepasan el 0.25, por lo que se consideraría que la red no emite respuesta, aunque en sus patrones de elevación y decaimiento de la activación baja registrada si se puede identificar cuándo se presentó cada una de las tres condiciones a las que fue sometida la red.

7.4 SIMULACIÓN 19

En esta simulación se exploró otra posibilidad de integración de las dos subredes que componen la red, esta vez sin emplear una "capa" más de unidades de tipo "ma", sino que se interconectaron las unidades de la primera capa "ma" directamente con la unidad integradora "CR/UR".

Todas las corridas se hicieron considerando 300 ensayos compuestos cada uno, de 8 momentos temporales. Se registró la actividad de la unidad CR/UR en el momento temporal 7 y cuando el refuerzo fue administrado, se hizo en el momento temporal 8, a través de la unidad S*, la que se activó con un valor de 1.0

Se realizaron 10 replicaciones de cada simulación.

7.4.1 ESTRUCTURA DE LA RED

La red empleada se compone de 42 elementos en total, de los cuales 25 son elementos de entrada y 17 son unidades de procesamiento. El número total de conexiones en la red es de 114. En esta simulación se buscó incorporar un elemento de procesamiento que permitiera tener una arquitectura más "cerrada". La red está estructurada de la siguiente manera: las unidades de entrada 1 a la 13 tienen conexiones completas con las unidades 26, 27 y 28 (unidades de tipo "sa"), y las unidades de entrada 13 a la 25 tienen conexiones completas con las unidades 29, 30 y 31 (también unidades del tipo "sa"), a partir de aquí se pueden identificar dos subredes, ya que las unidades "sa" 26, 27 y 28, mantienen conexión con la unidad "ca1" número 32 y con las unidades de tipo "ma" 34, 36 y 36. Asimismo, las unidades de tipo "sa" 29, 30 y 31, mantienen conexiones solamente con las unidades de tipo "ma" 37, 38 y 39, así como con la unidad de tipo "ca1" número 33. Siguiendo con la primera subred, las unidades "ma" 34, 35 y 36, tienen conexión solamente con la unidad "vta" número 40 y con la única unidad de salida "CR/UR" que integra a la red, la número 42. La segunda subred tiene a sus unidades "ma" 37, 38 y 39 conectadas solamente con la unidad "vta" 41 y con la unidad integradora "CR/UR" 42. Se registró la salida en esta unidad "CR/UR". En esta arquitectura la unión entre las dos subredes se da de tres maneras, la primera, a través de las unidades de entrada, en donde la unidad de entrada número 13 tiene conexión con las unidades "ma" de ambas subredes, la segunda, a partir de la unidad integradora "CR/UR" número 42 que recibe la conexión de las unidades "ma" de ambas subredes, y tercero, a partir de los cálculos de las discrepancias tanto en las unidades "ca1" como "vta" que sirven para retroalimentar los pesos de toda la red.

La arquitectura empleada se presenta en la Figura 58:

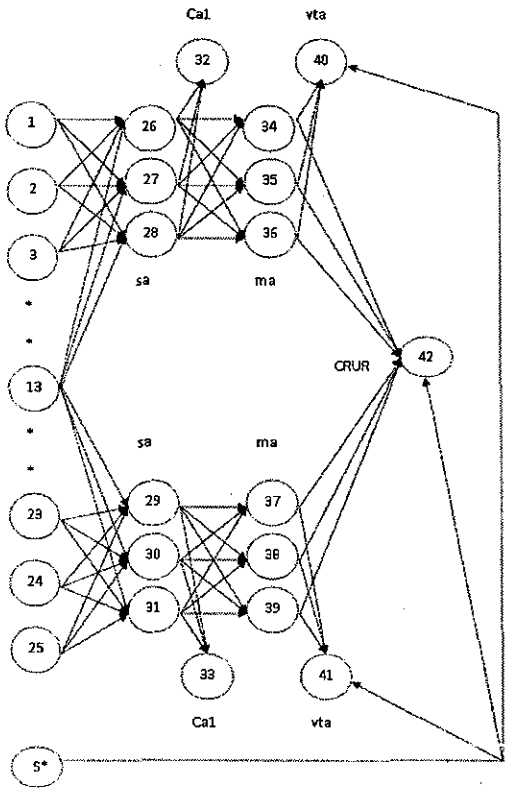


Figura 58. Arquitectura de la Red empleada en la simulación 19

7.4.2 PATRONES DE ENTRADA USADOS

En esta simulación se sometió la red a tres condiciones distintas. En la primera condición se reforzó el estímulo A, en la segunda condición se presentó el mismo estímulo A, pero sin refuerzo, y en la tercera condición se presentó un estímulo diferente, denominado B, el cual recibió refuerzo. El esquema de condiciones para esta simulación fue A+|A-|B+.

Se realizaron 300 ensayos para cada condición y no se empleó re-inicialización de pesos entre la condición A- y la condición B+.

A las unidades de entrada que no recibían activación con un valor de 1.0, les fue asignado un valor de 0.0.

A lo largo de los momentos temporales, los patrones empleados en esta simulación fueron los siguientes:

Ent	Ts1	Ts2	Ts3	Ts4	Ts5	Ts6	Ts7.	Ts8
1	1	1	1	1	1	1	1	1
2	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0
11	0	0	0	0	0	0	0	0
12	0	0	0	0	0	0	0	0
13	0	0	0	0	0	0	0	0
14	0	0	0	0	0	0	0	0
15	0	0	0	0	0	0	0	0
16	0	0	0	0	0	0	0	0
17	0	0	0	0	0	0	0	0
18	0	0	0	0	0	0	0	0
19	0	0	0	0	0	0	0	0
20	0	0	0	0	0	0	0	0
21	0	0	0	0	0	0	0	0
22	0	0	0	0	0	0	0	0
23	0	0	0	0	0	0	0	0
24	0	0	0	0	0	0	0	0
25	0	0	0	0	0	0	0	0
S*	0	0	0	0	0	0	0	1

Tabla 46. Patrones de entrada por momento temporal correspondientes al estímulo A+ presentados en la simulación 19

Ent	Ts1	Ts2	Ts3	Ts4	Ts5	Ts6	Ts7	Ts8
1	1	1	1	1	1	1	1	1
2	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0
11	0	0	0	0	0	0	0	0
12	0	0	0	0	0	0	0	0
13	0	0	0	0	0	0	0	0
14	0	0	0	0	0	0	0	0
15	0	0	0	0	0	0	0	0
16	0	0	0	0	0	0	0	0
17	0	0	0	0	0	0	0	0
18	0	0	0	0	0	0	0	0
19	0	0	0	0	0	0	0	0
20	0	0	0	0	0	0	0	0
21	0	0	0	0	0	0	0	0
22	0	0	0	0	0	0	0	0
23	0	0	0	0	0	0	0	0
24	0	0	0	0	0	0	0	0
25	0	0	0	0	0	0	0	0
S*	0	0	0	0	0	0	0	0

Tabla 47. Patrones de entrada por momento temporal correspondientes al estímulo A- presentados en la simulación 19

Ent	Ts1	Ts2	Ts3	Ts4	Ts5	Ts6	Ts7	Ts8
1	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0
11	0	0	0	0	0	0	0	0
12	0	0	0	0	0	0	0	0
13	0	0	0	0	0	0	0	0
14	0	0	0	0	0	0	0	0
15	0	0	0	0	0	0	0	0
16	0	0	0	0	0	0	0	0
17	0	0	0	0	0	0	0	0
18	0	0	0	0	0	0	0	0
19	0	0	0	0	0	0	0	0
20	0	0	0	0	0	0	0	0
21	0	0	0	0	0	0	0	0
22	0	0	0	0	0	0	0	0
23	0	0	0	0	0	0	0	0
24	0	0	0	0	0	0	0	0
25	1	1	1	1	1	1	1	1
S*	0	0	0	0	0	0	0	1

Tabla 48. Patrones de entrada por momento temporal correspondientes al estímulo B+ presentados en la simulación 19

7.4.3 RESULTADOS OBTENIDOS

En la Figura 59 se muestran los valores de activación promedio en rangos de 50 ensayos, de la unidad "CR/UR" (unidad número 42):

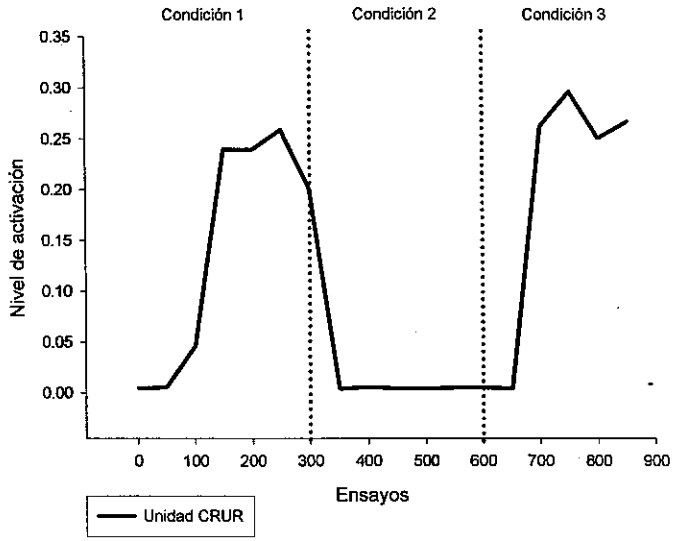


Figura 59. Niveles de activación promedio de la unidad 42 después de la simulación 19

Se puede observar en la Figura 59, que la arquitectura empleada, empleando dos subredes, unidas por la unidad "CR/UR" integradora, recibiendo directa conexión de las unidades "ma" de la red, no permite que la red configurada responda con niveles altos de activación. En ambos casos, los valores no sobrepasan el 0.35, por lo que se consideraría que la red no emite respuesta, aunque en sus patrones de elevación y decaimiento de la activación baja registrada, si se puede identificar cuándo se presentó cada una de las tres condiciones a las que fue sometida la red.

7.5 SIMULACIÓN 20

En esta simulación se probó una posible arquitectura más, dejando de usarse subredes a lo largo de la red configurada y conservando la separación de estímulos, pero solamente a nivel de las entradas. Para ello se empleó una red "pequeña" muy simple. Asimismo, se emplearon diferentes unidades "ca1" y "vta" para cada parte de la red, dividida ahora solamente por las entradas.

Todas las corridas se hicieron considerando 300 ensayos compuestos cada uno, de 8 momentos temporales. Se registró la actividad de la unidad "CR/UR" en el momento temporal 7 y cuando el refuerzo fue administrado, se hizo en el momento temporal 8, a través de la unidad S*, la que se activó con un valor de 1.0

Se realizaron 10 replicaciones de cada simulación.

7.5.1 ESTRUCTURA DE LA RED

La red empleada se compone de 11 elementos en total, de los cuales 2 son elementos de entrada y 9 son unidades de procesamiento. El número total de conexiones en la red es de 12. En esta red, las unidades de entrada envían cada una, una sola conexión con una unidad de tipo "sa". Así, la entrada 1 tiene solamente conexión con la unidad "sa" número 3, y la entrada 2 tiene solamente conexión con la unidad "ma" número 4. La unidad "sa" número 3 tiene conexión con la unidad "ca1" número 5, que solamente recibe conexión de ésta. De igual manera, la unidad "sa" número 4 tiene conexión con la unidad "ca1" número 6, que sólo recibe conexión de ésta. Las unidades "sa" y las unidades "ma" están completamente interconectadas, pero solo la unidad "ma" número 7 tiene conexión con la unidad "vta" número 9 y solo la unidad "ma" número 8 tiene conexión con la unidad "vta" número 10. Ambas unidades "ma" tienen conexión con la única unidad "CR/UR" número 11.

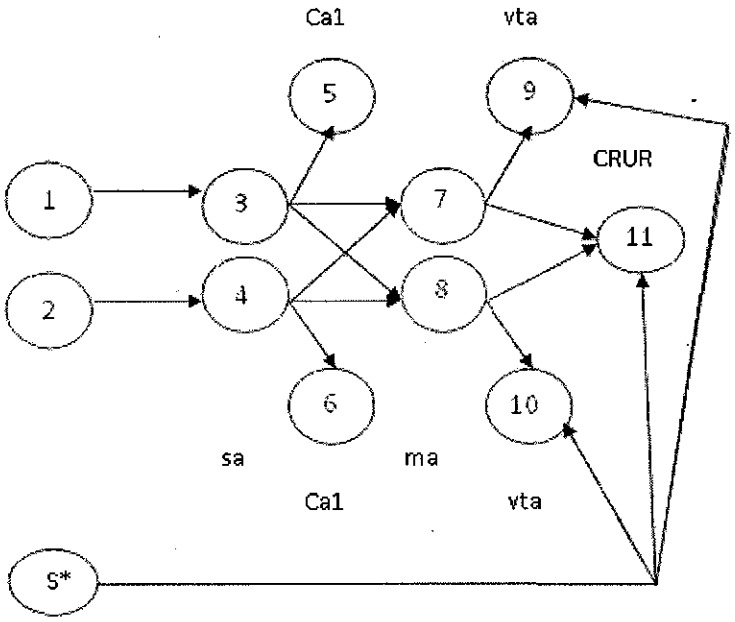


Figura 60. Arquitectura de la Red empleada en la simulación 20

7.5.2 PATRONES DE ENTRADA USADOS

En esta simulación se sometió la red a tres condiciones distintas. En la primera condición se reforzó el estímulo A, en la segunda condición se presentó el mismo estímulo A, pero sin refuerzo, y en la tercera condición se presentó un estímulo diferente, denominado B, el cual recibió refuerzo. El esquema de condiciones para esta simulación fue A+|A-|B+.

Se realizaron 300 ensayos para cada condición y no se empleó re-inicialización de pesos entre la condición A- y la condición B+.

A las unidades de entrada que no recibían activación con un valor de 1.0, les fue asignado un valor de 0.0.

A lo largo de los momentos temporales, los patrones empleados en esta simulación fueron los siguientes:

Ent	Ts1	Ts2	Ts3	Ts4	Ts5	Ts6	Ts7	Ts8
1	1	1	1	1	1	1	1	1
2	0	0	0	0	0	0	0	0
S*	0	0	0	0	0	0	0	1

Tabla 49. Patrones de entrada por momento temporal correspondientes al estímulo A+ presentados en la simulación 20

Ent	Ts1	Ts2	Ts3	Ts4	Ts5	Ts6	Ts7	Ts8
1	1	1	1	1	1	1	1	1
2	0	0	0	0	0	0	0	0
S*	0	0	0	0	0	0	0	0

Tabla 50. Patrones de entrada por momento temporal correspondientes al estímulo A- presentados en la simulación 20

Ent	Ts1	Ts2	Ts3	Ts4	Ts5	Ts6	Ts7	Ts8
1	0	0	0	0	0	0	0	0
2	1	1	1	1	1	1	1	1
S*	0	0	0	0	0	0	0	1

Tabla 51. Patrones de entrada por momento temporal correspondientes al estímulo B+ presentados en la simulación 20

7.5.3 RESULTADOS OBTENIDOS

En la Figura 61 se muestran los valores de activación promedio en rangos de 50 ensayos, de la unidad "CR/UR" (unidad número 11):

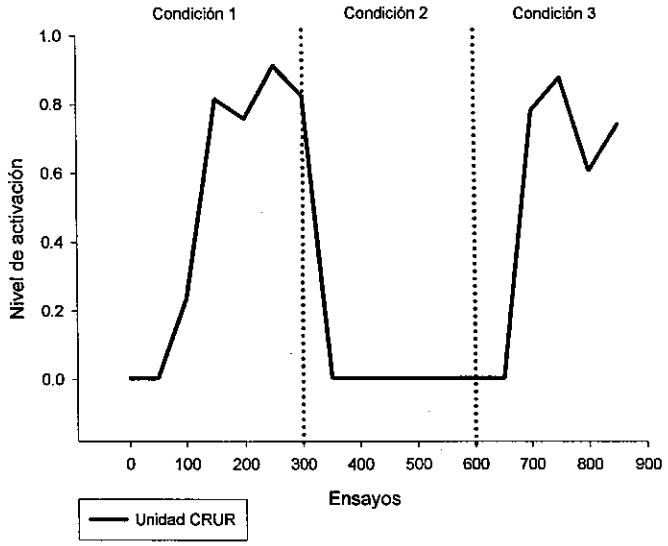


Figura 61. Niveles de activación promedio de la unidad 11 después de la simulación 20

Se puede observar en la Figura 61, que la arquitectura empleada, empleando la separación solamente de las unidades de entrada a través de su conexión con las unidades "sa", y la separación de las conexiones de las unidades "sa" con las unidades "ca1", así como la separación de las unidades "ma" con las unidades "vta", produce niveles elevados de activación ante ambas condiciones en las que se da el reforzamiento de un estímulo. En ambos casos la activación que se presenta está por arriba del 0.6, con lo que se considera que sí hay respuesta ante los dos estímulos: A y B.

7.6 SIMULACIÓN 21

Con base en los resultados de la simulación anterior, se explora la posibilidad de escalar esta arquitectura de red a un mayor número de unidades de proceso y entradas. Se retoma la idea de las subredes, pero bajo esta nueva arquitectura.

Todas las corridas se hicieron considerando 300 ensayos compuestos cada uno, de 8 momentos temporales. Se registró la actividad de la unidad "CR/UR" en el momento temporal 7 y cuando el refuerzo fue administrado, se hizo en el momento temporal 8, a través de la unidad S*, la que se activó con un valor de 1.0

Se realizaron 10 replicaciones de cada simulación.

7.6.1 ESTRUCTURA DE LA RED

La red empleada se compone de 21 elementos en total, de los cuales 4 son elementos de entrada y 17 son unidades de procesamiento. El número total de conexiones en la red es de 24. Esta red se conforma a partir de dos subredes, unidas a nivel de conexiones, solamente por la unidad integradora "CR/UR", número 21.

La estructura de la red empleada se muestra en la Figura 62.

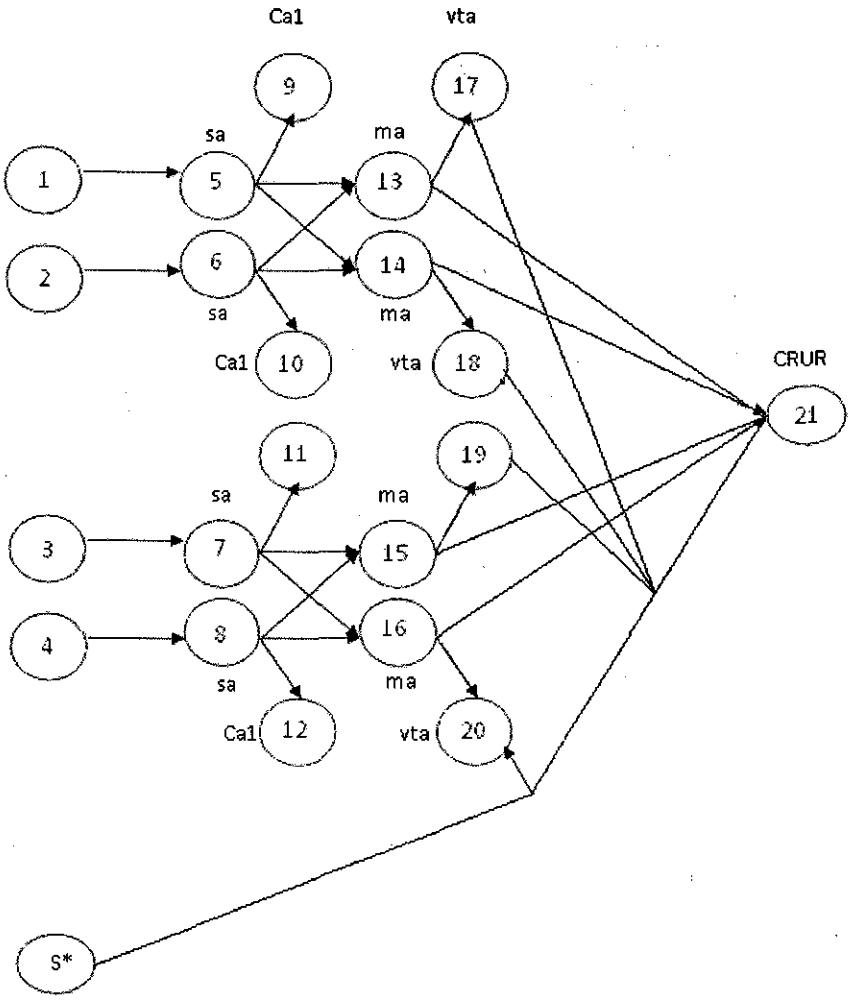


Figura 62. Arquitectura de la Red empleada en la simulación 21

7.6.2 PATRONES DE ENTRADA USADOS

En esta simulación se sometió la red a tres condiciones distintas. En la primera condición se reforzó el estímulo A, en la segunda condición se presentó el mismo estímulo A, pero sin refuerzo, y en la tercera condición se presentó un estímulo diferente, denominado B, el cual recibió refuerzo. El esquema de condiciones para esta simulación fue A+|A-|B+.

Se realizaron 300 ensayos para cada condición y no se empleó re-inicialización de pesos entre la condición A- y la condición B+.

A las unidades de entrada que no recibían activación con un valor de 1.0, les fue asignado un valor de 0.0.

A lo largo de los momentos temporales, los patrones empleados en esta simulación fueron los siguientes:

Ent	Ts1	Ts2	Ts3	Ts4	Ts5	Ts6	Ts7	Ts8
1	1	1	1	1	1	1	1	1
2	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0
S*	0	0	0	0	0	0	0	1

Tabla 52. Patrones de entrada por momento temporal correspondientes al estímulo A+ presentados en la simulación 21

Ent	Ts1	Ts2	Ts3	Ts4	Ts5	Ts6	Ts7	Ts8
1	1	1	1	1	1	1	1	1
2	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0
S*	0	0	0	0	0	0	0	0

Tabla 53. Patrones de entrada por momento temporal correspondientes al estímulo A- presentados en la simulación 21

Ent	Ts1	Ts2	Ts3	Ts4	Ts5	Ts6	Ts7	Ts8
1	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0
4	1	1	1	1	1	1	1	1
S*	0	0	0	0	0	0	0	1

Tabla 54. Patrones de entrada por momento temporal correspondientes al estímulo B+ presentados en la simulación 21

7.6.3 RESULTADOS OBTENIDOS

En la Figura 63 se muestran los valores de activación promedio en rangos de 50 ensayos, de la unidad "CR/UR" (unidad número 21):

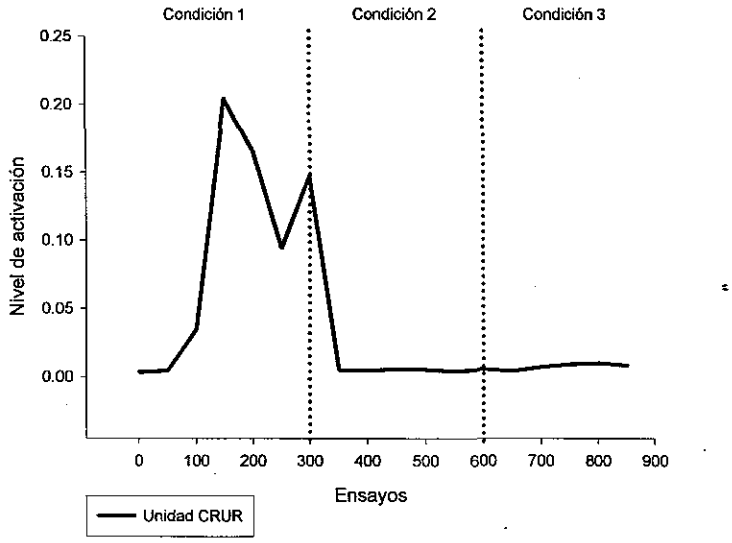


Figura 63. Niveles de activación promedio de la unidad 21, después de la simulación 21

Se puede observar en la Figura 63, que la arquitectura empleada, empleando la separación solamente de las unidades de entrada a través de su conexión con las unidades "sa", y la separación de las conexiones de las unidades "sa" con las unidades "ca1", así como la separación de las unidades "ma" con las unidades "vta", en una red más grande, compuesta por subredes de este tipo, produce niveles muy bajos de activación. Tampoco se puede distinguir, aún con los niveles bajos, que haya algún tipo de efecto en la activación de la unidad "CR/UR" ante la presentación del segundo estímulo reforzado.

7.7 SIMULACIÓN 22

En esta simulación se empleó la misma red que en la simulación anterior, solamente se cambiaron los patrones de entrada, para ver si los resultados ante este tipo de arquitectura dependen de los patrones de entrada presentados.

Todas las corridas se hicieron considerando 300 ensayos compuestos cada uno, de 8 momentos temporales. Se registró la actividad de la unidad "CR/UR" en el momento temporal 7 y cuando el refuerzo fue administrado, se hizo en el momento temporal 8, a través de la unidad S*, la que se activó con un valor de 1.0.

Se realizaron 10 replicaciones de cada simulación.

7.7.1 ESTRUCTURA DE LA RED

La red empleada se compone de 21 elementos en total, de los cuales 4 son elementos de entrada y 17 son unidades de procesamiento. El número total de conexiones en la red es de 24. Esta red se conforma a partir de dos subredes, unidas a nivel de conexiones, solamente por la unidad integradora "CR/UR", número 21. La estructura de la red empleada se muestra en la Figura 64.

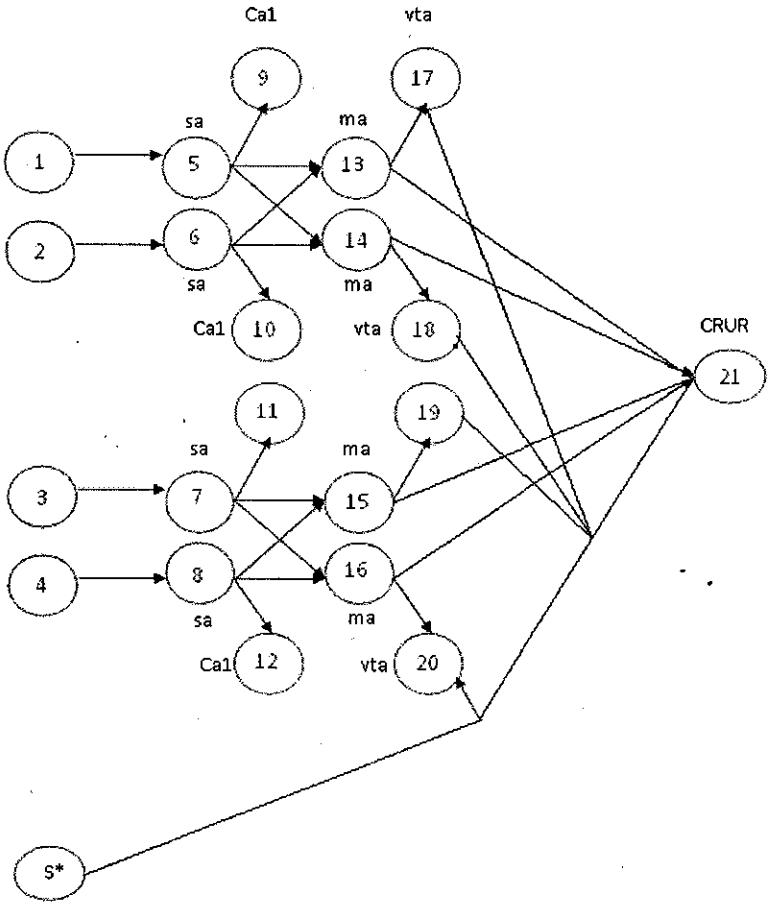


Figura 64. Arquitectura de la Red empleada en la simulación 22

7.7.2 PATRONES DE ENTRADA USADOS

En esta simulación se sometió la red a tres condiciones distintas. En la primera condición se reforzó el estímulo A, en la segunda condición se presentó el mismo estímulo A, pero sin refuerzo, y en la tercera condición se presentó un estímulo diferente, denominado B, el cual recibió refuerzo. El esquema de condiciones para esta simulación fue A+|A-|B+.

Se realizaron 300 ensayos para cada condición y no se empleó re-inicialización de pesos entre la condición A- y la condición B+.

A las unidades de entrada que no recibían activación con un valor de 1.0, les fue asignado un valor de 0.0.

Los patrones específicos de entrada para cada condición, difieren de los empleados en la simulación anterior, debido a que se enfocan a mantener el mismo sub-patrón de entrada para cada subred. Estos nuevos patrones se muestran en las siguientes tablas.

A lo largo de los momentos temporales, los patrones empleados en esta simulación fueron los siguientes:

Ent	Ts1	Ts2	Ts3	Ts4	Ts5	Ts6	Ts7	Ts8
1	1	1	1	1	1	1	1	1
2	0	0	0	0	0	0	0	0
3	1	1	1	1	1	1	1	1
4	0	0	0	0	0	0	0	0
S*	0	0	0	0	0	0	0	1

Tabla 55. Patrones de entrada por momento temporal correspondientes al estímulo A+ presentados en la simulación 22

Ent	Ts1	Ts2	Ts3	Ts4	Ts5	Ts6	Ts7	Ts8
1	1	1	1	1	1	1	1	1
2	0	0	0	0	0	0	0	0
3	1	1	1	1	1	1	1	1
4	0	0	0	0	0	0	0	0
S*	0	0	0	0	0	0	0	0

Tabla 56. Patrones de entrada por momento temporal correspondientes al estímulo A- presentados en la simulación 22

Ent	Ts1	Ts2	Ts3	Ts4	Ts5	Ts6	Ts7	Ts8
1	0	0	0	0	0	0	0	0
2	1	1	1	1	1	1	1	1
3	0	0	0	0	0	0	0	0
4	1	1	1	1	1	1	1	1
S*	0	0	0	0	0	0	0	1

Tabla 57. Patrones de entrada por momento temporal correspondientes al estímulo B+ presentados en la simulación 22

7.7.3 RESULTADOS OBTENIDOS

En la Figura 65 se muestran los valores de activación promedio en rangos de 50 ensayos, de la unidad "CR/UR" (unidad número 21):

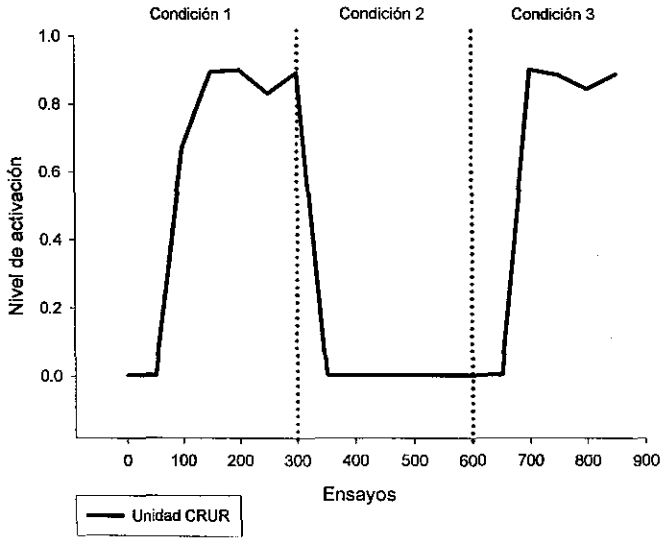


Figura 65. Niveles de activación promedio de la unidad 21 después de la simulación 22

Se puede observar en la Figura 65, que la arquitectura empleada, empleando la separación de las unidades de entrada a través de su conexión con las unidades "sa", y la separación de las conexiones de las unidades "sa" con las unidades "ca1", así como la separación de las unidades "ma" con las unidades "vta", pero manteniendo conexiones completas entre las unidades "sa" y las unidades "ma" de cada subred, no así entre las dos subredes, sí produce niveles elevados de activación ante ambas condiciones en las que se da el reforzamiento de un estímulo, una vez cambiando la configuración de los patrones de entrada para hacerlos iguales para cada una de las subredes. En ambos casos la activación que se presenta está por arriba del 0.8, con lo que se considera que sí hay respuesta ante los dos estímulos: "A" y "B".

7.8 SIMULACIÓN 23

En esta simulación se explora la posibilidad de mejorar el resultado obtenido en la simulación anterior, ya no solamente manteniendo las conexiones completas entre las unidades "sa" y las "ma" de cada sub red, sino sosteniendo una interconexión completa entre todas las unidades "sa" y las unidades "ma" en la red.

Todas las corridas se hicieron considerando 300 ensayos compuestos cada uno, de 8 momentos temporales. Se registró la actividad de la unidad "CR/UR" en el momento temporal 7 y cuando el refuerzo fue administrado, se hizo en el momento temporal 8, a través de la unidad "S*", la que se activó con un valor de 1.0

Se realizaron 10 replicaciones de cada simulación.

7.8.1 ESTRUCTURA DE LA RED

La red empleada se compone de 21 elementos en total, de los cuales 4 son elementos de entrada y 17 son unidades de procesamiento. El número total de conexiones en la red es de 32. En esta red, las entradas tienen solamente una conexión con una unidad "sa", así la entrada 1 tiene conexión solamente con la unidad "sa" número 5, la entrada 2 con la "sa" 6, la entrada 3 con la "sa" 7, y la entrada 4 con la "sa" número 8; la unidad "sa" 5 tiene conexión con la "ca1" 9, la unidad "sa" tiene conexión con la "ca1" 10, la unidad "sa" 7 tiene conexión con la "ca1" 11, y la unidad "sa" 8 tiene conexión con la "ca1" 12. Entre las unidades "sa" 5, 6, 7 y 8, y las unidades "ma" 13, 14, 15 y 16 hay interconexión completa. La unidad "ma" 13 tiene conexión con la unidad "vta" 17, la "ma" 14 tiene conexión con la "vta" 18, la "ma" 15 tiene conexión con la "vta" 19 y la "ma" 16 tiene conexión con la "vta" 20. Todas las unidades "ma" tienen conexión con la única unidad "CR/UR", la número 21.

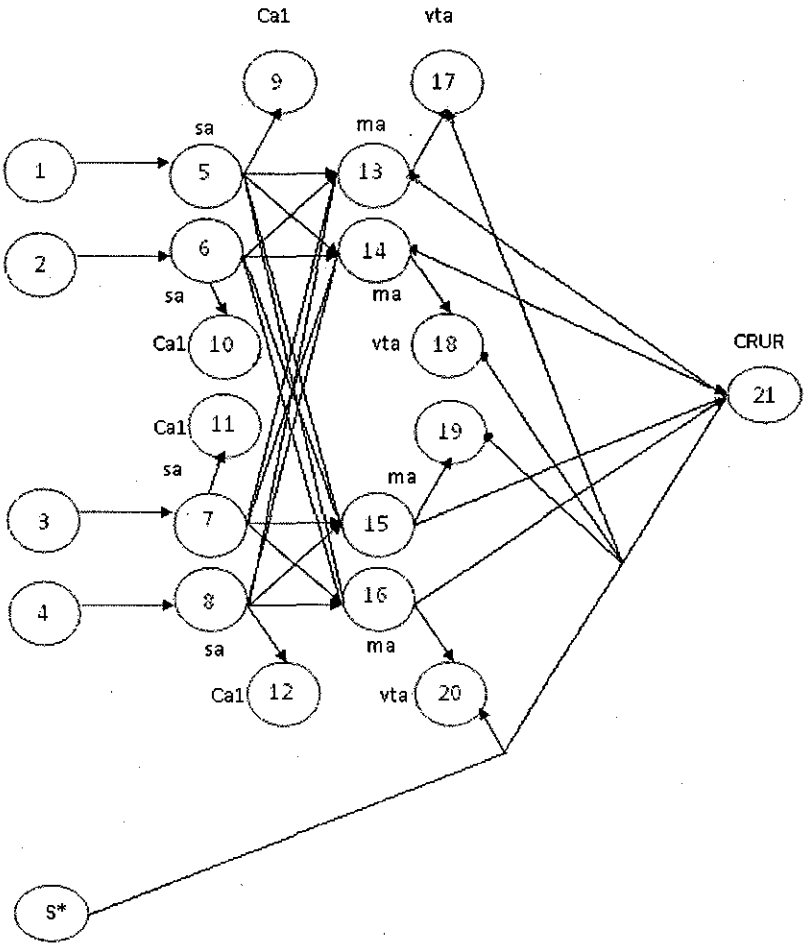


Figura 66. Arquitectura de la Red empleada en la simulación 23

7.8.2 PATRONES DE ENTRADA USADOS

En esta simulación se sometió la red a tres condiciones distintas. En la primera condición se reforzó el estímulo A, en la segunda condición se presentó el mismo estímulo A, pero sin refuerzo, y en la tercera condición se presentó un estímulo diferente, denominado B, el cual recibió refuerzo. El esquema de condiciones para esta simulación fue A+|A-|B+.

Se realizaron 300 ensayos para cada condición y no se empleó re-inicialización de pesos entre la condición A- y la condición B+.

A las unidades de entrada que no recibían activación con un valor de 1.0, les fue asignado un valor de 0.0.

Los patrones específicos de entrada para cada condición, son los mismos empleados en la simulación 11.

Las tablas siguientes muestran dichos patrones.

Ent	Ts1	Ts2	Ts3	Ts4	Ts5	Ts6	Ts7	Ts8
1	1	1	1	1	1	1	1	1
2	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0
S*	0	0	0	0	0	0	0	1

Tabla 58. Patrones de entrada por momento temporal correspondientes al estímulo A+ presentados en la simulación 23

Ent	Ts1	Ts2	Ts3	Ts4	Ts5	Ts6	Ts7	Ts8
1	1	1	1	1	1	1	1	1
2	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0
S*	0	0	0	0	0	0	0	0

Tabla 59. Patrones de entrada por momento temporal correspondientes al estímulo A- presentados en la simulación 23

Ent	Ts1	Ts2	Ts3	Ts4	Ts5	Ts6	Ts7	Ts8
1	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0
4	1	1	1	1	1	1	1	1
S*	0	0	0	0	0	0	0	1

Tabla 60. Patrones de entrada por momento temporal correspondientes al estímulo B+ presentados en la simulación 23

7.8.3 RESULTADOS OBTENIDOS

En la Figura 67 se muestran los valores de activación promedio en rangos de 50 ensayos, de la unidad "CR/UR" (unidad número 21):

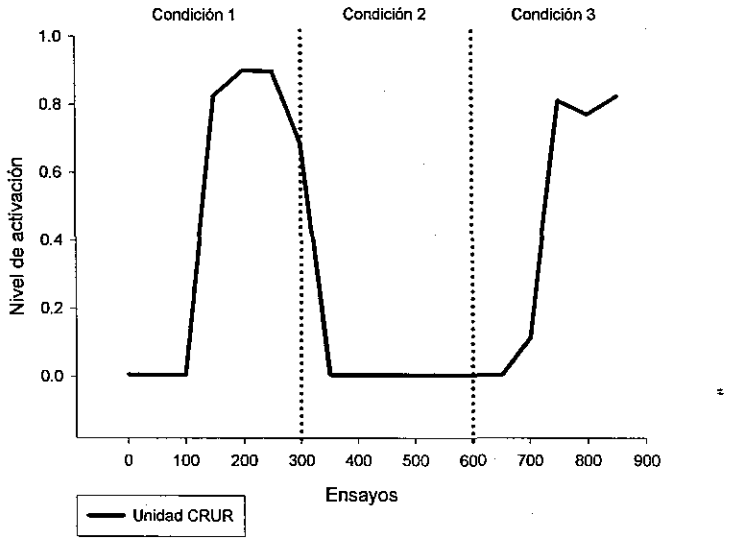


Figura 67. Niveles de activación promedio de la unidad 21 después de la simulación 23

Se puede observar en la Figura 67, que la arquitectura empleada, empleando la separación de las unidades de entrada a través de su conexión con las unidades "sa", y la separación de las conexiones de las unidades "sa" con las unidades "ca1", así como la separación de las unidades "ma" con las

unidades "vta", pero manteniendo conexiones completas entre todas las unidades "sa" y las unidades "ma" de la red, sí produce niveles elevados de activación ante ambas condiciones en las que se da el reforzamiento de un estímulo, sin tener que hacer similares los estímulos que entran a cada subred. En ambos casos la activación que se presenta está por arriba del 0.8, con lo que se considera que sí hay respuesta ante los dos estímulos: "A" y "B".

El problema de la dependencia de la configuración de los patrones de entrada, encontrado en la simulación 12, se resuelve entonces interconectando completamente todas las unidades "sa" con las unidades "sa" en la red.

7.9 SIMULACIÓN 24

En esta simulación se escaló la misma arquitectura usada en la simulación 23, para ver si funciona con redes de mayor tamaño.

Todas las corridas se hicieron considerando 300 ensayos compuestos cada uno, de 8 momentos temporales. Se registró la actividad de la unidad "CR/UR" en el momento temporal 7 y cuando el refuerzo fue administrado, se hizo en el momento temporal 8, a través de la unidad "S*", la que se activó con un valor de 1.0

Se realizaron 10 replicaciones de cada simulación.

7.9.1 ESTRUCTURA DE LA RED

La red empleada se compone de 51 elementos en total, de los cuales 10 son elementos de entrada y 41 son unidades de procesamiento. El número total de conexiones en la red es de 140. En esta red cada unidad de entrada se conecta solamente con una unidad de tipo "sa", cada unidad "ca1" recibe conexión solamente de una unidad "sa", cada unidad "vta" recibe conexión solamente de una unidad "ma", y existe interconexión completa entre todas las unidades "sa" y las unidades "ma", así como entre las unidades "ma" y la unidad de salida "CR/UR". La red creada se muestra en la Figura 68:

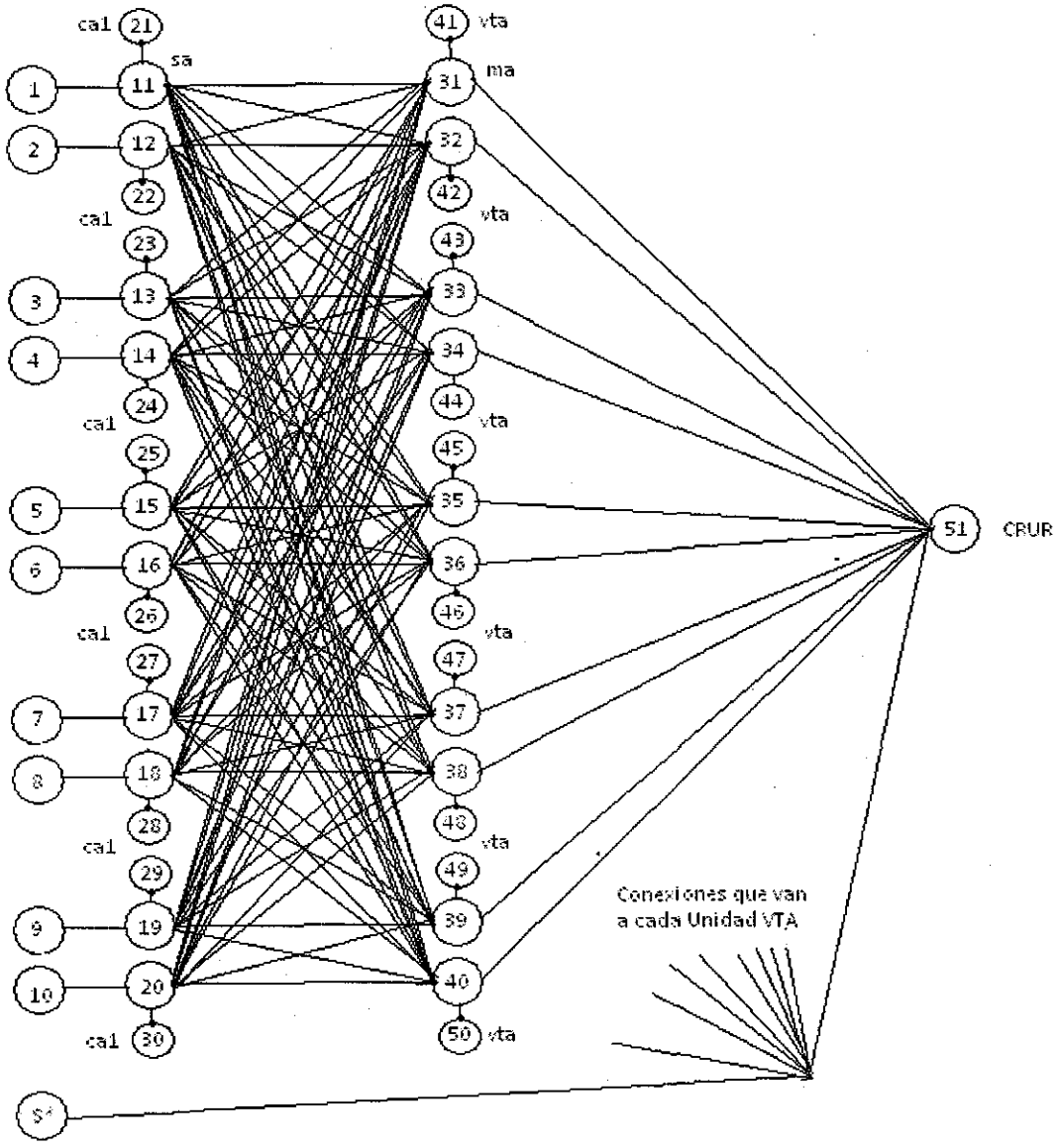


Figura 68. Arquitectura de la Red empleada en la simulación 24

7.9.2 PATRONES DE ENTRADA USADOS

En esta simulación se sometió la red a tres condiciones distintas. En la primera condición se reforzó el estímulo A, en la segunda condición se presentó el mismo estímulo A, pero sin refuerzo, y en la tercera condición se presentó un estímulo diferente, denominado B, el cual recibió refuerzo. El esquema de condiciones para esta simulación fue A+|A-|B+.

Se realizaron 300 ensayos para cada condición y no se empleó re-inicialización de pesos entre la condición A- y la condición B+. A las unidades de entrada que no recibían activación con un valor de 1.0, les fue asignado un valor de 0.0.

A lo largo de los momentos temporales, los patrones empleados en esta simulación fueron los siguientes:

Ent	Ts1	Ts2	Ts3	Ts4	Ts5	Ts6	Ts7	Ts8
1	1	1	1	1	1	1	1	1
2	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0
S*	0	0	0	0	0	0	0	1

Tabla 61. Patrones de entrada por momento temporal correspondientes al estímulo A+ presentados en la simulación 24

Ent	Ts1	Ts2	Ts3	Ts4	Ts5	Ts6	Ts7	Ts8
1	1	1	1	1	1	1	1	1
2	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0
S*	0	0	0	0	0	0	0	1

Tabla 62. Patrones de entrada por momento temporal correspondientes al estímulo A- presentados en la simulación 24

Ent	Ts1	Ts2	Ts3	Ts4	Ts5	Ts6	Ts7	Ts8
1	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0
10	1	1	1	1	1	1	1	1
S*	0	0	0	0	0	0	0	1

Tabla 63. Patrones de entrada por momento temporal correspondientes al estímulo B+ presentados en la simulación 24

7.9.3 RESULTADOS OBTENIDOS

En la Figura 69 se muestran los valores de activación promedio en rangos de 50 ensayos, de la unidad "CR/UR" (unidad número 51):

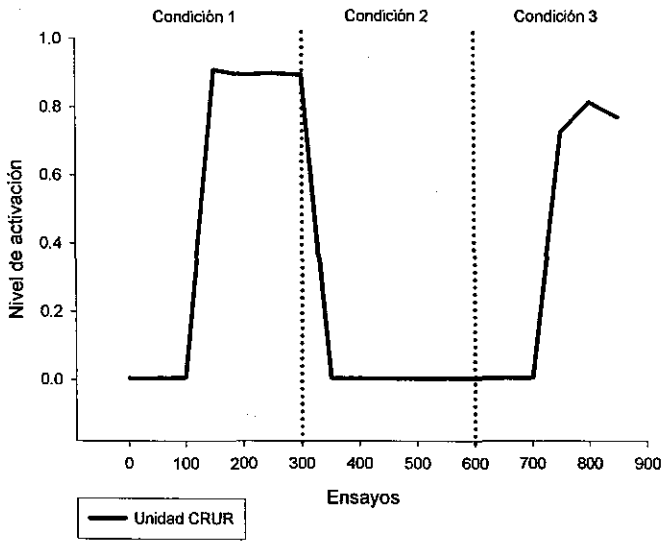


Figura 69. Niveles de activación promedio de la unidad 51 después de la simulación 24

En la Figura 69 se muestra que esta arquitectura permitió obtener niveles altos de activación de la unidad "CR/UR" ante el estímulo A reforzado y ante el estímulo "B" reforzado una vez que se ha extinguido el nivel de activación alcanzado ante el estímulo "A". En ambos casos, los niveles de activación alcanzados sobrepasan el 0.7.

7.10 SIMULACIÓN 25

En esta simulación se exploró la posibilidad de que una red, bajo la arquitectura empleada en las dos simulaciones anteriores, pueda alcanzar niveles de activación altos ante la presentación de dos estímulos reforzados, sin que haya habido una condición de extinción para el primero, antes de presentar el segundo.

Todas las corridas se hicieron considerando 300 ensayos compuestos cada uno, de 8 momentos temporales. Se registró la actividad de la unidad "CR/UR" en el momento temporal 7 y cuando el refuerzo fue administrado, se hizo en el momento temporal 8, a través de la unidad "S*", la que se activó con un valor de 1.0

Se realizaron 10 replicaciones de cada simulación.

7.10.1 ESTRUCTURA DE LA RED

La red empleada se compone de 51 elementos en total, de los cuales 10 son elementos de entrada y 41 son unidades de procesamiento. El número total de conexiones en la red es de 140. La forma en la que estas unidades se conectan ha sido especificada en el apartado correspondiente a la simulación anterior. En la Figura 70 se muestra la arquitectura empleada para esta red.

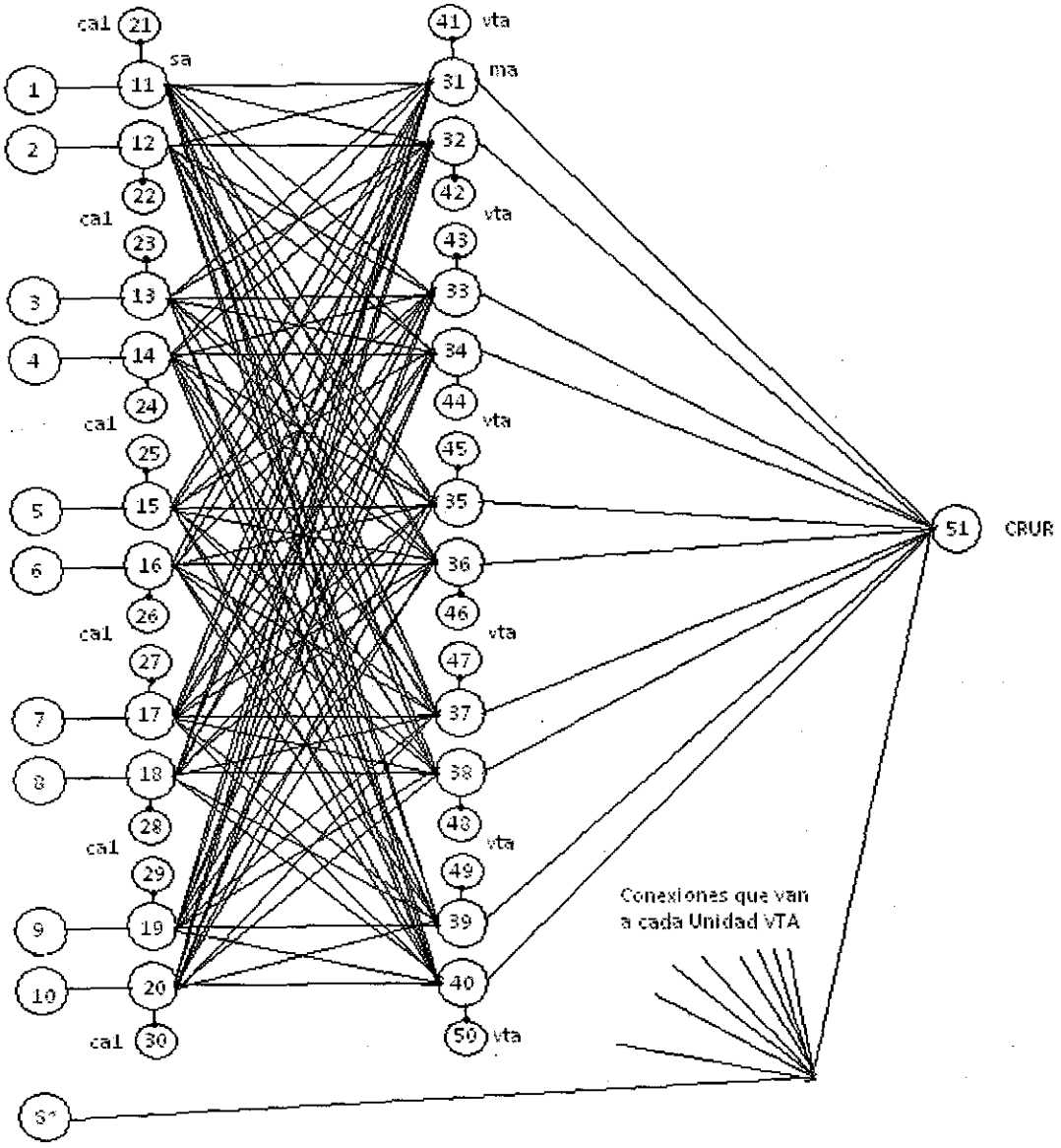


Figura 70. Arquitectura de la Red empleada en la simulación 25

7.10.2 PATRONES DE ENTRADA USADOS

En esta simulación se sometió la red a dos condiciones distintas. En la primera condición se reforzó el estímulo A, mientras que en la segunda condición se presentó un estímulo diferente, denominado B, el cual recibió refuerzo. El esquema de condiciones para esta simulación fue A+|A-|B+.

Se realizaron 300 ensayos para cada condición y no se empleó re-inicialización de pesos entre la condición A+ y la condición B+. A las unidades de entrada que no recibían activación con un valor de 1.0, les fue asignado un valor de 0.0.

A lo largo de los momentos temporales, los patrones empleados en esta simulación fueron los siguientes:

Ent	Ts1	Ts2	Ts3	Ts4	Ts5	Ts6	Ts7	Ts8
1	1	1	1	1	1	1	1	1
2	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0
S*	0	0	0	0	0	0	0	1

Tabla 64. Patrones de entrada por momento temporal correspondientes al estímulo A+ presentados en la simulación 25

Ent	Ts1	Ts2	Ts3	Ts4	Ts5	Ts6	Ts7	Ts8
1	1	1	1	1	1	1	1	1
2	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0
S*	0	0	0	0	0	0	0	0

Tabla 65. Patrones de entrada por momento temporal correspondientes al estímulo A- presentados en la simulación 25

Ent	Ts1	Ts2	Ts3	Ts4	Ts5	Ts6	Ts7	Ts8
1	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0
10	1	1	1	1	1	1	1	1
S*	0	0	0	0	0	0	0	1

Tabla 66. Patrones de entrada por momento temporal correspondientes al estímulo B+ presentados en la simulación 25

7.10.3 RESULTADOS OBTENIDOS

En la Figura 71 se muestran los valores de activación promedio en rangos de 50 ensayos, de la unidad "CR/UR" (unidad número 51):

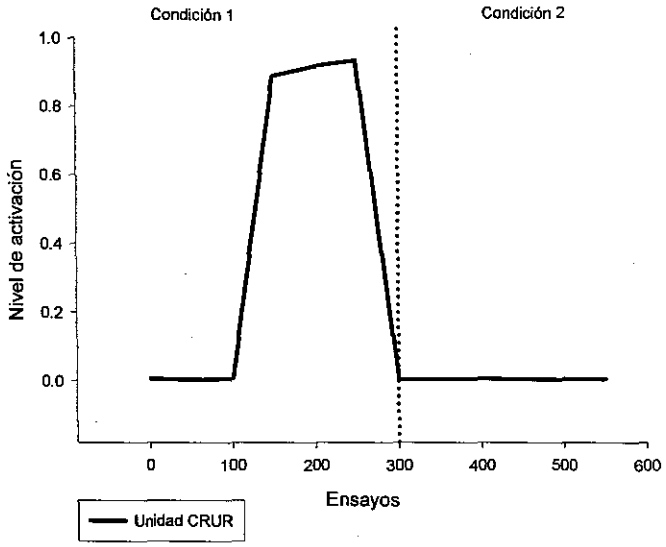


Figura 71. Niveles de activación promedio de la unidad 51 (CR/UR) después de la simulación 25

En la Figura 71 se puede observar un nivel elevado de activación de la unidad "CR/UR" ante la presentación del estímulo A reforzado, pero los niveles de activación no se elevan ante la presentación del estímulo B reforzado en la segunda condición. Relacionando este resultado con el resultado de la simulación anterior, se puede afirmar entonces, que para una red con esta arquitectura no es posible presentar niveles de activación ante un segundo estímulo reforzado, sin que previamente haya una condición en la que se presente el primer estímulo sin refuerzo, es decir, sin haber extinguido la activación lograda ante un estímulo previamente reforzado.

7.11 SIMULACIÓN 26

Ante la imposibilidad presentada por la arquitectura anterior para aprender a presentar niveles elevados de activación en la unidad "CR/UR" correspondiente, ante un segundo estímulo sin haber extinguido un estímulo previamente aprendido, se diseñó la presente simulación para averiguar si la arquitectura anteriormente usada, con base en subredes, era capaz de aprender un segundo estímulo sin una condición previa en la que se extinguiera la respuesta ante el estímulo ya aprendido, y para verificar los niveles de activación obtenidos en las unidades "CR/UR" ante la presentación alternada de los estímulos previamente entrenados.

Todas las corridas se hicieron considerando 300 ensayos compuestos cada uno, de 8 momentos temporales. Se registró la actividad de las unidades "CR/UR" en el momento temporal 7 y cuando el refuerzo fue administrado, se hizo en el momento temporal 8, a través de la unidad "S*", la que se activó con un valor de 1.0. Se realizó una sola corrida de esta simulación.

7.11.1 ESTRUCTURA DE LA RED

La red empleada en esta simulación se compone de 43 elementos en total, de los cuales 25 son elementos de entrada y 18 son unidades de procesamiento. El número total de conexiones en la red es de 114. Esta red se estructuró con base en una arquitectura de subredes, con dos unidades de

respuesta "CR/UR", ya empleada en simulaciones anteriores en este mismo trabajo. La red configurada se presenta en la Figura 72:

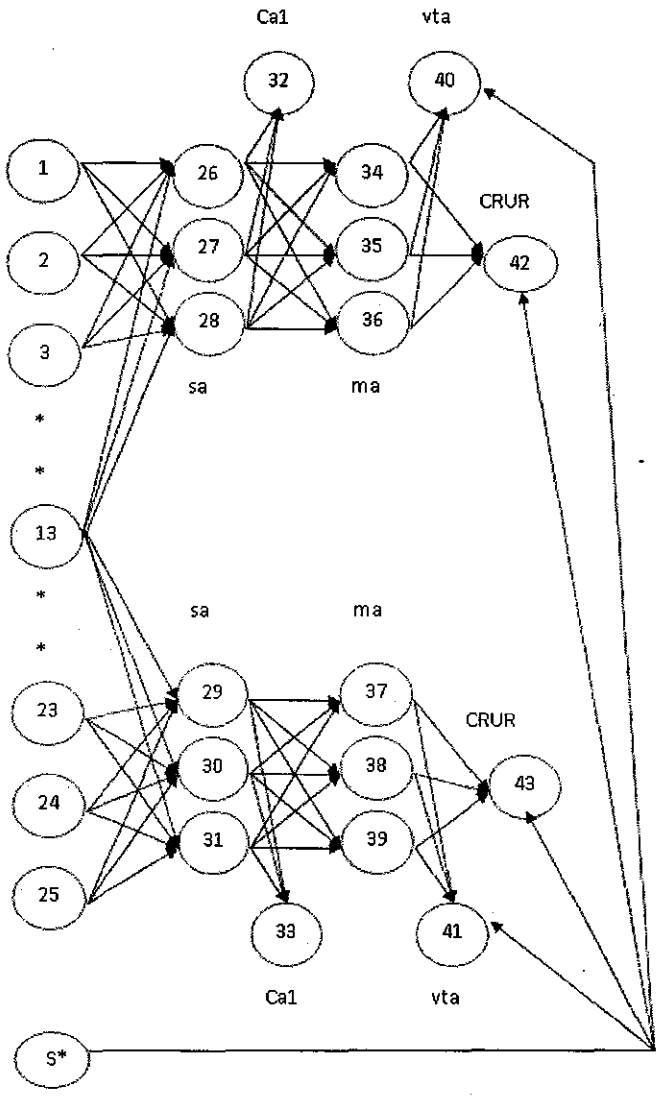


Figura 72. Arquitectura empleada en la simulación 26

7.11.2 PATRONES DE ENTRADA USADOS

En esta simulación se sometió la red a tres condiciones distintas. En la primera condición se reforzó el estímulo A, mientras que en la segunda condición se presentó un estímulo diferente, denominado B, el cual recibió refuerzo. En la tercera condición se presentaron ambos estímulos alternadamente.

El esquema de las condiciones en la simulación fue el siguiente: A+|B+ |A+/B+.

Se realizaron 300 ensayos para cada condición y no se empleó re-inicialización de pesos entre el final de una condición y el inicio de la siguiente.

A las unidades de entrada que no recibían activación con un valor de 1.0, les fue asignado un valor de 0.0.

A lo largo de los momentos temporales, los patrones empleados en esta simulación fueron los siguientes:

Ent	Ts1	Ts2	Ts3	Ts4	Ts5	Ts6	Ts7	Ts8
1	1	1	1	1	1	1	1	1
2	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0
11	0	0	0	0	0	0	0	0
12	0	0	0	0	0	0	0	0
13	0	0	0	0	0	0	0	0
14	0	0	0	0	0	0	0	0
15	0	0	0	0	0	0	0	0
16	0	0	0	0	0	0	0	0
17	0	0	0	0	0	0	0	0
18	0	0	0	0	0	0	0	0
19	0	0	0	0	0	0	0	0
20	0	0	0	0	0	0	0	0
21	0	0	0	0	0	0	0	0
22	0	0	0	0	0	0	0	0
23	0	0	0	0	0	0	0	0
24	0	0	0	0	0	0	0	0
25	0	0	0	0	0	0	0	0
S*	0	0	0	0	0	0	0	1

Tabla 67. Patrones de entrada por momento temporal correspondientes al estímulo A+ presentados en la simulación 26

Ent	Ts1	Ts2	Ts3	Ts4	Ts5	Ts6	Ts7	Ts8
1	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0
11	0	0	0	0	0	0	0	0
12	0	0	0	0	0	0	0	0
13	0	0	0	0	0	0	0	0
14	0	0	0	0	0	0	0	0
15	0	0	0	0	0	0	0	0
16	0	0	0	0	0	0	0	0
17	0	0	0	0	0	0	0	0
18	0	0	0	0	0	0	0	0
19	0	0	0	0	0	0	0	0
20	0	0	0	0	0	0	0	0
21	0	0	0	0	0	0	0	0
22	0	0	0	0	0	0	0	0
23	0	0	0	0	0	0	0	0
24	0	0	0	0	0	0	0	0
25	1	1	1	1	1	1	1	1
S*	0	0	0	0	0	0	0	1

Tabla 68. Patrones de entrada por momento temporal correspondientes al estímulo B+ presentados en la simulación 26

7.11.3 RESULTADOS OBTENIDOS EN LAS DOS PRIMERAS CONDICIONES

En la Figura 73 se muestran los valores de activación alcanzados por las unidades "CR/UR" (unidades número 42 y 43):

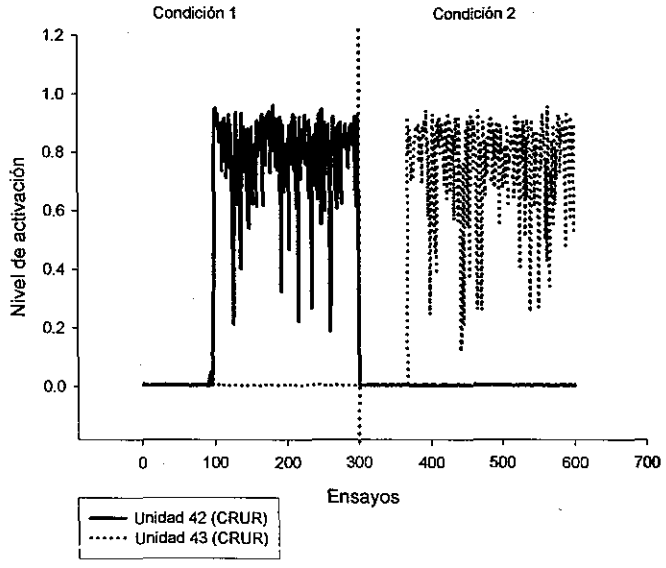


Figura 73. Niveles de activación de las unidades 42 y 43 ("CR/UR") en las dos primeras condiciones de la simulación 26

En la Figura 73 se pueden observar niveles altos de activación en las unidades de respuesta "CR/UR" en la condición correspondiente, tras la presentación del estímulo reforzado. La unidad "CR/UR" número 42, muestra niveles de activación elevados ante la presentación del estímulo A reforzado, mientras que sus niveles permanecen cercanos a 0.0 durante la segunda condición. De la misma manera, la unidad "CR/UR" número 43, presenta niveles de activación cercanos al 0.0 durante la primera condición y presenta un elevamiento en el nivel de activación en la segunda condición, cuando se presentó el estímulo "B" reforzado.

7.11.4 RESULTADOS OBTENIDOS EN LA TERCERA CONDICION

En esta tercera condición se presentaron 300 ensayos con la presentación alternada del estímulo "A", reforzado y el estímulo B reforzado.

7.11.5 RESULTADOS OBTENIDOS

En las Figuras 74 y 75 se muestran los valores de activación alcanzados por las unidades "CR/UR" (unidades número 42 y 43), en esta condición adicional, en la que se presentaron alternadamente ambos estímulos reforzados:

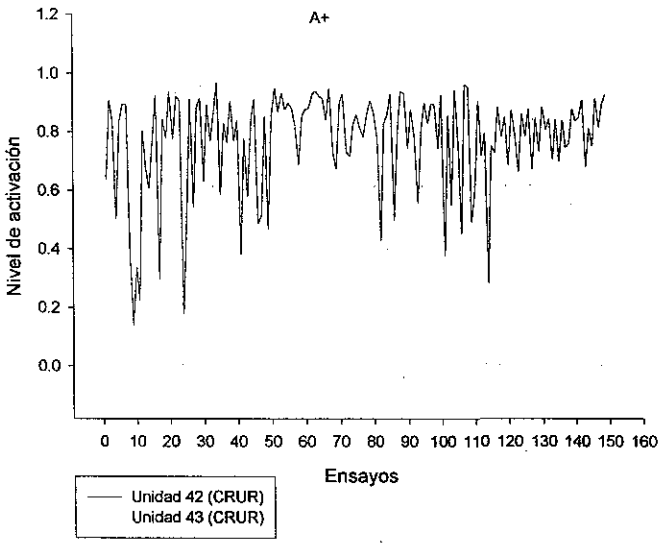


Figura 74. Niveles de activación de las unidades 42 y 43 ("CR/UR") después de la tercera condición de la simulación 26

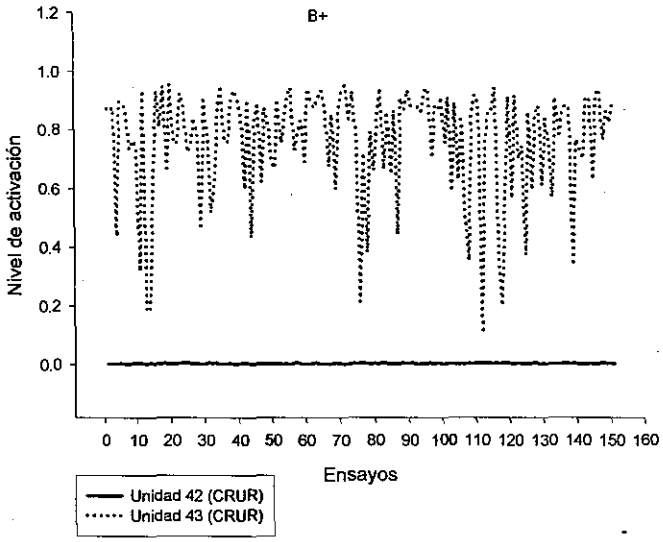


Figura 75. Niveles de activación de las unidades 42 y 43 ("CR/UR") después de la tercera condición de la simulación 26

En las Figuras 74 y 75, se muestra que los niveles de activación permanecieron altos en esta condición de presentaciones alternadas de ambos estímulos. Cuando el estímulo "A" reforzado se presentaba, el nivel de activación de la unidad "CR/UR" 42 se presentaba alto, mientras que el nivel de activación de la unidad "CR/UR" 43 permanecía en valores cercanos al 0.0, y ante la presentación del estímulo "B" reforzado, los niveles de activación de la unidad 42 permanecian cercanos a 0.0, mientras que los de la unidad 43 mostraban niveles elevados.

En esta simulación se comprueba que la arquitectura con base en subredes no solamente es capaz de aprender a responder ante un segundo estímulo sin haberse extinguido la respuesta ante un estímulo distinto previamente entrenado, sino que también es capaz de discriminar entre un estímulo y otro cuando son presentados alternadamente.

7.12 SIMULACIÓN 27

Esta simulación se realizó para ver si la arquitectura empleada en la simulación anterior, se puede escalar a mayores dimensiones para poder aprender a mostrar valores de activación altos ante tres estímulos condicionados distintos.

Todas las corridas se hicieron considerando 300 ensayos compuestos cada uno, de 8 momentos temporales. Se registró la actividad de las unidades "CR/UR" en el momento temporal 7 y cuando el refuerzo fue administrado, se hizo en el momento temporal 8, a través de la unidad "S*", la que se activó con un valor de 1.0

Se corrieron 10 replicaciones de esta simulación.

7.12.1 ESTRUCTURA USADA

En la Figura 76 se muestra la arquitectura empleada, compuesta por tres subredes: la primera compuesta por una unidad "sa" (unidad número 12), una unidad "ca1" (la número 15), una unidad "ma" (la número 18), una unidad "vta" (la número 21) y una unidad "CR/UR" (la número 24). Las otras dos subredes están compuestas por las unidades 13, 16, 19 22 y 25, así como por las unidades 14, 17, 20, 23 y 26 respectivamente. La respuesta se registró en las 3 unidades de salida "CR/UR", es decir, en las unidades 24, 25 y 26. En este caso, se usan las unidades de entrada 1 y 11, como unidades de "unión" de las tres subredes, estableciendo conexiones con todas las unidades "sa" de la red.

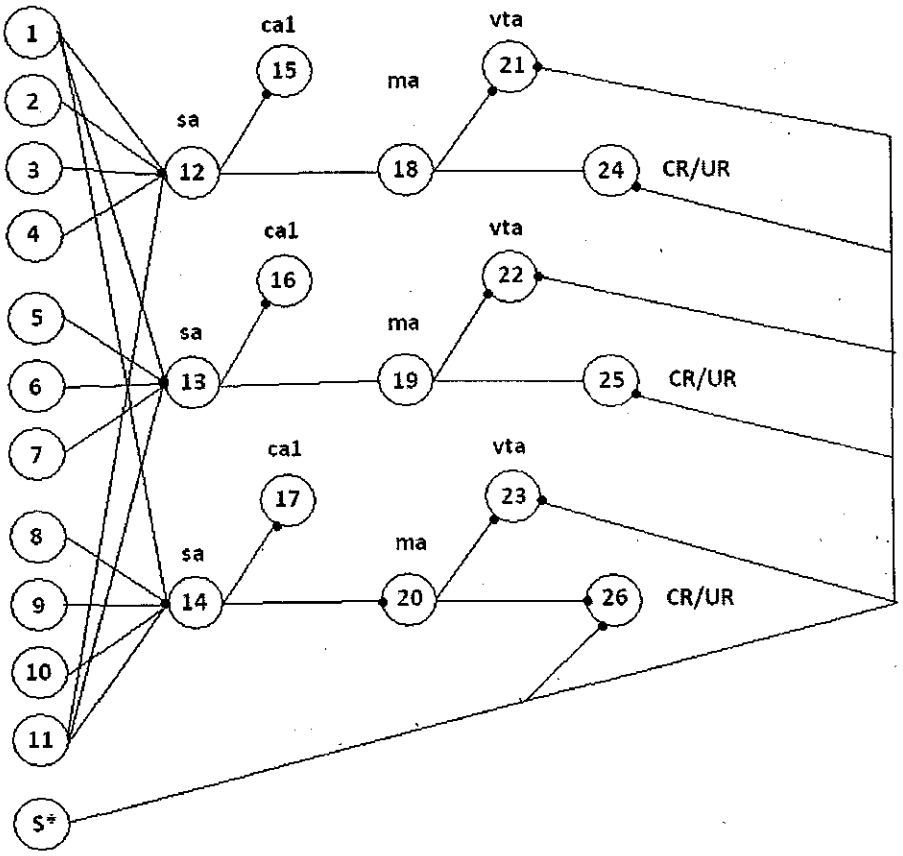


Figura 76. Arquitectura empleada en la simulación 27

7.12.2 PATRONES DE ENTRADA USADOS

En esta simulación se sometió la red a tres condiciones distintas. En la primera condición se reforzó el estímulo A, mientras que en la segunda condición se presentó un estímulo diferente, denominado B, el cual recibió refuerzo. En la tercera condición se presentó un tercer estímulo reforzado distinto a los anteriores, el estímulo C.

El esquema de las condiciones en la simulación fue el siguiente: A+|B+|C+.

Se realizaron 300 ensayos para cada condición y no se empleó re-inicialización de pesos entre la condición A+ y la condición B+, ni entre la condición B+ y la condición C+. A las unidades de entrada que no recibían activación con un valor de 1.0, les fue asignado un valor de 0.0.

A lo largo de los momentos temporales, los patrones empleados en esta simulación fueron los siguientes:

Ent	Ts1	Ts2	Ts3	Ts4	Ts5	Ts6	Ts7	Ts8
1	0	0	0	0	0	0	0	0
2	1	1	1	1	1	1	1	1
3	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0
11	0	0	0	0	0	0	0	0
S*	0	0	0	0	0	0	0	1

Tabla 69. Patrones de entrada por momento temporal correspondientes al estímulo A+ presentados en la simulación 27

Ent	Ts1	Ts2	Ts3	Ts4	Ts5	Ts6	Ts7	Ts8
1	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0
5	1	1	1	1	1	1	1	1
6	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0
11	0	0	0	0	0	0	0	0
S*	0	0	0	0	0	0	0	1

Tabla 70. Patrones de entrada por momento temporal correspondientes al estímulo B+ presentados en la simulación 27

Ent	Ts1	Ts2	Ts3	Ts4	Ts5	Ts6	Ts7	Ts8
1	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0
8	1	1	1	1	1	1	1	1
9	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0
11	0	0	0	0	0	0	0	0
S*	0	0	0	0	0	0	0	1

Tabla 71. Patrones de entrada por momento temporal correspondientes al estímulo C+ presentados en la simulación 27

7.12.3 RESULTADOS OBTENIDOS

En la Figura 77 se muestran los valores promedio de activación de las unidades "CR/UR" (unidades "CR/UR" números 24, 25 y 26):

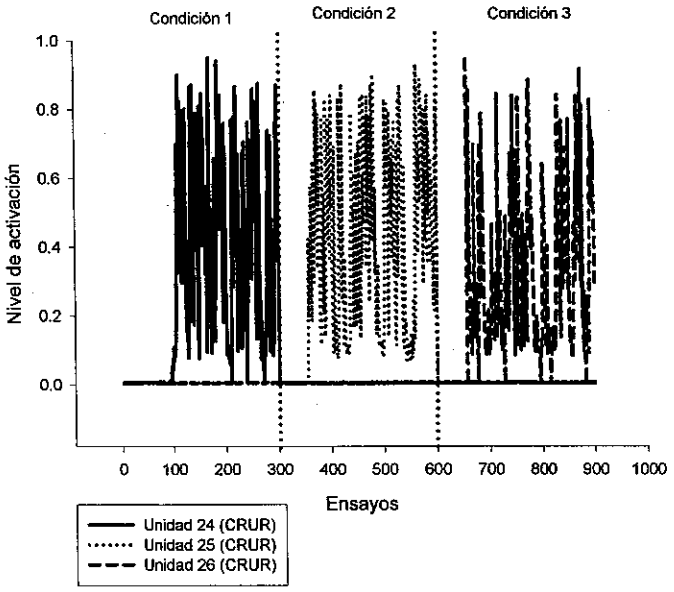


Figura 77. Niveles de activación de las unidades 24,25 y 26 (CR/UR) en la simulación 27

En la Figura 77 se observa, que aunque es posible manejar tres estímulos en la misma red sin necesidad de extinguir ninguno de ellos, la arquitectura produce niveles elevados de activación, aunque los niveles no son tan elevados los alcanzados en otras simulaciones.

7.13 DISCUSIÓN GENERAL DEL CAPÍTULO

Este capítulo exploró dos tipos generales de arquitecturas, ambos tipos se caracterizaron por el uso de subredes para procesar estímulos distintos y por no usar interconexiones completas entre sus elementos, variando el grado de interconexión parcial de una instancia a otra.

El primer tipo de arquitecturas empleó tres tipos de estrategias de integración de las subredes a nivel arquitectónico: 1) Dos de las redes emplearon integración a partir de las conexiones de las unidades de entrada con las unidades de tipo "sa" (marcadas en la Figura 78 como a) y b); 2) Todas las redes usaron a la unidad de salida "CR/UR" como medio para integrar las subredes establecidas para cada tipo de estímulo a manejar en la red completa. Así, se esperaba la salida de la red en sólo en la unidad de respuesta integradora, sin importar cuantas subredes constituyeran el circuito neural completo; y, 3) Tres redes (las marcadas en la Figura 78 como c), d) y e) proyectaban conexiones a otras subredes a través de las unidades "sa" y "ma". Sin embargo, la integración de las subredes también se dio a nivel funcional, ya que los cálculos de las discrepancias para el posterior ajuste de los valores de los pesos, se realizó con base en la totalidad de las unidades "ca1" y de las unidades "vta".

La Figura 78 muestra un concentrado de las arquitecturas empleadas en las simulaciones de este capítulo, que emplearon como elemento arquitectónico integrador a la unidad de salida "CR/UR".

La red marcada como a) emplea subredes que emplean al final una sola unidad de tipo "ma" la cual envía conexión a la unidad integradora de tipo "CR/UR". Esta red es distinta al resto de las empleadas en este trabajo, ya que usa dos capas de unidades "ma" en cada una de las subredes,

una compuesta por 3 unidades y la otra, compuesta por una sola unidad. La red marcada como b) emplea al final de cada subred, una sola capa de unidades de tipo "ma", las cuales envían conexiones directamente a la única unidad integradora de la red (tipo "CR/UR").

En el caso de las redes marcadas en la Figura 78 como c), d) y e) se puede observar que conservan aún las subredes dedicadas al proceso de cada estímulo distinto de entrada, sin embargo, estas subredes se encuentran parcialmente interconectadas entre sí, ya que se mantienen conexiones entre las unidades "sa" y "ma" en toda la red. Este tipo de redes contó una unidad "ca1" por cada unidad de tipo "sa" y una unidad de tipo "vta" por cada unidad de tipo "ma". En estos casos cada unidad de tipo "sa" contaba con una conexión a una unidad de tipo "ca1" que solo recibía conexión de ésta, y cada unidad de tipo "ma" enviaba una conexión a una unidad de tipo "vta" que sólo recibía conexión de ésta. Asimismo, se estableció una diferencia más en relación con el resto de las redes empleadas, ya que cada unidad de entrada mantenía solamente una conexión con su correspondiente unidad de tipo "sa".

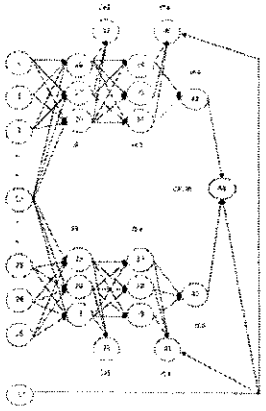
Sólo las redes marcadas en la Figura 78 como c), d) y e) fueron capaces de aprender a responder ante un segundo EC, pero solamente existiendo un proceso previo de extinción del EC anteriormente entrenado. Las redes a) y b) fueron incapaces de hacerlo, bajo las condiciones presentadas en las simulaciones.

El segundo tipo de arquitectura usó salidas distintas para cada uno de los estímulos procesados en la red; así, si se requería que la red pudiese responder ante dos estímulos distintos, se establecían dos subredes integradas en un solo sistema, al igual que en el tipo anterior, a dos niveles: a) arquitectónico, a partir de las conexiones que las unidades de entrada mantenían con las unidades tipo "sa", y b) funcional, a partir de los cálculos generales de las discrepancias tanto en las unidades tipo "ca1", como en las unidades tipo "vta" y el posterior reajuste de los valores de los pesos en consecuencia de estos cálculos.

En la Figura 79 se muestra un concentrado de las arquitecturas probadas en este capítulo que se constituyeron a partir de subredes, usando unidades de respuesta especiales para cada uno de los estímulos a manejar. Las arquitecturas marcadas en la Figura 79 como a), b) y c) difieren solamente en el tamaño, pudiéndose observar claramente al interior de éstas, que existen subredes que en su estructura son muy similares a la arquitectura clásica usada en el modelo DBP, ya descrita en el capítulo anterior: integran 3 unidades de tipo "sa", 1 unidad de tipo "ca1", 3 unidades de tipo "ma", 1 unidad tipo "vta" y 1 unidad de tipo "CR/UR". Cada subred se encuentra completamente interconectada con los elementos que la componen, excepto entre las entradas y las unidades tipo "sa" que es donde se da la integración de las subredes a nivel arquitectónico.

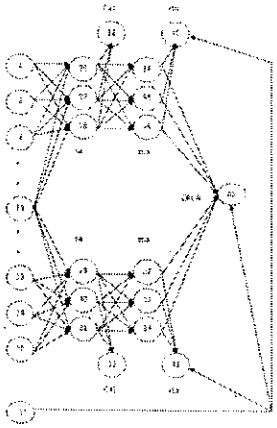
En la arquitectura marcada como d) en la misma Figura 79, se observan cambios en cuanto al número de unidades. En este caso, se trata de subredes compuestas por 1 sola unidad de cada tipo, en donde cada unidad, por consecuencia, recibe solamente una conexión, excepto entre las unidades de entrada y la unidad de tipo "sa" de cada subred.

a)



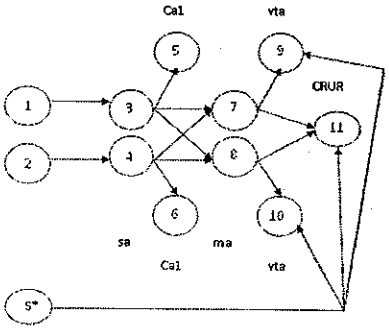
Red usada en la simulación 18

b)



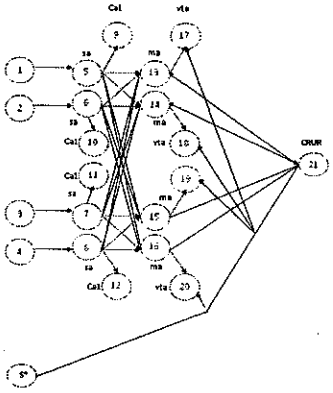
Red usada en la simulación 19

c)



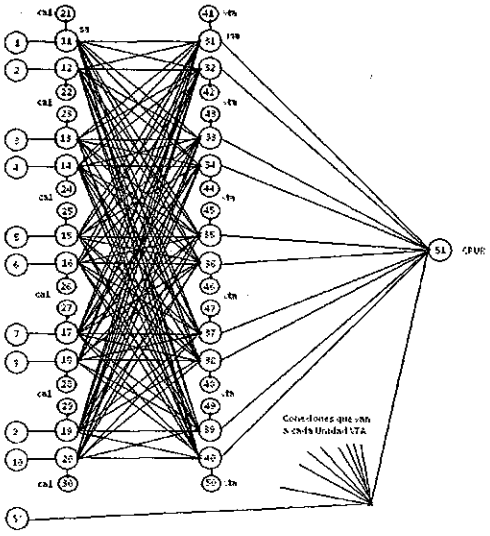
Red usada en la simulación 20

d)



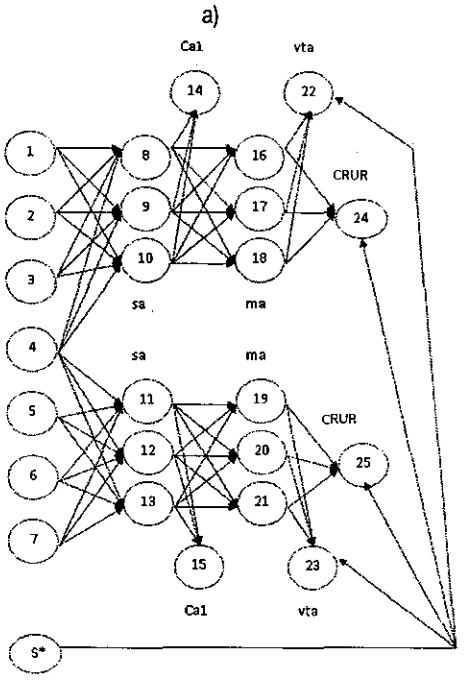
Red usada en las simulaciones 21, 22 y 23

e)

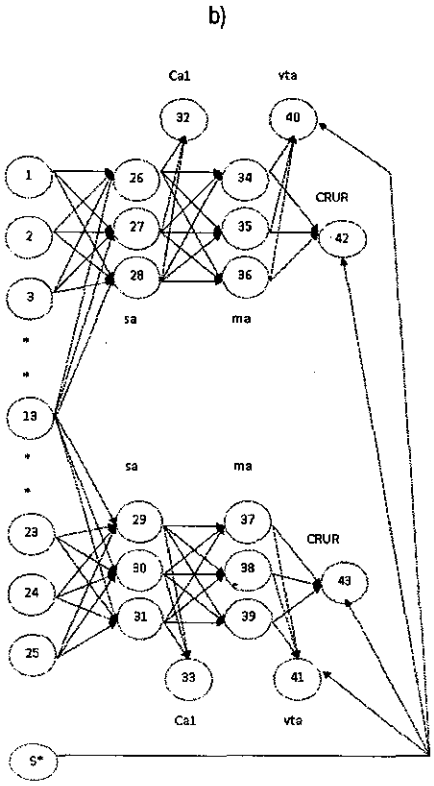


Red usada en las simulaciones 24 y 25

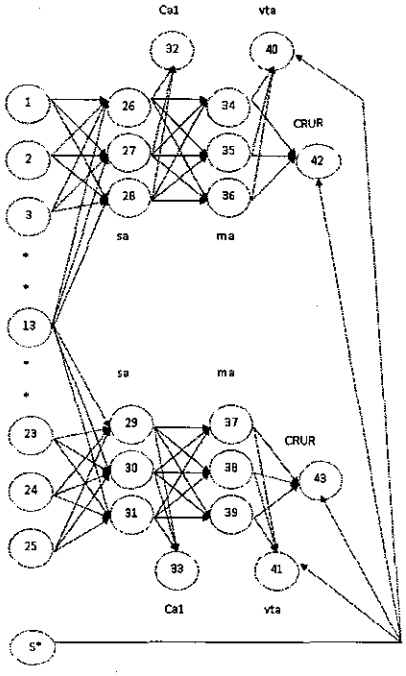
Figura 78. Arquitecturas empleadas con la unidad "CR/UR" como elemento integrador de las subredes.



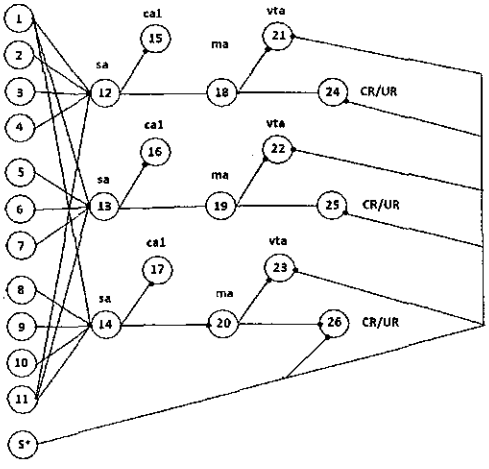
Red usada en la simulación 16



Red usada en la simulación 17



Red usada en la simulación 26



Red usada en la simulación 27

Figura 79. Arquitecturas empleadas que no usaron a la unidad "CR/UR" como elemento integrador de las subredes.

Todas las arquitecturas de este tipo presentaron resultados efectivos aprendiendo a responder ante dos o más EC, aún sin necesidad de pasar por un proceso de extinción previo al entrenamiento ante el segundo o tercer EC.

Del conjunto de simulaciones realizadas en este capítulo se desprende que la estrategia consistente en el empleo de subredes para el manejo de varios estímulos en una red bajo el Modelo DBP es efectiva para manejar varios EC a la vez, bajo las siguientes restricciones:

1) La red general deberá integrar tantas subredes como estímulos condicionados deba manejar al mismo tiempo.

   Este hecho podría hacer que la estrategia pudiese ser vista como "artificiosa", en virtud de que tendría que establecerse una red especial para cada problema que se quiera manejar, dependiendo de cuántos estímulos por aprender sean planteados en el mismo. Sin embargo, la red marcada en la Figura 79 como "d)", usada en la simulación 27, emplea la versión más simple del circuito neural posible para cada subred, demostrando ser efectivo para el manejo de varios EC simultáneos. Este circuito básico (integrado por 1 unidad "sa", 1 unidad "ca1", 1 unidad "ma", 1 unidad "vta" y 1 unidad "CR/UR") podría verse como una "unidad" a emplearse en cualquier arquitectura que pretendiera manejar varios estímulos, con lo que la percepción de "artificio" podría verse reducida.

2) Cada sub red deberá incluir al menos una unidad de tipo "CR/UR" en donde se registrará exclusivamente la respuesta de la red ante el estímulo determinado para la subred correspondiente.

3) Los cálculos de las discrepancias de las unidades "ca1" y "vta" entre su valor actual y el valor en el momento temporal anterior, deberán ser realizados con base en la totalidad de las unidades de estos tipos. Asimismo, el reajuste de pesos de la red se debe realizar con base en los cálculos anteriores, de acuerdo con la regla de aprendizaje definida para el modelo.

4) En las redes empleadas se usan unidades de entrada que mantienen a la red unida a nivel arquitectónico entre la capa de entrada y las unidades de tipo "sa". Estas unidades no participan plenamente en los estímulos con una activación mayor a 0.0, por lo que puede prescindirse de ellas. En este capítulo se emplearon solamente como una estrategia más para la unión de las subredes en un sistema integrado.

Finalmente, es necesario remarcar que estos dos tipos de arquitecturas permiten el aprendizaje efectivo de dos o más EC sin haber sido necesaria una modificación al modelo en sí, sino solamente a la arquitectura que usualmente se empleaba. No se exploró el uso de conexiones inhibitorias en estas arquitecturas, por lo que se vislumbra la posibilidad de que no sean las únicas que permiten el manejo de varios EC. Tampoco se exploraron otras modificaciones a la arquitectura, como la inclusión de más capas de unidades "sa" y "ma", o la dotación de más EI. De igual manera, se debe especificar que las simulaciones fueron hechas en computadoras secuenciales, implicando una inversión de recursos de procesamiento relativamente baja. En estas simulaciones, debido a la plataforma computacional empleada, no se realizó en ningún momento, algún tipo de procesamiento en paralelo, o por lo menos concurrente a nivel de los procesos implicados en la operación de la red.

Considerar la futura exploración de estas posibilidades, hace que los resultados de las simulaciones realizadas en este capítulo sean solamente un paso inicial hacia nuevas simulaciones que contemplen redes con arquitecturas más complejas.

8. SIMULACION DE LA INHIBICIÓN CONDICIONADA CON EL MODELO DBP

Las simulaciones realizadas en el presente capítulo pretendieron simular la inhibición condicionada, con base en el uso de la preparación clásica reportada por Pavlov (1927), consistente en el uso de un esquema de tres condiciones: B+|A+/AC-|AB. Asimismo, para poder constatar que el estímulo condicionado, en este caso el estímulo "A" es efectivamente un inhibidor, se planteó el uso de las dos pruebas establecidas por Rescorla (1969): la prueba de sumación y la prueba de retardo.

La inhibición condicionada es un fenómeno que implica, de acuerdo con la definición del esquema de entrenamiento ya referido, la intervención de 3 estímulos diferentes, por lo que se requería de una red que pudiese manejarlos adecuadamente. No habría sido posible intentar esta simulación si no se hubiesen obtenido resultados efectivos en las simulaciones realizadas en el capítulo anterior, para encontrar una arquitectura que permitiera manejar varios EC al mismo tiempo. Por ello, las simulaciones en este capítulo se basan en la arquitectura que demostró ser la más efectiva de todas las que fueron revisadas: basada en el uso de una subred para el manejo de cada estímulo presentado a la red, con una unidad "CR/UR" en la que se espera la respuesta correspondiente a cada estímulo, y con cálculo global de discrepancias para la realización del reajuste de los pesos de la red.

8.1 SIMULACIÓN 28

Para tratar de simular este fenómeno, todas las corridas se hicieron considerando 300 ensayos compuestos cada uno, de 8 momentos temporales. Se registró la actividad de las unidades "CR/UR" en el momento temporal 7 y cuando el refuerzo fue administrado, se hizo en el momento temporal 8, a través de la unidad S*, la que se activó con un valor de 1.0

8.1.1 ESTRUCTURA USADA PARA LA RED

Para esta simulación se empleó la misma arquitectura y configuración de la red empleada en la simulación anterior.

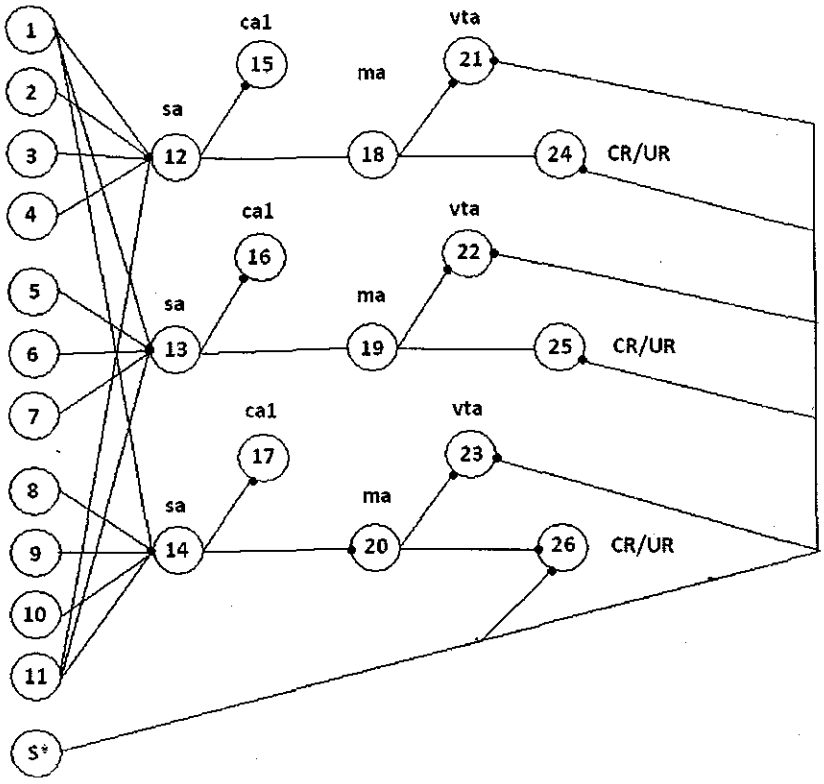


Figura 80. Arquitectura empleada en la simulación 28

8.1.2 PATRONES DE ENTRADA USADOS EN LA SIMULACIÓN

En esta simulación se sometió la red a tres condiciones distintas. En la primera condición se reforzó el estímulo B, mientras que en la segunda condición se presentaron dos estímulos alternados, el estímulo C reforzado, y el estímulo CA sin refuerzo, (este estímulo CA es la unión de los estímulos C y A en un solo patrón de entrada simultáneo), en la tercera condición, se presentó el estímulo AB reforzado.

El esquema de las condiciones en la simulación fue el siguiente: B+|C+/CA-|AB+.

De acuerdo con la preparación original para la inhibición condicionada, establecida por Pavlov (1927) se espera que el estímulo A se convierta en un inhibidor condicionado, y que al presentarlo en la tercer condición simultáneamente al estímulo B, lo inhiba (es decir, se esperaría una disminución en el nivel de activación de la unidad "CR/UR" correspondiente a la respuesta ante B)

Se realizaron 300 ensayos para cada condición y no se empleó re-inicialización de pesos entre cada una de las condiciones.

A las unidades de entrada que no recibían activación con un valor de 1.0, les fue asignado un valor de 0.0.

Para esta simulación, se emplearon los patrones de entrada que se presentan en las siguientes tablas.

Ent	B+	C+	CA-	AB+
1	0	0	0	0
2	0	0	1	1
3	0	0	0	0
4	0	0	0	0
5	1	0	0	1
6	0	0	0	0
7	0	0	0	0
8	0	1	1	0
9	0	0	0	0
10	0	0	0	0
11	0	0	0	0
S*	1	1	0	1

Tabla 72. Patrones de entrada por estímulo empleados en la simulación 28

8.1.3 RESULTADOS OBTENIDOS

En la Figura 81 se presentan los resultados de la simulación de la inhibición condicionada:

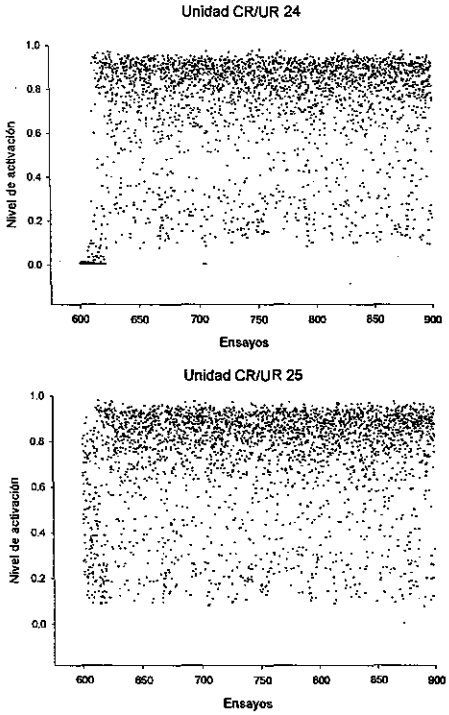


Figura 81. Activaciones de las unidades 24 y 25 ("CR/UR"), durante la tercera condición de la simulación bajo las condiciones AB+|AB-|AB+.

Durante los primeros ensayos de la tercera condición se observan niveles altos en las activaciones de las unidades "CR/UR" números 24 y 25, sin embargo, se presentan diferencias en estos niveles. Mientras que en la unidad "CR/UR" número 24, es decir, la unidad en la que se esperaba la respuesta ante el estímulo "A", se observa un inicio con niveles de activación muy cercanos a cero, para después sufrir un incremento que llega hasta niveles superiores al 0.8. Por otra parte, la unidad "CR/UR" 25, es decir, la unidad en la que se espera la respuesta ante el estímulo "B", mantiene niveles de activación cercanos al 0.6 hasta alcanzar niveles muy similares a los que presenta la unidad 24.

En la Figura 82 se presentan, los niveles de activación presentados por las unidades 24 y 25, durante las 10 replicaciones de la misma simulación y en la Figura 83 se muestra la activación promedio durante los primeros 50 ensayos.

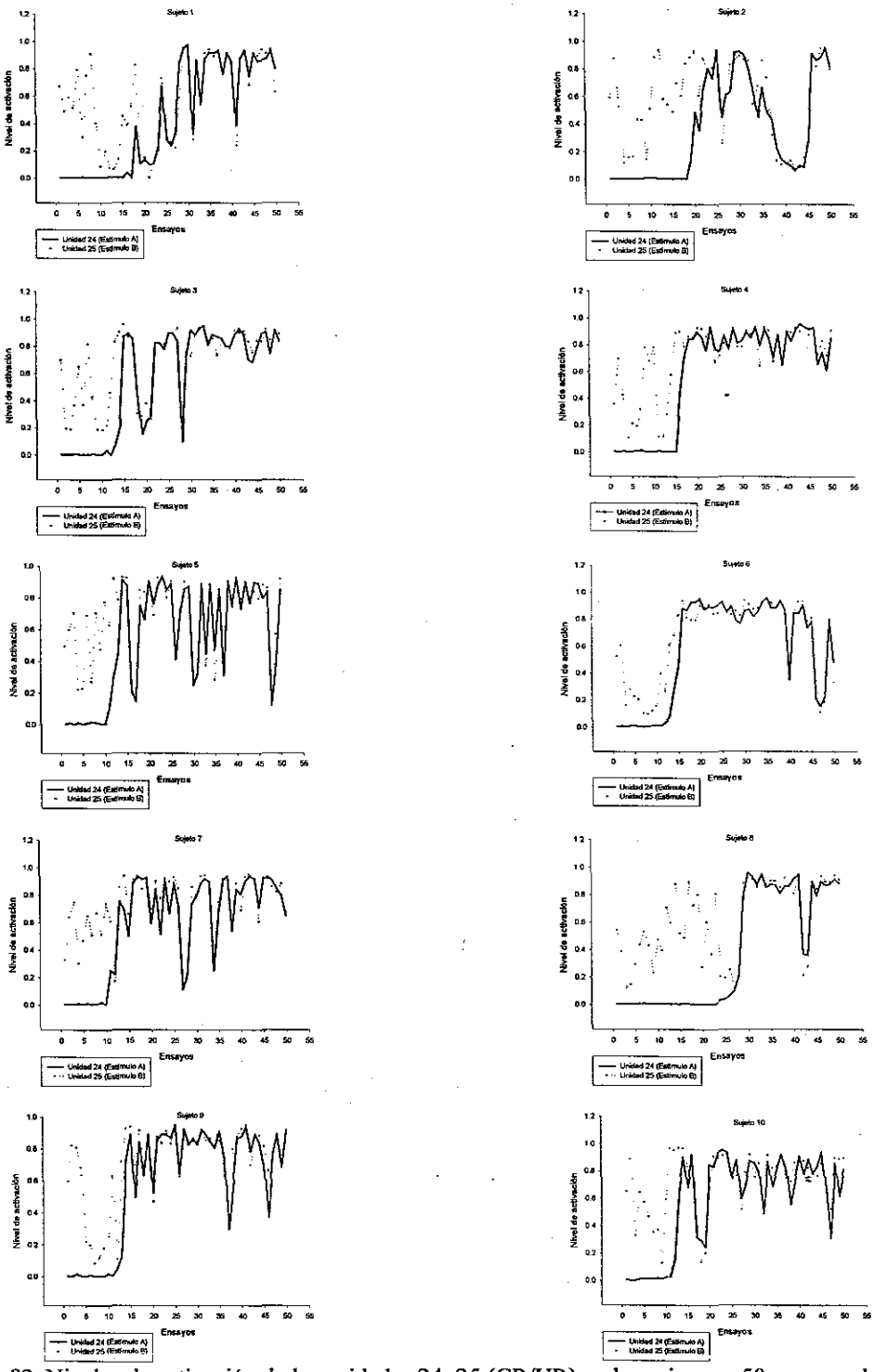


Figura 82. Niveles de activación de las unidades 24, 25 (CR/UR) en los primeros 50 ensayos de la tercera condición (AB+) de las 10 replicaciones de la simulación 18 (B+ | C+/CA- | AB+).

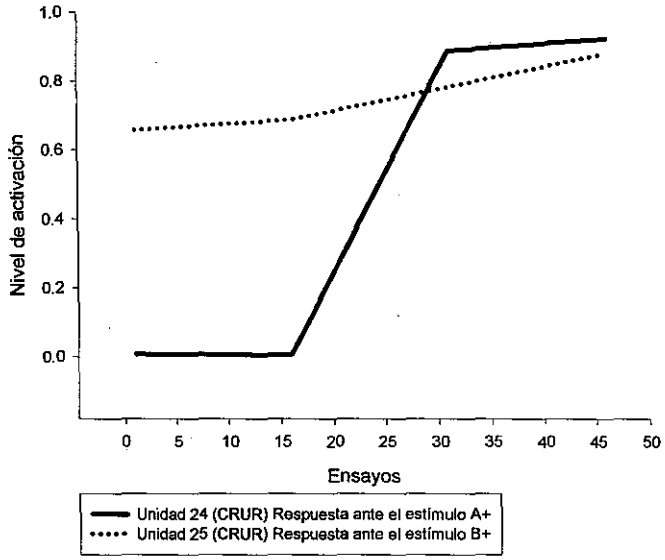


Figura 83. Niveles de activación promedio de las unidades 24 y 25 (CR/UR), (en rangos de 15 ensayos) en las primeros 50 ensayos de la tercera condición de la simulación 28.

Las Figuras 82 y 83 muestran que, mientras que la activación de la unidad 24, ante el estímulo "A" permanece en niveles bajos, la activación en la unidad 25, ante el estímulo "B" permanece en niveles inferiores a los que presenta, cuando la activación en la unidad 24, se incrementa a valores cercanos al 1.0. De la misma manera, en las Figuras anteriores se puede observar que en la unidad 24, es decir aquella en la que se espera la respuesta ante el estímulo "A", presenta niveles de activación cercanos a cero durante los primeros ensayos de esta condición, para luego incrementarse hasta llegar a niveles cercanos a 1.0. Este incremento se da de manera similar al que se observa regularmente en la adquisición o la readquisición de un estímulo, como las observadas en las simulaciones 1 y 2 en este mismo capítulo. Sin embargo, esta elevación del nivel de activación se da en un número menor de ensayos que el que generalmente se ha empleado en las simulaciones anteriormente realizadas.

Los resultados en esta tercera condición (AB+) podrían explicarse de la manera expuesta a continuación:

8.2 ACERCA DE LAS DIFERENCIAS ENCONTRADAS EN LOS NIVELES DE ACTIVACIÓN

Mientras el nivel de activación en la unidad "CR/UR" número 24, permanece en valores cercanos a 0, podría tener un efecto sobre el nivel de activación que se observa en la unidad "CR/UR" número 25, unidad en la que se espera la respuesta ante el estímulo B. Si el estímulo "A" se ha convertido efectivamente en un inhibidor condicionado, entonces estaríamos observando el fenómeno de sumación del que hablaba Rescorla (1969), observando niveles de activación menores a los que presentaría esta unidad ante un estímulo que ha sido previamente expuesto a presentaciones reforzadas y en las cuales se ha obtenido ya un nivel de activación elevado.

Sin embargo, para verificar que efectivamente se habla de una sumación, en la que el estímulo inhibidor supuesto, es decir el estímulo "A" tiene injerencia, es necesario comparar los niveles

presentados durante la parte inicial de esta tercera condición con aquellos niveles presentados por la misma unidad 25 durante la primera condición, en donde se presentó el estímulo B reforzado (B+). Estos niveles de activación se presentan en la Figura 84, en la que se pueden observar tres niveles distintos de activación de esta unidad: el más bajo de los niveles se presenta durante la primera condición, mientras que en la tercera condición se identifican los dos tipos de niveles ya descritos.

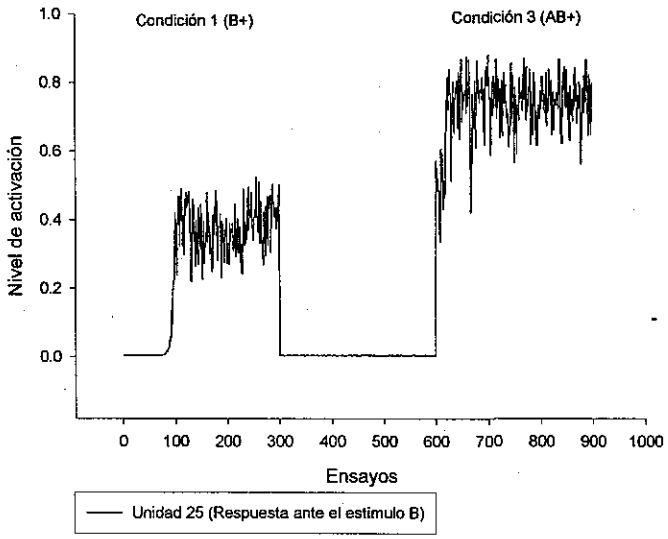


Figura 84. Niveles de activación promedio de la unidad 25 (Respuesta ante B), durante las tres condiciones de la simulación 28.

Se realizó un conjunto de simulaciones adicionales para averiguar si los niveles de activación mostrados en la parte inicial de la condición 3 corresponden a la activación inicial mostrada por esta misma unidad 25 durante la condición 1.

8.2.1 SIMULACIÓN 29

En esta simulación, se buscó incrementar el nivel de activación ante el estímulo "B" reforzado durante la primera condición. Para ello, se emplearon 10 redes exactamente iguales a las empleadas en la simulación original, pero con las condiciones siguientes: AB+|AB-|AB+. El resultado se muestra en la Figura 85, en la que se muestran los niveles de activación obtenidos durante la tercera condición en las 10 replicaciones:

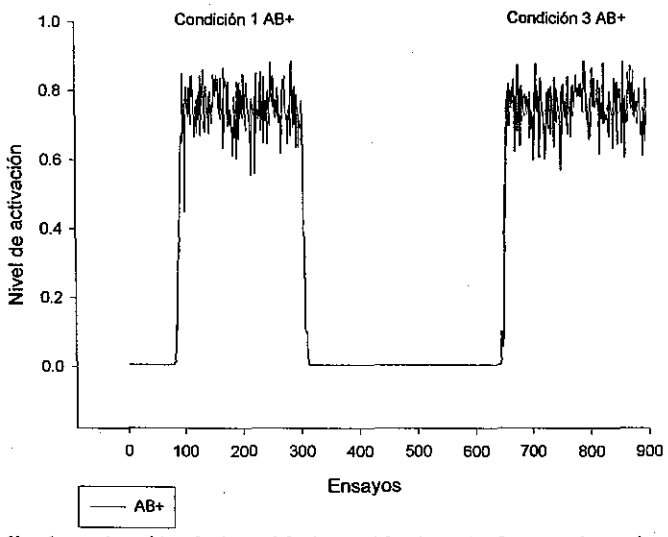


Figura 85. Nivel medio de activación de la unidad 25 ("CR/UR"), durante las primera y tercera condiciones de la simulación bajo las condiciones AB+|AB-|AB+.

Como puede observarse en la Figura 85, los niveles medios de activación de la unidad 25 (CR/UR) son muy similares después de la inclusión de un componente más en el estímulo inicial, con la consecuente activación de un segundo circuito de la red.

8.2.2 SIMULACIÓN 30

Una vez establecido, que el nivel de activación durante la primera condición se puede elevar mediante un estímulo que activara dos subredes, entonces se corrieron 10 simulaciones más, usando exactamente la misma configuración de red, pero ahora bajo el siguiente esquema de condiciones: AB+| C+/CA- | AB+. Los niveles de activación promedio para la unidad 25, en donde se espera la respuesta ante el estímulo "B", se presentan a continuación.

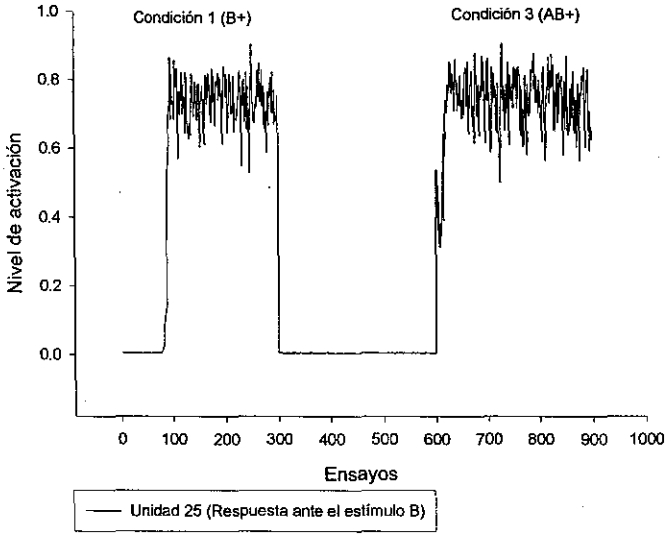


Figura 86. Niveles de activación promedio de la unidad 25 (Respuesta ante B), durante las tres condiciones AB+ | C+/CA- | AB+.

En la Figura 86, se pueden observar ahora dos niveles distintos de activación, el nivel de activación alcanzado en la primera condición y presentado en la mayor parte de la tercera condición, y un nivel más bajo de activación presentado durante los ensayos iniciales de la tercera condición.

Ello podría sugerir que el nivel bajo de activación que presenta la unidad "CR/UR" número 25 se presenta tanto con niveles de activación bajos y altos, alcanzados por la propia unidad, durante la primer condición. Sin embargo, ahora habría que buscar si este nivel de activación bajo inicial, pero no cercano a cero, se presenta ante otro tipo de condiciones. Para ello se corrieron simulaciones adicionales, bajo condiciones diferentes, en las que pudiese observarse un incremento de la activación de esta unidad en la tercera condición. No se probaron todas las posibilidades (2744), sino que se eligieron aquellas consideradas como ilustrativas del posible comportamiento de la activación de la unidad "CR/UR" en la tercera condición. Se consideraron así, los siguientes conjuntos de arreglos:

1) Arreglos en donde el estímulo B, sólo o apareado con otro estímulo, es reforzado en la primera condición, se presenta sólo, sin refuerzo en la segunda condición y es reforzado finalmente en la tercera condición, solo o apareado con otro estímulo:

   a) B+|B-|B+

   b) AB+|B-|AB+

   c) ABC+|B-|ABC+

2) Arreglos en donde el estímulo B, sólo o apareado con otro estímulo, es reforzado en la primera condición, se presenta en conjunto con otro estímulo, sin refuerzo en la segunda condición y es reforzado finalmente en la tercera condición, solo o apareado con otro estímulo:

   d) AB+|AB-|AB+

   e) ABC+|ABC-|ABC+

3) Arreglos en los que en la primera condición se refuerza el estímulo B sólo o apareado con otro estímulo; en la segunda condición no participa y en la tercera condición vuelve a presentarse sólo o apareado con otro estímulo:

   f) B+|A-|B+

   g) AB+|A-|AB+

   h) ABC+|A-|ABC+

   i) ABC+|AC-|ABC+

En todos los casos se corrieron 10 replicaciones bajo las condiciones especificadas.

A continuación se presentan los resultados de estas simulaciones bajo las condiciones descritas.

8.2.3 SIMULACIÓN 31

En esta simulación se probó el arreglo B+|B-|B+. Los resultados se muestran a en la Figura 87:

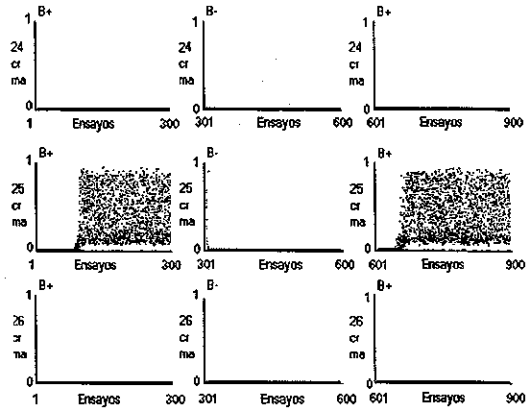


Figura 87. Niveles de activación de todas las replicaciones, durante las tres condiciones de la simulación B+ | B- | B+.

Los niveles de activación presentados por la unidad "CR/UR" durante la condición 3, bajo este conjunto de condiciones, no es similar al encontrado en la simulación de la inhibición condicionada, ni en el número de ensayos empleados, ni en los niveles de activación logrados. Por lo que podría suponerse que lo observado en la simulación de la inhibición condicionada no se trata de una readquisición.

8.2.4 SIMULACIÓN 32

En esta simulación se probó el arreglo AB+|B-|AB+

Los resultados se muestran a en la Figura 88:

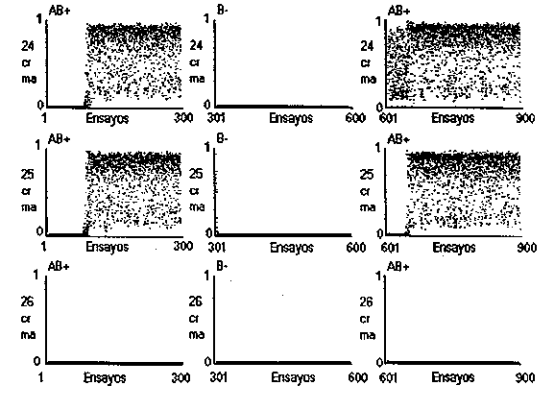


Figura 88. Niveles de activación de todas las replicaciones, durante las tres condiciones de la simulación AB+ | B- | AB+.

En este caso, la Figura 88 muestra niveles de activación en la unidad 24 "CR/UR" que son similares a los que se presentan en la simulación de la inhibición condicionada, en la unidad "CR/UR" 25, y al mismo tiempo, se observa un nivel menor de activación, en tanto la unidad 25 no alcanza niveles altos de activación. Sin embargo, el número de ensayos que dura este comportamiento es mayor que en el observado en la simulación de la inhibición condicionada. Este caso se retomará un poco más adelante en este mismo capítulo.

8.2.5 SIMULACIÓN 33

En esta simulación se probó el arreglo ABC+|B-|ABC+

Los resultados se muestran a en la Figura 89:

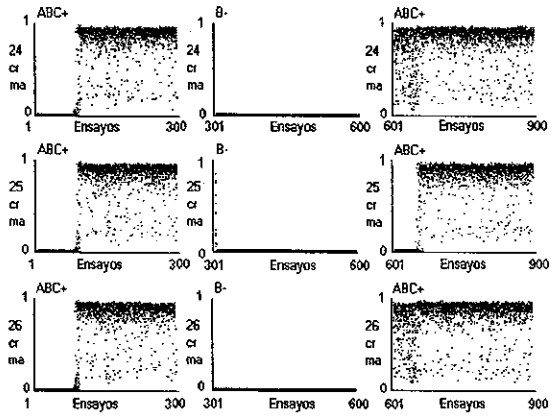


Figura 89. Niveles de activación de todas las replicaciones, durante las tres condiciones de la simulación ABC+ | B- | ABC+.

En la Figura 89, se observa el mismo fenómeno que en la simulación b), solo que con dos de las unidades "CR/UR", las correspondientes a la respuesta ante el estímulo "A" (unidad 24) y al estímulo "C" (unidad 25). Asimismo, el número de ensayos durante el cual se observa este comportamiento requerido, es similar al de la simulación b), pero mucho mayor al presentado en la simulación de la inhibición condicionada. Este caso también será retomado un poco más adelante en este mismo documento.

8.2.6 SIMULACIÓN 34

En esta simulación se probó el arreglo AB+|AB-|AB+. Los resultados se muestran a en la Figura 90:

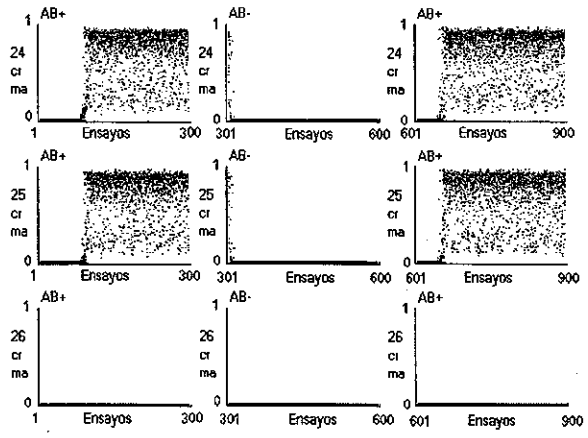


Figura 90. Niveles de activación de todas las replicaciones, durante las tres condiciones de la simulación AB+ | AB- | AB+.

En la Figura 90, los niveles de activación presentados por las unidades 24 y 25 ("CR/UR") durante la condición 3, bajo este conjunto de condiciones, no es similar al encontrado en la simulación de la

inhibición condicionada, ni en el número de ensayos empleados, ni en los niveles de activación logrados. De nuevo, podría sugerirse que lo observado en la simulación de la inhibición condicionada, no se trata de una readquisición.

8.2.7 SIMULACIÓN 35

En esta simulación se probó el arreglo ABC+|ABC-|ABC+. Los resultados se muestran a en la Figura 91:

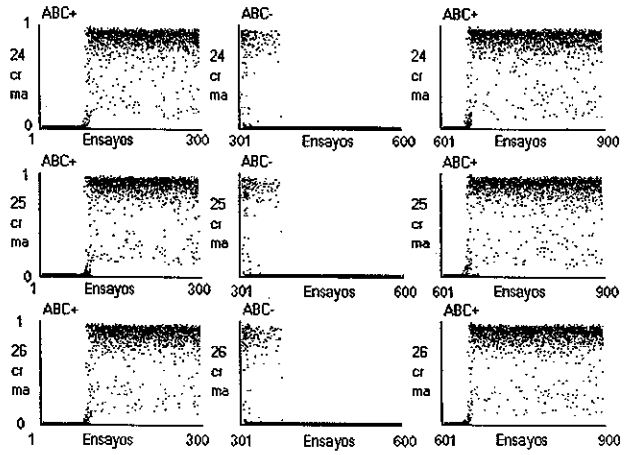


Figura 91. Niveles de activación de todas las replicaciones, durante las tres condiciones de la simulación ABC+ | ABC- | ABC+.

De la misma manera, en la Figura 91, se presentan resultados muy similares a los observados en las simulaciones anteriores en las que interviene la readquisición.

8.2.8 SIMULACIÓN 36

En esta simulación de probó el arreglo B+|A-|B+. Los resultados se muestran a en la Figura 92:

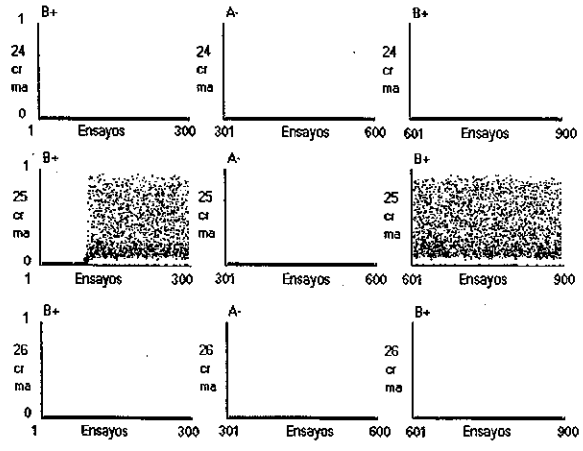


Figura 92. Niveles de activación de todas las replicaciones, durante las tres condiciones de la simulación B+ | A- | B+.

En este caso, no hubo extinción del estímulo "B" entre la condición 1 y la 3, y el estímulo "B" no interviene en la segunda condición. No se observan en la Figura 92, incrementos notables en el nivel de activación de la unidad "CR/UR" número 25, sino que los niveles alcanzados en la condición 1 se sostienen en la condición 3. El comportamiento en estos niveles de activación puede sugerir que las activaciones observadas en la simulación de la inhibición condicionada, pueden no deberse a un caso de sostenimiento de un nivel de activación previamente alcanzado, ya que en caso de ser así debería ser similar y no superior a los previamente alcanzados.

8.2.9 SIMULACIONES 37, 38 Y 39

En este conjunto de simulaciones se probaron los arreglos AB+|A-|AB+, ABC+|A-|ABC+ y ABC+|AC-|ABC+.

Los resultados se muestran a en las Figuras 93, 94 y 95, respectivamente:

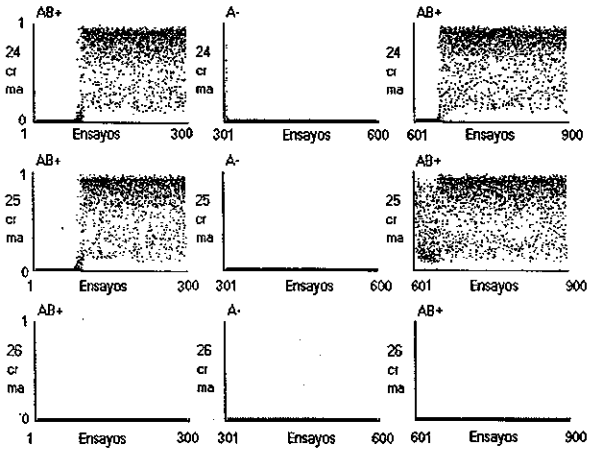


Figura 93. Niveles de activación de todas las replicaciones, durante las tres condiciones de la simulación AB+ | A- | AB+.

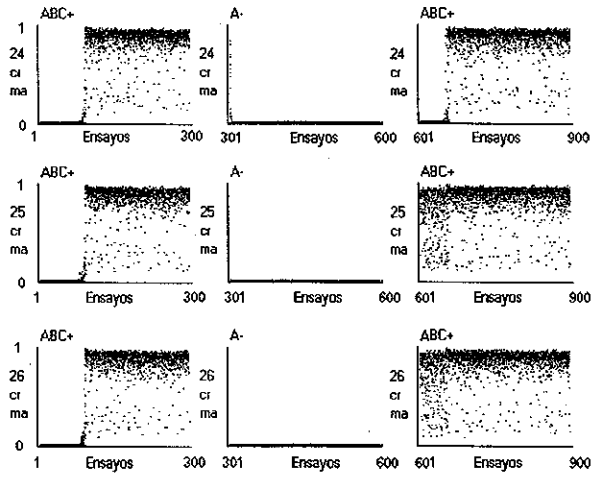


Figura 94. Niveles de activación de todas las replicaciones, durante las tres condiciones de la simulación ABC+ | A- | ABC+.

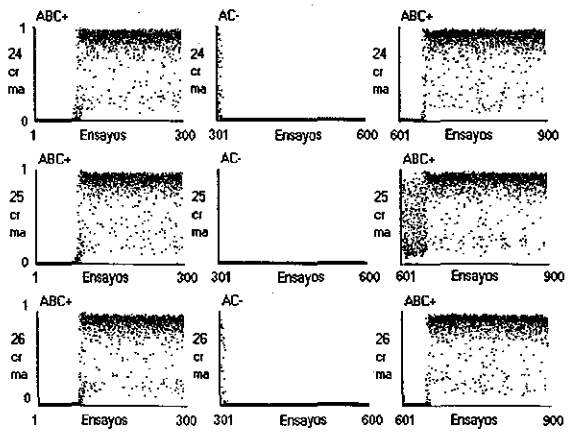


Figura 95. Niveles de activación de todas las replicaciones, durante las tres condiciones de la simulación ABC+ | AC- | ABC+.

En los tres casos anteriores se observa un comportamiento similar a los obtenidos en la simulación original de la inhibición condicionada, sin embargo, el número de ensayos durante el cual puede observarse es mayor. Asimismo, se observa que en los casos en los que se ven involucrados más de dos estímulos con este tipo de comportamiento, en la tercera condición, el nivel de activación es mayor al presentado cuando solo se trata de un estímulo involucrado.

En los arreglos AB+|B-|AB+, ABC+|B-|ABC+, AB+|A-|AB+, ABC+|A-|ABC+ y ABC+|AC-|ABC+ se encontraron resultados similares a los encontrados en la simulación de la inhibición condicionada, en lo referente a la existencia de dos niveles de activación en la condición 3 y la permanencia del primero de los dos niveles observados, durante los ensayos durante los cuales el estímulo que era presentado en la condición inmediata anterior sin refuerzo, lograba que la unidad correspondiente alcanzara niveles altos de activación.. Es decir, se observó que un comportamiento similar al observado en las activaciones de la condición 3 de la simulación de la inhibición condicionada, se presentó en simulaciones en las que un subcomponente de un estímulo compuesto, pero no todo, era presentado sin refuerzo en la condición anterior previa, es decir, en la tercera condición; y este mismo componente o componentes eran reforzados en la tercera condición en apareo con otros. Estos resultados podrían ser considerados como simulaciones de la inhibición condicionada resultante de la extinción previa del estímulo condicionado de acuerdo con Savastano y otros (1999).

En los casos en los que se presentó el estímulo "B" sin refuerzo en la segunda condición, este mismo estímulo fue reforzado en la tercera condición, apareado con el estímulo "A" o los estímulos "A" y "C". En los casos en los que se presentó el estímulo "A" sin refuerzo en la segunda condición, este mismo estímulo era presentado con refuerzo en la tercera condición, apareado con los estímulos "B", o "B" y "C". Asimismo, en el caso en el que se presentaron apareados los estímulos "A" y "C" sin refuerzo en la segunda condición, durante la tercera condición fueron presentados con refuerzo, apareados con el estímulo "B".

En todos los casos, el o los estímulos que no participaron en la segunda condición presentaron, durante la tercera, niveles medios de activación durante los primeros ensayos de la tercera condición, para alcanzar niveles altos después. Asimismo, en todos los casos, el o los estímulos que habían participado en la segunda condición, presentaron un comportamiento de las activaciones en sus unidades correspondientes, similares a los que se observan durante la readquisición simulada al inicio

de este mismo capítulo. Este es un comportamiento similar al observado en la simulación de la inhibición condicionada, excepto porque permanece en esta última durante menos ensayos.

Cabe otra posibilidad aún para buscar explicar el hecho de que en la simulación de la inhibición condicionada, se observe durante un número menor de ensayos al inicio de la condición 3, y es el hecho de que el estímulo que se presenta sin refuerzo, el "AC", tiene una presentación durante menos ensayos (150) ya que se presenta alternadamente con una parte de este mismo estímulo, el estímulo "C" que sí se presenta reforzado.

8.2.10 SIMULACIONES 40, 41, 42, 43 Y 45

Para indagar acerca del efecto que podría tener sobre el número de ensayos durante los que se presenta este comportamiento reportado, en la tercera condición de la simulación de la inhibición condicionada, se realizaron 5 simulaciones más: AB+|A-|AB+, con 300 ensayos en la condición 1 y 3, y con 50, 150, 600, 2000 y 5000 ensayos, en la condición 2.

. Los resultados de estas dos simulaciones se presentan en la Figura 96.

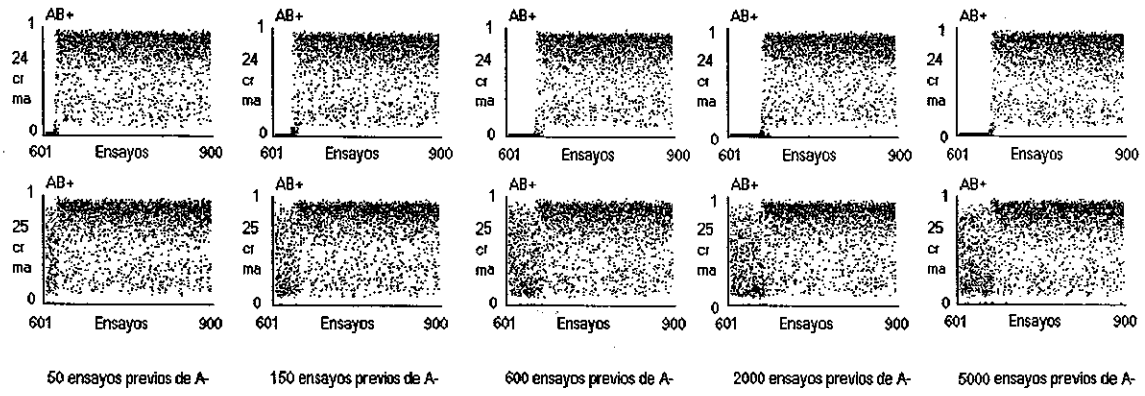


Figura 96. Niveles de activación de las unidades 24 y 25 ("CR/UR") en todas las replicaciones, durante tercera condición de la simulación AB+ | A- | AB+.

El número de ensayos empleados en la condición 2 influye sobre la duración del comportamiento observado en las activaciones, durante la tercera condición, solamente hasta los 600 de ahí en adelante parece no haber diferencia alguna.

8.2.11 SIMULACIONES 46, 47, 48 Y 49

Con base en estos resultados, se realizó una nueva simulación de la inhibición condicionada, variando el número de ensayos durante la segunda condición para ver si se obtenían resultados similares. En la Figura 97, en el inciso a) se muestran los resultados de las simulaciones que en la segunda condición (C+/CA-) emplearon números distintos de ensayos, pero en la misma proporción: 75/75, 150/150 y 300/300; en el inciso b) de la misma Figura 97 se muestra el resultado al cambiar la proporción de los componentes del compuesto a 75/300.

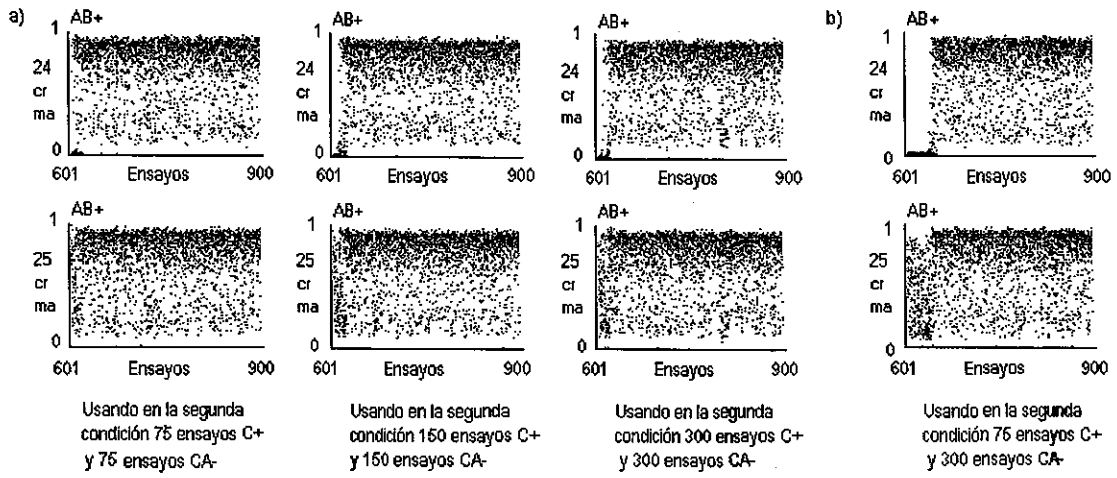


Figura 97. Niveles de activación de las unidades 24 y 25 en todas las replicaciones, durante tercera condición de la simulación B+ |C+/CA- | AB+, usando números de ensayos diferentes en la segunda condición.

Como puede observarse en la Figura 97, reducir de 150 a 75 el número de ensayos C+/CA- en la misma proporción, acorta el número de ensayos durante los cuales se presenta el comportamiento reportado en las activaciones de las unidades "CR/UR", pero no hay diferencia entre 150 y 300 si se guarda la proporción de uno a uno. En cambio, usar una proporción 75/300, ocasiona que sea mayor el número de ensayos en los que se presenta el comportamiento de activaciones reportado.

8.3 ACERCA DE LA POSIBLE EXISTENCIA DE UN RETARDO OCASIONADO POR EL INHIBIDOR PUTATIVO

Apartándose de los niveles distintos de activación observados y centrándose ahora en el número de ensayos que se observa este comportamiento, y con base en las 20 simulaciones adicionales realizadas a propósito del intento por simular la inhibición condicionada, no es posible afirmar si se observa un retardo en el periodo observado, durante el cual, los niveles de activación en la unidad o las unidades en donde se espera la respuesta para el o los componentes del estímulo que sí participan en la segunda condición (para el caso de la simulación de la inhibición condicionada A+, o la unidad 24).

Si se asume que efectivamente el estímulo "A" es un candidato viable para ser considerado como un inhibidor condicionado, se observa derivado de los mismos resultados presentados anteriormente, que la adquisición de la respuesta se presenta en un número de ensayos, menor a los que son requeridos para la adquisición de una respuesta sin haberse sometido a las condiciones anteriores. Bajo esta perspectiva no se da un retardo, que es la segunda condición establecida por Rescorla (1969) para establecer la existencia de un inhibidor condicionado. Sin embargo, la variación en la proporción de ensayos en la presentación del compuesto C+/CA- está muy cercana a alcanzar el número de ensayos requeridos para el caso de la readquisición, como el observado en la simulación g) en este mismo apartado de la simulación 18. La Figura 98 muestra las activaciones de la unidad 24, en la tercer condición de la simulación g) (ABC+|ABC-|ABC+), y las activaciones de la misma unidad 24 en la tercer condición de la simulación de la inhibición condicionada con una proporción 75/300 en la segunda condición.

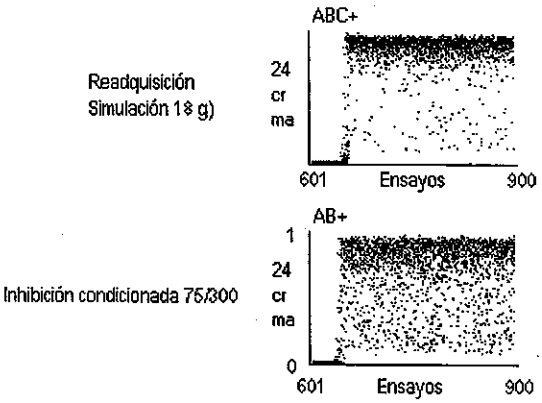


Figura 98. Niveles de activación de las unidades 24 y 25 en todas las replicaciones, durante tercera condición de la simulación ABC+ |ABC- | ABC+ y la simulación B+|C+/CA-|AB+ con una proporción de 75/300 en la segunda condición.

9. CONCLUSIONES

El trabajo realizado puede ser resumido de la manera siguiente:

- Primero, se diseñó y produjo una implementación computacional del Modelo DBP que fuese lo suficientemente flexible como para permitir la realización de simulaciones, con base en un conjunto de datos de entrada y de acuerdo con las funciones establecidas en las especificaciones del modelo, pero con parámetros cuyos valores son proporcionados por el usuario. Esta implementación debía ofrecer al usuario, resultados confiables.

- Segundo, se debía comprobar que la arquitectura clásica de red, empleada por el Modelo DBP, no permite que se aprenda a responder ante dos o más EC. Para lo cual se tuvo que probar que la imposibilidad para el manejo de varios estímulos, no depende, en esta arquitectura clásica, del tamaño de la red empleada; ni del número de unidades de entrada provistas a la red; ni de los valores de entrada definidos para cada estímulo procesado en la red; ni del esquema de entrenamiento elegido para la presentación de los estímulos.

- Tercero, se debía buscar una arquitectura de red para el Modelo DBP, que permitiese manejar varios EC al mismo tiempo.

- Cuarto y último, una vez encontrada la nueva arquitectura, se debía, con base en ella, realizar una simulación del fenómeno de inhibición condicionada, reportado por Pavlov (1927) y para el cual, Rescorla (1969) estableció dos pruebas, sumación y retardo, que debían ser aplicadas para verificar si a partir de los resultados presentados se podía hablar de la existencia de un inhibidor condicionado en la simulación realizada.

A continuación se presentan las conclusiones derivadas de los resultados en la realización de estas acciones enlistadas.

9.1 ACERCA DE LA IMPLEMENTACIÓN COMPUTACIONAL DEL MODELO DBP

En la actualidad son muy pocos los simuladores existentes para crear y ejecutar redes neurales bajo el Modelo DBP, e igualmente escasas son las herramientas de implementación de redes neurales disponibles en el mercado que sean lo suficientemente flexibles como para poder ser usadas en la simulación de una red bajo el Modelo DBP. Es por ello que se optó por diseñar una aplicación propia para hacer simulaciones bajo el modelo DBP.

A diferencia del primer y más amigable de los simuladores creados para el Modelo DBP, el "Selnet", que fue programado en Borland Delphi, por el Dr. Burgos Triano, el simulador propio fue programado en Microsoft Visual C# (Sharp) para ambiente Windows XP.

En el simulador creado para este trabajo, no se incluyeron las facilidades con las que cuenta el simulador original "Selnet" para la configuración de una red neural y de las simulaciones a correr, sino que se tienen que establecer estas configuraciones a partir de su especificación en el propio código fuente. Lo anterior fue hecho con la finalidad de obtener los mismos resultados, pero con mayor

rapidez, ahorrando tiempo en la programación de estos módulos que habrían hecho el simulador más amigable. Una vez que el núcleo del simulador propio ha sido ya definido, el trabajo requerido a futuro, implicaría el diseño de una interfaz amigable para hacer más cómodo el trabajo con éste.

En el simulador realizado en este trabajo se emplean en general, estrategias de cálculo similares a las empleadas en "Selnet", sin embargo, existieron algunas diferencias:

a) La generación de números aleatorios. El simulador propio empleó una estrategia basada en la creación de una lista secuencial con números enteros, a partir de la cual se hace una selección aleatoria; los números seleccionados se van eliminando de la lista secuencial y ésta se va compactando. "Selnet", en cambio, emplea un mecanismo basado en la generación aleatoria de números enteros, los cuales son metidos en una pila; la distinción entre números generados y números no generados aún, se realiza con base en el uso de "banderas". No existen diferencias en cuanto al resultado final, pero sí se encontró que la estrategia empleada en el simulador propio posibilita un considerable ahorro en términos de recursos de cómputo.

b) La generación de números aleatorios gaussianos. El simulador realizado emplea un método de generación diferente al que emplea "Selnet", con lo que se prueba que las simulaciones son independientes del método empleado para ello.

Una vez realizado el simulador, se sometió a una simulación de prueba, comparándose sus resultados con aquellos arrojados por "Selnet" en una simulación ante las mismas condiciones, sin que se observaran diferencias importantes entre ellos.

Esta parte del trabajo se reviste especial importancia, debido a que son pocos los intentos exitosos por realizar un simulador del Modelo DBP y en este reporte se ofrece, no solamente el código fuente de un simulador que demostró ser confiable, sino también, una especificación en pseudocódigo, de las funciones principales que estructuran la implementación del Modelo. La bibliografía especializada en el Modelo ofrece las ecuaciones básicas a partir de las cuales funciona el modelo, pero no existe una referencia que incluya aspectos prácticos para su implementación.

En el proceso de diseño del simulador no se hizo un estudio algorítmico que asegurara que las funciones implementadas fueran las más económicas en términos de procesamiento, o en términos de complejidad. El código fuente que se ofrece en este documento es entonces, efectivo, pero dista mucho de ser el mejor o el más adecuado y las estructuras de datos empleadas no son las más económicas, ni se ajustan a límites reales en la dimensión de los datos a procesar. Aún así, la ejecución se realiza en un tiempo relativamente rápido y el nivel de procesamiento requerido puede ser alcanzado sin problemas por una computadora de tipo personal, con características de vigencia en el mercado actual.

Se deja así, para trabajos posteriores la refinación del código para volverlo más simple y eficiente, así como la inclusión de mejores características de compatibilidad, portabilidad y usabilidad.

9.2 ACERCA DE LAS SIMULACIONES REALIZADAS

El conjunto de simulaciones presentadas en este trabajo puede dividirse en tres grupos:

a) Un conjunto de simulaciones realizadas para comprobar que la arquitectura clásica del Modelo DBP no permite que la red aprenda a responder ante un nuevo estímulo, una vez que previamente ha aprendido a responder a uno, (ver capítulo 6).

b) Un conjunto de simulaciones realizadas para comprobar que haciendo modificaciones a la arquitectura clásica de empleada por el Modelo DBP, es posible manejar más de un EC al mismo tiempo (Ver capítulo 7), y

c) La simulación de la inhibición condicionada bajo la preparación original de Pavlov (1927): B+|C+/AC-|AB+, y un conjunto de simulaciones adicionales, realizadas para probar los resultados obtenidos (Ver Capítulo 8).

A continuación se discuten los resultados encontrados en cada uno de estos conjuntos de simulaciones.

9.2.1. SIMULACIONES REALIZADAS PARA COMPROBAR QUE LA ARQUITECTURA CLÁSICA NO PERMITE AL DBP EL MANEJO DE DOS O MAS EC AL MISMO TIEMPO

La arquitectura que hasta ahora se ha empleado en las redes creadas para funcionar bajo el Modelo DBP, basadas en una interconexión completa de sus elementos, no permite que se puedan presentar niveles de activación altos en las unidades "CR/UR" ante un segundo estímulo, una vez que se ha entrenado a la red para que presente niveles de activación elevados ante un primer estímulo. Esta imposibilidad se mantiene aún cuando se somete a la red a un proceso de extinción del estímulo anterior, previo a la presentación del estímulo nuevo. Las simulaciones presentadas en el Capítulo 6 muestran que, bajo esta arquitectura:

- Es imposible que la red aprenda a responder a más de un EC al mismo tiempo. Las simulaciones realizadas permitieron comprobar que esta limitante de la arquitectura clásica no depende del esquema de entrenamiento presentado. Se probaron tres tipos distintos de esquemas de entrenamiento distintos: A+|B+, A+/B+, y A+|A-|B+, sin que en alguno de los casos hubiese habido diferencias importantes.

- La limitación no depende, tampoco del tamaño de la red, ni del número de unidades de entrada, ya que todos los esquemas enunciados en el párrafo anterior fueron presentados a redes de tamaños distintos, e incluso a redes que en su interior eran idénticas, pero que contenían en la capa de entrada diferente número de unidades, sin que se hayan observado diferencias importantes entre sus resultados.

- Tampoco depende de los valores de entrada de los patrones presentados a la red, ya que se emplearon no sólo valores binarios discretos [0,1], sino valores continuos acotados, que en algunos casos, incluso, fueron establecidos en forma aleatoria, sin que hubiese tampoco diferencias importantes.

Las pruebas presentadas en el Capítulo 6, representan también el primer intento formal reportado, de exploración de esta imposibilidad presentada por la arquitectura clásica usada en el Modelo DBP, hasta ahora, se trataba de un hecho conocido por los autores del Modelo, pero no estudiado.

9.2.2. SIMULACIONES PARA EXPLORAR NUEVAS ARQUITECTURAS QUE PERMITIERAN MANEJAR DOS O MAS EC

En el capítulo 7 se exploraron dos tipos generales de arquitecturas distintas a la arquitectura clásica. En ambos tipos se usaron subredes para procesar estímulos distintos y no se emplearon interconexiones completas entre sus elementos, variando el grado de interconexión parcial de una instancia a otra a lo largo de las simulaciones.

Ambas propuestas se establecieron con base en la búsqueda de un punto de unión en el que dos redes dejaran de considerarse sistemas separados, para comenzar a considerarse como uno solo: esta fue la estrategia de búsqueda de la arquitectura más adecuada para resolver el problema. Sin embargo, aún a pesar de que ambas propuestas se derivan del mismo principio, los resultados demuestran que aquellas arquitecturas que realizaban un "cierre" de las subredes, a partir de una sola unidad integradora de tipo "CR/UR" que concentrara los impulsos provenientes de todos los sub-circuitos, fueron menos efectivas en el manejo simultáneo de varios EC, a diferencia de aquellas redes que integraron subredes con salidas independientes a unidades de tipo "CR/UR" separadas para cada uno de estos sub-circuitos o subredes.

Aquellas arquitecturas que se conformaron con una unidad "CR/UR" integradora de los sub-circuitos, demostraron ineficiencia para aprender a responder con niveles altos de activación ante la presencia de un EC nuevo; sin embargo, una vez que se sometía a la red a un proceso de extinción del EC previo, la red era capaz de aprender a responder efectivamente ante un nuevo EC.

Por otra parte, las redes que usaron salidas independientes de tipo "CR/UR" para cada uno de los sub-circuitos que integran la red, demostraron la capacidad para responder efectivamente ante un nuevo EC sin que haya habido un proceso de extinción previo.

Del conjunto de simulaciones realizadas en este capítulo se desprende que la estrategia consistente en el empleo de subredes para el manejo de varios estímulos en una red bajo el Modelo DBP es efectiva para que la red pueda aprender a responder ante varios EC a la vez, bajo las siguientes restricciones:

1) Para cada EC debe haberse dedicado una subred en la arquitectura general de la red a emplear. Con la ventaja de que son altamente efectivos, aún los circuitos más simples, compuestos por una unidad de tipo "sa", una unidad de tipo "ca1", una unidad de tipo "ma", una unidad de tipo "vta" y una unidad de tipo "CR/UR".

2) Cada una de estas subredes debe incluir una unidad "CR/UR" independiente, como salida de la subred. Es en esta unidad en donde se registra entonces, la salida de la red correspondiente a un estímulo determinado: aquel que fue procesado por esa subred específicamente.

3) El cálculo de discrepancias y la subsecuente actualización de pesos de la red, se debe realizar exactamente como el modelo lo plantea. Es decir, el cambio en la arquitectura, usando

subredes para el proceso de patrones, no implica que se tenga que hacer el cálculo de discrepancias y actualización de pesos en forma separada.

4) Las entradas pueden estar conectando a la totalidad de los sub-circuitos o subredes a partir de conexiones, siempre y cuando éstas no estén activas en el conjunto de patrones de entrada presentados.

Finalmente, es necesario remarcar que estos dos tipos de arquitecturas permiten efectivamente a la red, aprender a responder ante dos o más EC, hecho al que se deben sumar las siguientes consideraciones:

- No fue necesario hacer modificación alguna al modelo en alguno de sus dos submodelos.
- No fue necesario el uso de una estrategia de actualización de pesos distinta.
- No se usaron conexiones inhibitorias.
- No fue necesario el uso de procesadores paralelos para lograr la simulación de este tipo de procesamiento.
- La inversión de recursos necesarios para lograr el presente trabajo es muy baja.

9.2.3 EN CUANTO A LA SIMULACIÓN DE LA INHIBICIÓN CONDICIONADA

Las simulaciones realizadas en Capítulo 8 pretendieron simular la inhibición condicionada, con base en el uso de la preparación clásica reportada por Pavlov (1927), consistente en el uso de un esquema de tres condiciones: B+|A+/AC-|AB. Asimismo, para poder constatar que el estímulo condicionado, en este caso el estímulo "A" es efectivamente un inhibidor, se planteó el uso de las dos pruebas establecidas por Rescorla (1969): la prueba de sumación y la prueba de retardo.

La inhibición condicionada es un fenómeno que implica, de acuerdo con la definición del esquema de entrenamiento ya referido, la intervención de 3 estímulos diferentes, por lo que se requería de una red que pudiese manejarlos adecuadamente. No habría sido posible intentar esta simulación si no se hubiesen obtenido resultados efectivos en las simulaciones realizadas en el Capítulo 7, para encontrar una arquitectura que permitiera manejar varios EC al mismo tiempo.

Los resultados no son contundentes en este sentido. Parece ser que el estímulo supuestamente inhibidor putativo, pasa efectivamente la prueba de sumación, pero no pasa la prueba de retardo. Sin embargo, estos resultados deben ser analizados con más detenimiento, considerando el hecho de que es posible que en esta ausencia de retardo intervengan variables del funcionamiento del modelo que no hayan sido contempladas.

Para el caso de la sumación, uno de los primeros problemas que se tuvieron que enfrentarse en la realización de la simulación, fue saber si el efecto observado no era producto de la presentación de niveles bajos de activación en la primera condición de la simulación realizada. Es por ello que se tuvieron que ensayar otro tipo de combinaciones de estímulos que activaran no solamente una subred, sino dos o más, de manera que se pudieran tener valores de activación del mismo nivel que los

observados durante la tercera condición de la simulación. Una vez elevados los niveles de activación en la primera condición de la simulación se encontró que el efecto de sumación observado en la simulación inicial, no variaba, por lo que pudo inferirse que no se trataba de la combinación de estímulos inicial y en consecuencia, de los niveles de activación producidos por éstos.

Asimismo, con base en el ensayo de las posibilidades de combinación de estímulos, más significativas para los efectos del presente trabajo, se observó que la sumación se observaba en todas aquellas preparaciones en donde parte de los estímulos que habían sido aprendidos en una primera condición, era presentada sin refuerzo en la segunda condición.

La prueba de retardo no tuvo resultados exitosos como la prueba de la sumación. El retardo se midió con base en el número de ensayos que se requerían para poder aprender a responder ante un estímulo, coincidiendo este número de ensayos, con los ensayos durante los cuales era posible observar el efecto de sumación.

Si se asume que efectivamente el estímulo "A" es un candidato viable para ser considerado como un inhibidor condicionado, se observa derivado de los mismos resultados presentados anteriormente, que la adquisición de la respuesta se presenta en un número de ensayos menor a los que son requeridos para la adquisición de una respuesta sin haberse sometido a las condiciones anteriores. Bajo esta perspectiva no se da un retardo. Sin embargo, se realizaron varias simulaciones en las que fue variada la proporción de ensayos en la presentación del compuesto C+/CA- y se llegó a alcanzar un número cercano al número de ensayos requeridos para la readquisición.

En este trabajo se presenta solamente una exploración de la simulación de la inhibición condicionada, empleando como criterio para el establecimiento de la existencia o no de retardo, el número de ensayos requeridos para la presentación de niveles altos de activación en la unidad de respuesta "CR/UR" correspondiente, comparados con la cantidad de ensayos requeridos para la instalación de una respuesta o para la readquisición de la misma. Sin embargo, existe la posibilidad de que existan otros criterios para establecer la existencia o no del retardo y que no fueron considerados en el presente trabajo, como por ejemplo, en función no solamente del número de ensayos requeridos, sino también de los propios niveles de activación presentados, es decir, usando "criterios de respuesta" que sería una forma más cercana a la que se emplea en organismos vivos. Se sugiere para estudios posteriores el estudio de esta posibilidad.

La exploración realizada de otras preparaciones experimentales derivadas de las condiciones originales empleadas por Pavlov (1927) y reportadas en el Capítulo 8 como "combinaciones significativas" del conjunto de combinaciones posibles, sugieren que la simulación de la inhibición condicionada es posible, a partir de los resultados similares observados en el comportamiento de las activaciones en las unidades "CR/UR" obtenidos en el uso de condiciones cercanas a las reportadas por Savastano y otros (1999) como otras preparaciones experimentales que producen inhibidores condicionados. Sin embargo, estas preparaciones no fueron estudiadas con más detalle. Habría que realizar estudios con mayor profundidad empleando estas preparaciones, articulándolas con las otras posibles evidencias del retardo que se han mencionado en el párrafo anterior.

9.5 EN CUANTO AL TRABAJO REALIZADO

Este trabajo demuestra que es posible el manejo de varios estímulos condicionados en una red neural bajo el Modelo DBP, sin que se requiera hacer alguna modificación al Sub-Modelo de red o al Sub-Modelo Neurocomputacional. La posibilidad de que se requiriera hacer algún tipo de modificación de este tipo era considerada como el último recurso a seguir, en caso de que no se encontrara alguna arquitectura capaz de resolver el problema.

El empleo de conexiones inhibitorias estaba considerado como segunda opción, en caso de que no se hubieran encontrado arquitecturas que posibilitaran a una red aprender a responder ante varios EC simultáneamente. Las conexiones inhibitorias no fueron empleadas en este trabajo y constituyen un tema que requiere ser estudiado en combinación con las arquitecturas encontradas.

Se presenta una propuesta para la búsqueda de la simulación de la inhibición condicionada y establece que es factible producir, con ciertas consideraciones, inhibidores condicionados sin el uso de conexiones inhibitorias.

Otro de los aportes del el presente trabajo, aunque de índole más práctica, representa el hecho de que haya sido documentado el grupo de algoritmos empleados en cada función específica que conforma el simulador; hecho que en la práctica representó varios problemas prácticos al momento de la realización del mismo. Estos problemas prácticos no están documentados en la bibliografía existente a la fecha y sin embargo, son muy importantes al tratar de diseñar un simulador bajo este modelo.

REFERENCIAS

Aizawa K, *History of connectionism*, documento en internet, dirección: http://artsci.wustl.edu/~philos/MindDict/connectionismhistory.html. Revisión: 6 de Julio del 2002.

Alkon, D L., T P. Vogl y D Tam, Memory Function in Neural and Artificial Networks, en: *Neural Network Models of Conditioning and Action*, Nueva York, Lawrence Erlbaum Associates Publishers, 1991, páginas 1-11.

Baxter, D. A., D. V. Buonomano, J. L. Raymond, D. G. Cook, F. M. Kuenzi, T. J. Carew y J.H. Byrne, Empirically derived adaptative elements and networks simulate associative learning, en: *Neural Network Models of Conditioning and Action*, Nueva York, Lawrence Erlbaum Associates Publishers, 1991, páginas 13-52.

Burgos, J. E. (1997). Evolving artificial neural networks in Pavlovian environments. In J. W. Donahoe & V. Packard-Dorsel (Eds.), Neural-network models of cognition (pp. 58-79). Holland: Elsevier Science.

Burgos, J. E.(2000) Superstition in Artificial Neural Networks, en: *Revista Mexicana de Análisis de la Conducta*, Volumen 26, Número 2, 2000, páginas 159-190.

Cheng-Lian y Lin Chin-Teng, Reinforcement Learning for an ART-Based Fuzzy Adaptative Learning Control Network, en: *IEEE Transactions on Neural Networks*, Vol 7, No 3, Mayo de 1996, páginas 709- 731.

Chester, D. L. A Comparison of some Neural Network Models of Classical Conditioning, IEEE, Documento TH0333-5/90/0000/1163, 1990.

Churchland, P. *Matter and Consciousness*, Revised Edition, The MIT-press, 1993.

Cohen, Paul R. *Empirical Methods for Artificial Intelligence*, Cambridge Mass, The MIT Press, 1995.

Cole, R. C., R. C. Barnet y R. R. Miller, An evaluation of conditioned inhibition as defined by Rescorla's two-test strategy, en: *Learning and Motivation*, Num 28, 1997, páginas 323-341.

Donahoe, J W, J E. Burgos y D C. Palmer, A Selectionist approach to reinforcement, en: *Journal of The Experimental Analysis of Behavior*, Vol 60, Num 1, Julio de 1993, páginas 17- 40.

Donahoe, J, W., J. E. Burgos y D. C. Palmer, A selectionist approach to reinforcement, en: *Journal of the experimental analysis of behavior*, Volumen 60, Número 1, Julio de 1993, páginas 17 a la 40.

Donahoe, J. W., Burgos, J. E., & Palmer, D. C. (1993). A selectionist approach to reinforcement. Journal of the Experimental Analysis of Behavior, 60, 17-40.

Donahoe, J. W., & Palmer, D. C. (1994). Learning and complex behavior.Boston: Allyn & Bacon.

Donahoe, J. W., Palmer, D. C., & Burgos, J. E. (1997a). The S-R issue: Its status in behavior analysis and in Donahoe and Palmer's Learning and Complex Behavior. Journal of the Experimental Analysis of Behavior, 67, 193-211.

Donahoe, J. W., Palmer, D. C., & Burgos, J. E. (1997b). The unit of selection: What do reinforcers reinforce? Journal of the Experimental Analysis of Behavior,67, 259-273.

Freeman J.A. y D.M. Skapura, *Neural Networks, Algorithms, Applications and Programming Techinques*, Addison-Wesley, 1991.

Fritzke, B. (1997) *Some Competitive Learning Methods*, Reporte de Investigación, Institute for Neural Computation, Ruhr-Universität Bochum, Abril 5 de 1997.

Fukushima, K. (1975). Cognitron: A self-organizing multilayered neural network. Biological Cybernetics, 20, 121-136.

Gallant, Stephen I. *Neural Network Learning and Expert Systems*, Cambridge, The MIT Press, 1994.

Gluck, Mark A y Catherine E. Myers, A Neural Network Approach to Adaptative Similarity and Stimulus Representations in Cortico-hippocampal function en: *Neural Networks Models of Cognition*, edición de J. W. Donahoe y V. Packard Dorsel, 1997, Elsevier Science, B.V. páginas 220-243.

Gray R. M. (1992) *Vector Quantitization and Signal Compression.* Kluwer Academic Press, 1992.

Grossberg S. y N Schmajuk. Neural Dynamics of adaptativo timing and Temporal Discrimination during Associative Learning. Neural Networks, 2(2), 79-102, 1989.

Haykin, S. *Neural networks. A Comprehensive fundation*, Nueva York, McMillan College Publishing Company, 1994.

Hebb, D. O. *The Organization of Behavior*, Nueva York, Wiley. Introducción, Capítulo III y Capítulo IV, reimpresos en: *Neurocomputing*, J. Anderson y E. Rosenfeld (editores), The MIT-Press.

Hecht-Nielsen, R. Neurocomputing. Reading, MA: Addison-Wesley. 1990

Hilera, J. R. y V J. Martínez, *Redes Neurales Artificiales, Fundamentos, modelos y aplicaciones*, México, Editorial Alfa Omega, 2000.

Hopfield, J.J. Neural Networks and Physical systems with emergent collective computational habilitéis, en: *Proceedings of The National Academy of Sciences*, No. 79, 1982, págs. 2554- 2558.

Humprey, W. S. *Managing the software process*, Addison-wesley publishing company, 1990.

James, W, *Principles of Psychology*, Henry Holt, New York, 1890

Kaski, S. *Data exploration using self-organizing maps*, Número 82 de la serie Acta Polytechnica Scandinavica, Mathematics, Computing and Management in Engineering Series, (English version) Finnish Academy of Technology, Espoo, 1997

Kehoe, E. James, A layered model of associative learning: learning to learn and configuration, en: *Psychological Review*, 1988, Vol 95, No. 4, págs 411-433.

Klopf A. H. Classical conditioning phenoinena predicted by a drive-reinforcement model of neuronal function. In John H. Byrne and William 0. Berry, editors, Neural Models of Plasticity: Experimental and Theoretical Approaches, chapter 7, pages 104-132, Academic Press Inc, 1989.

Kohonen, Teuvo, Self Organized formation of topologically correct feature maps, en: *Biological Cybernetics*, No. 43, 1982, págs. 59-69.

Lpimann, R. P. An introduction to computing with Neural Nets, en: *IEEE ASSP Magazine*, IEEE, Abril de 1987, págs 4 -22.

LoLordo, V. M. y J. L. Fiarles, Pavlovian Conditioned Inhibition: The literature since 1969, en: *Information processing in animals: conditioned inhibition*, R. R. Miller y N. E. Spear (editores), Lawrence Erlbaum Editors, páginas 1-49.

Maravall Gómez_Allende, D, *Reconocimiento de formas y visión artificial*, Wilmington, Delaware, Addison-Wesley Iberoamericana/RAMA, 1994.

McClelland, J. L., D. E. Rummelhart y G. Hinton, Representaciones distribuidas, en: Filosofía *de la Inteligencia Artificial*, compilación de Margaret Boden, México, Fondo de Cultura Económica, 1994.

McCulloch, Warren S. Y Walter Pitts, A logical Calculus of the ideas immanent in nervous activity, en: *Bulletin of Mathematical Biophysics*, Numero 5, 1943 págs 115-133 reimpreso en Neurocomputing, J. Anderson y E. Rosenfeld (editores), The MIT-Press.

Mensik, Ger-Jan y Jeroen G.W.Raaijmakers, A model for interference and forgetting en: *Psychological Review*, 1988, Vol 95, No. 4, págs 434-455.

Minsky, M y S. Papert, Perceptrons, 1969, reimpreso en: *Neurocomputing*, J. Anderson y E. Rosenfeld (editores), The MIT-Press.

Minsky, M, Logical vs. Analogical or Symbolic vs. Connectionist or Neat vs. Scruffy, en: *Artificial Intelligence at MIT. Expanding Frontiers*, Patrick H. Winston (editor), Vol 1,Masachussetts, MIT Press, 1990. Reimpreso en AI Magazine, 1991

Moore, J W. Implementing connectionist Algorithms for Classical Conditioning in the brain, en: *Neural Network Models of Conditioning and Action*, Nueva York, Lawrence Erlbaum Associates Publishers, 1991, páginas 181-199.

Olmsted P, *Brief History of Neural Networks*, McMillan, Nueva York, 1999

Pagels, H. R. *Los sueños de la razón. El ordenador y los nuevos horizontes de las ciencias de la complejidad*, México, CONACyT/GEDISA, 1991.

Papini, M.R y M.E. Bitterman, The two test strategy in the study of inhibitory conditioning, en: *Journal of Experimental Psychology: Animal Behavior Processes*, Número 19, 1993, páginas 342-352.

Parker, D.B. *Learning Logic* Technical Report 47, Center for Computational Research in Economics and Management Science, MIT, Cambridge, 1985, reimpreso en: *Neurocomputing*, J. Anderson y E. Rosenfeld (editores), The MIT-Press.

Pavlov, I., Conditioned Reflexes, 1927, traducción al idioma inglés realizada por G.V. Anrep, 1927, disponible en internet, en la siguiente dirección: http://pavlov.psyc.queensu.ca/~ron/325/3/mainpav.htm, consultada en junio del 2002.

Pedrycz, W y J Waletzky, Neural Network front Ends in Unsupervised Learning en: *IEEE Transactions on Neural Networks*, Vol 8, No 2, 1997, páginas 390-402.

Ponce, A y J Burgos, *Simulación de la Generalización Pavloviana con redes neurales,* ponencia presentada en la 25 Conferencia Anual de la SQAB, Tornoto Canadá, mayo 24 y 25 del 2002.

Ponce A. Presentación de avances de trabajo de tesis en proseminarios del Doctorado en Ciencias del Comportamiento, Documento de trabajo, 2003.

Ponce A. Presentación de avances de trabajo de tesis en proseminarios del Doctorado en Ciencias del Comportamiento, Documento de trabajo, 2004.

Ponce A. Presentación de avances de trabajo de tesis en proseminarios del Doctorado en Ciencias del Comportamiento, Documento de trabajo, 2005.

Rescorla, R. A. Pavlovian conditioned inhibition, en: *Psychological Bulletin*, Num 72, 1969, páginas 77-94.

Rashevsky, N, *Mathematical Biophysics*, University of Chicago Press, Chicago, IL, 1938

Rosenblatt, F. The perceptron: a probabilistic model for information storage and organization in the brain, en: *Psichological Review* No. 65, 1958, págs 386-408, reimpreso en: *Neurocomputing*, J. Anderson y E. Rosenfeld (editores), The MIT-Press.

Rummelhart D. y J. McClelland (editores) *Paralell distribuited processing: Explorations in the microstructure of cognition*, Vol 1 Foundations, The MIT-Press, 1986.

Savastano, H. I., R. P. Cole, R. C. Barnet y R. R. Miller, Reconsidering conditioned inhibition, en: *Learning and Motivation*, Num 30, 1999, páginas 102-127.

Schmajuck, N. A. *Animal Learning and Cognition. A Neural Network approach*, Cambridge University Press, 1997.

Skapura, D. M. *Building Neural Networks*, Nueva York, Nueva York, ACM Press, 1995.

Sutton R.S. y A. G. Barto. Toward a modern theory of adaptive networks: expectation and prediction. Psychological Review,88(2):135-170, 1981.

Thompson, R. F., N. H. Donegan, G. A. Clark, D. G. Lavond, J. S. Lincoln, J. Madden IV, L Mamounas, M D. Mauk y D A. McCormick, Neural substrates of discrete, defensive conditioned reflexes, conditioned fear states and their interactions in the rabbit, en: *Classical Conditioning*, 3rd Edition, Isidore Gormezano, William F. Prokasy y Richard F. Thompson (editores), Hillsdale Nueva Jersey, Lawrence Erlbaum Associates, publishers, 1987, páginas 371-399

Turban, E, *Expert Systems and Applied Artificial Intelligence*, Nueva York, McMillan Publishing Company, 1992.

Wasserman, E. A., S. R. Franklin y E. Hearst, Pavlovian appetitive contingencies and approach vs. Withdrawal to conditioned stimuli in pigeons, en: *Journal of Comparative and Psysiological Psychology*, Num 86, 1974, 616-627.

Werbos, P, *Beyond Regression: New Tools for Prediction and Analysis in Behavioral Sciences*, Cambridge, Ma. 1974, citado por Hecht-Nielsen, en: *Neurocomputing*, Reading, Addison-Wesley Publishing Company, 1990

ANEXO 1

Código fuente en lenguaje Visual C# de los archivos creados para el simulador

Archivo Program.cs

```
using System;
using System.Collections.Generic;
using System.Windows.Forms;

namespace DBP5000
{
    static class Program
    {
        /// <summary>
        /// The main entry point for the application.
        /// </summary>
        [STAThread]
        static void Main()
        {
            Application.EnableVisualStyles();
            Application.SetCompatibleTextRenderingDefault(false);
            Application.Run(new Form1());
        }
    }
}
```

Archivo Form1.Designer.cs:

```
namespace DBP5000
{
    partial class Form1
    {
        /// <summary>
        /// Required designer variable.
        /// </summary>
        private System.ComponentModel.IContainer components = null;

        /// <summary>
        /// Clean up any resources being used.
        /// </summary>
        /// <param name="disposing">true if managed resources should be disposed;
otherwise, false.</param>
        protected override void Dispose(bool disposing)
        {
            if (disposing && (components != null))
            {
                components.Dispose();
            }
            base.Dispose(disposing);
        }

        #region Windows Form Designer generated code

        /// <summary>
        /// Required method for Designer support - do not modify
        /// the contents of this method with the code editor.
        /// </summary>
        private void InitializeComponent()
```

```
{
    this.button1 = new System.Windows.Forms.Button();
    this.button3 = new System.Windows.Forms.Button();
    this.button2 = new System.Windows.Forms.Button();
    this.checkBox1 = new System.Windows.Forms.CheckBox();
    this.SuspendLayout();
    //
    // button1
    //
    this.button1.Location = new System.Drawing.Point(28, 23);
    this.button1.Name = "button1";
    this.button1.Size = new System.Drawing.Size(137, 37);
    this.button1.TabIndex = 2;
    this.button1.Text = "Crear Red";
    this.button1.UseVisualStyleBackColor = true;
    this.button1.Click += new System.EventHandler(this.button1_Click_1);
    //
    // button3
    //
    this.button3.Location = new System.Drawing.Point(28, 73);
    this.button3.Name = "button3";
    this.button3.Size = new System.Drawing.Size(136, 43);
    this.button3.TabIndex = 3;
    this.button3.Text = "Correr Simulación";
    this.button3.UseVisualStyleBackColor = true;
    this.button3.Click += new System.EventHandler(this.button3_Click);
    //
    // button2
    //
    this.button2.Location = new System.Drawing.Point(29, 276);
    this.button2.Name = "button2";
    this.button2.Size = new System.Drawing.Size(135, 50);
    this.button2.TabIndex = 4;
    this.button2.Text = "Guardar en Archivo";
    this.button2.UseVisualStyleBackColor = true;
    this.button2.Click += new System.EventHandler(this.button2_Click);
    //
    // checkBox1
    //
    this.checkBox1.AutoSize = true;
    this.checkBox1.Checked = true;
    this.checkBox1.CheckState = System.Windows.Forms.CheckState.Checked;
    this.checkBox1.Location = new System.Drawing.Point(227, 34);
    this.checkBox1.Name = "checkBox1";
    this.checkBox1.Size = new System.Drawing.Size(81, 17);
    this.checkBox1.TabIndex = 5;
    this.checkBox1.Text = "Aprendizaje";
    this.checkBox1.UseVisualStyleBackColor = true;
    //
    // Form1
    //
    this.AutoScaleDimensions = new System.Drawing.SizeF(6F, 13F);
    this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;
    this.ClientSize = new System.Drawing.Size(598, 372);
    this.Controls.Add(this.checkBox1);
    this.Controls.Add(this.button2);
    this.Controls.Add(this.button3);
    this.Controls.Add(this.button1);
    this.Name = "Form1";
    this.Text = "DBP5000";
    this.ResumeLayout(false);
    this.PerformLayout();
```

```
        }

        #endregion

        private System.Windows.Forms.Button button1;
        private System.Windows.Forms.Button button3;
        private System.Windows.Forms.Button button2;
        private System.Windows.Forms.CheckBox checkBox1;
    }
}
```

Archivo Form1.cs:

```
using System;
using System.Collections; // para la nueva version del create random list
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;
using System.IO;
using System.Threading;


namespace DBP5000
{
    public partial class Form1 : Form
    {
        public int Numero_Ensayos = 300;
        public int Numero_Time_Steps = 8;
        public int Numero_Unidades =14;  // Va de la Unidad 0 a la 13

        public int Ensayo;

        int Tot_CA1_NPEs = 1; // Total de Unidades CA1
        double Tot_CA1_d_t;   // Valor Total de la suma de las activaciones de
las unidades CA1
        double Avg_CA1_d_t;   // Valor promedio de las activaciones de las CA1 en
el tiempo t

        int Tot_VTA_NPEs = 1; // Total de Unidades VTA
        double Tot_VTA_d_t;  // Valor total de la suma de las activaciones de las
unidades VTA
        double Avg_VTA_d_t;   // Valor promedio de las activaciones de las VTA en
el tiempo t

        double Nr = 1;  // Maximum proportion of receptors on postsynaptic NPE
        double d_thr = 0.001; // Umbral de discrepancia


        public int Numero_Unidades_Entrada = 3;     // Numero de unidadesde
entrada en la red

        public int LimiteBajoNeuronaElegida = 1;     // Generacion de numeros
aleatorios unicos

        public int LimiteBajoConexionElegida = 1;    // Generacion de numeros
aleatorios unicos

        public int GenerarlistaNeuronas = 1;         // Generacion de numeros
aleatorios unicos Neuronas
```

```
        public int GenerarlistaConexiones = 2;      // Generacion de numeros
aleatorios unicos Conexiones

        int[] ArrayNeuronasElegidas;                // Generacion de numeros
aleatorios unicos array Neuronas
        int[] ArrayConexionesElegidas;              // Generacion de numeros
aleatorios unicos array conexiones


        public double cR;                                       // para la
generacion de aleatorios gaussianos
        public double cdR;                                      // para la
generacion de aleatorios gaussianos
        public double cmR;                                      // para la
generacion de aleatorios gaussianos
        public int i97R;                                        // para la
generacion de aleatorios gaussianos
        public int j97R;                                        // para la
generacion de aleatorios gaussianos
        public int testR = 0;                                   // para la
generacion de aleatorios gaussianos
        public double[] uR = new double[97];        // para la generacion de
aleatorios gaussianos


        public double TAU = 0.1;
        public double KAPPA = 0.1;
        public double ALPHA = 0.5;
        public double BETA = 0.1;
        public double LOGISSIGMA = 0.1;
        public double LOGISMU = 0.5;
        public double PESOINOCENTE = 0.01;      // original 0.01 , en algunas
simulaciones es de 0.001
        public double Thr_mu = 0.2;             // Valor original 0.2
        public double Thr_sigma = 0.15;         // Valor original 0.15


        struct Unidad_Struct
        {
            public int Numero_Unidad;
            public int Tipo_Unidad;
            public int Tipo_Activacion;
            public double Valor_Activacion;
            public double Valor_Activacion_Tmenos1;
            public double p_epsp_t; // logistica de exc

            public double Tau;
            public double Kappa;
            public double Alpha;
            public double Beta;
            public double LogisSigma;
            public double LogisMu;

            public int Numero_Conexiones_Entrantes;
            public int[] Unidad_Presinaptica;
            public double[] Peso_Presinaptico;

        }

        Unidad_Struct[] Unidad;

        public double[,] Entrada;
```

```
        public double[,,] Registro_de_Activaciones; // Para guardar todas las
activaciones de la simulación

        public double[, ,] Registro_de_Pesos; // Para guardar todos los pesosde
la simulación

        public Form1()
        {
            InitializeComponent();
        }

        public void Crear_Red()
        {
            // Defino arrays para guardar datos

            Registro_de_Activaciones = new double[Numero_Unidades,
Numero_Time_Steps, Numero_Ensayos];


            // Defino las entradas

            Entrada = new double[(Numero_Unidades_Entrada + 1),
Numero_Time_Steps]; // Entradas vs TimeSteps

            // Defino las unidades

            Unidad = new Unidad_Struct[Numero_Unidades];

            Unidad[4].Unidad_Presinaptica = new int[4]; // SA
            Unidad[4].Peso_Presinaptico = new double[4];

            Unidad[5].Unidad_Presinaptica = new int[4]; // SA
            Unidad[5].Peso_Presinaptico = new double[4];

            Unidad[6].Unidad_Presinaptica = new int[4]; // SA
            Unidad[6].Peso_Presinaptico = new double[4];

            Unidad[7].Unidad_Presinaptica = new int[4]; // CA1
            Unidad[7].Peso_Presinaptico = new double[4];

            Unidad[8].Unidad_Presinaptica = new int[4]; // MA
            Unidad[8].Peso_Presinaptico = new double[4];

            Unidad[9].Unidad_Presinaptica = new int[4]; // MA
            Unidad[9].Peso_Presinaptico = new double[4];

            Unidad[10].Unidad_Presinaptica = new int[4]; // MA
            Unidad[10].Peso_Presinaptico = new double[4];

            Unidad[11].Unidad_Presinaptica = new int[4]; // VTA
            Unidad[11].Peso_Presinaptico = new double[4];

            Unidad[12].Unidad_Presinaptica = new int[4]; // R
            Unidad[12].Peso_Presinaptico = new double[4];

            Unidad[13].Unidad_Presinaptica = new int[4]; // CR/UR
            Unidad[13].Peso_Presinaptico = new double[4];
        }


        public void Cargar_Red()
        {
```

```
//   U1      U9         U13
//                                U17
//   U2      U10        U14
//                                U18
//   U3      U11        U15
//
//   U4
//
//   U5
//
//   U6
//
//   U7
//
//   U8
//
//                U12         U16
//   U0

// TimeStep 0

Entrada[0, 0] = 0.0; // Entrada S* NO REFUERZO
Entrada[1, 0] = 1.0; // Entrada 1
Entrada[2, 0] = 1.0; // Entrada 2
Entrada[3, 0] = 1.0; // Entrada 3

// TimeStep 1

Entrada[0, 1] = 0.0; // Entrada S* NO REFUERZO
Entrada[1, 1] = 1.0; // Entrada 1
Entrada[2, 1] = 1.0; // Entrada 2
Entrada[3, 1] = 1.0; // Entrada 3

// TimeStep 2

Entrada[0, 2] = 0.0; // Entrada S* NO REFUERZO
Entrada[1, 2] = 1.0; // Entrada 1
Entrada[2, 2] = 1.0; // Entrada 2
Entrada[3, 2] = 1.0; // Entrada 3

// TimeStep 3

Entrada[0, 3] = 0.0; // Entrada S* NO REFUERZO
Entrada[1, 3] = 1.0; // Entrada 1
Entrada[2, 3] = 1.0; // Entrada 2
Entrada[3, 3] = 1.0; // Entrada 3

// TimeStep 4

Entrada[0, 4] = 0.0; // Entrada S* NO REFUERZO
Entrada[1, 4] = 1.0; // Entrada 1
Entrada[2, 4] = 1.0; // Entrada 2
Entrada[3, 4] = 1.0; // Entrada 3

// TimeStep 5

Entrada[0, 5] = 0.0; // Entrada S* NOREFUERZO
Entrada[1, 5] = 1.0; // Entrada 1
Entrada[2, 5] = 1.0; // Entrada 2
Entrada[3, 5] = 1.0; // Entrada 3

// TimeStep 6
```

```
            Entrada[0, 6] = 0.0; // Entrada S*  NO REFUERZO
            Entrada[1, 6] = 1.0; // Entrada 1
            Entrada[2, 6] = 1.0; // Entrada 2
            Entrada[3, 6] = 1.0; // Entrada 3

            // TimeStep 7   Este es el TimeStep en el que se da el refuerzo

            Entrada[0, 7] = 1.0; // Entrada S*  REFUERZO
            Entrada[1, 7] = 1.0; // Entrada 1
            Entrada[2, 7] = 1.0; // Entrada 2
            Entrada[3, 7] = 1.0; // Entrada 3

            Unidad[1].Numero_Unidad = 1;
            Unidad[1].Tipo_Unidad = 0;       // 0 Entrada, 1 SA, 2 MA, 3 CA1, 4
VTA, 5 R, 6 CR/UR
            Unidad[1].Tipo_Activacion = 1;  // 0 Inhibitoria, 1 Excitatoria

            Unidad[2].Numero_Unidad = 2;
            Unidad[2].Tipo_Unidad = 0;       // 0 Entrada, 1 SA, 2 MA, 3 CA1, 4
VTA, 5 R, 6 CR/UR
            Unidad[2].Tipo_Activacion = 1;  // 0 Inhibitoria, 1 Excitatoria

            Unidad[3].Numero_Unidad = 3;
            Unidad[3].Tipo_Unidad = 0;       // 0 Entrada, 1 SA, 2 MA, 3 CA1, 4
VTA, 5 R, 6 CR/UR
            Unidad[3].Tipo_Activacion = 1;  // 0 Inhibitoria,1 Excitatoria


            // Unidad 4 SA Capa 1

            Unidad[4].Numero_Unidad = 4;
            Unidad[4].Tipo_Unidad = 1;       // 0 Entrada, 1 SA, 2 MA, 3 CA1, 4
VTA, 5 R, 6 CR/UR
            Unidad[4].Tipo_Activacion = 1;  // 0 Inhibitoria, 1 Excitatoria

            Unidad[4].Tau = TAU;
            Unidad[4].Kappa = KAPPA;
            Unidad[4].Alpha = ALPHA;
            Unidad[4].Beta = BETA;
            Unidad[4].LogisSigma = LOGISSIGMA;
            Unidad[4].LogisMu = LOGISMU;

            Unidad[4].Numero_Conexiones_Entrantes = 3;

            Unidad[4].Unidad_Presinaptica[1] = 1;
            Unidad[4].Peso_Presinaptico[1] = PESOINOCENTE;
            Unidad[4].Unidad_Presinaptica[2] = 2;
            Unidad[4].Peso_Presinaptico[2] = PESOINOCENTE;
            Unidad[4].Unidad_Presinaptica[3] = 3;
            Unidad[4].Peso_Presinaptico[3] = PESOINOCENTE;

            // Unidad 5 SA Capa 1

            Unidad[5].Numero_Unidad = 5;
            Unidad[5].Tipo_Unidad = 1;       // 1 SA, 2 MA, 3 CA1, 4 VTA, 5 R, 6
CR/UR
            Unidad[5].Tipo_Activacion = 1;  // 0 Inhibitoria, 1 Excitatoria

            Unidad[5].Tau = TAU;
            Unidad[5].Kappa = KAPPA;
            Unidad[5].Alpha = ALPHA;
            Unidad[5].Beta = BETA;
            Unidad[5].LogisSigma = LOGISSIGMA;
```

```
Unidad[5].LogisMu = LOGISMU;

Unidad[5].Numero_Conexiones_Entrantes = 3;

Unidad[5].Unidad_Presinaptica[1] = 1;
Unidad[5].Peso_Presinaptico[1] = PESOINOCENTE;
Unidad[5].Unidad_Presinaptica[2] = 2;
Unidad[5].Peso_Presinaptico[2] = PESOINOCENTE;
Unidad[5].Unidad_Presinaptica[3] = 3;
Unidad[5].Peso_Presinaptico[3] = PESOINOCENTE;

// Unidad 6 SA Capa 1

Unidad[6].Numero_Unidad = 6;
Unidad[6].Tipo_Unidad = 1;       // 1 SA, 2 MA, 3 CA1, 4 VTA, 5 R, 6
CR/UR
Unidad[6].Tipo_Activacion = 1;  // 0 Inhibitoria, 1 Excitatoria

Unidad[6].Tau = TAU;
Unidad[6].Kappa = KAPPA;
Unidad[6].Alpha = ALPHA;
Unidad[6].Beta = BETA;
Unidad[6].LogisSigma = LOGISSIGMA;
Unidad[6].LogisMu = LOGISMU;

Unidad[6].Numero_Conexiones_Entrantes = 3;

Unidad[6].Unidad_Presinaptica[1] = 1;
Unidad[6].Peso_Presinaptico[1] = PESOINOCENTE;
Unidad[6].Unidad_Presinaptica[2] = 2;
Unidad[6].Peso_Presinaptico[2] = PESOINOCENTE;
Unidad[6].Unidad_Presinaptica[3] = 3;
Unidad[6].Peso_Presinaptico[3] = PESOINOCENTE;

// Unidad 7 CA1

Unidad[7].Numero_Unidad = 7;
Unidad[7].Tipo_Unidad = 3;       // 1 SA, 2 MA, 3 CA1, 4 VTA, 5 R, 6
CR/UR
Unidad[7].Tipo_Activacion = 1;  // 0 Inhibitoria, 1 Excitatoria

Unidad[7].Tau = TAU;
Unidad[7].Kappa = KAPPA;
Unidad[7].Alpha = ALPHA;
Unidad[7].Beta = BETA;
Unidad[7].LogisSigma = LOGISSIGMA;
Unidad[7].LogisMu = LOGISMU;

Unidad[7].Numero_Conexiones_Entrantes = 3;

Unidad[7].Unidad_Presinaptica[1] = 4;
Unidad[7].Peso_Presinaptico[1] = PESOINOCENTE;
Unidad[7].Unidad_Presinaptica[2] = 5;
Unidad[7].Peso_Presinaptico[2] = PESOINOCENTE;
Unidad[7].Unidad_Presinaptica[3] = 6;
Unidad[7].Peso_Presinaptico[3] = PESOINOCENTE;

// Unidad 8 MA

Unidad[8].Numero_Unidad = 8;
Unidad[8].Tipo_Unidad = 2;       // 1 SA, 2 MA, 3 CA1, 4 VTA, 5 R, 6
CR/UR
Unidad[8].Tipo_Activacion = 1;  // 0 Inhibitoria, 1 Excitatoria
```

```
Unidad[8].Tau = TAU;
Unidad[8].Kappa = KAPPA;
Unidad[8].Alpha = ALPHA;
Unidad[8].Beta = BETA;
Unidad[8].LogisSigma = LOGISSIGMA;
Unidad[8].LogisMu = LOGISMU;

Unidad[8].Numero_Conexiones_Entrantes = 3;

Unidad[8].Unidad_Presinaptica[1] = 4;
Unidad[8].Peso_Presinaptico[1] = PESOINOCENTE;
Unidad[8].Unidad_Presinaptica[2] = 5;
Unidad[8].Peso_Presinaptico[2] = PESOINOCENTE;
Unidad[8].Unidad_Presinaptica[3] = 6;
Unidad[8].Peso_Presinaptico[3] = PESOINOCENTE;

// Unidad 9 MA

Unidad[9].Numero_Unidad = 9;
Unidad[9].Tipo_Unidad = 2;       // 1 SA, 2 MA, 3 CA1, 4 VTA, 5 R, 6
CR/UR
Unidad[9].Tipo_Activacion = 1;  // 0 Inhibitoria, 1 Excitatoria

Unidad[9].Tau = TAU;
Unidad[9].Kappa = KAPPA;
Unidad[9].Alpha = ALPHA;
Unidad[9].Beta = BETA;
Unidad[9].LogisSigma = LOGISSIGMA;
Unidad[9].LogisMu = LOGISMU;

Unidad[9].Numero_Conexiones_Entrantes = 3;

Unidad[9].Unidad_Presinaptica[1] = 4;
Unidad[9].Peso_Presinaptico[1] = PESOINOCENTE;
Unidad[9].Unidad_Presinaptica[2] = 5;
Unidad[9].Peso_Presinaptico[2] = PESOINOCENTE;
Unidad[9].Unidad_Presinaptica[3] = 6;
Unidad[9].Peso_Presinaptico[3] = PESOINOCENTE;

// Unidad 10 MA

Unidad[10].Numero_Unidad = 10;
Unidad[10].Tipo_Unidad = 2;       // 1 SA, 2 MA, 3 CA1, 4 VTA, 5 R, 6
CR/UR
Unidad[10].Tipo_Activacion = 1;  // 0 Inhibitoria, 1 Excitatoria

Unidad[10].Tau = TAU;
Unidad[10].Kappa = KAPPA;
Unidad[10].Alpha = ALPHA;
Unidad[10].Beta = BETA;
Unidad[10].LogisSigma = LOGISSIGMA;
Unidad[10].LogisMu = LOGISMU;

Unidad[10].Numero_Conexiones_Entrantes = 3;

Unidad[10].Unidad_Presinaptica[1] = 4;
Unidad[10].Peso_Presinaptico[1] = PESOINOCENTE;
Unidad[10].Unidad_Presinaptica[2] = 5;
Unidad[10].Peso_Presinaptico[2] = PESOINOCENTE;
Unidad[10].Unidad_Presinaptica[3] = 6;
Unidad[10].Peso_Presinaptico[3] = PESOINOCENTE;
```

```
// Unidad 11 VTA

Unidad[11].Numero_Unidad = 11;
Unidad[11].Tipo_Unidad = 4;       // 1 SA, 2 MA, 3 CA1, 4 VTA, 5 R, 6
CR/UR
Unidad[11].Tipo_Activacion = 1;  // 0 Inhibitoria, 1 Excitatoria

Unidad[11].Tau = TAU;
Unidad[11].Kappa = KAPPA;
Unidad[11].Alpha = ALPHA;
Unidad[11].Beta = BETA;
Unidad[11].LogisSigma = LOGISSIGMA;
Unidad[11].LogisMu = LOGISMU;

Unidad[11].Numero_Conexiones_Entrantes = 3;

Unidad[11].Unidad_Presinaptica[1] = 8;
Unidad[11].Peso_Presinaptico[1] = PESOINOCENTE;
Unidad[11].Unidad_Presinaptica[2] = 9;
Unidad[11].Peso_Presinaptico[2] = PESOINOCENTE;
Unidad[11].Unidad_Presinaptica[3] = 10;
Unidad[11].Peso_Presinaptico[3] = PESOINOCENTE;

// Unidad 12 R

Unidad[12].Numero_Unidad = 12;
Unidad[12].Tipo_Unidad = 5;       // 1 SA, 2 MA, 3 CA1, 4 VTA, 5 R, 6
CR/UR
Unidad[12].Tipo_Activacion = 1;  // 0 Inhibitoria, 1 Excitatoria

Unidad[12].Tau = TAU;
Unidad[12].Kappa = KAPPA;
Unidad[12].Alpha = ALPHA;
Unidad[12].Beta = BETA;
Unidad[12].LogisSigma = LOGISSIGMA;
Unidad[12].LogisMu = LOGISMU;

Unidad[12].Numero_Conexiones_Entrantes = 3;

Unidad[12].Unidad_Presinaptica[1] = 8;
Unidad[12].Peso_Presinaptico[1] = PESOINOCENTE;
Unidad[12].Unidad_Presinaptica[2] = 9;
Unidad[12].Peso_Presinaptico[2] = PESOINOCENTE;
Unidad[12].Unidad_Presinaptica[3] = 10;
Unidad[12].Peso_Presinaptico[3] = PESOINOCENTE;

// Unidad 13 CR/UR

Unidad[13].Numero_Unidad = 13;
Unidad[13].Tipo_Unidad = 6;       // 1 SA, 2 MA, 3 CA1, 4 VTA, 5 R, 6
CR/UR
Unidad[13].Tipo_Activacion = 1;  // 0 Inhibitoria, 1 Excitatoria

Unidad[13].Tau = TAU;
Unidad[13].Kappa = KAPPA;
Unidad[13].Alpha = ALPHA;
Unidad[13].Beta = BETA;
Unidad[13].LogisSigma = LOGISSIGMA;
Unidad[13].LogisMu = LOGISMU;

Unidad[13].Numero_Conexiones_Entrantes = 3;

Unidad[13].Unidad_Presinaptica[1] = 8;
```

```
            Unidad[13].Peso_Presinaptico[1] = PESOINOCENTE;
            Unidad[13].Unidad_Presinaptica[2] = 9;
            Unidad[13].Peso_Presinaptico[2] = PESOINOCENTE;
            Unidad[13].Unidad_Presinaptica[3] = 10;
            Unidad[13].Peso_Presinaptico[3] = PESOINOCENTE;
        }

        public double Sum_Exc_Wgts(int Cur_NPE)
        {
            // Sumo los pesos entrantes de las Neuronas presinapticas con
conexion excitatoria
            double Tot;
            Tot = 0.0;
            // Si la conexion entrante es exitatoria Tot = Tot + Peso entrante de
conexion excitatoria

            for (int i = 1; i <= Unidad[Cur_NPE].Numero_Conexiones_Entrantes;
i++)
            {
                if
(Unidad[Unidad[Cur_NPE].Unidad_Presinaptica[i]].Tipo_Activacion == 1)
                {
                    Tot = Tot + Unidad[Cur_NPE].Peso_Presinaptico[i];
                }
                else
                {
                    Tot = 0.0;
                }
            }
            return Tot;
        }

        public double Sum_Inh_Wgts(int Cur_NPE)
        {
            // Sumo los pesos entrantes de las Neuronas presinapticas con
conexion inhibitoria
            double Tot;
            Tot = 0.0;
            // Si la conexion entrante es inhibitoria Tot = Tot + Peso entrante
de conexion inhibitoria

            for (int i = 1; i <= Unidad[Cur_NPE].Numero_Conexiones_Entrantes;
i++)
            {
                if
(Unidad[Unidad[Cur_NPE].Unidad_Presinaptica[i]].Tipo_Activacion == 0)
                {
                    Tot = Tot + Unidad[Cur_NPE].Peso_Presinaptico[i];
                }
                else
                {
                    Tot = 0.0;
                }
            }
            return Tot;
        }

        public double Sum_Exc_ActWgt(int Cur_NPE)
        {
            // Sumo los pesos excitatorios por la Actiwcion excitatoria
presinapticas
            double Tot;
```

```
            // Si la conexion entrante es exitatoria Tot = Tot + Peso entrante *
Activacion Excitatorias
            Tot = 0.0;

            for (int i = 1; i <= Unidad[Cur_NPE].Numero_Conexiones_Entantes;
i++)
            {
                if
(Unidad[Unidad[Cur_NPE].Unidad_Presinaptica[i]].Tipo_Activacion == 1)
                {
                    Tot = Tot +

Unidad[Unidad[Cur_NPE].Unidad_Presinaptica[i]].Valor_Activacion *
                        Unidad[Cur_NPE].Peso_Presinaptico[i];
                }
                else
                {
                    Tot = 0.0;
                }
            }
            return Tot;
        }

        public double Sum_Inh_ActWgt(int Cur_NPE)
        {
            // Sumo los pesos Inhibitorios por la Activacion inhibitoria
presinapticas
            double Tot;
            // Si la conexion entrante es inhibitoria Tot = Tot + Peso entrante *
Activacion Inhibitoria
            Tot = 0.0;

            for (int i = 1; i <= Unidad[Cur_NPE].Numero_Conexiones_Entrantes;
i++)
            {
                if
(Unidad[Unidad[Cur_NPE].Unidad_Presinaptica[i]].Tipo_Activacion == 0)
                {
                    Tot = Tot +

Unidad[Unidad[Cur_NPE].Unidad_Presinaptica[i]].Valor_Activacion *
                        Unidad[Cur_NPE].Peso_Presinaptico[i];
                }
                else
                {
                    Tot = 0.0;
                }
            }
            return Tot;
        }

        public double Logistic(double x, int Cur_NPE)
        {
            double Producto;

            Producto = 1 / (1 + Math.Exp((-x + Unidad[Cur_NPE].LogisMu) /
Unidad[Cur_NPE].LogisSigma));

            if ((-x + Unidad[Cur_NPE].LogisMu) / Unidad[Cur_NPE].LogisSigma < -(
11000))
            {
                return 1.0;
            }
```

```
            else
            {
                if ((-x + Unidad[Cur_NPE].LogisMu) / Unidad[Cur_NPE].LogisSigma >
11000)
                {
                    return 0.0;
                }
                else
                {
                    return Producto;
                }
            } // Fin del if ((-x + Unidad[Cur_NPE].LogisMu) /
Unidad[Cur_NPE].LogisSigma < (-11000))
        } // Fin de la función Logistic


        public double Calcula_p_ipsp(int Cur_NPE)
        {
            if (Sum_Inh_ActWgt(Cur_NPE) > 0.0)
            {
                return Logistic(Sum_Inh_ActWgt(Cur_NPE), Cur_NPE);
            }
            else
            {
                return 0.0;
            }
        }

        public double Activation(int Cur_NPE)
        {
            double thr, p_epsp_t_1, exc, inh;

            p_epsp_t_1 = Unidad[Cur_NPE].p_epsp_t;

            exc = Sum_Exc_ActWgt(Cur_NPE);
            Unidad[Cur_NPE].p_epsp_t = Logistic(exc, Cur_NPE);
            inh = Calcula_p_ipsp(Cur_NPE);

            thr = RandomGaussian(Thr_mu, Thr_sigma);    // Gauss_Rnd; Calculo de
theta

            if (Unidad[Cur_NPE].p_epsp_t > inh)
            {
                if (Unidad[Cur_NPE].p_epsp_t >= thr)
                    {
                    return (Unidad[Cur_NPE].p_epsp_t + Unidad[Cur_NPE].Tau *
p_epsp_t_1 * (1 - Unidad[Cur_NPE].p_epsp_t)) - inh;
                    }
                else
                    {
                    if ((Unidad[Cur_NPE].Kappa * Unidad[Cur_NPE].Valor_Activacion
< Unidad[Cur_NPE].Valor_Activacion)
                            && (Unidad[Cur_NPE].Valor_Activacion- =
Unidad[Cur_NPE].Kappa * Unidad[Cur_NPE].Valor_Activacion > inh))
                        {
                        return (Unidad[Cur_NPE].Valor_Activacion-
Unidad[Cur_NPE].Kappa * Unidad[Cur_NPE].Valor_Activacion)- inh;
                        }
                    else
                        {
                        return 0.0;
                        }
                    }
```

```
            }
            else  // Si (Unidad[Cur_NPE].p_epsp_t =< inh)
            {
                return 0.0;
            } // Fin del if (Unidad[Cur_NPE].p_epsp_t > inh)
        } // Fin de la funcion Activation(int Cur_NPE)

        private void button1_Click_1(object sender, EventArgs e) // Crear la red
        {
            Crear_Red();
            Cargar_Red();

            MessageBox.Show("Red Creada y cargada con valores iniciales por
Default!", "DBP5000 16 julio 2006");

        }

        private void button3_Click(object sender, EventArgs e) // Un paso
adelante
        {
            UpdateNet();
        }

        public void Compute_CA1_Discrepancies()
        {
           int Neurona;
           Tot_CA1_d_t = 0.0;

           for (Neurona = 0; Neurona< Numero_Unidades; Neurona++)
           {
               if( Unidad[Neurona].Tipo_Unidad == 3) // Si la unidad es CA1
               {
                   Tot_CA1_d_t = Tot_CA1_d_t +
Math.Abs(Unidad[Neurona].Valor_Activacion
-
Unidad[Neurona].Valor_Activacion_Tmenos1);
               }
           }
           Avg_CA1_d_t = Avg_CA1_d(); // CA1 diffuse signal
        }

        public void Compute_VTA_Discrepancies()
        {
            int Neurona;
            Tot_VTA_d_t = 0.0;

            for (Neurona = 0; Neurona< Numero_Unidades; Neurona++)
           {
               if( Unidad[Neurona].Tipo_Unidad == 4) // Si la unidad es VTA
               {
                   Tot_VTA_d_t = Tot_VTA_d_t + (Unidad[Neurona].Valor_Activacion
-
Unidad[Neurona].Valor_Activacion_Tmenos1);
               }
           }
           Avg_VTA_d_t = Avg_VTA_d(); // VTA diffuse signal
        }

        public double Avg_CA1_d() // Señal de refuerzo para los NPES
polisensoriales
        {
            if (Tot_CA1_NPEs > 0)
            {
```

```
                return (Tot_CA1_d_t / Tot_CA1_NPEs);
            }
            else
            {
                return (0.0);
            }
        }

        public double Avg_VTA_d() // Señal de refuerzo para los NPES
polisensoriales
        {
            if (Tot_VTA_NPEs > 0)
            {
                return (Tot_VTA_d_t / Tot_VTA_NPEs);
            }
            else
            {
                return (0.0);
            }
        }

        public void Amplify_CA1_d()
        {
         if ((Avg_CA1_d_t > 0.0) && (Avg_VTA_d_t > 0.0))
            {
             Avg_CA1_d_t = Avg_CA1_d_t + (Avg_VTA_d_t * (1- Avg_CA1_d_t));
            }
        }

        public void UpdateActivations()
        {
            int TimeStep;
            int Neurona;

            for (TimeStep = 0; TimeStep < Numero_Time_Steps; TimeStep++)
                {
                // Meto los valores del array de entrada como activacion de las
unidades de entrada
                Unidad[0].Valor_Activacion = Entrada[0, TimeStep];
                Registro_de_Activaciones[0, TimeStep, Ensayo] =
Unidad[0].Valor_Activacion;
                Unidad[1].Valor_Activacion = Entrada[1, TimeStep];
                Registro_de_Activaciones[1, TimeStep, Ensayo] =
Unidad[1].Valor_Activacion;
                Unidad[2].Valor_Activacion = Entrada[2, TimeStep];
                Registro_de_Activaciones[2, TimeStep, Ensayo] =
Unidad[2].Valor_Activacion;
                Unidad[3].Valor_Activacion = Entrada[3, TimeStep];
                Registro_de_Activaciones[3, TimeStep, Ensayo] =
Unidad[3].Valor_Activacion;

                // Creo la lista con las neuronas elegidas aleatoriamente
                create_random(LimiteBajoNeuronaElegida, (Numero_Unidades-1),
GenerarlistaNeuronas);

                for (Neurona = 0; Neurona < ArrayNeuronasElegidas.Length;
Neurona++)
                    {
                        if (ArrayNeuronasElegidas[Neurona] >
Numero_Unidades_Entrada)
                        {
                            // Paso las activaciones de t a t menos 1
```

```
Unidad[ArrayNeuronasElegidas[Neurona]].Valor_Activacion_Tmenos1 =
Unidad[ArrayNeuronasElegidas[Neurona]].Valor_Activacion;
                            // Calculo las nuevas activaciones

Unidad[ArrayNeuronasElegidas[Neurona]].Valor_Activacion =
Activation(ArrayNeuronasElegidas[Neurona]);


Registro_de_Activaciones[ArrayNeuronasElegidas[Neurona], TimeStep, Ensayo] =
Unidad[ArrayNeuronasElegidas[Neurona]].ValorActivacion;
                            // Si la Unidad es VTA o CRUR y el TimeStep es
Reforzado
                            if
(((Unidad[ArrayNeuronasElegidas[Neurona]].Tipo_Unidad == 4) ||
(Unidad[ArrayNeuronasElegidas[Neurona]].Tipo_Unidad == 6))
                                && (Unidad[0].Valor_Activacion > 0.0))
                            {

Unidad[ArrayNeuronasElegidas[Neurona]].Valor_Activacion =
Unidad[0].Valor_Activacion;

Registro_de_Activaciones[ArrayNeuronasElegidas[Neurona], TimeStep, Ensayo] =
Unidad[ArrayNeuronasElegidas[Neurona]].Valor_Activacion;
                            }
                        } // fin del if
                    } // Fin del for Neruona = 0
                } // Fin del for de timesteps
            // MessageBox.Show("Paso Terminado!");
        }

        public void Str_Wgts(double d, int Cur_NPE)
        {
            double Tot_Wgt, N_t, pi_t, dWij_t;
            int dd;

            dWij_t = 0.0;

            if (Sum_Exc_ActWgt(Cur_NPE) > 0) // Si la suma de los pesos por las
activaciones entrantes es mayor que 0
                {
                Tot_Wgt = Sum_Exc_Wgts(Cur_NPE);
                N_t = Nr - Tot_Wgt;

                create_random(LimiteBajoConexionElegida,
Unidad[Cur_NPE].Numero_Conexiones_Entrantes, GenerarlistaConexiones);

                for (dd = 0; dd < ArrayConexionesElegidas.Length; dd++)
                    {
                    if
(Unidad[Unidad[Cur_NPE].Unidad_Presinaptica[ArrayConexionesElegidas[dd]]].Tipo_Ac
tivacion == 1)
                        {
                        pi_t =
(Unidad[Unidad[Cur_NPE].Unidad_Presinaptica[ArrayConexionesElegidas[dd]]].Valor_A
ctivacion *

Unidad[Cur_NPE].Peso_Presinaptico[ArrayConexionesElegidas[dd]]) /
Sum_Exc_ActWgt(Cur_NPE);

                        dWij_t = Unidad[Cur_NPE].Alpha *
Unidad[Cur_NPE].Valor_Activacion * d * pi_t * N_t;
```

```
Unidad[Cur_NPE].Peso_Presinaptico[ArrayConexionesElegidas[dd]] =
Unidad[Cur_NPE].Peso_Presinaptico[ArrayConexionesElegidas[dd]] + dWij_t;
                        }

                    } // Fin del for
                } // Fin del if (Sum_Exc_ActWgt(Cur_NPE) > 0)

            dWij_t = 0.0;

            if (Sum_Inh_ActWgt(Cur_NPE) > 0) // Si la suma de los pesos por las
activaciones entrantes es mayor que 0
            {
                Tot_Wgt = Sum_Inh_Wgts(Cur_NPE);
                N_t = Nr - Tot_Wgt;

                create_random(LimiteBajoConexionElegida,
Unidad[Cur_NPE].Numero_Conexiones_Entrantes, GenerarlistaConexiones);

                for (dd = 0; dd < ArrayConexionesElegidas.Lagth; dd++)
                {
                    if
(Unidad[Unidad[Cur_NPE].Unidad_Presinaptica[ArrayConexionesElegidas[dd]]].Tipo_Ac
tivacion == 0)
                        {
                        pi_t =
(Unidad[Unidad[Cur_NPE].Unidad_Presinaptica[ArrayConexionesElegidas[dd]]].Valor_A
ctivacion *

Unidad[Cur_NPE].Peso_Presinaptico[ArrayConexionesElegidas[dd]]) /
Sum_Inh_ActWgt(Cur_NPE);

                        dWij_t = Unidad[Cur_NPE].Alpha *
Unidad[Cur_NPE].Valor_Activacion * d* pi_t * N_t;

Unidad[Cur_NPE].Peso_Presinaptico[ArrayConexionesElegidas[dd]] =

Unidad[Cur_NPE].Peso_Presinaptico[ArrayConexionesElegidas[dd]] + dWij_t;
                        }
                } // Fin del for
            } // Fin del if
        } // Fin de la función Str_Wgts(double d, int Cur_NPE)

        public void Weak_Wgts(double d, int Cur_NPE)
        {
            double dWij_t;
            int dd;

            dWij_t = 0.0;

            create_random(LimiteBajoConexionElegida,
Unidad[Cur_NPE].Numero_Conexiones_Entrantes, GenerarlistaConexiones);

            for (dd = 0; dd < ArrayConexionesElegidas.Length; dd++)
                {
                if
(Unidad[Unidad[Cur_NPE].Unidad_Presinaptica[ArrayConexionesElegidas[dd]]].Tipo_Ac
tivacion == 1)
                    {
                    dWij_t = Unidad[Cur_NPE].Beta *

Unidad[Unidad[Cur_NPE].Unidad_Presinaptica[ArrayConexionesElegidas[dd]]].Valor_Ac
tivacion *
```

```
                        Unidad[Cur_NPE].Valor_Activacion *
Unidad[Cur_NPE].Peso_Presinaptico[ArrayConexionesElegidas[dd]];

Unidad[Cur_NPE].Peso_Presinaptico[ArrayConexionesElegidas[dd]] =
Unidad[Cur_NPE].Peso_Presinaptico[ArrayConexionesElegidas[dd]]- dWij_t;
                    }
                } // Fin del for
        } // Fin del Weak_Wgts(double d, int Cur_NPE)

        public void Update_Wgts()
        {
            int Neurona;

            create_random(LimiteBajoNeuronaElegida, (Numero_Unidades-1),
GenerarlistaNeuronas);

            for (Neurona = 0; Neurona < ArrayNeuronasElegidas.Length; Neurona++)
            {
                if (ArrayNeuronasElegidas[Neurona] > Numero_Unidades_Entrada)
                {
                    if ((Unidad[ArrayNeuronasElegidas[Neurona]].Tipo_Unidad == 1)
||
                        (Unidad[ArrayNeuronasElegidas[Neurona]].Tipo_Unidad ==
3))
                        {
                            if (Avg_CA1_d_t >= d_thr)
                            {
                                Str_Wgts(Avg_CA1_d_t,
ArrayNeuronasElegidas[Neurona]); // increment weights
                            }
                            else
                            {
                                Weak_Wgts(Avg_CA1_d_t,
ArrayNeuronasElegidas[Neurona]); // decrement weights
                            }
                        }
                    else
                        {
                            if (Avg_VTA_d_t >= d_thr)
                            {
                                Str_Wgts(Avg_VTA_d_t,
ArrayNeuronasElegidas[Neurona]); // increment weights
                            }
                            else
                            {
                                Weak_Wgts(Avg_VTA_d_t,
ArrayNeuronasElegidas[Neurona]); // decrement weights
                            }
                        }
                } // Fin del if array neuronas elegidas == 3
            } // Fin del for neurona
        } // Fin de la funcion update weights

        public void Reset_Acts()
        {
            int Neurona;

            for (Neurona = (Numero_Unidades_Entrada + 1); Neurona <
Numero_Unidades; Neurona++)
            {
```

```
                Unidad[Neurona].Valor_Activacion = (Logistic(0, Neurona));
            }
        }

        public void UpdateNet()
        {
            for (Ensayo = 0; Ensayo < Numero_Ensayos; Ensayo++)
            {
                UpdateActivations();
                Compute_CA1_Discrepancies();
                Compute_VTA_Discrepancies();
                Amplify_CA1_d();
                if (checkBox1.Checked == true) // Si el Check de aprendizaje esta
activado
                {
                    Update_Wgts();
                }
                Reset_Acts();
            }

            MessageBox.Show("Corrida Terminada!");

        }

        private void button2_Click(object sender, EventArgs e) // Guardar en
Archivo
        {
            TextWriter tw0 = new StreamWriter("Act_0.xls");
            TextWriter tw1 = new StreamWriter("Act_1.xls");
            TextWriter tw2 = new StreamWriter("Act_2.xls");
            TextWriter tw3 = new StreamWriter("Act_3.xls");
            TextWriter tw4 = new StreamWriter("Act_4.xls");
            TextWriter tw5 = new StreamWriter("Act_5.xls");
            TextWriter tw6 = new StreamWriter("Act_6.xls");
            TextWriter tw7 = new StreamWriter("Act_7.xls");
            TextWriter tw8 = new StreamWriter("Act_8.xls");
            TextWriter tw9 = new StreamWriter("Act_9.xls");
            TextWriter tw10 = new StreamWriter("Act_10.xls");
            TextWriter tw11 = new StreamWriter("Act_11.xls");
            TextWriter tw12 = new StreamWriter("Act_12.xls");
            TextWriter tw13 = new StreamWriter("Act_13.xls");
            TextWriter tw13ts7 = new StreamWriter("Act_13_ts7.xls");


            int TimeStep, Ensayos;

            for (Ensayos = 0; Ensayos < Numero_Ensayos; Ensayos++)
            {
                for (TimeStep = 0; TimeStep < Numero_Time_Steps; TimeStep++)
                {
                  tw0.WriteLine(Registro_de_Activaciones[0, TimeStep, Ensayos]);
                  tw1.WriteLine(Registro_de_Activaciones[1, TimeStep, Ensayos]);
                  tw2.WriteLine(Registro_de_Activaciones[2, TimeStep, Ensayos]);
                  tw3.WriteLine(Registro_de_Activaciones[3, TimeStep, Ensayos]);
                  tw4.WriteLine(Registro_de_Activaciones[4, TimeStep, Ensayos]);
                  tw5.WriteLine(Registro_de_Activaciones[5, TimeStep, Ensayos]);
                  tw6.WriteLine(Registro_de_Activaciones[6, TimeStep, Ensayos]);
                  tw7.WriteLine(Registro_de_Activaciones[7, TimeStep, Ensayos]);
                  tw8.WriteLine(Registro_de_Activaciones[8, TimeStep, Ensayos]);
                  tw9.WriteLine(Registro_de_Activaciones[9, TimeStep, Ensayos]);
                 tw10.WriteLine(Registro_de_Activaciones[10, TimeStep, Ensayos]);
                 tw11.WriteLine(Registro_de_Activaciones[11, TimeStep,Ensayos]);
                 tw12.WriteLine(Registro_de_Activaciones[12, TimeStep, Ensayos]);
```

```
                tw13.WriteLine(Registro_de_Activaciones[13, TimeStep, Ensayos]);
                    if (TimeStep == 6)
                    {
              tw13ts7.WriteLine(Registro_de_Activaciones[13, TimeStep,Ensayos]);
                    }
                }

            }


            tw0.Close();
            tw1.Close();
            tw2.Close();
            tw3.Close();
            tw4.Close();
            tw5.Close();
            tw6.Close();
            tw7.Close();
            tw8.Close();
            tw9.Close();
            tw10.Close();
            tw11.Close();
            tw12.Close();
            tw13.Close();
            tw13ts7.Close();

            MessageBox.Show("Archivo Guardado!");
        }


        // Generaciones aleatorias, de listas y de números aleatorios gaussianos
        public void create_random(int aaa, int bbb, int ccc)  //1,13,1,neuronas
1,2,2,Conexiones Nueva Version
        {
            // En los arrays respectivos ArrayNeuronasElegidas los elementos van
de 0 hasta bbb-1
            // Esta función requiere que se use el componente using
System.Collections;

            if (ccc == GenerarlistaNeuronas) // Si la seleccion es par generar
una lista de neuronas
            {
                ArrayNeuronasElegidas = new int[bbb];

                ArrayList lista_de_neuronas = new ArrayList();
                for (int i = aaa; (i <= bbb); i++)
                {
                    // Add the numbers to the collection.
                    lista_de_neuronas.Add(i);
                }
                Random rand_N = new Random();
                int index_N;
                object item_N;
                int a_N = 0; // Contador para almacenar numeros elegidos en el
array
                // Display the items in random order.
                while ((lista_de_neuronas.Count > 0))
                {
                    // Choose a random index.
                    index_N = rand_N.Next(0, lista_de_neuronas.Count);
                    // Get the item at that index.
                    item_N = lista_de_neuronas[index_N];
                    // Remove the item so that it cannot be chosen again.
```

```
                    lista_de_neuronas.RemoveAt(indexN);
                    // Display the item.
                    // MessageBox.Show(item.ToString());
                    ArrayNeuronasElegidas[a_N] =
Convert.ToInt32(item_N.ToString());
                    a_N = a_N + 1; // incremento el contador en 1
                } // Fin del while
            } // Fin del If ccc = 1
            if (ccc == GenerarlistaConexiones) // Si la seleccion es para generar
una lista de conexiones
            {
                ArrayConexionesElegidas = new int[bbb];

                ArrayList lista_de_Conexiones = new ArrayList();
                for (int i = aaa; (i <= bbb); i++)
                {
                    // Add the numbers to the collection.
                    lista_de_Conexiones.Add(i);
                }
                Random rand_C = new Random();
                int index_C;
                object item;
                int a_C = 0; // Contador para almacenar numeros elegidos en el
array
                // Display the items in random order.
                while ((lista_de_Conexiones.Count > 0))
                {
                    // Choose a random index.
                    index_C = rand_C.Next(0, lista_de_Conexiones.Count);
                    // Get the item at that index.
                    item = lista_de_Conexiones[index_C];
                    // Remove the item so that it cannot be chosen again.
                    lista_de_Conexiones.RemoveAt(index_C);
                    // Display the item.
                    // MessageBox.Show(item.ToString());
                    ArrayConexionesElegidas[a_C] =
Convert.ToInt32(item.ToString());
                    a_C = a_C + 1; // incremento el contador en 1
                } // Fin del while
            } // Fin del If ccc = 2
        }

}
```

ANEXO 2

Código para la generación de números aleatorios gaussianos, realizado por Karl-L. Noell y Helmut Weber en 1989. Traducido a c#, por el autor. La versión original en C++ está disponible fue proporcionada por los autores mediante correo electrónico. Se conservaron los comentarios originales en inglés, a petición de los autores.

```
        public void RandomInitialise(int ij, int kl) // Generacion de aleatorios
gaussianos
        {
        // ****** GENERACION DE NUMEROS ALEATORIOS GAUSSIANOS **********
       /*  RandomGaussian(Thr_mu,Thr_sigma)

       This Random Number Generator is based on the algorithm in a FORTRAN
       version published by George Marsaglia and Arif Zaman, Florida State
       University; ref.: see original comments below.
       At the fhw (Fachhochschule Wiesbaden, W.Germany), Dept. of Computer
       Science, we have written sources in further languages (C, Modula2
       Turbo-Pascal(3.0, 5.0), Basic and Ada) to get exactly the same test
       results compared with the original FORTRAN version.
       April 1989
       Karl-L. Noell <NOELL@DWIFH1.BITNET>
          and  Helmut  Weber <WEBER@DWIFH1.BITNET>

       This random number generator originally appeared in "Toward a Universal
       Random Number Generator" by George Marsaglia and Arif Zaman.
       Florida State University Report: FSUSCRI-87-50 (1987)
       It was later modified by F. James and published in "A Review of Pseudo
       random Number Generators"
       THIS IS THE BEST KNOWN RANDOM NUMBER GENERATOR AVAILABLE.
       (However, a newly discovered technique can yield
       a period of 10^600. But that is still in the development stage.)
       It passes ALL of the tests for random number generators and has a period
       of 2^144, is completely portable (gives bitidentical results on all
       machines with at least 24-bit mantissas in the floating point
       representation).
       The algorithm is a combination of a Fibonacci sequence (with lags of 97
       and 33, and operation "subtraction plus one, modulone") and an
       "arithmetic sequence" (using subtraction).

       Use IJ = 1802 & KL = 9373 to test the random number generator. The
       subroutine RANMAR should be used to generate 20000 random numbers.
       Then display the next six random numbers generated multiplied by 4096*4096
       If the random number generator is working properly, the random numbers
       should be:
               6533892.0  14220222.0  7275067.0
               6172232.0  8354498.0   10633180.0

       This is the initialization routine for the random number generator.
       NOTE: The seed variables can have values be//tween:    0 <= IJ <= 31328
                                                              0 <= KL <= 30081
       The random number sequences created bythese //     two seeds are of
sufficient
           length to complete an entire calculation with. For example, if sveral
           different groups are working on different parts of the same
calculation,
           each group could be assigned its own IJ seed. This would leave each
group
```

```
        with 30000 choices for the second seed. That is to say, this random
        number generator can create 900 million different subsequences-- with
        each subsequence having a length of approximately 10^30.

    public double[] uR = new double[97];        // para la generacion de
aleatorios gaussianos
    public double cR;                                   // para la
generacion de aleatorios gaussianos
    public double cdR;                                  // para la
generacion de aleatorios gaussianos
    public double cmR;                                  // para la
generacion de aleatorios gaussianos
    public int i97R;                                    // para la
generacion de aleatorios gaussianos
    public int j97R;                                    // para la
generacion de aleatorios gaussianos
    public int testR = 0;                               // para la
generacion de aleatorios gaussianos

    */
        double s, t;
        int ii, i, j, k, l, jj, m;

        /*
           Handle the seed range errors
              First random number seed must be be//    tween 0 and 31328
              Second seed must have a value be//tween 0 and 30081
        */
        if (ij < 0 || ij > 31328 || kl < 0 || kl > 30081)
        {
            ij = 1802;
            kl = 9373;
        }

        i = (ij / 177) % 177 + 2;
        j = (ij % 177) + 2;
        k = (kl / 169) % 178 + 1;
        l = (kl % 169);

        for (ii = 0; ii < 97; ii++)
        {
            s = 0.0;
            t = 0.5;
            for (jj = 0; jj < 24; jj++)
            {
                m = (((i * j) % 179) * k) % 179;
                i = j;
                j = k;
                k = m;
                l = (53 * l + 1) % 169;
                if (((l * m % 64)) >= 32)
                    s += t;
                t *= 0.5;
            }
            uR[ii] = s;
        }

        cR = 362436.0 / 16777216.0;
        cdR = 7654321.0 / 16777216.0;
        cmR = 16777213.0 / 16777216.0;
        i97R = 97;
        j97R = 33;
        testR = 1; // Test = TRUE
```

```
        }
        /*
           This is the random number generator proposed by George Marsaglia in
           Florida State University Report: FSUSCRI-87-50
        */
        public double RandomUniform()// Generacion de aleatorios gaussianos
        {
            double uni;

            /* Make sure the initialisation routine has been called */

            if (testR == 0)
                RandomInitialise(1802, 9373);

            uni = uR[i97R - 1] - uR[j97R - 1];
            if (uni <= 0.0)
                uni++;
            uR[i97R - 1] = uni;
            i97R--;
            if (i97R == 0)
                i97R = 97;
            j97R--;
            if (j97R == 0)
                j97R = 97;
            cR -= cdR;
            if (cR < 0.0)
                cR += cmR;
            uni -= cR;
            if (uni < 0.0)
                uni++;

            return (uni);
        }
        /*
               ALGORITHM 712, COLLECTED ALGORITHMS FROM ACM.
               THIS WORK PUBLISHED IN TRANSACTIONS ON MATHEMATICAL SOF/tWARE,
               VOL. 18, NO. 4, DECEMBER, 1992, PP. 434435.
               The function returns a normally distributed pseuderandom number
               with a given mean and standard devaiation.  Calls are made to a
               function subprogram which must return independent random
               numbers uniform in the interval (0,1).
               The algorithm uses the ratio of uniforms method of A.J. Kinderman
               and J.F. Monahan augmented with quadratic bounding curves.
            */
        double RandomGaussian(double mean, double stddev) // Generacion de
aleatorios gaussianos
        {
            double q, u, v, x, y;

            /*
                Generate P = (u,v) uniform in rect. enclosing acceptance region
              Make sure that any random numbers <= 0 are rejected, since
              gaussian() requires uniforms > 0, but RandomUniform() delivers >=
0.
            */
            do
            {
                u = RandomUniform();
                v = RandomUniform();
                if (u <= 0.0 || v <= 0.0)
                {
                    u = 1.0;
                    v = 1.0;
```

```
                }
                v = 1.7156 * (v - 0.5);

                /*  Evaluate the quadratic form */
                x = u - 0.449871;
                y = Math.Abs(v) + 0.386595;
                q = x * x + y * (0.19600 * y- 0.25472 * x);

                /* Accept P if inside inner ellipse */
                if (q < 0.27597)
                    break;

                /*  Reject P if outside outer ellipse, or outside acceptance
region */
            } while ((q > 0.27846) || (v * v >-4.0 * Math.Log(u) * u * u));

            /*  Return ratio of P's coordinates as the normaldeviate */
            return (mean + stddev * v / u);
        }
        /*
                Return random integer within a range, lower-> upper INCLUSIVE
         */
        int RandomInt(int lower, int upper)// Generacion de aleatorios gaussianos
        {
            return ((int)(RandomUniform() * (upper- lower + 1)) + lower);
        }
        /*
                Return random float within a range, lower-> upper
         */
        double RandomDouble(double lower, double upper) // Generacion de
aleatorios gaussianos
        {
            return ((upper - lower) * RandomUniform() + lower);
        }
        //******* FIN DE LA GENERACION DE NUMEROS ALEATORIOS
GAUSSIANOS*********************
    }
```