ESTUDIO Y APLICACIÓNES DE LA REDES ART
(TEORIA DE LA RESONANCIA ADAPTIVA)

GABRIEL CUBAS GLENN
ORLANDO BORRE DEL RIO

UNIVERSIDAD TECNOLOGICA DE BOLIVAR
FACULTAD DE INGENIERIA ELECTONICA
CARTAGENA DE INDIAS D.T. Y C.
2007

ESTUDIO Y APLICACIÓNES DE LA REDES ART
(TEORIA DE LA RESONANCIA ADAPTIVA)

GABRIEL CUBAS GLENN
ORLANDO BORRE DEL RIO

DIRECTOR
ING. EDUARDO GOMEZ VÀSQUEZ

UNIVERSIDAD TECNOLOGICA DE BOLIVAR
FACULTAD DE INGENIERIA ELECTONICA
CARTAGENA DE INDIAS D.T. Y C.
2007

ESTUDIO Y APLICACIÓNES DE LA REDES ART
(TEORIA DE LA RESONANCIA ADAPTIVA)

GABRIEL CUBAS GLENN
ORLANDO BORRE DEL RIO

Trabajo de grado presentado como requisito para obtener el certificado del Minor en Automatización Industrial

DIRECTOR
ING. EDUARDO GOMEZ VÀSQUEZ
Magíster en Ciencias Computacionales

UNIVERSIDAD TECNOLOGICA DE BOLIVAR
FACULTAD DE INGENIERIA ELECTONICA
CARTAGENA DE INDIAS D.T. Y C.
2007

Nota de aceptación

_____

_____

_____

_____

Jurado

_____

Jurado

_____

Cartagena D..T. Y C., Mayo de 2007

Señores
COMITÉ CURRICULAR
UNIVERSIDAD TECNOLOGICA DE BOLIVAR
La ciudad

Respetados señores:

Con toda la atención me dirijo a ustedes con el fin de presentarles a su consideración, estudio y aprobación la monografía tilutala **ESTUDIO Y APLICACIONES DE LAS REDES ART (TEORIA DE LA RESONANCIA ADAPTATIVA)** como requisito para obtener el titulo de Ingeniero Electrónico

Atentamente,

_____

GABRIEL CUBAS GLENN

Cartagena D..T. Y C., Mayo de 2007

Señores
COMITÉ CURRICULAR
UNIVERSIDAD TECNOLOGICA DE BOLIVAR
La ciudad

Respetados señores:

Con toda la atención me dirijo a ustedes con el fin de presentarles a su consideración, estudio y aprobación la monografía tilutala **ESTUDIO Y APLICACIONES DE LAS REDES ART (TEORIA DE LA RESONANCIA ADAPTATIVA)** como requisito para obtener el titulo de Ingeniero Electrónico

Atentamente,

_____

ORLANDO MARIO BORRE DEL RIO

Cartagena D..T. Y C., Mayo de 2007

Señores
COMITÉ CURRICULAR
UNIVERSIDAD TECNOLOGICA DE BOLIVAR
La ciudad

Respetados señores:

A través de la presente me permito entregar la monografía titulada **ESTUDIO Y APLICACIONES DE LAS REDES ART (TEORIA DE LA RESONANCIA ADAPTATIVA)** par su estudio y evaluación, la cual fue realizada por los estudiantes GABRIEL CUBAS GLENN y ORLANDO MARIO BORRE DEL RIO, de la cual acepto ser su director

Atentamente,

_____

ING. EDUARDO GOMEZ VÀSQUEZ
Magíster en Ciencias Computacionales

AUTORIZACION

Yo, ORLANDO MARIO BORRE DEL RIO, identificado con la cedula de ciudadanía numero 1.128.044.203 de Cartagena, autorizo a al Universidad Tecnológica De Bolívar, par hacer uso de mi trabajo de monografía y publicarlo en el catalogo on-line de la biblioteca.

_____

ORLANDO MARIO BORRE DEL RIO

AUTORIZACION

Yo, GABRIEL CUBAS GLENN, identificado con la cedula de ciudadanía numero 73.208.573 de Cartagena, autorizo a al Universidad Tecnológica De Bolívar, par hacer uso de mi trabajo de monografía y publicarlo en el catalogo on-line de la biblioteca.

_____

GABRIEL CUBAS GLENN

TABLA DE CONTENIDO

.

LISTA DE FIGURAS

- **LISTA DE FIGURAS**

LISTA DE TABLAS

- **LISTA DE TABLAS**

GLOSARIO

Pesos sinápticos: Representa la intensidad de la interacción entre cada neurona presináptica *j* y la neurona postsináptica *i*

Lógica difusa: La lógica borrosa o difusa se basa en lo relativo de lo observado. Este tipo de lógica toma dos valores aleatorios, pero contextualizados y referidos entre sí, En la vida real se puede entender con que la lógica clásica sólo definiría si la persona es alta o no, en difusa se puede decir si es muy alta, muy baja etc.

Cluster: Establecimiento de categorías, Categorizar

Iterativo: Posee la cualidad de reiterarse

Mapeo: Hacer un mapa de las características mas relevantes de los datos de entrada

Plasticidad: poder aprender nuevos patrones

Estabilidad: poder retener nuevos patrones.

Bottom up: Activación de la entrada

Top – Down: Comparación entre F_1 Y F_2

Memoria a corto plazo STM (short-term memory): Son los patrones de actividad que se desarrollan arriba de los nodos en la dos capas del sistema de atención, solo existen en asociación con una sola aplicación de la neurona de entrada

Memoria de largo plazo LTM (large-term memory): Son los pesos asociados con el Bottom up y el Top - Down entre F_1 Y F_2 , codifican la información que queda por un largo periodo de tiempo

INTRODUCCION

El cerebro humano ha sido considerado por el hombre como una maquina perfecta con capacidades inimaginables, que si se llegara a imitar alcanzaría grandes aplicaciones en ámbitos económicos, científicos e industriales y además múltiples beneficios en entornos sociales. Los científicos en la necesidad de crear diferentes tipos de mecanismos para la simulación del aprendizaje y razonamiento humano han creado distintos métodos entre lo cuales se encuentran las Redes Neuronales Artificiales.

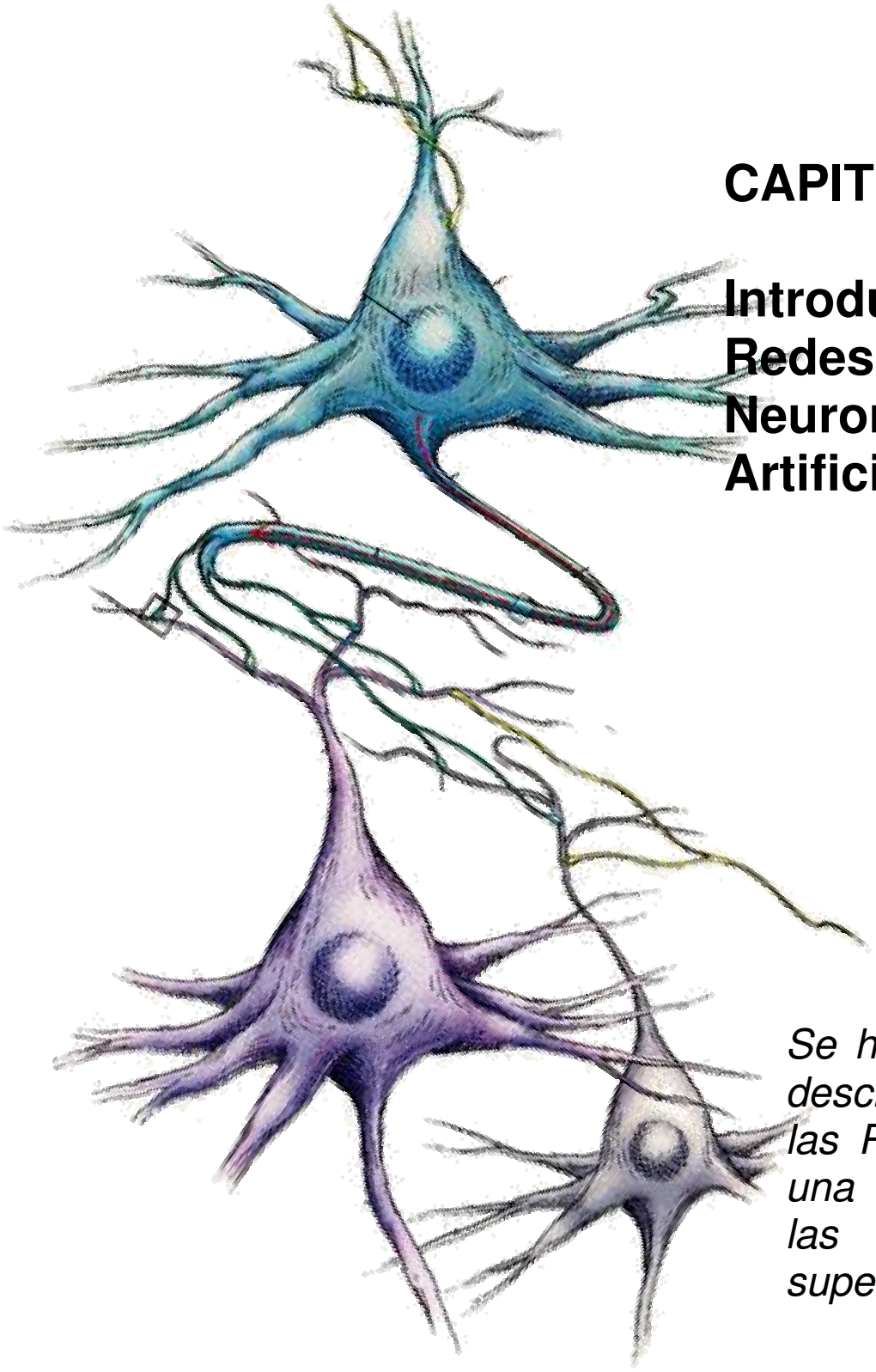
Las Redes Neuronales Artificiales *(RNA)* son hoy en día un método muy utilizado como aprendizaje en el proceso de señales e imágenes. En donde una red para una determinada aplicación presenta una arquitectura muy concreta, que comprende elementos de procesado adaptativo masivo paralelo combinadas con estructuras de interconexión de red jerárquica.

En Muchas arquitecturas de Redes Neuronales, la red debe ser entrenada y si se desea ingresar un patrón distinto de entrada hay que cambiar la red completa, lo cual no simula realmente el comportamiento del cerebro ya que no es autodependiente. Las arquitecturas creadas para tener un aprendizaje no supervisado poseen el problema de no tener un aprendizaje estable

(Dilema de la Estabilidad y Plasticidad). Al hablar de estabilidad nos referimos a poder retener los patrones aprendidos, y a la plasticidad nos referimos a poder aprender nuevos patrones.

El conseguir un modelo de RNA capáz de resolver uno de estos problemas es sencillo, conseguir un modelo que sea capaz de dar respuesta a ambos no lo es. Las redes más conocidas son capaces de aprender como tienen que responder ante unos patrones de entrada, pero, una vez entrenados, el intentar que aprendan nuevos patrones puede suponer el "olvido" de lo aprendido previamente. Debido a esto Gail A. Carpenter y Sthepen Grossberg crearon un tipo de RNA llamada Teoría de la Resonancia Adaptiva (ART), cuyo objetivo es proponer una solución a este dilema mediante un mecanismo de realimentación entre las neuronas competitivas de la capa de salida de una Red Neuronal.

En esta investigación, en el primer capitulo, nos presentara una Introducción a las redes neuronales artificiales, en el segundo capitulo se realiza descripción completa de las redes ART, en el 3 capitulo, se realizan unos ejemplos de aplicaciones de ART en Matlab, y para finalizar, en el cuarto capitulo, se presentaran unas aplicaciones de las redes ART.



CAPITULO 1

Introducción a las Redes Neuronales Artificiales

Se hace una breve descripción sobre las RNA, así como una introducción a las redes no supervisadas.

1. INTRODUCCIÓN A LAS REDES NEURONALES ARTIFICIALES

1.1 QUE SON LA REDES NEURONALES ARTIFICIALES?[1]

Es difícil pensar que los grandes computadores capaces de realizar inmensas cantidades de operaciones en coma flotante por segundo, no puedan entender el significado de las formas visuales o de distinguir entre distintas clases de objetos.

Los sistemas de computación secuencial, son exitosos en la resolución de problemas numéricos, en la manipulación, creación y mantenimiento de bases de datos, en comunicaciones electrónicas, en el procesamiento de textos, gráficos y auto edición, en funciones de control de electrodomésticos (domótica), incluso en ámbitos robustos como lo es la robótica a gran escala para la creación de maquinarias pesada, haciéndolos más eficientes y fáciles de usar, pero definitivamente tienen una gran incapacidad para interpretar el mundo.

En el afán por encontrar un nuevo sistema de tratamiento de la información que permita solucionar problemas cotidianos, tal como lo hace el cerebro humano; debido a que este órgano biológico cuenta con varias

[1] Complementar con la pagina web http://ingenieria.udea.edu.co/investigacion/mecatronica/mectronics/redes.htm

características deseables para cualquier sistema de procesamiento digital, tales como:

- Es robusto y tolerante a fallas, diariamente mueren neuronas sin afectar su desempeño.
- Es flexible, se ajusta a nuevos ambientes por aprendizaje, no hay que programarlo.
- Puede manejar información difusa, con ruido o inconsistente.
- Es altamente paralelo.
- Es pequeño, compacto y consume poca energía.

Se ha desarrollado desde hace más de 30 años la teoría de las Redes Neuronales Artificiales (RNA), las cuales emulan las redes neuronales biológicas, y que se han utilizado para aprender estrategias de solución basadas en ejemplos de comportamiento típico de patrones; estos sistemas no requieren que la tarea a ejecutar se programe, ellos generalizan y aprenden de la experiencia.

Las aplicaciones más exitosas de las RNA son:

- Procesamiento de imágenes y de voz.
- Reconocimiento de patrones.
- Planeamiento.
- Interfaces adaptativas para sistemas Hombre/máquina.

- Predicción.
- Control y optimización.
- Filtrado de señales.

1.2 PRINCIPALES CARACTERISTICAS DE LA RNA

Se denomina *neurona* a un dispositivo simple de cálculo que, a partir de un vector de entrada procedente del exterior o de otras neuronas, proporciona una única respuesta o salida.

Los elementos que constituyen la neurona de etiqueta *i* son los siguientes (*figura 1):*

- Conjunto de **entradas**, $x_j(t)$.
- **Pesos sinápticos** de la neurona *i*, w_{ij} que representa la intensidad de la interacción entre cada neurona presináptica *j* y la neurona postsináptica *i.*
- **Regla de propagación** $\sigma(w_{ij}, x_j(t))$, que proporciona el valor del potencial postsináptico $h_i(t) = \sigma(w_{ij}, x_j(t))$ de la neurona *i* en función de sus pesos y entradas.
- **Función de activación** $fi(a_i(t-1), h_i(t))$, que proporcione el estado de activación actual $a_i(t) = fi(a_i(t-1), h_i(t))$ de la neurona *i*, en función de su estado anterior $a_i(t-1)$ y de su potencial post sináptico actual.

- **Función de salida** $F_i(a_i(t))$**,** que proporciona la salida actual $y_i(t) = F_i(a_i(t))$ de la neurona *i* en función de su estado de activación.

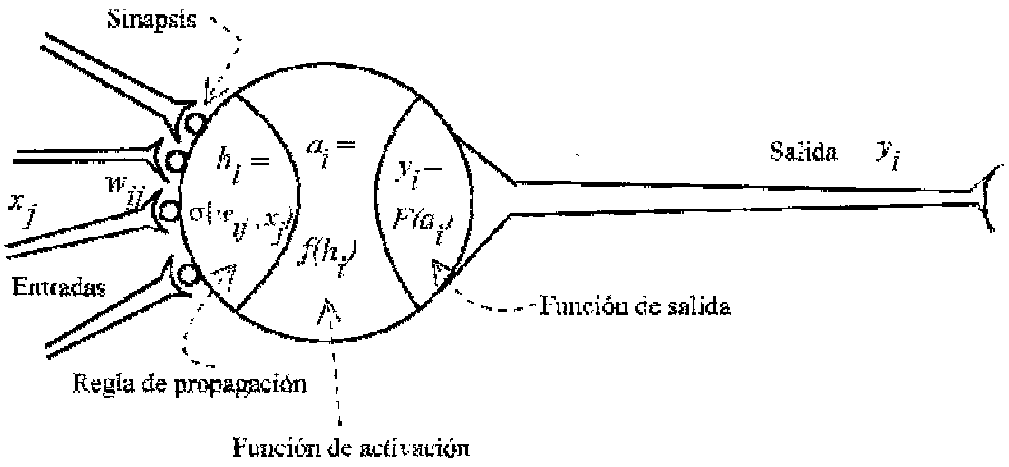


Figura 1. Modelo genérico de una neurona artificial[2].

1.3 ARQUITECTURA DE LAS RNA[3]

Se le denomina arquitectura a la topología, estructura o patrón de conexión de una red neuronal. En una RNA los nodos se conectan por medio de la sinapsis, esta estructura de conexiones sinápticas determina el comportamiento de la red. Las conexiones sinápticas son direccionales (*Figura 2*), es decir, la información solamente puede propagarse en un único sentido (desde la neurona presináptica a la postsináptica).

[2] Tomado de Bibliografía [1]. Capitulo 1, introducción a las RNA

[3] Complementar con la pagina web http://www.monografias.com/trabajos12/redneur/redneur.shtml

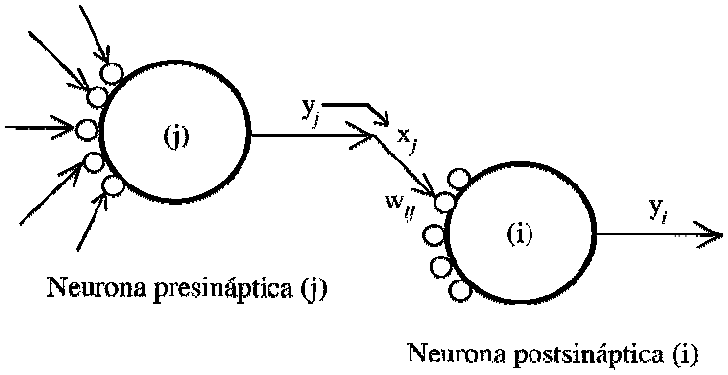


Figura 2. Interacción entre una neurona presináptica y otra postsináptica[4]

En general, las neuronas se suelen agrupar en unidades estructurales que denominaremos capas. Las neuronas de una capa pueden agruparse, a su vez, formando grupos neuronales *(clusters).* Dentro de un grupo, o de una capa si no existe este tipo de agrupación, las neuronas suelen ser del mismo tipo. Finalmente, el conjunto de una o más capas constituye la red neuronal.

Se distinguen tres tipos de capas[5]: de entrada, de salida y ocultas. Una capa de entrada o sensorial está compuesta por neuronas que reciben datos o señales procedentes del entorno (por ejemplo, proporcionados por sensores). Una capa de salida es aquella cuyas neuronas proporcionan la respuesta de la red neuronal (sus neuronas pueden estar conectadas a efectores). Una capa oculta es aquella que no tiene una conexión directa con el entorno, es decir, que no se conecta directamente ni a órganos

[4] Tomado de Bibliografía [1]. Capitulo 1, introducción a las RNA
[5] Complementar con el siguiente articulo http://www.uta.cl/revistas/charlas/volumen16/Indice/Ch-csaavedra.pdf

sensores ni a efectores. Este tipo de capa proporciona a la red neuronal grados de libertad adicionales, gracias a los cuales puede encontrar representaciones internas correspondientes a determinados rasgos del entorno, proporcionando una mayor riqueza computacional.

Las conexiones entre las neuronas pueden ser excitatorias o inhibitorias: un peso sináptico negativo define una conexión inhibitoria, mientras que uno positivo determina una conexión excitatoria. Habitualmente, no se suele definir una conexión como de un tipo o de otro, sino que por medio del aprendizaje se obtiene un valor para el peso, que incluye signo y magnitud.

Por otra parte, se puede distinguir entre conexiones intra-capa e inter-capa. Las conexiones intra-capa, también denominadas laterales, tienen lugar entre las neuronas pertenecientes a una misma capa, mientras que las conexiones inter-capa se producen entre las neuronas de diferentes capas. Existen además conexiones realimentadas, que tienen un sentido contrario al de entrada-salida. En algunos casos puede existir realimentación incluso de una neurona consigo misma.

De acuerdo a distintos conceptos, pueden establecerse diferentes tipos de arquitecturas neuronales. En relación a su estructura en capas se puede hablar de ***Redes Monocapas***, que son aquellas compuestas por una única capa. Las ***Redes Multicapa***, que son aquellas neuronas que se organizan en varias capas. En relación al flujo de datos en la red neuronal, se puede

hablar de ***Redes Unidireccionales***, el cual la información circula en un único sentido, desde la neurona de entrada hacia la de salida. Las ***Redes Recurrentes o realimentadas***, la información puede circular entre las capas en cualquier sentido, incluido el de salida-entrada. Por ultimo, ya que se interpreta la operación de una RNA como la de una *memoria asociativa,* se clasifican debido a la asociación entre la información de entrada y salida las **Redes Autoasociativas,** el cual se dice que si una red es entrenada para que asocie un patrón **A** consigo mismo, entonces pertenece a este tipo. Pero si una red se entrena para que ante la presentación de un patrón **A** responda con otro diferente **B**, se dice que la red es **Heteroasociativas.**

1.4 MODOS DE OPERACIÓN[6]

Claramente se distinguen dos tipos: el *modo recuerdo* o *ejecución*, y el *modo aprendizaje* o *entrenamiento*. Este último es de particular interés, pues una característica fundamental de los RNA es que se trata de sistemas entrenables, capaces de realizar un determinado tipo de procesamiento o cómputo aprendiéndolo a partir de un conjunto de patrones de aprendizaje o ejemplos.

1.4.1 Fase de Aprendizaje. Convergencia

En el contexto de las redes neuronales puede definirse el **aprendizaje** como

[6] Complementar http://www.guru-games.org/people/pedro/aad/ivan_martinez.pdf - Pag 7

el proceso por el que se produce el ajuste de los parámetros libres de la red a partir de un proceso de estimulación por el entorno que rodea la red. El tipo de aprendizaje vendrá determinado por la forma en la que dichos parámetros son adaptados. En la mayor parte de las ocasiones el aprendizaje consiste simplemente en determinar un conjunto de pesos sinápticos que permita a la red realizar correctamente el tipo de procesamiento deseado.

Cuando se construye un sistema neuronal, se parte de un cierto modelo de neurona y de una determinada arquitectura de red, estableciéndose los pesos sinápticos iníciales como nulos o aleatorios. Para que la red resulte operativa es necesario entrenarla, lo que constituye el **modo aprendizaje.** El entrenamiento o aprendizaje se puede llevar a cabo a dos niveles. El más convencional es el de modelado de las sinapsis, que consiste en modificar los pesos sinápticos siguiendo una cierta regla de aprendizaje, construida normalmente a partir de la optimización de una función de error o coste, que mide la eficacia actual de la operación la red.

El proceso de aprendizaje es usualmente **iterativo,** actualizándose los pesos de la manera anterior, una y otra vez, hasta que la red neuronal alcanza el rendimiento deseado. Algunos modelos neuronales incluyen otro nivel en el aprendizaje, la creación o destrucción de neuronas, en el cual se modifica la propia arquitectura de la red.

Los dos tipos básicos de aprendizaje son el supervisado y el no supervisado, cuya distinción proviene en origen del campo del reconocimiento de patrones. Ambas modalidades pretenden estimar funciones entrada/salida multivariable o densidades de probabilidad, pero mientras que en el aprendizaje supervisado se proporciona cierta información sobre estas funciones (como la distribución de las clases, etiquetas de los patrones de entrada o salidas asociadas a cada patrón), en el autoorganizado no se proporciona información alguna. Las reglas de aprendizaje supervisadas suelen ser computacionalmente más complejas, pero también más exactos sus resultados.

Además de las dos formas básicas anteriores pueden distinguirse el aprendizaje híbrido y el reforzado.

El **Aprendizaje Hibrido,** en este caso coexisten en la red los dos tipos básicos de aprendizaje, el supervisado y el no supervisado, los cuales tienen lugar normalmente en distintas capas de neuronas. El modelo de contra-propagación y las RBF son ejemplos de redes que hacen uso de este tipo de aprendizaje.

El **Aprendizaje Reforzado,** Se sitúa a medio camino entre el supervisado y el autoorganizado. Como en el primero de los citados, se emplea información sobre el error cometido, pero en este caso existe una única señal de error, que representa un índice global del rendimiento de la red (solamente le indicamos lo bien o lo mal que está actuando, pero sin

proporcionar más detalles). Como en el caso del no supervisado, no se suministra explícitamente la salida deseada. En ocasiones se denomina *aprendizaje por premio-castigo.*

1.4.2 Fase de Ejecución

La Fase de ejecución consiste generalmente (aunque no en todos los modelos), cuando una vez que el sistema ha sido entrenado, el aprendizaje "se desconecta", por lo que los pesos y la estructura quedan fijos, estando la red neuronal ya dispuesta para procesar datos. Este modo de operación se denomina *modo recuerdo o de ejecución*.

En las redes unidireccionales, ante un patrón de entrada, las neuronas responden proporcionando directamente la salida del sistema. Al no existir bucles de realimentación no existe ningún problema en relación con su estabilidad. Por el contrario, las redes con realimentación son sistemas dinámicos no lineales, que requieren ciertas condiciones para que su respuesta acabe convergiendo a un estado estable o punto fijo. Una serie de teoremas generales indican las condiciones que aseguran la estabilidad de la respuesta en una amplia gama de redes neuronales, bajo determinadas condiciones.

1.5 CLASIFICACION DE LAS RNA[7]

Esta clasificación se hace de acuerdo a:

1.5.1 Arquitectura

Se puede realizar una clasificación de acuerdo a los tipos de redes neuronales existentes

TIPOS DE CONEXIONES		MODELO DE RED
CONEXIONES LATERALES EXPLÍCITAS	CONEXIONES AUTORRE-CURRENTES	BRAIN-SATATE-IN-A-BOX
		ADDITIVE GROSSBERG (AG)
		SHUNTING GROSSBERG (SG)
		OPTIMAL LINEAR ASOCIATIVE MEMORY
	NO AUTO-RECURRENTES	HOPFIELD
		BOL TZMANN MACHINE
		CAUCHY MACHINE
CROSSBAR		LEARNING MATRIX (LM)

Tabla 1. Redes Neuronales Monocapa[8]

[7] Complementar con http://es.tldp.org/Presentaciones/200304curso-glisa/redes_neuronales/curso-glisa-redes_neuronales-html/x84.html y igualmente http://lorien.die.upm.es/insn/docs/capitulo22-RedesNeuronales.pdf

[8] Tomado de bibliografía [2]. Capitulo 2, clasificación de las RNA

N DE CAPAS	TIPOS DE CONEXIONES		MODELOS DE RED
2 CAPAS	CONEXIÓN HACIA ADELANTE FEEFORWARD		ADALINE/MADALINE
			PERCEPTRON
			LINEAR/ASSOC REWAR. PENALTY
			LINEAR ASSOCIATIVE MEMORY
			OPTIMAL LINEAR ASSOC. MEM.
			DRIVE-REINFORCEMENT (DR)
		CONEXIONES LATERALES IMPLICITAS Y AUTORREC.	LEARNING VECTOR QUANTIZER
			TOPOLOGY PRESERVING MAP (TPM)
	CONEXIÓN ADELANTE / ATRÁS FEEDFORWARD / FEEDBACK	SIN CONEXIONES LATERALES	BIDIRECTIONAL ASSOC. MEM. (BAM
			ADAPTIVE BAM.
			TEMPORAL ASSOC. MEMORY (TAM)
			FUZZY ASSOCIATIVE MEMORY (FAM)
		CON CONEXIONES LATERALES Y AUTORREC.	COMPETITIVE ADAPTIVE BAM
			ADAPTIVE RESONANCE THEORY (ART)
3	CONEXIONES HACIA ADELANTE (FEEDFORWARD)	SIN CONEXIONES LATERALES	ADAPTIVE HEURISTIC CRITIC (AHC)
			BOLTZMANN/CAUCHY MACHINE
		CON CONEXIONES LATERALES Y AUTORREC.	COUNTERPROPAGATION
			BOLTZMANN/CAUCHY MACHINE
	CONEXIONES ADELANTE/ATRÁS Y LATERALES		BOLTZMANN/CAUCHY MACHINE
N	CONEXIONES HACIA ADELANTE		BACK-PROPAGATION (BPN)
	FEEDFORWARD-FEEDBACK JERARQUIA DE NIVELES DE CAPAS BIDIRECCIONALES		COGNITRON-NEOCOGNITRON

Tabla 2. Redes Neuronales Multicapa[9]

[9] Tomado de bibliografía [2]. Capitulo 2, clasificación de las RNA

1.5.2 Modo de Operación

TIPO DE APRENDIZAJE NO SUPERVISADO		MODELO DE RED
APRENDIZAJE HEBIANO	OFF LINE	HOPFIELD
		LEARNING MATRIX
		TEMPORAL ASSOC. MEMORY
		LINEAR ASSOCIATIVE MEMORY (LAM)
		OPTIMAL LAM
		DRIVE - REINFORCEMENT
		FUZZY ASSOCIATIVE MEMORY
	ON LINE	ADDITIVE GROSSBERG
		SHUNTING GROSSBERG
		BIDIRECTIONAL ASSOCIATIVE MEMORY (BAM)
		ADAPTIVE BAM
APRENDIZAJE COMPETITIVO/ COOPERATIVO	OFF LINE	LEARNING VECTOR QUANTIZER
		COGNITRON/ NEOCOGNITRON
		TOPOLOGY PRESERVING MAP
	ON	ADAPTIVE RESONANCE THEORY

Tabla 3. Tipo de redes con aprendizaje no supervisado más conocido[10]

[10] Tomado de bibliografía [2]. Capitulo 2, clasificación de las RNA

TIPO DE APRENDIZAJE SUPERVISADO		MODELO DE RED
APRENDIZAJE POR CORRECCIÓN DE ERROR	OFF LINE	PERCEPTRON
		ADALINE/MADALINE
		BACKPROP AGA TION
		BRAIN-ST A TE-IN-A-BOX
		COUNTERPROP AGA TION
APRENDIZAJE POR REFUERZO	ON LINE	LINEAR REWARD PENALTY
		ASSOCIATIVE REW. PENALTY
		ADAPTIE HEURISTIC CRITIC
APRENDIZAJE ESTOCÁSTICO	OFF LINE	BOLTZMANN MACHINE
		CAUCHY MACHINE

Tabla 4. Tipo de redes con aprendizaje supervisado más conocido[11]

[11] Tomado de bibliografía [2]. Capitulo 2, clasificación de las RNA

1.5.3 Tipo de Asociación Entre la Información de Entrada y Salida

REDES HETEROASOCIATIVAS	REDES AUTOASOCIATIVAS
PERCEPTRON	BRAIN-STATE-IN-A-BOX
ADALINE/MADALINE	HOPFIELD
BACKPROPAGA TION	OPTIMAL LINEAR ASSOCIA TIVE MEMORY
LINEAR REWARD PENALTY	
ASSOCIATIVE REWARD PENALTY	ADDITIVE GROSSBERG
ADAPTIVE HEURISTIC CRITIC	SHUNTING GROSSBERG
BOLTZMANN MACHINE	
CAUCHY MACHINE	
LEARNING MATRIX	
TEMPORAL ASSOCIATIVE MEMORY	
LINEAR ASSOCIATIVE MEMORY	
OPTIMAL LINEAR ASSOCIATIVE MEMORY	
DRIVE-REINFORCEMENT	
FUZZY ASSOCIATIVE MEMORY	
COUNTERPROPAGATION	
BIDIRECTIONAL ASSOCIA TIVE MEMORY	
ADAPTIVE BIDIRECTIONAL ASSOCIATIVE MEMORY	
COGNITRONOCOGNITRON	
TOPOLOGY PRESERVING MAP	
LEARNING VECTOR QUANTIZER	
ADAPTIVE RESONANCE THEORY	

Tabla 5. Clasificación de las redes neuronales en función del tipo de asociación[12]

[12] Tomado de bibliografía [2]. Capitulo 2, clasificación de las RNA

1.5.4 Representación de la Información de Entrada y Salida

REDES CONTINUAS --- E: ANALÓGICA S: ANALÓGICA	REDES HÍBRIDAS --- E: ANALÓGICA S: BINARIA	REDES DISCRETAS --- E: BINARIA S: BINARIA
BACKPROPAGATION	PERCEPTRON	DISCRETE HOPFIELD
BRAIN-STATE-IN-BOX	ADALINE/MADALINE	LEARNING MATRIX
CONTINOUS HOPFIELD	LINEAR ASSOCIA TIVE REWARD PENALTY	TEMPORAL ASSOCIATIVE MEMORY
LINEAR ASSOCIA TIVE MEMORY	ADAPTIVE HEURISTIC CRITIC	BIDIRECTIONAL ASSOCIATIVE MEMORY
OPTIMAL LINEAR ASSOCIATIVE MEMORY		COGNITRON/ NEOCOG NITRON
DRIVE-REINFORCEMENT		ADAPTIVE RESONANCE THEORY 1
COUNTERPROPAGATION		BOL TZMANN MACHINE
ADDITIVE GROSSBERG		CAUCHY MACHINE
SHUNTING GROSSBERG		
ADAPTIVE BAM		
LEARNINF VECTOR QUANTIZER		
TOPOLOGY PRESERVING MAP		
ADAPTIVE RESONANCE THEORY 2		

Tabla 5. Clasificación de las redes neuronales en función del tipo de representación de las informaciones de entrada y salida[13]

[13] Tomado de bibliografía [2]. Capitulo 2, clasificación de las RNA

1.6 REDES CON APRENDIZAJE NO SUPERVISADAS

Las redes con aprendizaje no supervisado no requieren influencia externa para ajustar los pesos de las conexiones entre sus neuronas. La red no recibe ninguna información por parte del entorno que le indique si la salida generada en respuesta a una determinada entrada es o no correcta; por ello, suele decirse que estas redes son capaces de *autoorganizarse.*

Estas redes deben encontrar las características, correlaciones, o categorías que se puedan establecer entre los datos que se presenten en su entrada. Debido a que no hay un supervisor que indique a la red la respuesta que debe crear ante una entrada específica, cabria preguntarse precisamente por lo que la red genera en estos casos. Existen diferentes posibilidades en cuanto a la interpretación de la salida de este tipo de redes, dependiendo de su estructura y algoritmo.

En algunos casos, la salida puede representar el grado de familiaridad o similitud entre la información que esta presente en la entrada y la información que se le ha presentado en el pasado. En otro caso puede realizar una clusterización (*Clustering*) o establecimiento de categorías, indicando la red a la salida a que categoría pertenece la información presentada a la entrada, siendo la propia red quien deba encontrar las categorías apropiadas a partir de la correlación entra las informaciones presentadas.

Otro aspecto a tener en cuenta es que en los aprendizajes no supervisados se realiza una codificación de los datos de entrada, que genera a la salida una versión con menos bits, pero manteniendo la información mas relevante.

Otras redes de este tipo realizan un mapeo de características (feature mapping), obteniendo en la salida una disposición geométrica (mapa topográfico) de las características de los datos de entrada.

En cuanto a los algoritmos de aprendizaje no supervisados, suelen comúnmente considerarse dos tipos:

1.6.1 Aprendizaje Hebbiano

En este caso suele medirse la familiaridad o extraer características de los datos de entrada y se basa en un postulado formulado por Donald O. Hebb.

Este aprendizaje consiste básicamente en el ajuste de los pesos de las conexiones de acuerdo con la correlación (valores binarios +1 y -1) de los valores de activación de las dos neuronas conectadas. Esto quiere decir que si las dos unidades son activadas (+) se produce un reforzamiento de las conexiones. Por el contrario, cuando una es activa y la otra es pasiva (-), se produce un debilitamiento de la conexión.

Dentro de este tipo de aprendizaje se encuentra la RED HOPFIELD, utilizada en reconstrucción de patrones y optimización, pero con limitaciones de capacidad y estabilidad. Memoria asociativa bidireccional (BAM), se aplica principalmente en memorias heteroasocitiva de acceso por contenido, con

aprendizaje y arquitectura simple, pero con limitaciones de baja capacidad de almacenamiento.

1.6.2 Aprendizaje Competitivo y Cooperativo

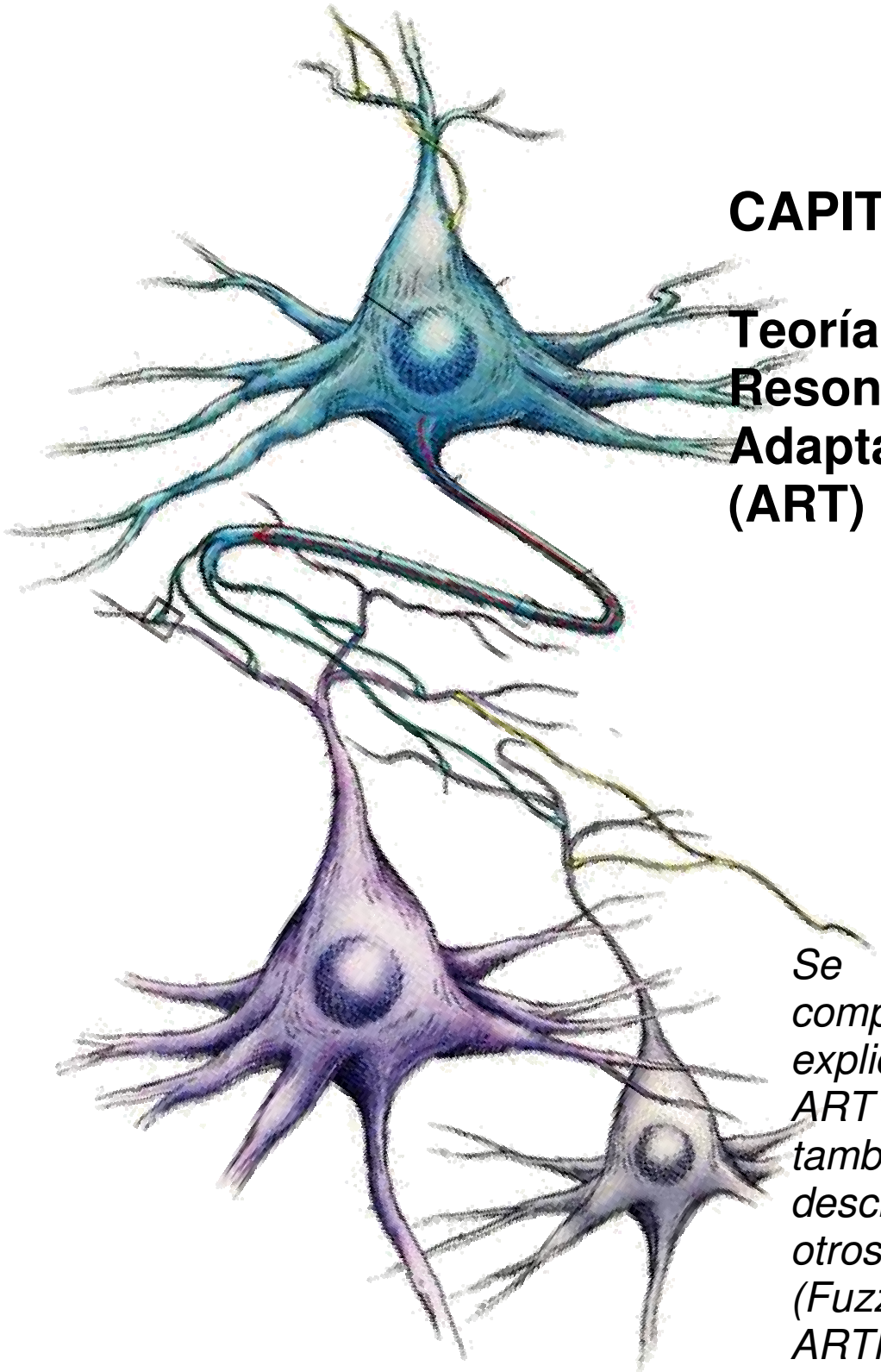
Estas redes se puede decir que las neurona se compiten (y cooperan) unas con otras con el fin de llevar a cabo una tarea dada. Con el fin de que cuando se presente a la red cierta información de entrada, solo una de las neuronas de salida de la red, o una por cierto grupo de neuronas, se active (alcance su valor de respuesta máximo). Por tanto estas neuronas compiten por activarse, quedando una como vencedora, anulando al resto.

El objetivo de este aprendizaje es categorizar (Clusterizar) los datos q se introducen en la red. La información similar es clasificada en una misma categoría, y por tanto activan la misma neurona de salida.

En este tipo de redes, cada neurona tiene asignado un peso total, suma de todos los pesos de las conexiones que tiene a su entrada. El aprendizaje afecta solo a las neuronas ganadoras (activas), redistribuyendo este peso total entre sus conexiones, sustrayendo una porción a los peos de todas las conexiones que llegan a la neurona vencedora y repartiendo esta cantidad por igual entre todas las conexiones procedentes de unidades activas.

Dentro de este tipo de aprendizaje se encuentra La SOM, el cual realiza mapas de características comunes de los datos aprendidos, tiene la limitación de que requiere mucho entrenamiento. Su aplicación más

importante son el reconocimiento de patrones, codificación de datos y optimización. Otro tipo de red es LVQ, aplicado a redes feedforward de dos capas. También se puede ver la red llamada NEOCOGNITRON, la cual es insensible a la translación, rotación y escala, pero requiere muchos elementos de proceso, niveles y conexión; y su aplicación más importante en el reconocimiento caracteres manuscritos



CAPITULO 2

Teoría de la Resonancia Adaptativa (ART)

Se realiza una completa explicación de ART1, y ART2, y también una breve descripción de otros tipos (FuzzyART, ART3, ARTMAP).

2. TEORIA DE LA RESONANCIA ADAPTATIVA

2.1 GENERALIDADES[14]

Stephen Grossberg y Gail A. Carpenter desarrollaron la denominada Teoría de la Resonancia Adaptiva (ART) en el periodo de 1976-1986, un sistema capaz de organizarse a si mismo.

ART se aplica a sistema competitivos, y lo que se pretende es categorizar (clusterizar). Es un modelo de aprendizaje sin supervisar, lo cual es capaz automáticamente de encontrar categorías y crear nuevas cuando se necesitan.

ART fue creado para arreglar el problema de estabilidad y plasticidad en redes competitivas de aprendizaje.

El dilema de la estabilidad y plasticidad consiste en como un sistema de aprendizaje puede preservar su conocimiento adquirido previamente mientras tenga la capacidad de aprender nuevos patrones; Los pesos, el cual han capturado algún conocimiento del pasado, continúan cambiando así como nuevos conocimientos entran, entonces se crea un peligro de perder el conocimiento en el tiempo. Los pesos deben ser lo bastantes flexible para acomodar el nuevo conocimiento pero no mucho como para perder el viejo.

Plasticidad: poder aprender nuevos patrones.

[14] Complementar con http://www.answers.com/topic/adaptive-resonance-theory

Estabilidad: poder retener nuevos patrones.

Los modelos ART se pueden organizarse a si mismos en tiempo real, produciendo reconocimiento estable y mantener los patrones de entrada dentro las categorías originalmente guardados.

Para solucionar el dilema de la plasticidad y estabilidad, el modelo ART propone añadir a las redes un mecanismo de realimentación entre las neuronas competitivas de la capa de salida de la red y la capa de entrada. Este mecanismo facilita el aprendizaje de nueva información sin destruir la ya almacenada. La teoría de la resonancia adaptiva se basa en la idea de hacer Resonar la información de entrada con los representantes o prototipos de las categorías que reconoce la red. Si entra en resonancia con alguno, es suficientemente similar, la red considera que pertenece a dicha categoría y únicamente realiza una pequeña adaptación del prototipo almacenado representante de la categoría para que incorpore algunas características del dato presentado. Cuando no resuena con ninguno, no se parece a ninguno de los existentes, recordados por la red hasta ese momento, la red se encarga de crear una nueva categoría con el dato de entrada como prototipo de la misma.

Existen 6 tipos de ART: ART1, ART2, ART3, Fuzzy ART y ARTMAP

2.2 DESCRIPCION DE LA RED ART[15]

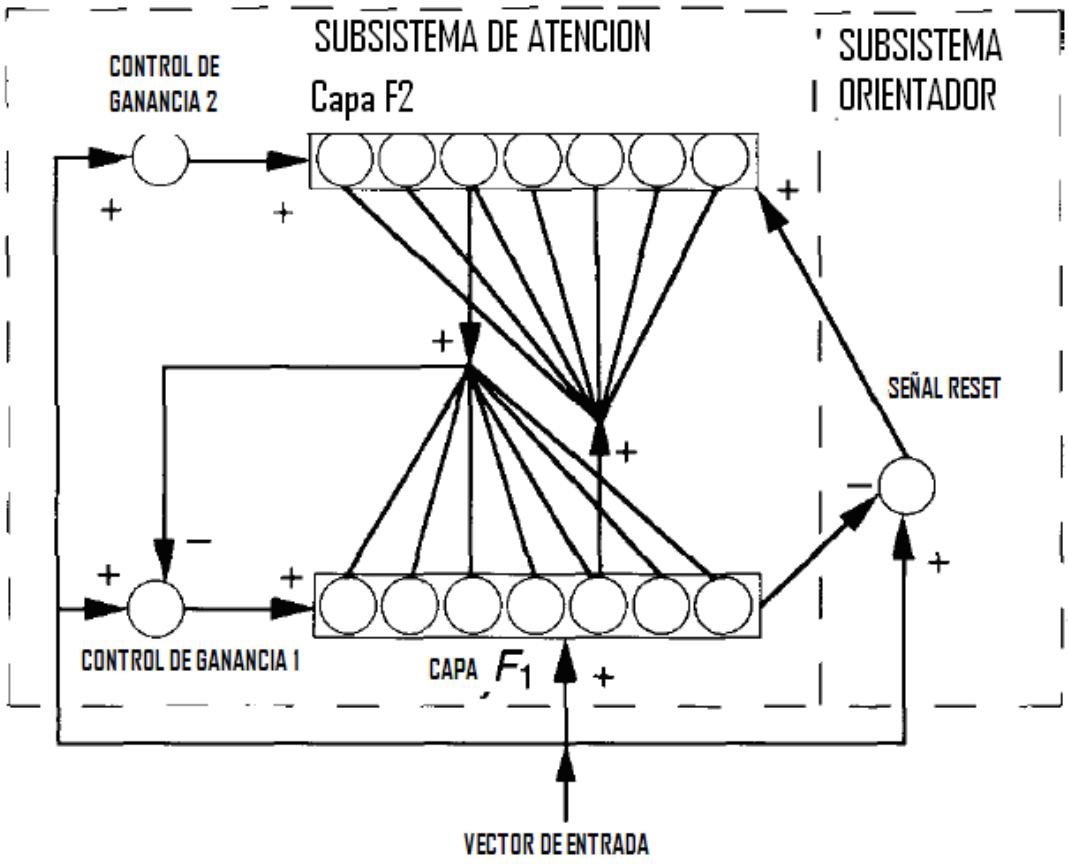


Figura 3. Arquitectura ART[16]

Las redes ART están conformadas por un subsistema de atención y un subsistema orientador (*Figura 3*).

El subsistema de atención contiene: Dos neuronas competitivas: la capa de comparación F_1 y la capa de reconocimiento F_2, y dos controles de ganancia.

[15] Comparar http://cannes.itam.mx/Alfredo/English/Publications/Nslbook/MitPress/157_170.CH08.pdf figura 8.1

[16] Tomado de bibliografía [3]. Capitulo 8, Adaptive Resonace Theory

El subsistema orientador contiene la capa de reset que se encarga de controlar la dinámica del subsistema de atención.

El signo mas indica una conexión excitatoria y el signo menos una conexión inhibidora. Los patrones de actividad que se desarrollan arriba de los nodos en la dos capas del sistema de atención se llaman memoria de corto plazo STM (short-term memory) porque solo existen en asociación con una sola aplicación de la neurona de entrada. Los pesos asociados con el Bottom up (activación de la entrada) y el Top - Down (comparación)entre F_1 Y F_2 se llaman memoria de largo plazo LTM (large-term memory) porque codifican la información que queda por un largo periodo de tiempo

La Estabilización del aprendizaje y la activación ocurren en el subsistema de atención, con la activación de la entrada (Bottom-Up) y la comparación (Top-Down)

2.2.1 Dinamica

En la figura 4(a) , un patrón de entrada I, se introduce a las neuronas F_1, un componente del vector va a cada nodo. Un patrón de activación, X, es producida a través de F_1. El mismo patrón de entrada excita el subsistema orientador A, y la ganancia de control G. El patrón de salida, S, resultan en una señal inhibidora que también es enviada a A. La red esta estructurada para que esta señal inhibidora cancele el efecto excitatorio de la señal de I, así que A permanece inactivo. Nótese que G envía una señal excitatorio a F_1.

La misma señal es aplicada a cada nodo de la capa y se conoce como la señal no específica (se explicara mas a delante).

La aparición de X en F_1 resulta en un patrón de salida S, la cual es enviada a través de las conexiones a F2. Cada unidad de F_2 recibe el vector de entrada S de F_1. Las unidades F_2 calculan sus valores red-entrada sumando los productos de los valores de entrada y los pesos de conexión. En respuestas a las entradas F_1, un patrón de actividad Y, se crea a través de los nodos F_2. F_2 es una capa competitiva que hace un realce de contraste en la señal de entrada. La ganancia de control de F_2 son omitidas aquí para simplicidad.

Figura 4(b). El patrón de actividad Y, resulta en un patrón de actividad U, de F_2. Este patrón de salida es enviado como una señal inhibidora al sistema de control de ganancia. El control de ganancia esta configurado que si recibe una señal inhibidora de F_2, este cesa la actividad. U se convierte en el segundo patrón de entrada para F1. U es transformado por los rastro de LTM en las conexiones Top-Down de F_2 a F_1

Notemos que hay tres posibles entradas en F1, pero solo dos se usan en un tiempo. Las unidades de F_1 y F_2 están construidas para que se activen si solo dos de las tres fuentes de entradas están activas. Esto se llama la regla 2/3 y juega un papel importante en ART. La combinación de Ganancia y la regla 2/3, permite a la capa F_1 distinguir una señal de las capas superiores, o una señal de entrada; esto quiere decir, que es la encargada de controlar los procesos de la red.

Por la regla 2/3, solamente los nodos F_1 recibiendo las señales de I y V permanecerán activos. El patrón que queda en F_1 es $I \cap V$, La intersección de I y V. En la figura 4(b), los patrones no concuerdan y un nuevo patrón de actividad X*, se crea en F_1. Como un nuevo patrón de salida S*, es diferente que el patrón original S, la señal inhibidora de A ya no cancela la excitación proveniente de I.

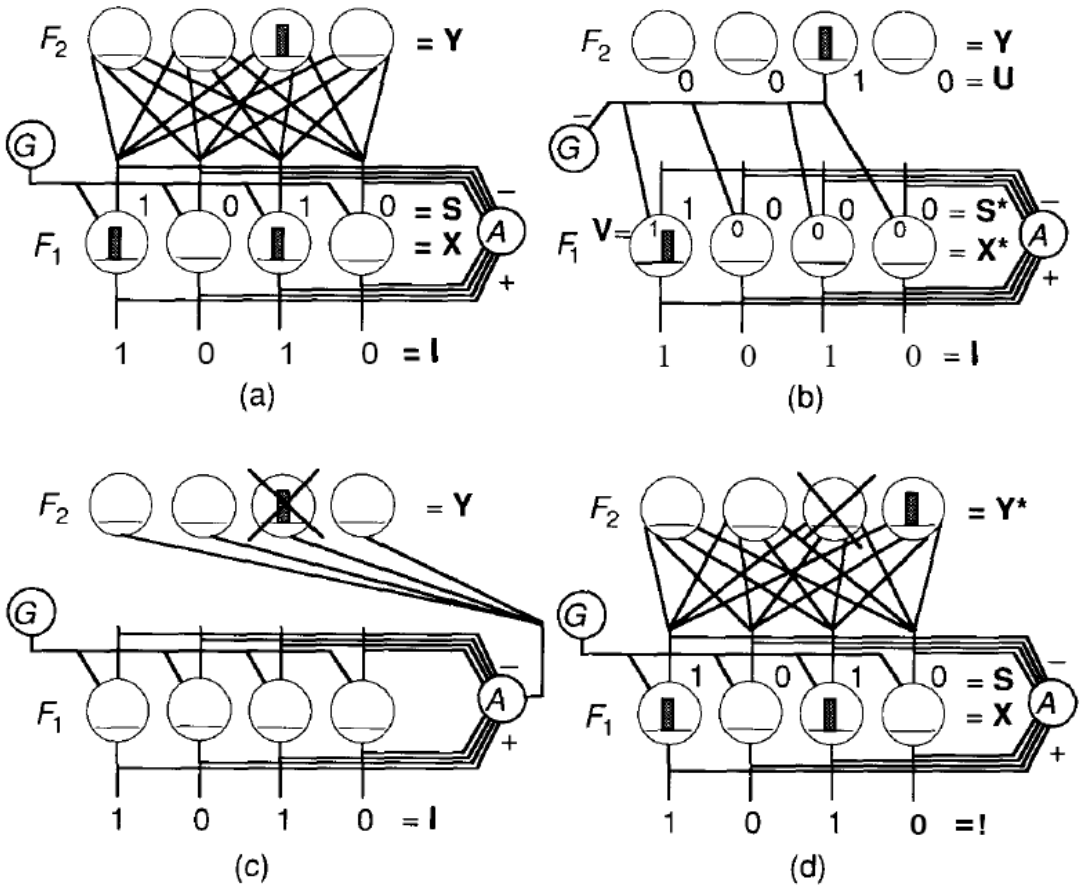


Figura 4. Ciclo de reconocimiento de patrones de una red ART[17]

[17] Tomado de bibliografía [3]. Capitulo 8, Adaptive Resonace Theory

En la figura 4(c), A se convierte activo en respuesta de la no concordancia de los patrones en F_1. A envía una señal no especifica a la señal reset a todos los nodos en F_2, esos nodos responden de acuerdo con su estado presente. Si ellas están inactivas, ellas no responden. Si están activas, se convierten en inactivas y se mantienen así por un periodo de tiempo extendido. Esta inhibición es necesaria para prevenir que el mismo nodo gane la competencia durante el próximo ciclo de concordancia. Mientras Y no aparezca, la salida Top-Down de F_2 y la señal inhibidora de la ganancia desaparece.

En la figura 4(d), el patrón original X, es reinsertado en F_1, y un nuevo ciclo de concordancia de patrones comienza. Esta ves un nuevo patrón Y*, aparece en F_2. Los nodos participando en el patrón original Y se mantienen inactivos debido a el largo efecto de la señal reset de A.

El ciclo de concordancia de patrones seguirá hasta que se encuentre una concordancia, o hasta que a F_2 se le acaben los patrones previamente guardados. Si no se encuentra concordancia, la red asignara nodos sin compromisos en F_2 y comenzara a aprender el nuevo patrón. El aprendizaje toma parte a través de la modificación de los pesos, o los rastros LTM. Es importante entender que este proceso de aprendizaje no comienza o se detiene, siempre continua incluso en el proceso de concordancia de patrones. Siempre se envían señales en las conexiones, los pesos asociados a esas conexiones están sujetos a modificaciones. La no concordancia de patrones no resulta en una perdida de conocimiento debido a que el tiempo

requerido para que cambios significantes ocurran en los pesos es muy largo respecto al tiempo requerido a un ciclo completo de concordancia. Las conexiones participando en no concordancias están inactivos por mucho tiempo con el fin de que no afecten los pesos asociados seriamente.

Cuando una concordancia ocurre, no hay señal de reset y la red se adquiere un estado resonante como se describió al principio. Durante este estado estable, las conexiones se mantienen activas por un tiempo suficientemente largo lo que hace que los pesos sean reforzados. Este estado resonante puede presentarse cuando una concordancia de patrones ocurren, o durante el reclutamiento de nuevas unidades de F_2 con el fin de guardar un patrón desconocido.

2.3 DESCRIPCION DE LA RED ART1[18]

La arquitectura ART1 comparte la misma estructura mostrada en la figura 1. Todas las entradas de ART1 deben ser vectores binarios (0 , 1). Empezaremos analizando el subsistema de atención. Incluyendo las capas de STM de F_1 y F_2, y el mecanismo de control de ganancia G.

2.3.1 El Subsistema de Atención

Las ecuaciones dinámicas para el procesamiento de las actividades en las capas F_1 y F_2 tienen la forma:

$$ex_k = -x_k + (1 - Ax_k)J_k^+ - (B + Cx_k)J_K^- \qquad (1)$$

[18] Comparar con http://www.cns.bu.edu/Profiles/Grossberg/Gro1987CogSci.pdf

J_K^+, es una entrada excitatoria para la k_{esima} unidad

J_K^-, es una entrada inhibidora. Las definiciones precisas para estos términos así como los parámetros A, B y C, depende de cual capa estamos hablando, pero en todos los términos ellos son asumidos que son mayores que cero. De aquí en adelante nosotros usaremos que x_{1i} se refieren a las actividades en la capas F_1 y X_{2J} para actividades en la capa F_2. Similarmente nosotros añadimos números a los nombres de los parámetros para identificar a que capa pertenece: por ejemplo B_1, A_2 . Para conveniencia nosotros debemos nombrar los nodos de F_1 con el símbolo v_i, y los de F_2 con V_j. Los subíndices *i* y *j* serán usados exclusivamente para referirnos a las capas F_1 y F_2 respectivamente.

El factor e se refiere a que las actividades de concordancia de patrón de F_1 y F_2 deben ocurrir mucho mas rápido que el tiempo requerido para que los pesos de las conexiones cambien significativamente. Si nosotros insistimos en que $0 < e << 1$, entonces x_K va a ser un valor muy largo; así que x_K alcanzara su valor de equilibrio rápidamente. Como x_K pasa el mayor de su tiempo cerca de su valor de equilibrio, nosotros no nos debemos preocupar por el tiempo de evolución de los valores de actividad: nosotros debemos automáticamente colocar esas actividades a Valores asintóticos. Debajo de estas condiciones el factor e se convierte en superfluo (sobrante), y debemos quitarlo de la ecuación 1.

2.3.2 Procesamiento en F_1

La figura 3 muestra un elemento F_1 de procesamiento con sus varias entradas y sus vectores de peso. Las unidades calculan un valor de entrada de red viniendo de F_2[19]:

$$V_I = \sum_j u_j z_{ij} \tag{2}$$

Nosotros asumimos que la salida de la unidad de la función rápidamente alcanza a 1 para actividades que no son cero. Además nosotros podemos aproximar la unidad de salida S con una función binaria de escalón.

$$s_i = h(x_{1i}) = \begin{cases} 1\, x_{1i} > 0 \\ 0\, x_{1i} \le 0 \end{cases} \tag{3}$$

La entrada total excitatoria J_i^+ , esta dada por:

$$J_I^+ = I_I + D_1 V_I + B_1 G \tag{4}$$

Donde D_1 y B_1 son constantes[20]. El término inhibitorio J_i^- se debe colocar igual a 1. Con estas definiciones la ecuación para el procesamiento de F1 es

$$\dot{x}_{1i} = -x_{1i} + (1 - A_1 x_{1i})(I_i + D_1 V_I + B_1 G) - (B_1 + C_1 X_{1I}) \tag{5}$$

La salida G del sistema de ganancia de control depende de las actividades en otra parte de la red. Se puede describir G con la ecuación

$$G = \begin{cases} 1 \text{ Si } I \neq 0 \text{ y } U = 0 \\ 0 \text{ De otra manera} \end{cases} \tag{6}$$

[19] Las convenciones de los índices de subpesos q se ha utilizado en esta investigación es opuesta por el usado en Carpenter y Grossberg. En nuestras notaciones, *z i j*, refiere a los pesos en las conexiones desde la unidad Jesima hasta la unidad Iesima. En la notación de Carpenter y Grossberg, *z i j*, se refiere a los pesos en las conexiones desde Iesima hasta Jesima.

[20] Carpenter y Grossber incluye en los cálculos de Vi el parámetros D1-.

En otras palabras, si hay un vector de entrada, y F_2 no esta produciendo un vector de salida entonces G = 1. Si hay otra combinación de actividades en F_1 y F_2 inhibe el control de ganancia a producir una situación no excitación a las unidades F_1.

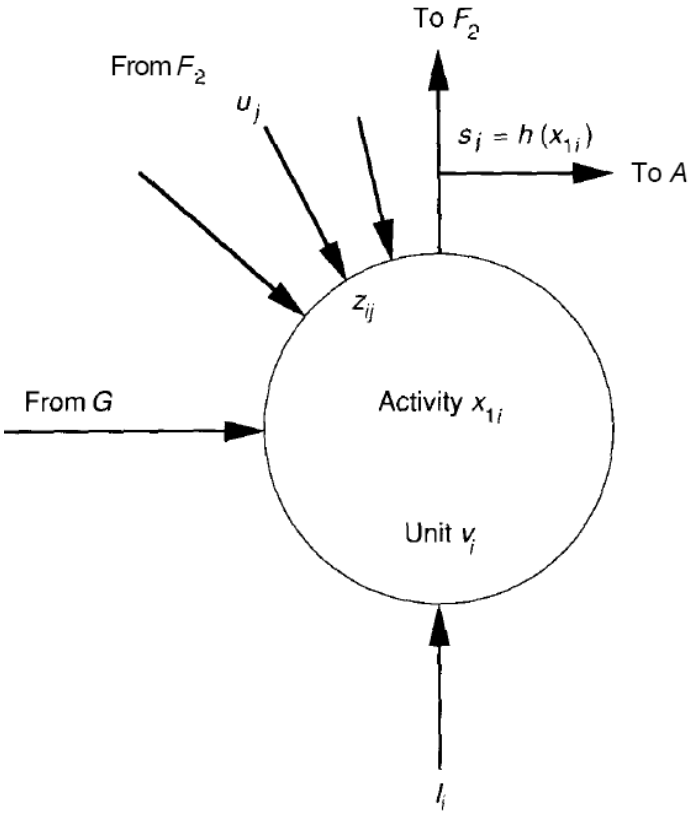


Figura 5. Elemento de procesamiento V_i en la capa F_1[21]

En la *figura 5* muestra un elemento de procesamiento V_i, en la capa F_1 de una red ART1. La actividad de esta unidad es x_{1i}. este recibe una entrada binaria de $I_{i,}$, y una señal excitatoria de la ganancia de control. También se encuentran las señales Top- Down $u_{j,}$, de F_2 que son multiplicadas por los

[21] Tomado de bibliografía [3]. Capitulo 8, Adaptive Resonace Theory
Comparar con http://cannes.itam.mx/Alfredo/English/Publications/Nslbook/MitPress/157_170.CH08.pdf - fig 8.2

pesos Z_{jj}. La salida S_i del elemento de procesamiento va a F_2 y a través del subsistema de orientador A.

Examinando la ecuación 5 para las cuatro posibles combinaciones de actividad de I y F_2, tenemos. Primero, considérese el caso en que no exista vector de entrada y F_2 este inactivo. La ecuación 5 se reduce a

$$x_{1i} = -x_{1i} - (B_1 + C_1 x_{1i})$$

En equilibrios $x_{1I} \to 0$

$$x_{1i} = \frac{-B}{1 + C_1} \tag{7}$$

Las unidades que no tienen entradas, se mantienen en un estado de actividad negativo.

Segundo, Ahora apliquemos un vector de entrada I, pero mantendremos F_2 inactivo por un momento. En este caso F_1 y la ganancia de control G reciben señales de entrada. Como F_2 esta inactivo, G no esta inhibido. Las ecuaciones de las unidades de F1 se convierten en:

$$x_{1i} = -x_{1i} + (1 - A_1 x_{1i})(I_i + B_1 G) - (B_1 + C_1 x_{1i})$$

Cuando las unidades alcanzan sus actividades equilibradas:

$$x_{1i} = \frac{I_i}{1 + A_1 (I_i + B_1) + C_1} \tag{8}$$

En donde hemos usado G = 1. en este caso, las unidades que reciben una entrada distinta a cero, también generan una actividad mas grande que cero

y tienen una salida distinta a a cero. Las unidades que reciben una entrada a cero tienen sus actividades por encima del nivel cero por la señal de excitación no específica de G.

Para el tercer escenario, nosotros examinaremos el caso en donde un patrón de entrada I, y un patrón de Top – Down V de F_2 esta entrando en la capa F_1. las ecuaciones para las unidades de actividades son:

$$x_{1i} = -x_{1i} + (1 - A_1 x_{1i})(I_i + D_1 V_i) - (B_1 + C_1 x_{1i})$$

Y con los valores de equilibrio

$$x_{1i} = \frac{I_i + D_1 V_I - B_1}{1 + A_1 (I_i + D_1 V_I) + C_1} \tag{9}$$

Esto significa que si x_{li} es más grande, igual, o menor que cero depende de los valores relativos en el numerador. Podemos discutir tres casos de interés el cual están determinados por la regla 2/3. Si una unidad tiene un valor de entrada positivo I_i y una entrada positiva de V_i, entonces la regla 2/3 dice que la unidad de actividad debe ser más grande que cero. Para este caso debe ser verdad que $I_I + D_1 V_I - B_1 > 0$ en la Eq. 9. para analizar esta relación mas adelante, nos debemos anticipar algunos resultados de la discusión del procesamiento en la capa F_2. Específicamente, debemos asumir que solo un nodo de F_2 tiene una salida distinta a cero a un determinado tiempo, la salida máxima de un nodo F_2 es uno, y el máximo peso en una conexión Top - Down es también 1.

Como $V_i = \sum_j u_j Z_{ij}$, y solamente un u_j es diferente de cero, entonces Vi < 1.

Entonces, en el caso mas extremo con $V_i = 1$ y $x_{li} = 1$, debemos tener que $1 + D_1 - B_1 > 0$, o

$$B_1 < D_1 + 1 \tag{10}$$

Si una unidad no recibe una señal Top-Down de F_2 debe tener una actividad negativa, aunque el reciba una entrada inferior. En este caso, debemos tener $I_i - B_1 < 0$ o

$$B_1 > 1 \tag{11}$$

Suponemos que F_2 esta produciendo una salida Top- Down, pero no hay aun un vector de entrada I. G esta aun inhibido en este caso. El estado de equilibrio es

$$x_{1i} = \frac{D_1 V_I - B_1}{1 + A_1 D_1 V_I + C_1} \tag{12}$$

Si una unidad no recibe entrada de F2 $V_i = 0$, entonces este se mantiene a su mayor nivel de actividad negativa, como en la ecuación 7 si $V_i > 0$, entonces la actividad de la unidad alcanza algún valor mayor que la Eq. 7, pero debe mantenerse negativa por que no se quiere que la unidad tenga una salida distinta a cero basado en las entradas Top-Down solamente. Entonces del numerador de la Ecuación 12, debemos tener $D_1 - B_1 > 0$, o

$$B_1 > D_1 \tag{13}$$

Combinando las ecuaciones 10, 11 y 13 tenemos la condición global

$$\max\{D_1,1\} < B_1 < D_1 + 1 \tag{14}$$

Los parámetro de ART1 deben satisfacer el contraste de la ecuación 14 para implementar la regla de 2/3 satisfactoriamente y distinguir entre una entrada Top-Down y Botton-Up.

Satisfaciendo los contrastes de la ecuación 14 no se garantizan que la unidad que recibe 2 entradas, una inferior I_i, y una entrada superior V_i, tengan un valor positivo de activación. Se debe considerar el caso en que V_i es menor que su máximo valor 1 (sin embargo $U_j = 1$, z_{ij} deben ser menos que uno, resultando en $V_i < 1$). En este caso la condición para que la ecuación 9 entregue un valor positivo es.

$$I_I + D_1 V_I - B_1 > 0$$

Mientras, $I_i = 1$, esta relación define a la condición en V_i:

$$V_I > \frac{B_1 - 1}{D_1} \tag{15}$$

La ecuación anterior nos dice que la entrada a una unidad V_i debido a una señal Top-Down de F_2 debe alcanzar cierto rango de condición para asegurar una activación positiva de v_i, aunque v_i, recibe una fuerte entrada inferior, I_i, regresaremos este resultado cuando se discutan los pesos, con los rastros LTM de F_2 a F_1.

2.3.3 Procesamiento de F_2

La figura 6 nos muestra un elemento de procesamiento típico en la capa F_2. la entrada de control de ganancia y la conexión del subsistema orientador son mostradas pero no se deben incluir explícitamente.

Las actividades se calculan por medio de la misma ecuación general (ecuación 1); La entrada de red recibida de F1 se calcula con :

$$T_J = \sum_I s_i z_{ji} \tag{16}$$

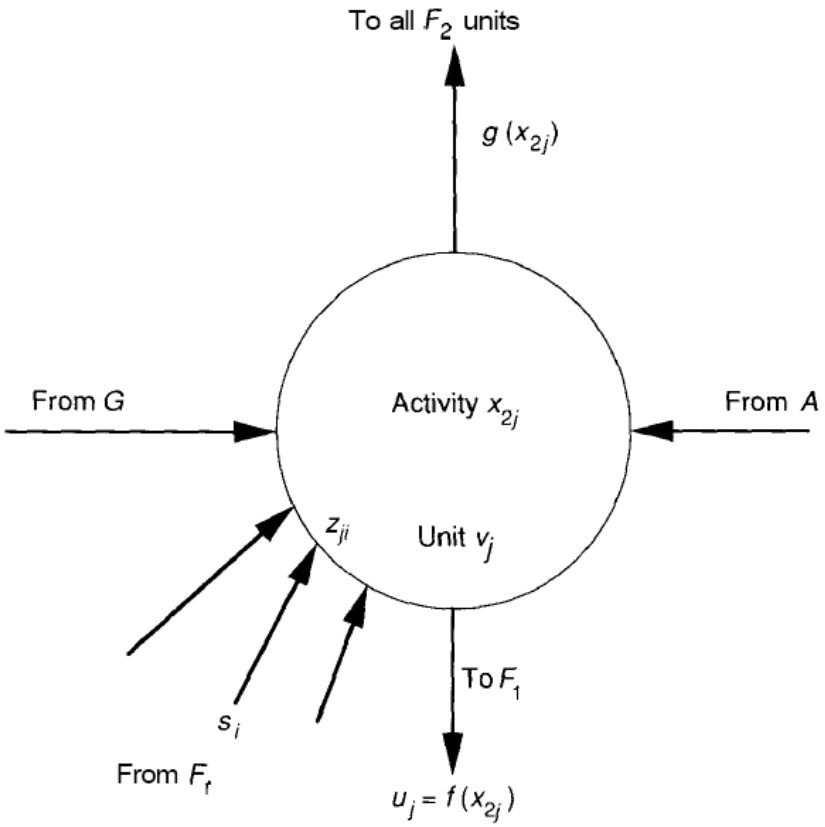


Figura 6. Elemento de procesamiento en la capa F_2[22]

[22] Tomado de bibliografía [3]. Capitulo 8, Adaptive Resonace Theory
Comparar con http://cannes.itam.mx/Alfredo/English/Publications/Nslbook/MitPress/157_170.CH08.pdf - fig 8.3

En la *Figura 6* un elemento de procesamiento v_j, es mostrado en la capa F_2 de una red ART1. la actividad de la unidad es x_{2j} la unidad V_j recibe entradas de la capa F_1, el sistema de control de ganancias G, y el subsistema orientador A.. La señales Botton –Up s_i, de F_1 son multiplicados por los pesos z_{ji}. Las salidas u_j son devueltas a F_1. Además, cada unidad recibe un termino de retroalimentación positiva de el mismo, $g(x_{2j})$ y las envía con una señal idéntica a través de una conexión inhibidora a las otras unidades en la capa.

La entrada total excitatoria a V_j es:

$$J_j^+ = D_2 T_j + g(x_{2j}) \tag{17}$$

La entrada inhibidora es:

$$J_j^- = \sum_{k \neq j} g(x_{2k}) \tag{18}$$

Sustituyendo estos Valores en la Ecuación 1

$$\dot{x}_{2j} = -x_{2j} + (1 - A_2 x_{2j})(D_2 T_j + g(x_{2j})) - (B_2 + C_2 x_{2j}) \sum_{k \neq j} g(x_{2k}) \tag{19}$$

Se asume que los valores de varios parámetros en al ecuación 19, y la forma funcional de g(x) han sido escogidas para aumentar la actividad de un solo nodo de F2 con un valor de entrada de red mas larga de F1 de acuerdo a la ecuación 16. las actividades de los otros nodos son eliminadas a cero. La salida de este nodo ganador se le da un valor de 1. Entonces se puede expresar los valores de salida de los nodos de F2 como: símbolos lógicos

$$u_j = f(x_{2j}) = \begin{cases} 1 & T_i = \max_K \{T_k\} \forall k \\ 0 & \text{otro valor} \end{cases} \tag{20}$$

Se debe clarificar un punto final en la figura 6. El elemento de procesamiento en la figura aparenta violar nuestro estándar de una sola salida por nodo:
El nodo envía una salida a $g(x_{2j})$ a las unidades de F_2, y una salida de $f(x_{2j})$ a las unidades F_1. se puede reconciliar esa discrepancia permitiendo que la figura 6 represente la estructura compuesta. Se puede ordenar la unidad V_j para que tenga un solo valor de salida de x_{2j}. Esta salida puede ser enviada a otros dos elementos de procesamiento; uno que da una salida de $g(x_{2j})$, y otro que da a una salida de $f(x_{2j})$. Asumiendo la existencia de estos nodos intermedios, o ínter neuronas, se puede evitar violar el estándar de una sola salida. El nodo de la figura 6 representa un compuesto de nodos de V_j y los 2 nodos intermedios.

A. RASTROS DE TOP-DOWN LTM:

las ecuaciones que describen los rastros Top-Down LTM (Pesos en las conexiones de las unidades de F_2 a la de F_1) son.

$$\dot{z}_{ij} = (-z_{ij} + h(x_{1i}))f(x_{2j}) \qquad (21)$$

Como $f(x_{2j})$ es diferente de cero para solo un valor de j (para un nodo de F_2, v_j), la ecuación 21 es diferente a cero solamente para las conexiones que provienen de la unidad ganadora. Si un nodo j de F_2 es activado y un nodo i de F_1 esta también activo, entonces $\dot{z}_{ij} = -z_{ij} + 1$ y z_{ij} asintóticamente se

acerca a 1. si el nodo j de F_2 es activo y el nodo i de F_1 no esta activo, entonces $z_{ij} = -z_{ij}$ y z_{ij} decae a cero. Se puede resumir el comportamiento de z_{ij} como lo siguiente.

$$\dot{z}_{ij} = \begin{cases} -z_{ij} + 1 & V_j \text{ activo y } v_i \text{ activo} \\ -z_{ij} & V_j \text{ activo y } v_i \text{ inactivo} \\ 0 & V_j \text{ inactivo y } v_i \text{ inactivo} \end{cases} \tag{22}$$

Retomando de la ecuación 15 que, si F_2 está activo, entonces v_i , puede estar activa si solamente está recibiendo una entrada, I_i, de abajo, o una suficientemente larga entrada de red V_i, proveniente de la capa F_2. Puesto que solo una unidad de F_2 esta activa al tiempo, $V_j - u_j z_{ij} - z_{ij}$. La ecuación 15 ahora se convierte en una condición para los pesos en la conexión de V_j a v_i :

$$z_{ij} > \frac{B_1 - 1}{D_1} \tag{23}$$

A menos que z_{ij} tenga un valor mínimo dado por la ecuación 23, este ira decayendo a cero aunque v_j este activo y v_i este recibiendo una entrada I_i.

Desde un punto practico, todos los pesos de conexiones Top-Down deben ser inicializados a un valor más grande que el mínimo entregado en la ecuación 23 para que cualquier aprendizaje tenga lugar en la red. De otra forma, cualquier tiempo en que V_j este activo en F_2, todas las conexiones de el a cualquier unidad en F_1 decaerán a cero, eventualmente todos los pesos de las conexiones serán cero y el sistema será inútil.

Si a una condición resonante se le permite continuar por un periodo extendido, los pesos Top-Down se acercaran a sus valores asintóticos de uno o cero, de acuerdo a la ecuación 22. Así, tan pronto como se detecten un estado resonante, se puede inmediatamente colocar el peso Top-Down apropiado para sus valores asintóticos. Si v_j es el nodo ganador, entonces.

$$z_{ij} = \begin{cases} 1 & v_i \text{ activo} \\ 0 & \text{Otro Valor} \end{cases} \tag{24}$$

Los pesos en las conexiones de otros nodos, v_j. $j \neq 1$ no cambian. Se refiere a este modelo como rápido aprendizaje. Los pesos en las conexiones Botton-Up también tienen un modelo de rápido aprendizaje, las cuales discutiremos a continuación.

B. RASTROS BOTTON-UP

Estas ecuaciones son más complicadas que los rastros Top-Down. El peso en las conexiones de v_i en F_1 a V_j en F_2 están determinados por

$$z_{ji} = Kf(x_{2j})\left[(1 - z_{ji})Lh(x_{1i}) - z_{ji}\sum_{k \neq i} h(x_{1k})\right] \tag{25}$$

Donde K y L son constantes, $f(x_{2j})$ es la salida de V_j, y $h(x_{1i})$ es la salida de v_i. Como fue el caso con los rastros LTM Top-Down, el factor $f(x_{2j})$ asegura que solo los pesos del nodo ganador son los que se le permiten cambiar. La ecuación 25 es una ecuación para un sistema competitivo con interacciones.

En este caso, sin embargo, son los pesos individuales los que compiten uno con otro, en vez de unidades individuales.

Se asume que V_j es la unidad ganadora de F_2. Hay otros dos casos para considerar. Si v_i es activo en F_1 entonces $h(x_{1i})$ es igual a 1; de otra forma x_{1i} es igual a cero. Los pesos en las conexiones a otras unidades F_2 no cambian hasta que $f(x_{2j}) = 0$, $k \neq j$.

Antes de seguir adelante de la ecuación 25 se quiere incluir una nueva notación. Si el patrón de entrada es I, entonces se debe definir la magnitud de I como

$|I| = \sum_i I_i$, puesto que I_i es cero o uno, la magnitud de I es igual a un número de entradas diferentes a cero. La salida de F_1 es el patrón S: esta magnitud es $|S| = \sum_i h(x_{1i}),$ el cual es también exactamente el número de salidas diferentes de cero en F_1. El patrón de salida actual S, depende de las condiciones en la red

$$S = \begin{cases} I & F_2 \text{ es inactiva} \\ I \cap V^J & F_2 \text{ es activa} \end{cases} \qquad (26)$$

Donde la intercepción en V^J significa que v_j fue el nodo ganador en F_2. Se puede interpretar V^J como el significado del patrón binario con un 1 en esas posiciones donde la entrada, V_i, de arriba es tan grande para apoyar la activación V_i cada vez que V_i reciba una entrada I_i de abajo.

Debido a que $|S| = \sum_i h(x_{1i}),$, entonces $\sum_{k \neq i} h(x_{1k}) = \sum_k h(x_{1k}) - h(x_{1i})$ el cual va a ser igual a $|S| - 1$ o $|S|$, dependiendo en si v_i esta activo. Se puede concluir los tres casos de la ecuación 25 como sigue:

$$\dot{z}_{ji} = \begin{cases} K\left[(1 - z_{ji})L - z_{ji}(|S| - 1)\right] & V_j \text{ activo y } V_i \text{ activo} \\ -K\left[z_{ij}|S|\right] & v_j \text{ activo y } v_i \text{ inactivo} \\ 0 & V_j \text{ inactivo} \end{cases} \tag{27}$$

Recordemos, v_i puede mantenerse activo solamente si recibe una entrada debido a una señal Top- Down V_i, suficientemente alta; y una entrada I_i. en un caso de rápido aprendizaje, los pesos en el nodo ganador F_2, v_j, toman valores asintóticos dados por:

$$z_{ji} = \begin{cases} \dfrac{L}{L - 1 + |s|} & \text{si } v_i \text{ es activa} \\ 0 & \text{si } V_i \text{ es inactivo} \end{cases} \tag{28}$$

Donde se tiene L > 1 con el fin de mantener L – 1 > 0.

2.3.3 El Subsistema Orientador

El subsistema orientador en una red ART es responsable de censar discordancias entre los patrones Botton-Up y Top-Down en la capa F_1. Esta operación puede ser moderada añadiéndole términos a la ecuación dinámica que describe las actividades en el elemento de procesamiento F_2. Desde que nuestra discusión ha evolucionado desde las ecuaciones dinámicas hasta

sus ecuaciones asintóticas y el caso de rápido aprendizaje, no se regresara a las educaciones dinámicas en este punto.

Hay muchas formas para moderar las dinámicas del subsistema orientador. Nuestra aproximación ira a describir los detalles del proceso de concordancia y reset y los efectos en las unidades F_2.

Se puede moderar el subsistema orientador como un solo elemento de procesamiento A, con una salida a cada unidad en la capa F_2, las entradas a A son las salidas de las unidades F_1 (S), y el vector de entrada I. Los pesos en las conexiones del el vector de entrada son igual a un valor P, las que están en las conexiones de F_1 son igual al valor de -Q. la entrada de red a A es entonces $P|I| - Q|S|$. La salida de A se activa si las entradas de red se convierten diferentes de cero.

$$P|I| - Q|S| > 0$$

O,

$$P|I| > Q|S|$$

$$\frac{P}{Q} > \frac{|S|}{|I|}$$

La cantidad P/Q se le da el nombre de parámetro de vigilancia y es usualmente identificado por el símbolo, *p.* Además, la activación del subsistema orientador es prevenido si:

$$\frac{|S|}{|I|} > \rho \qquad (29)$$

Recordemos que la magnitud de $|S| = |I|$ cuando F_2 esta inactivo. El subsistema orientador no debe enviar una señal de reset a F_2 en ese tiempo. De la ecuación 29 se tiene una condición en el parámetro de vigilancia:

$$p \leq 1$$

También se obtiene una condición subsiguiente en P y Q:

$$P \leq Q$$

El valor del parámetro de Vigilancia mide el grado al cual el sistema discrimina entre diferentes clases de patrones de entrada. Debido a la forma en que *p* es definida, esta implementa un patrón de concordancia Auto-Escala. Por Auto-Escalamiento, se quiere decir que la presencia o la abstinencia de una cierta característica en dos patrones debe o no debe causar un reset dependiendo en la importancia total de esa característica en definir las clases de patrones. La figura 7 ilustra un ejemplo de este Auto-Escalamiento.

Por un número de patrones dados para ser clasificados, un valor grande de *p* ira a resultar en una discriminación fina entra las clases, que un valor de *p* menor

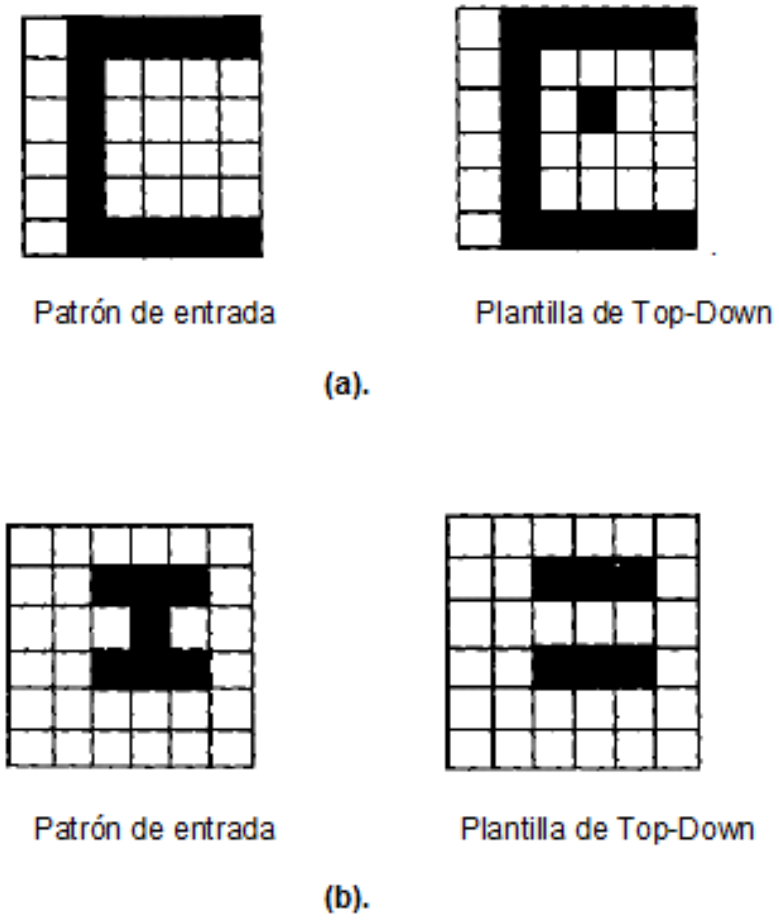


Figura 7. Propiedad de Autoescalamiento[23]

En la *figura 7* se ilustran la propiedad de Auto-Escalamiento de la red ART. (a). para el de p ≅ 0.8, la existencia de la característica extra del centro del patrón Top-Down a la derecha es ignorada por el subsistema orientador, el cual considera los dos patrones como de la misma clase. (b) para el mismo valor de p estos patrones Botton-Up y Top-Down causarán que el subsistema orientador envíe un reset a F_2.

[23] Tomado de bibliografía [3]. Capitulo 8, Adaptive Resonace Theory
Comparar con Ejemplo de Matlab Algoritmo ART1 Capitulo 3

Teniendo un valor de p que es menor que 1 también permite la posibilidad que el patrón Top-Down que esta codificado por F_2 para representar una clase particular deba cambiar así como nuevos vectores son presentados en la red.

Se puede ver esto referenciado en la figura 7(a). Implícitamente se asume que el patrón Top-Down en la derecha (con el punto extra) ha sido previamente codificado (aprendido) por unos de los nodos F_2. la aparición del patrón Botton-Up en la izquierda (sin el punto extra) no causa un reset, entonces se establece una resonancia entre la capa F_1 y el nodo ganador F_2 que produjo el patrón Top-Down. Durante esta resonancia, los pesos pueden ser cambiados. El nodo F_1 correspondiente a la característica del punto del centro no esta activa desde que la característica hace falta en el vector de entrada. De acuerdo a la ecuación 22, los pesos del Top-Down en esa conexión irán decayendo lejos; en el modo de rápido aprendizaje, simplemente se coloca igual a cero. Similarmente, el peso Botton-Up en el nodo F_2 ira decayendo de acuerdo a ala ecuación 27.

La recodificación de los patrones de plantillas Top-Down descritas en el párrafo anterior se pudo llevar a inestabilidades en el proceso de aprendizaje. Estas inestabilidades pudieron manifestarse como un cambio continuo en las clases de los vectores de entrada reconocidos, o codificados, por cada unidad F_2. Afortunadamente, ART fue creado para combatir esta inestabilidad. Mas no se probara aquí, pero puede ser mostrado que el

aprendizaje de categoría se estabilizará en un red ART1, después de unas pocas recodificaciones. La estabilidad del aprendizaje es un resultado directo del uso de la regla 2/3 descrita anteriormente.

Para completar el modelo del subsistema orientador, se debe considerar sus efectos en la unidad F_2. Cuando un patrón de no concordancia ocurre, el subsistema orientador debe inhibir la unidad F_2 que resulto en un patrón de no concordancia, y debe mantener esa inhibición a través de el ciclo de concordancia restante.

Se concluye con esto la presentación de ART1. Antes de proceder al modelo ART2 se presentará un resumen del modelo ART1 para el caso asintótico y de rápido aprendizaje.

2.3.4 Resumen De Procesamiento ART1:

Para este resumen se debe emplear las soluciones asintóticas a las ecuaciones dinámicas y el caso de rápido aprendizaje para los pesos. También se debe presentar paso a paso los cálculos mostrando como la red ART1 aprende y recuerda a los patrones.

Para empezar se debe determinar el tamaño de las capas F_1 y F_2 y los valores de los diversos parámetros en el sistema. Se coloca que M sea el numero de unidades en F_1 y N el numero de unidades en F_2.

Otros parámetros deben ser escogidos de acuerdo a los siguientes contrastes

$$A_1 > 0$$
$$C_1 > 0$$
$$D_1 > 0$$

$$\max\{D_1, 1\} < B_1 < D_1 + 1$$

$$L > 1$$
$$O < \rho < 1$$

Los pesos Top-Down ($v_j \rightarrow v_i$) son iniciados de acuerdo a:

$$z_{ij}(0) > \frac{B_1 - 1}{D_1}$$

Y los pesos Botton-Up ($v_i \rightarrow v_j$) son iniciados de acuerdo a:

$$0 < z_{ji}(0) < \frac{L}{L - 1 + M}$$

Las actividades en F_2 son iniciados en cero, pero, de acuerdo a nuestro modelo escogido, las actividades son iniciadas como:

$$x_{1i}(0) = \frac{-B_1}{1 + C_1}$$

Todos los patrones de entrada son binarios $I_i \in \{0,1\}$. La magnitud de un vector es igual a la suma de sus componentes: por ejemplo, $|I| = \sum_i^M I_I$. Mientras se esté interesado en la magnitud de vectores binarios solamente, esta suma va a ser igual al número de componentes diferente de cero del vector.

Ahora se está en condición para procesar los datos en la red. Procedemos de acuerdo al siguiente algoritmo:

1. Aplicar un vector de entrada I a F_1. Las actividades de F_1 son calculadas de acuerdo a

$$x_{1i} = \frac{I_i}{1 + A_1(I_I + B_1) + C_1}$$

2. Calcular el vector de salida para F_1

$$s_i = h(x_{1i}) = \begin{Bmatrix} 1\, x_{1i} > 0 \\ 0\, x_{1i} \le 0 \end{Bmatrix}$$

3. Propagar S hacia F_2 y calcular las actividades de acuerdo a:

$$T_j = \sum_{i=1}^{M} s_i z_{ji}$$

4. Solamente el nodo F_2 ganador tiene una salida diferente a cero

$$u_j = f(x_{2j}) \begin{cases} 1 & T_i = \max_K \{T_k\} \forall k \\ 0 & \text{otro valor} \end{cases}$$

Se asume que el nodo ganador es v_j

5. Propagar las salidas de F_2 hacia F_1. Calcular las entradas red de F_2 hacia las unidades F_1

$$V_I = \sum_{j=1}^{N} u_j z_{ij}$$

6. Calcular las nuevas actividades de acuerdo a

$$x_{1i} = \frac{I_i + D_1 V_i - B_1}{1 + A_1(I_i + D_1 V_i) + C_1}$$

7. Determinar los nuevos valores de salida, s_j como en el paso 2

8. Determinar el grado de concordancia entre el patrón de entrada y la plantilla Top-Down

$$\frac{|S|}{|I|} = \frac{\sum_{i=1}^{M} s_i}{\sum_{i=1}^{M} I_i}$$

9. Si $|S| / |I| < p$ señala a v_j como inactivo, las salidas de F_2 cero, y regresa al paso uno usando el patrón original de entrada. Si $|S| / |I| > p$, continuar.

10. Actualizando los pesos en Botton-Up solamente en las unidades F_2

$$z_{ji} = \begin{cases} \dfrac{L}{L-1+|s|} & \text{si } v_i \text{ es activa} \\ 0 & \text{si } V_i \text{ es inactivo} \end{cases}$$

11. Actualizando los pesos Top-Down solamente a todas las unidades F_1:

$$z_{ij} = \begin{cases} 1 & \text{si } v_i \text{ es activa} \\ 0 & \text{si } V_i \text{ es inactivo} \end{cases}$$

12. Remueva el patrón de entrada. Restaure todas las unidades F_2 inactivas. Regrese al paso uno con un nuevo patrón de entrada.

2.4 DESCRIPCION DE LA RED ART 2[24]

ART2 difiere de ART1 solamente en la naturaleza de los patrones de entrada: ART2 acepta componentes de vectores análogos (escala de gris), así como también componentes binarios. Esta capacidad representa un realce al sistema.

La diferencia entre ART1 y ART2 se basan en diferencias de arquitectura que le da a ART2 su estabilidad para enfrentarse a patrones análogos. Estas diferencias a veces son mas complejas, y a veces menos complejas que las estructuras ART1.

ART2 debe enfrentarse con complicaciones adicionales. Por ejemplo, ART2 debe poder reconocer la similitud subyacente de patrones idénticos sobrexpuestos en fondos constantes teniendo diferentes niveles. Comparado en un sentido común, dos patrones similares deben aparecer enteramente diferentes cuando en realidad, se debe estar clasificado el mismo patrón.

El precio de esta capacidad adicional es primeramente un incremento en la complejidad del procesamiento del nivel de F_1. El nivel en ART1 consiste en diferentes subniveles y varios sistemas de control de ganancia. El procesamiento en F_2 es igual tanto para ART2 como para ART1. Como una compensación parcial para la complejidad añadida, las ecuaciones LTM son un poco más simples para ART2 que para ART1.

---

[24] Comparar con http://cns-web.bu.edu/Profiles/Grossberg/CarGro1987AppliedOptics.pdf

2.4.1 Arquitectura De ART2

Como se menciono en la introducción de esta sección, ART2 tiene un parecido superficial a ART1. los dos tienen un subsistema de atención y un subsistema orientador. El subsistema de atención para cada arquitectura consiste en dos capas de elementos de procesamiento, F_1 y F_2, y, un sistema de control de ganancia. El subsistema orientador para cada red realiza la función idéntica. Además, la ecuaciones diferenciales básicas que gobiernan las actividades de los elementos de procesamientos individuales son los mismos en los dos casos. Para enfrentarse exitosamente con patrones análogos en el ART2, se tuvo que separar la capa F_1 en un número de subcapas conteniendo las conexiones Feedforward y Feedback. La figura 8 muestra la estructura resultante.

Toda la estructura de la red ART2 es similar que la de ART1 (*Figura 8*). la capa F1 ha sido dividida en 6 subcapas w, x, u, v, p y q. Cada nodo marcado con G es una unidad de control de ganancia que envía una señal de inhibición no específica a cada unidad en la capa que este alimenta. Toda las subcapas en F_1, así como en la capa r del subsistema orientador, tiene el mismo numero de unidades. Las subcapas individuales en F_1 están conectadas de unidad a unidad; esto es, que las capas no están totalmente interconectada, con excepción de las conexiones Botton-Up a F_2 y las conexiones Top-Down de F_2.

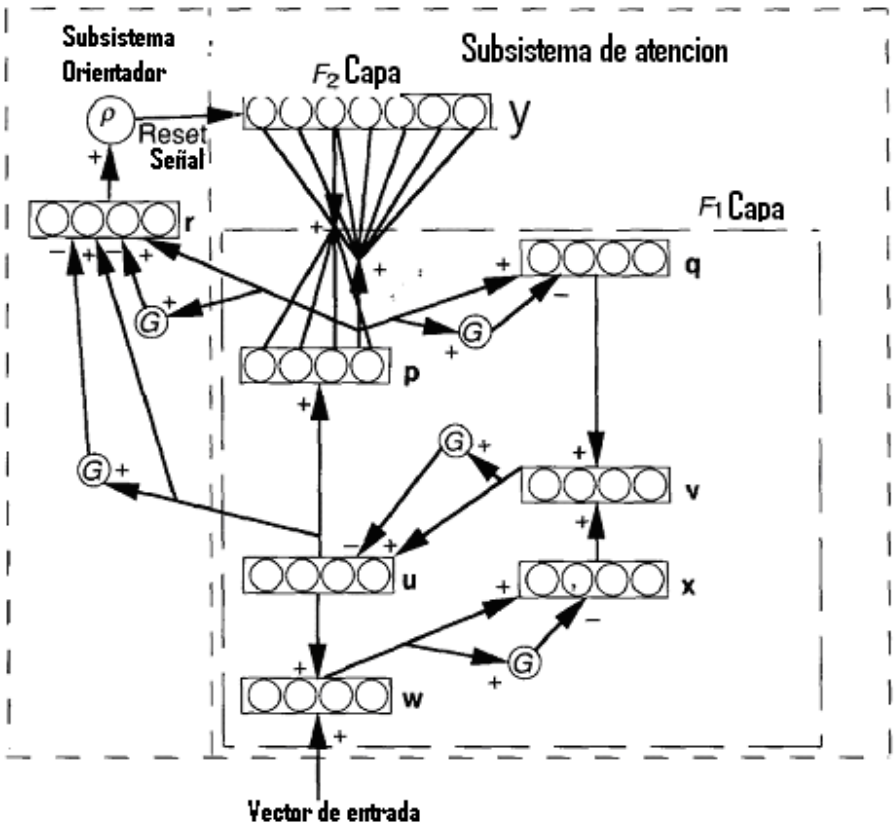


Figura 8. Arquitectura de ART2[25]

2.4.2 Procesamiento en F1

La actividad de cada unidad en cada subcapa de F_1 es gobernada por una ecuación de la forma

$$ex_k = -Ax_k + (1 - Bx_k)J_k^+ - (C + Dx_k)J_k^- \qquad (30)$$

Donde A, B, C, D son constantes. La ecuación 30 es casi idéntica a la ecuación 1 de ART1 la única diferencia es de un factor multiplicativo en el primer termino en el lado derecho de la ecuación 30. Para el modelo de ART2 presentado, se debe poner B y C idénticamente igual a cero. Igual que

[25] Tomado de bibliografía [3]. Capitulo 8, Adaptive Resonace Theory

en ART1, J_k^+ y J_k^- representa la excitación de la red y los factores inhibidores respectivamente. Igual que en ART1 se debe estar interesado en la solución asintótica, entonces

$$x_k = \frac{J_k^+}{A + DJ_k^-} \tag{31}$$

Los valores de las cantidades individuales en la ecuación 31 varían de acuerdo a la subcapa en que se esta considerando. Por conveniencia, se realizó la tabla 7, el cual muestra todas las cantidades apropiadas para cada subcapa F_1, así también, en capa *r* del subsistema orientador. Basada en la tabla 7, las actividades en cada una de las 6 subcapas pueden ser reducidas en las siguientes ecuaciones.

$$w_i = I_i + au_i \tag{32}$$

$$x_i = \frac{w_i}{e + \|w\|} \tag{33}$$

$$v_i = f(x_i) + bf(q_i) \tag{34}$$

$$u_i = \frac{v_i}{e + \|v\|} \tag{35}$$

$$p_i = u_i + \sum_j g(y_j) z_{ij} \tag{36}$$

$$q_i = \frac{p_i}{e + \|p\|} \tag{37}$$

CAPA	A	D	J^+	J^-
W	1	1	$I_i + au_i$	0
X	e	1	w_i	$\|w\|$
u	e	1	v_i	$\|v\|$
v	1	1	$f(x_i) + bf(q_i)$	0
p	1	1	$u_i + \sum_j g(y_j) z_{ij}$	0
q	e	1	ρ_i	$\|p\|$
r	e	1	$u + cp$	$\|u\| + c\|p\|$

Tabla 6. Cantidades apropiadas para cada subcapa F_1, la capa r del subsistema orientador[26]

Los factores en la ecuación 31 para cada subcapa F_1 y la capa *r*. Ii es el i_{teavo} componente del vector de entrada. Los parámetros a, b, c y e son constantes cuyos valores serán discutidos en esta investigación. Yj es la actividad de la $j_{enésima}$ unidad en la capa F_2 y g(y) es la función de salida de F_2. La función f(x) es descrita en esta investigación.

Se Explicará la capa *r* del subsistema orientador mas adelante. El parámetro *e* se le coloca un numero positivo considerablemente menor que uno. Este

[26] Tomado de bibliografía [3].

tiene el efecto de mantener las activaciones finitas cuando no hay entrada presente en el sistema. No se requiere la presencia de *e* para esta discusión por eso se colocará *e* = 0 para lo que resta de ART2.

Las tres unidades de control de ganancia en F_1 inhibe no específicamente la subcapa *x, u* y *q*. La señal inhibidora es igual a la magnitud del vector de entrada a esas capas. El efecto es que las actividades de estas tres capas son normalizadas por unidad por las señales de control de ganancia.

La forma de la función f(x), determina la naturaleza de realce de contraste que pasa en F_1.

$$f(x) = \begin{cases} 0 & 0 \le x \le \theta \\ x & x > \theta \end{cases} \tag{38}$$

Donde $\theta > 1$, se utilizará a $\theta = 0.2$

2.4.3 Procesamiento en F_2

El procesamiento de ART2 en F_2 es igual que en ART1. Las entradas Botton-Up son calculadas como en ART1:

$$T_j = \sum_i s_i z_{ji} \tag{39}$$

La competencia en F_2 resulta en un realce de contraste donde se escoge un solo nodo ganador.

La función de salida de F_2 esta dada por

$$g(y_j)=\begin{cases} d \;\; T_j = \max_K \{T_k\} \forall k \\ 0 \qquad \text{otro valor} \end{cases} \tag{40}$$

Esta ecuación presume que el ajuste de $\{T_K\}$ incluye solamente los nodos que no han sido reseteados recientemente por el subsistema orientador.

Ahora se puede reescribir la ecuación para el procesamiento en la capa *p* de F_1 como (Eq 36).

$$p_i = \begin{cases} u_{\mathrm{i}} & \text{si } \mathrm{F}_2 \text{ es inactivo} \\ u_{\mathrm{i}} + dz_{\mathrm{iJ}} & \text{si el jesimo nodo en } \mathrm{F}_2 \text{ es activo} \end{cases} \tag{41}$$

2.4.4 Ecuaciones LTM

Las ecuaciones LTM en ART2 son significativamente menos complejas que las presentadas en ART1. Las ecuaciones Botton-Up y Top-Down tienen la misma forma:

$$z_{ji} = g(y_j)(P_i - z_{ji}) \tag{42}$$

Para los pesos Botton-Up de v_i en F_1 a v_j en F_2, y

$$z_{ij} = g(y_j)(P_i - z_{ij}) \tag{43}$$

Para los pesos Top-Down de v_j en F_2 a v_i en F_1. Si v_j es el nodo F_2 ganador, se puede usar la ecuación 40 en la ecuación 41, 42 y 43 para mostrar que

$$z_{ji} = d(u_i + dz_{ij} - z_{ji})$$

Y similarmente

$$z_{ij} = d(u_i + dz_{ij} - z_{ij})$$

Con los otros $z_{iJ} = z_{Ji} = 0$ para $j \neq J$. Se debe estar interesado en el caso de rápido aprendizaje, entonces se puede resolver para los valores de equilibrio de los pesos

$$z_{iJ} = z_{Ji} = -\frac{u_i}{1-d} \tag{44}$$

Se asume que $0 < d < 1$.

Se debe posponer la discusión para los valores iníciales para los pesos hasta después de la discusión del subsistema orientador.

2.4.5 Subsistema Orientador de ART2

De la tabla 7 y la ecuación 31 se puede construir la ecuación para las actividades de los nodos en la capa *r* del subsistema orientador, como *e* es cero

$$r_i = \frac{u_i + cp_i}{\|u\| + \|cp\|} \tag{45}$$

La condición de reset es.

$$1 < \frac{p}{\|r\|} \tag{46}$$

Donde *p* es el parámetro de vigilancia.

Nótese que dos subcapas de F_1 p y u participan en el proceso de concordancia. Así como los pesos Top-Down cambian en la subcapa p durante el aprendizaje, la actividad de las unidades de la subcapa p también

cambian. La capa u se mantiene estable durante este proceso, entonces incluyendo a esta en proceso de aprendizaje previenen que ocurra el reset mientras el aprendizaje de un nuevo patrón se esta llevando a cabo.

Se puede reescribir la ecuación 45 en la forma de vector como.

$$r = \frac{u + cp}{\|u\| + c\|p\|}$$

Entonces, de $\|\mathbf{r}\| = (\mathbf{r} \bullet \mathbf{r})^{1/2}$, entonces se puede escribir

$$\|r\| = \frac{[1 + 2\|cp\|\cos(u, p) + \|cp\|^2]^{1/2}}{1 + \|cp\|} \tag{47}$$

Donde cos (u, p) es el cos del ángulo entre u y p. primero note que si u y p son paralelos, entonces la ecuación 47 se reduce a $\|r\| = 1$, y allí no existirá reset. Mientras no exista salida de F_2, la ecuación 36 muestra que u = p, y no existiría reset para este caso.

Suponga ahora que F_2 tiene una salida de alguna unidad ganadora, y que los patrones de entrada necesitan ser aprendidos o codificados por la unidad F_2. No se quiere un reset en este caso. De la ecuación 36 se tiene que $p = u + dz_j$, donde la J_{esima} unidad en F_2 es la ganadora y $z_J = (z_{iJ}, z_{iJ}, ..., z_{MJ})^t$. si se inicializa todos los pesos Top-Down z_{ij} en cero, entonces la salida inicial de F_2 no va a tener efecto en el valor de *p*; *p* se va a mantener igual a *u*.

Durante el proceso de aprendizaje, z_J se convierte paralelo a *u* de acuerdo a la ecuación 44. Además, *p* también se convierte paralelo a *u*, y otra vez $\|\mathrm{r}\| = 1$ y no hay reset.

Así como ART1, una suficiente diferencia entre el vector de entrada Botton-Up y la plantilla Top-Down da resultando un reset. En ART2, el patrón Botton-Up es tomado del subnivel *u* de F_1 y la plantilla Top-Down es tomado en *p*.

Ya se vio que los pesos Top-Down deben estar inicializados en cero. La inicialización de los pesos Botton-Up se verá a continuación.

2.4.6 Inicialización LTM Botton-Up

Se ha discutido la modificación de los rastros LTM, o pesos en el caso de rápido aprendizaje. Vamos a examinar el comportamiento dinámico de los pesos Botton-Up durante un proceso de aprendizaje. Asumimos que un nodo particular de F_2 ha previamente codificado un vector de entrada tal que $z_{ji} = u_i/(1-d)$, y, por lo tanto, $\|z_J\| = [\|u\|/(1-d)] - [1/1-d]$, donde z_j es el vector de los pesos Botton-Up en la J_{esimo} nodo F_2. Suponga que el mismo nodo gana por un patrón de entrada un poco diferente, uno el cual el grado de diferencia no es suficiente para causar un reset. Entonces, los pesos Botton-Up van a se decodificados para concordar el nuevo vector de entrada. Durante este proceso de recodificación dinámico, $\|z_J\|$ puede disminuir antes de regresar al valor $[1/1-d]$. Durante este periodo de disminución, $\|\mathrm{r}\|$ va a

disminuir también. Si otros nodos han tenido sus valores de pesos inicializados tal que $\|z_J(0)\| > 1/1-d$, entonces la red debe cambiar los ganadores en el medio del proceso de aprendizaje.

Se debe, por lo tanto, inicializar los vectores de peso Botton-Up como:

$$\|z\| < \frac{1}{1-d}$$

Se puede realizar esta inicialización, poniendo los pesos a números pequeños aleatorios. Alternativamente, podemos utilizar la inicialización

$$z_{ji}(0) < \frac{1}{(1-d)\sqrt{M}} \tag{48}$$

Este esquema anterior tiene una apariencia de una inicialización uniforme. Además, si se usa la igualdad, entonces los valores iníciales serán tan largos como sea posible. Haciendo los valores iníciales tan largos perjudica a la red hacia nodos sin compromisos. Aunque el parámetro de vigilancia este muy bajo para causar un reset, de otra manera la red va a escoger un nodo sin compromiso en vez de un nodo que tenga una mala concordancia. Este mecanismo ayuda a estabilizar la red en contra de la recodificación constante.

Similares argumentos llevan a un contraste en los parámetros *c, d*.

$$\frac{cd}{1-d} > 1 \tag{49}$$

Mientras el radio se acerca a uno, la red se vuelve mas sensitiva a discordancias por que el valor de $\|r\|$ disminuye a un valor pequeño, y las otras cosas siguen igual.

2.4.7 Resumen del Procesamiento de ART2

Se considera solamente las soluciones asintóticas para las ecuaciones dinámicas, y el modo de rápido aprendizaje. M va a ser el número de unidades en cada subcapa F_1, y N será el número de unidades en F_2. Los paramentos son escogidos de acuerdo a las siguientes restricciones

$$a, b > 0.$$
$$0 \le d \le 1$$
$$\frac{cd}{1-d} \le 1$$
$$0 \le c \le 1$$
$$0 < p < 1$$
$$e << 1$$

Los pesos Top-Down son inicializados en cero

$$z_{ij}(0) = 0$$

Los pesos Botton-Up son inicializados de acuerdo a

$$z_{ij}(0) \le \frac{1}{(1-d)\sqrt{M}}$$

Ahora estamos listos para procesar los datos

1. Inicializar todas las salidas de las capas y subcapas en vectores cero y establecer un ciclo de conteo inicializado a un valor de uno

2. Aplicar un patrón de entrada I a la capa *w* de F_1. La salida de esta capa es.

$$w_i = I_i + auj$$

3. Propagar hacia la subcapa x.

$$x_i = \frac{w_i}{e + \|w\|}$$

4. Propagar hacia la subcapa v.

$$v_i = f(x_i) + bf(qi)$$

Note que el segundo término es cero a través del primer paso, así como es cero para ese tiempo

5. Propagar hacia la subcapa u

$$u_i = \frac{v_i}{e + \|v\|}$$

6. Propagar hacia la subcapa p

$$p_i = u_i + dz_{iJ}$$

Donde el J_{esimo} nodo en F_2 es el ganador de la competencia en esa capa. Si F_2 es inactivó, $p_i = u_i$. Similarmente, si la red esta aun en su configuración inicial, $p_i = u_i$ porque $z_{ij}(0) = 0$

7. Propagar hacia la subcapa q.

$$q_i = \frac{p_i}{e + \|p\|}$$

8. Repetir desde el segundo paso hasta el séptimo tantas veces sea necesario hasta estabilizar los valores en F_1
9. Calcular la salida de capa r

$$r = \frac{u_i + cp_i}{e + \|u\| + \|cp\|}$$

10. Determinar si una condición de reset es indicada. Si $p/(e+\|r\|) > 1$, entonces enviar una señal de reset a F_2. Marcar cualquier nodo F_2 activo como inelegible para la competición, resetear el contador de ciclos a uno, y regresar al paso dos. Si no hay reset, y el contador de ciclos es uno, incrementar el contador de ciclos y continuar el paso 11. Si no hay reset, y el contador de ciclos es mas grande que uno, entonces se saltará al paso 14, en donde una resonancia ha sido establecida
11. Propagar la salida de la subcapa p a la capa F_2. Calcular las entradas de red para F_2

$$T_j = \sum p_i z_{ij}$$

12. Solamente el nodo F_2 ganador tiene una salida diferente a cero.

$$g(T_j) = \begin{cases} d & T_j = \max_K \{T_k\} \\ 0 & \text{otro valor} \end{cases}$$

Todos los nodos marcados como inelegible por las señales previas de reset no participan en la competencia.

13. Repetir pasos 6 hasta el 10

14. Unificar los pesos Botton-Up en la unidad F_2 ganadora.

$$z_{Ji} = \frac{u_i}{1-d}$$

15. Modificar los pesos Top-Down en la unidad F_2 ganadora.

$$z_{iJ} = \frac{u_i}{1-d}$$

16. Remover el vector de entrada. Restaurar todas las unidades F_2 inactivas.

    Regresar al paso uno con un nuevo patrón.

2.5 ART3[27]

Es Un modelo para implementar una búsqueda paralela de códigos de reconocimiento de patrones comprimidos o distribuidos en una arquitectura de red neuronal. El procesamiento de búsqueda funciona bien ya sea con rápido aprendizaje o de aprendizaje lento y robustamente puede con series de patrones de entradas asíncronas en tiempo real.

El procesamiento de búsqueda emerge cuando propiedades computacionales de sinapsis químicas, como una acumulación de transmisores, liberación, inactivación y modulación, pueden solucionarse con una arquitectura llamada ART3. Análogos formales de iones tales como Na^{-} y Ca^{2-} controla interacciones con retroalimentaciones no lineales que permite a los transmisores pre-sinápticos dinámicos modelar la representación pos-sináptica de la memoria a corto plazo STM de un código de reconocimiento de patrón. La retroalimentación reforzada puede modular el procesamiento de búsqueda alterando el parámetro de vigilancia de ART3 o directamente llamar el mecanismo de búsqueda. El proceso de búsqueda es una forma de probar hipótesis capas de descubrir representaciones apropiadas de un ambiente de entrada no estacionario.

[27] Comparar con el articulo http://cns.bu.edu/Profiles/Grossberg/CarGro1990NN.pdf

2.6 FUZZY ART[28]

Es un modelo capaz de un aprendizaje rápido y estable de reconocimiento de categorías en respuestas a secuencias arbitrarias de patrones de entradas análogos o binarios. Fuzzy ART incorpora cómputos de la teoría Fuzzy en la red ART1, el cual aprende a categorizar solamente patrones de entradas binarias. La generalización de aprender ambos patrones de entradas análogos y binarios es archivada por reemplazar apariencias del operador intersección $(\cap)$ en ART1 por el operador MIN $(\wedge)$ de la teoría Fuzzy. El operador mínimo reduce al operador intersección en el caso binario. La proliferación de categorías ens prevenida normalizando vectores de entrada en el estado de procesamiento. Un procedimiento de normalización llamado codificación complemento (Complement Coding), lleva a la teoría simétrica en el cual el operador MIN $(\wedge)$ y el operador MAX $(\vee)$ de la teoría FUZZY juega roles complementarios. La codificación complemento usa células ON y células OFF para representar el patrón de entrada, y preserva amplitudes de características individuales mientras se normaliza el vector total de células ON y células OFF. El aprendizaje es estable porque todos los pesos adaptivos pueden solamente decrecer en el tiempo. Decrecer los pesos corresponde a incrementar los tamaños de las categorías "Cajas". Un valor de vigilancia pequeño lleva a largas Cajas de categorías. El aprendizaje se define cuando el espacio de entrada es cubierta por Cajas. Con un rápido

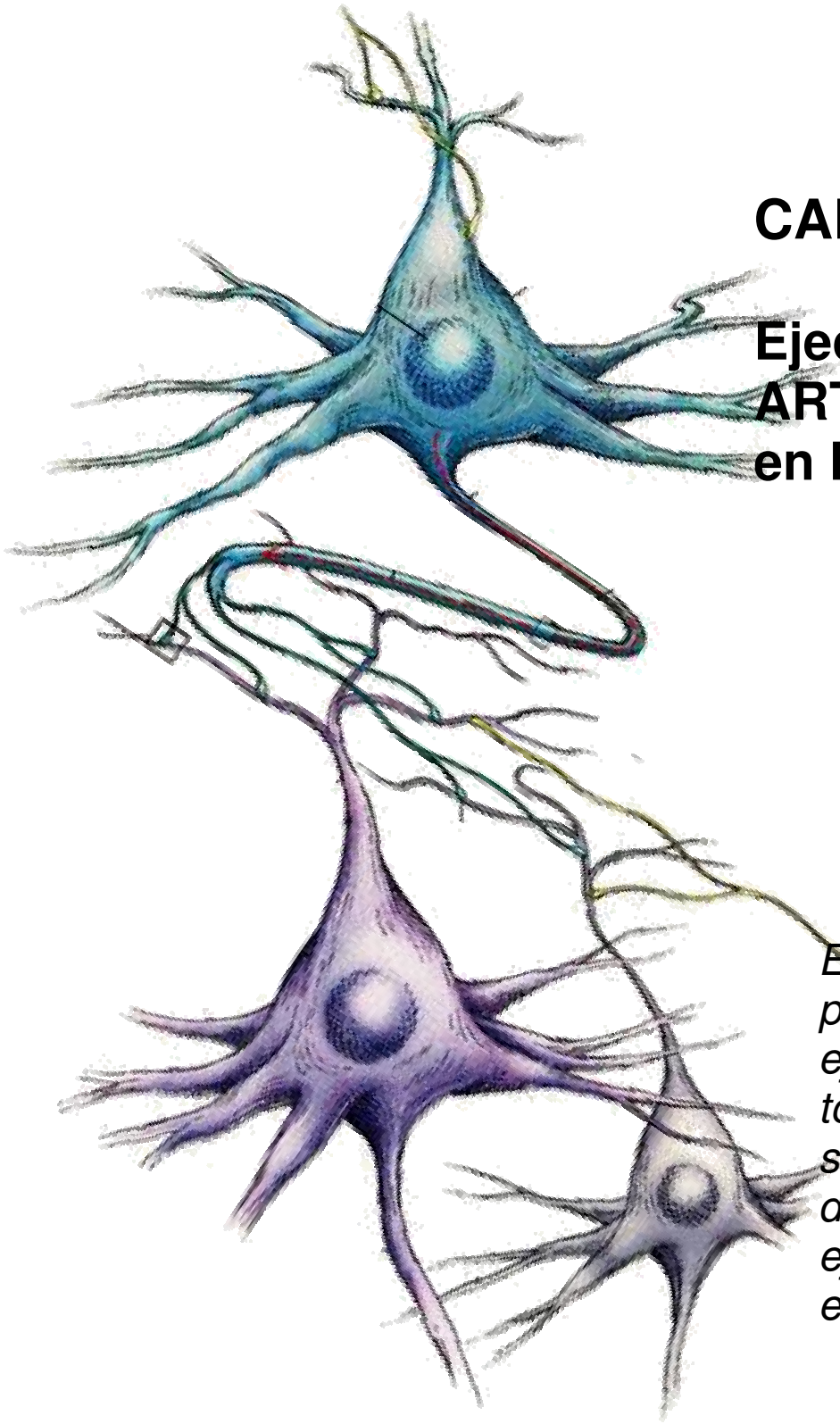
[28]Compara con el articulo http://cns.bu.edu/Profiles/Grossberg/CarGroRos1991NNFuzzyART.pdf

aprendizaje y una conjunto de entrada infinita de composición y tamaño arbitrario, el aprendizaje se estabiliza después de solamente una presentación de cada patrón de entrada. La opción rápida-confiabilidad y lenta-recodificación combina el rápido aprendizaje una regla olvidada que almacena intermediariamente la memoria del sistema en contra de ruido. Usando esta opción, raros eventos pueden ser rápidamente aprendidos y las memorias aprendidas previamente no son rápidamente borradas en respuestas a fluctuaciones de entrada estadísticamente no confiables.

2.7 ARTMAP[29]

ARTMAP autónomamente aprende a clasificar arbitrariamente a muchos vectores ordenados aleatoriamente en categorías de reconocimiento basado en predicciones exitosas. Este sistema de aprendizaje supervisado es construido de un par de módulos de ART_a y ART_b que son capaces del reconocimiento de categorías estables y autoorganizativas en respuesta de secuencias arbitrarias de patrones de entrada. Durante la prueba de entrenamiento, el modulo ART_a recibe una corriente de patrones de entrada $\left[a^p\right]$, y ART_b recibe una corriente de patrones de entrada $\left[b^p\right]$, donde b^p es la correcta predicción dada a ***p*.** Estos módulos de ART son unidos con una red de aprendizaje asociativa y un control interno que asegura la operación del sistema autónomo en tiempo real. Durante estas pruebas, el patrón sobrante a^p es presentado sin b^p y sus predicciones en ART_b son comparados con b^p.

[29]Comparar con el articulo http://cns.bu.edu/Profiles/Grossberg/CarGroRey1991NN.pdf



CAPITULO 3

Ejecuciones de ART Basadas en MATLAB

En este capitulo presentamos ejemplos del toolbox de matlab sobre algunos tipos de ART, así como ejemplos basados en MATLAB

3. EJECUCIONES BASADAS EN MATLAB

Para las ejecuciones basadas en Matlab se utilizo la versión 5.3

3.1 IMPLEMENTACIÓN DE UNA RED NEURONAL FUZZY ART.[30]

3.1.1 Descripción Del Sistema:

El conjunto de funciones es usado para crear, entrenar y usar una red Fuzzy ART para categorizar un conjunto de datos. Todas estas funciones fueron elaboradas utilizando Matlab 5.3, mas no se empleo ningún toolbox.

Mientras todas estas funciones son necesarias solamente la mitad de ellas son llamadas por el usuario, estas son las siguientes

Funciones permitidas por el usuario

ART_Categorize

ART_Complement_Code

ART_Create_Network

ART_Learn

Las cuatro funciones que quedan son usadas para modularizar la estructura del sistema. Estas funciones son relacionadas a diferentes componentes de la teoría de la resonancia adaptativa.

[30]Para descargar y observar los archivos de este ejemplo ver la pagina http://www.mathworks.com/matlabcentral/fileexchange/loadFile.do?objectId=4306&objectType=file

- **ART_Activate_Categories,** Esencialmente provee una activación Botton-Up a la capa F_2 para una entrada dada
- **ART_Add_New_Category,** Es usada después de una serie de reset de discordancias con el fin de crear una nueva neurona en F_2 para codificar la entrada actual.
- **ART_Calculate_Match**, es usada para determinar el grado de concordancia entre una entrada dada y la categoría codificada por la neurona F_2
- **ART_Update_Weights**, Es usado para actualizar la matriz peso durante el aprendizaje después que se ha alcanzado una resonancia.

3.1.2 Funciones Incluidas En Este Directorio:

ART_Activate_Categories – Realiza la activación de categorías en la red

ART_Add_New_Category – Añade un nuevo elemento de categoría a la red ART

ART_Calculate_Match – Calcula el grado de concordancia entre una entrada dada y una categoría

ART_Categorize – Usa una red ART entrenada para categorizar un conjunto de datos

ART_Complement_Code – Complementa códigos de la entrada dada

ART_Create_Network – Crea una red ART.

ART_Learn – Entrena una red ART dada en un conjunto de datos.

ART_Update_Weights – Actualiza la matriz peso de la red.

A. ART_ACTIVATE_CATEGORIES:

Esta función regresa un vector de activación de categorías para el vector de entrada dado, matriz peso y el valor *bias*.

Los parámetros de entrada son los siguientes:

- INPUT, es un vector de tamaños NumFeatures (Parámetro de tamaño) que contiene la señal de entrada en la red.
- WEIGHT, es una matriz de tamaño NumFeatures (Parámetro de tamaño) por NumCategories (Parámetro de rango) el cual mantiene los pesos de la red.
- BIAS, es la constante que es usada para diferenciar entre valores de activación de categorías muy similares.

El tamaño del vector INPUT debes ser igual al numero de filas en la matriz WEIGHT, y el valor BIAS debe estar dentro del rango[0,1] (sin embargo los valores mas cercanos a cero son los mejores).

El parámetro de retorno es el siguiente:

- El CATEGORYACTIVATION es un vector de tamaño NumCategories que mantiene el valor activación para cada categoría.

- PROGRAMA PRINCIPAL:

```
Function categoryActivation=ART_Activate_Categories(input,weight, bias)
```

Nos aseguramos que el usuario suministro los parámetros requeridos

```
if(nargin ~= 3)
error('You must specify the 3 input parameters.');
end
```

Chequeamos el tamaño y el rango de los parámetros

```
[numFeatures, numCategories] = size(weight);
if(length(input) ~= numFeatures)
    error('The length of the input and rows of the weights do not match.');
end
if((bias < 0) | (bias > 1))
    error('The bias must be within the range [0, 1].');
end
```

Establecer la variable retorno

```
categoryActivation = ones(1, numCategories);
```

Calcular la activación para cada categoría,. Esto es hecho de acuerdo a la ecuación siguiente

Activation(j) = |Input^Weight(j)| / (bias + |Weight(j)|)

```
for j = 1:numCategories
    matchVector = min(input, weight(:, j));
    weightLength = sum(weight(:, j));
    categoryActivation(1, j) = sum(matchVector) / (bias + weightLength);
end
return
```

B. ART_ADD_NEW_CATEGORY:

Esta función añade una nueva categoría a la matriz peso dada

RESIZEDWEIGHT = ART_Add_New_Category(WEIGHT), esta función retorna una nueva matriz peso el cual es idénticas a la matriz peso dada, excepto que este contiene una categoría mas el cual es iniciado a uno.

El parámetro de entrada es como sigue

- WEIGHT, es una matriz de tamaño NumFeatures (Parámetro de tamaño) por NumCategories (Parámetro de rango) el cual mantiene los pesos de la red.

El parámetro de retorno es el siguiente.

- RESIZEDWEIGHT, es una matriz de tamaño NumFeatures (Parámetro de tamaño) por NumCategories (Parámetro de rango) mas uno, el cual mantiene los pesos de la matriz vieja añadiéndole una categoría nueva para todos los valores de uno.

- PROGRAMA PRINCIPAL:

```
function resizedWeight = ART_Add_New_Category(weight)
```

Nos aseguramos que el usuario especifico la matriz peso

```
if(nargin ~= 1)
   error('You must specify the weight matrix parameter.');
end
```

Crear el retorno de la matriz WEIGHT con las dimensiones correctas

```
[numFeatures, numCategories] = size(weight);
newCategory = ones(numFeatures, 1);
resizedWeight = [weight, newCategory];
return
```

C. ART_CALCULATE_MATCH:

ART_Calculate_Match, Calcula el valor comparación de una entrada a una categoría

MATCH = ART_Calculate_Match(INPUT, WEIGHTVECTOR, esta función retorna un valor el cual representa la cantidad de concordancias entre una entrada y una categoría dada.

Los parámetros de entrada son los siguientes:

- INPUT, es un vector de tamaños NumFeatures (Parámetro de tamaño) que contiene la señal de entrada en la red.
- WEIGHTVECTOR, es una matriz de tamaño NumFeatures, el cual mantiene los pesos de la red para una categoría dada.

El tamaño del vector Input debe ser igual al tamaño del WeightVector.

Los parámetros de retorno son los siguientes

- MATCH es una medida del grado de concordancia de entrada y la categoría actual

- PROGRAMA PRINICIPAL

```
function match = ART_Calculate_Match(input, weightVector)
```

Inicializar las variables globales

```
match = 0;
numFeatures = length(input);
```

Se debe asegurar que el vector peso es apropiado para la entrada

```
if(numFeatures ~= length(weightVector))
   error('The input and weight vector lengths do not match.');
end
```

Calcular la concordancia entre la entrada dada y el vector peso. Esto es realizado de acuerdo a la siguiente ecuación

Match = |Input^WeightVector| / |Input|

```
matchVector = min(input, weightVector);
inputLength = sum(input);
if(inputLength == 0)
   match = 0;
else
   match = sum(matchVector) / inputLength;
end
return
```

D. ART_CATEGORIZE:

ART_Categorize, usa una red ART para categorizar el dato de entrada dada CATEGORIZATION = ART_Categorize(ART_NETWORK, DATA) esta función usa una red ART para categorizar la entrada de datos dada con el parámetro de vigilancia especificado. Cada muestra de datos es presentado a la red, el cual categoriza cada muestra. La función retorna la categorización de cada submuestra. Si la categorización de la submuestra requiere que una nueva categoría sea creada la categoría para esa muestra se coloca a -1.

Los parámetros de entrada son los siguientes:

- ART_NETWORK: Es una red ART entrenada. Esta debe ser creada con ART_Create_Network().
- DATA: Es el dato de categorización a ser presentado a la red. Esta es una matriz de tamaño NumFeatures por NumSamples

Lo parámetros de Retorno:

- CATEGORIZATION: Es un vector de tamaño NumSamples que mantiene la categoría en el cual la red ART coloco cada muestra.

- PROGRAMA PRINCIPAL:

```
function categorization = ART_Categorize(art_network, data)
```

Hay que asegurarse que el usuario especifica los parámetros de entrada

```
if(nargin ~= 2)
   error('You must specify both input parameters.');
end
```

Hay que asegurarse que los datos estén apropiados para la red dada

```
[numFeatures, numSamples] = size(data);
if(numFeatures ~= art_network.numFeatures)
   error('The data does not contain the same number of features as the network.');
end
```

Hay que asegurarse que la Vigilancia este dentro del rango [0,1]

```
if((art_network.vigilance <= 0) | (art_network.vigilance > 1))
   error('The vigilance must be within the range (0, 1].');
end
```

Cofigurar las Variables de retorno

```
    categorization = ones(1, numSamples);
```

Clasificar y aprender cada muestra

```
    for sampleNumber = 1:numSamples
```

Tomar la muestra de dato actual

```
 currentData = data(:, sampleNumber);
```

Activar las categorias para estas muestra

```
    bias = art_network.bias;
      categoryActivation = ART_Activate_Categories(currentData, art_network.weight, bias);
```

Categorizar las activaciones en orden desde más alto al más bajo. Esto nos permite un fácil acceso al paso a través de las categorías.

```
    [sortedActivations, sortedCategories] = sort(-categoryActivation);
```

Ir a través de cada categoría en la lista clasificada mirando para la mejor concordancia

```
     resonance = 0;
   match = 0;
   numSortedCategories = length(sortedCategories);
   currentSortedIndex = 1;
   while(~resonance)
```

Ir a la categoría basada en el índice clasificado

```
   currentCategory = sortedCategories(currentSortedIndex);
```

Obtener el vector peso actual de la lista de categorías clasificadas.

```
   currentWeightVector = art_network.weight(:, currentCategory);
```

Calcular la concordancia entregada de la muestra de dato actual y el vector peso.

```
     match = ART_Calculate_Match(currentData, currentWeightVector);
```

Chequear para ver si la concordancia es más grande que la vigilancia

```
if((match > art_network.vigilance) | (match >= 1))
```

Si es así, la categoría actual codifica la entrada.

Entonces, se debe inducir resonancia

```
categorization(1, sampleNumber) = currentCategory;
resonance = 1;
else
```

Si no, se escoge la siguiente categoría en la lista de categorías clasificadas.

Si la categoría actual es la ultima en la lista, se coloca la categoría para que el valor del retorno a -1 y se induce una resonancia

```
if(currentSortedIndex == numSortedCategories)
    categorization(1, sampleNumber) = -1;
    resonance = 1;
else
    currentSortedIndex = currentSortedIndex + 1;
     end
  end
 end
end
     return
```

E. ART_COMPLEMENT_CODE:

Complementa códigos de datos para el uso con una red ART.

COMPLEMENTCODEDDATA = ART_Complement_Code(DATA), esta función complementa los códigos de los datos dados donde el complemento de x es 1 – x. Por ejemplo, supongamos que los datos son los siguientes.

Data = 0.3 0.2 0.6
0.4 0.1 0.8

Entonces la clasificación complementaria será la siguiente

complementCodedData = 0.3 0.2 0.6
0.7 0.8 0.4
0.4 0.1 0.8
0.6 0.9 0.2

Los parámetros de entrada son los siguientes:

- DATA: es una matriz de tamaño NumFeatures por NumSamples que mantiene los datos a ser complementados. La codificación de complemento trabaja en cada columna (cada muestra).

Los parámetros de retorno es el siguiente:

- COMPLEMENTCODEDDATA: Es el dato que va a ser complementado. Esta es una matriz de tamaño 2*NumFeatures-by-NumSamples.

- PROGRAMA PRINCIPAL:

```
function complementCodedData = ART_Complement_Code(data)
```

Determina el tamaño del dato.

```
    [numFeatures, numSamples] = size(data);
```

Crea la variable de retorno

```
    complementCodedData = ones(2*numFeatures, numSamples);
```

Realiza la codificación complemento para cada muestra

```
    for j = 1:numSamples
      count = 1;
```

```
        for i = 1:2:(2*numFeatures)
            complementCodedData(i, j) = data(count, j);
            complementCodedData(i + 1, j) = 1 - data(count, j);
            count = count + 1;
        end
    end
return
```

F. ART_CREATE_NETWORK:

Crea una nueva red ART.

ART_NETWORK = ART_Create_Network(NUMFEATURES), esta función crea una nueva red ART con el numero especificado de características. La red es creada para expandir el número de categorías tanto como se necesita. Por esto, la red ART puede crecer para abarcar nuevos datos de acuerdo al parámetro de vigilancia, el cual por defecto esta es 0.75. El número inicial de categorías es establecido a 1. El máximo número de categorías es por defecto 100. El bias por defecto es 0.000001, el numero de épocas por defecto es 100, y la rata de aprendizaje es por defecto 1.0 (rápido aprendizaje).

El parámetro de entrada es el siguiente:

- NUMFEATURES: Es el número de características que la red espera del dato de entrada. Este valor debe ser un número entero positivo.

Los parámetros de retorno son:

- ART_NETWORK: Es la estructura que mantiene toda la información para la red. Estés debe ser transmitida a ART_LEARN y ART_CATEGORIZE. El campo de esta estructura es numCategories, maxNumCategories, weight, vigilance, bias, numEpochs y learningRate.

- PROGRAMA PRINCIPAL:

```
function art_network = ART_Create_Network(numFeatures)
```

Hay que asegurarse que el usuario especifique los parámetros requeridos.

```
if(nargin ~= 1)
   error('You must specify a number of features.');
end
```

Comprobar los rangos de los parámetros de entrada

```
numFeatures = round(numFeatures);
if(numFeatures < 1)
   error('The number of features must be a positive integer.');
end
```

Crear e inicializar los pesos de la matriz

```
weight = ones(numFeatures, 0);
```

Crear las estructuras y el retorno.

```
art_network = struct('numFeatures', {numFeatures}, 'numCategories', {0},
'maxNumCategories', {100}, 'weight', {weight}, ...
                'vigilance', {0.75}, 'bias', {0.000001}, 'numEpochs', {100}, 'learningRate',
{1.0});
return
```

G. ART_LEARN:

Entrena una red ART en la entrada de datos dada

[NEW_ART_NETWORK, CATEGORIZATION] = ART_Learn(ART_NETWORK, DATA), esta función entrena una red ART en la entrada de datos dada. Cada muestra de datos es presentada a la red, el cual categoriza y aprende para cada muestra. La función retorna una nueva red ART el cual ha aprendido el dato de entrada, junto con la categorización de cada muestra. Si el máximo número de categorías es alcanzado y un elemento debe ser clasificado con una nueva categoría que no puede ser creada, la categoría es colocada a -1. Los parámetros de entrada son los siguientes:

- ART_NETWORK: Es la red ART a ser entrenada. Esta debe ser creada con ART_Create_Network().
- DATA: Es el dato de entrenamiento a ser presentado a la red. Es una matriz de tamaño NumFeatures por NumSamples.

Los parámetros de retorno son los siguientes.

- NEW_ART_NETWORK: Es una nueva red ART el cual ha aprendido el dato de entrada.
- CATEGORIZATION: Es un vector de tamaño NumSamples que mantiene la categoría en el cual la red ART coloco cada muestra.

- PROGRAMA PRINICPAL:

```
function [new_art_network, categorization] = ART_Learn(art_network, data)
```

Hay que asegurarse que el usuario especifique los parámetros de entrada

```
if(nargin ~= 2)
    error('You must specify both input parameters.');
end
```

Hay que asegurarse que el dato sea apropiado por la red dada.

```
[numFeatures, numSamples] = size(data);
if(numFeatures ~= art_network.numFeatures)
    error('The data does not contain the same number of features as the network.');
end
```

Hay que asegurarse que la vigilancia se encuentre dentro del rago [0 , 1].

```
if((art_network.vigilance <= 0) | (art_network.vigilance > 1))
    error('The vigilance must be within the range (0, 1].');
end
```

Hay que asegurarse que el numero de apocas sea un entero positivo

```
if(art_network.numEpochs < 1)
    error('The number of epochs must be a positive integer.');
end
```

Configurar las variables de retorno.

```
new_art_network = {};
categorization = ones(1, numSamples);
```

Ir a través del dato una por cada epoca

```
for epochNumber = 1:art_network.numEpochs
```

Esta variable nos permitirá mantener con el cambio de la red total debido al aprendizaje. Se inicializa el número de cambios a cero

```
numChanges = 0;
```

Clasifica y se aprende en cada muestra.

```
for sampleNumber = 1:numSamples
```

Se obtiene la muestra de dato actual.

```
currentData = data(:, sampleNumber);
```

Se activa las categorías para esta muestra. Esto es equivalente al proceso de Botton – Up en ART

```
bias = art_network.bias;
categoryActivation = ART_Activate_Categories(currentData,
                art_network.weight, bias);
```

Se clasifica las activaciones en orden de mayor a menor. Esto nos permitirá un acceso mas facil para pasar a través de las categorías

```
[sortedActivations, sortedCategories] = sort(-categoryActivation);
```

Ir a través de ada categoría en la lista clasificada buscando por la mejor concordancia. Este es un equivalente del procesamiento Bottom - Up – Top - Down en ART.

```
resonance = 0;
match = 0;
numSortedCategories = length(sortedCategories);
currentSortedIndex = 1;
while(~resonance)
```

Si aun no hay categorias, se debe crear alguna.

```
if(numSortedCategories == 0)
resizedWeight = ART_Add_New_Category(art_network.weight);
                [resizedWeight, weightChange] =
            ART_Update_Weights(currentData, resizedWeight, ...
```

```
            1, art_network.learningRate);
            art_network.weight = resizedWeight;
            art_network.numCategories = art_network.numCategories+1;
            categorization(1, sampleNumber) = 1;
            numChanges = numChanges + 1;
            resonance = 1;
            break;
        end
```

Obtener la categoría basado en el índice clasificado

```
        currentCategory = sortedCategories(currentSortedIndex);
```

Obtener el vector de peso actual proveniente de la lista de categorías clasificada

```
    currentWeightVector = art_network.weight(:, currentCategory);
```

Calcular la concordancia dada por la muestra de dato actual y el vector peso.

```
    match = ART_Calculate_Match(currentData, currentWeightVector);
```

Comprobar para ver si la concordanca es mas grande que la vigilancia.

```
    if((match > art_network.vigilance) | (match >= 1))
```

Si es así, la categoría actual debe codificar la, entonces, nosotros debemos actualizar los pesos y inducir una resonancia

```
    [art_network.weight, weightChange] = ART_Update_Weights(currentData,
    art_network.weight, currentCategory, art_network.learningRate);
    categorization(1, sampleNumber) = currentCategory;
```

Si ocurrió un cambio, incrementamos nuestro contador

```
            if(weightChange == 1)
                numChanges = numChanges + 1;
```

```
        end
        resonance = 1;
      else
```

Si no, escogemos la próxima categoría en la lista clasificada.

Si la categoría actual es la ultima de la lista, nos aseguramos que el máximo numero de categorías no halla sido alcanzado. Si es así, asignamos a la entrada una categoría de -1. Si el máximo no ha sido alcanzado, se crea una nueva categoría para la entrada, se actualizan los pesos, y se induce una resonancia.

```
        if(currentSortedIndex == numSortedCategories)
          if(currentSortedIndex == art_network.maxNumCategories)
            categorization(1, sampleNumber) = -1;
            resonance = 1;
          else
            resizedWeight =
            ART_Add_New_Category(art_network.weight);
            [resizedWeight, weightChange] =
            ART_Update_Weights(currentData, resizedWeight, ...
            currentSortedIndex + 1, art_network.learningRate);
            art_network.weight = resizedWeight;
            art_network.numCategories =
            art_network.numCategories + 1;
          categorization(1, sampleNumber)=currentSortedIndex+ 1;
            numChanges = numChanges + 1;
            resonance = 1;
          end
```

```
            else
              currentSortedIndex = currentSortedIndex + 1;
            end
          end
        end
      end
```

Si la red no cambio del todo durante su ultima época, entonces nosotros alcanzamos un equilibrio. Ahora, nosotros podemos detenernos de entrenar

```
      if(numChanges == 0)
        break;
      end
    end
    fprintf('The number of epochs needed was %d\n', epochNumber);
```

Llenamos la nueva red con los valores apropiados

```
    new_art_network = art_network;
    return
```

H. ART_UPDATE_WEIGHTS:

Actualiza la matriz peso de una red ART

[UPDATEDWEIGHT, WEIGHTCHANGE] = ART_Update_Weights(INPUT, WEIGHT, CATEGORYNUMBER, LEARNINGRATE), esta función retorna una nueva matriz peso que ha aprendido la entrada, en una categoría dada, así como el valor correspondiente a si ha cambiado o no la matriz peso (0 = no cambio; 1 = cambio).

Los parámetros de entrada son.

- INPUT: es un vector de tamaño NumFeatures que contiene la señal de entrada a la red.
- WEIGHT, es una matriz de tamaño NumFeatures por NumCategories el cual mantiene los pesos de la red.
- CATEGORYNUMBER, es el numero de la categoría que codifica la entrada actual
- LEARNINGRATE, es la rata al cual la red debe aprender nuevas entradas.

El tamaño del vector INPUT debe ser igual al numero de filas en la matriz WEIGHT, el CATEGORYNUMBER debe estar en el rango entre [1, numcategories], y el LEARNINGRATE debe estar en el rango [0,1].

Los parámetros de retorno son los siguientes.

- UPDATEDWEIGHT, es una matriz del tamaño NumFeatures por NumCategories que mantiene los nuevos pesos de la red
- WEIGHTCHANGE, es un valor (0 o 1) el cual retransmite si la matriz peso cambio o no durante su actualización. 0 representa que no hay cambios y uno que si hay cambios.

- PROGRAMA PRINCIPAL:

```
function [updatedWeight, weightChange] = ART_Update_Weights(input, weight,
categoryNumber, learningRate)
```

Obtenemos el numero de características de la matriz peso

```
[numFeatures, numCategories] = size(weight);
```

Observamos los parámetros de entrada para rangos correctos

```
if(length(input) ~= numFeatures)
    error('The length of the input and rows of the weights do not match.');
end
if((categoryNumber < 1) | (categoryNumber > numCategories))
    error('The category number must be in the range [1, NumCategories].');
end
if((learningRate < 0) | (learningRate > 1))
error('The learning rate must be within the range [0, 1].');
end
```

Se modifica apropiadamente la categoría de la matiz peso de acuerdo con la regla siguiente:

```
FOR EACH i IN input
    IF input(i) < weight(i)(j) THEN
      weight(i)(j) = (a * input(i)) + ((1 - a) * weight(i)(j))
    ELSE
      weight(i)(j) does not change
    END IF
END FOR
```

%donde “a” es la rata de aprendizaje y “j representala categoria apropiada

```
weightChange = 0;
for i = 1:numFeatures
  if(input(i) < weight(i, categoryNumber))
```

```
        weight(i, categoryNumber) = (learningRate * input(i)) + ((1 - learningRate) * weight(i,
    categoryNumber));
        weightChange = 1;
      end
    end
```

Retornamos la matriz peso actualizada.

```
    updatedWeight = weight;
    return
```

3.1.3 Ejemplo De Funcionamiento

Para ver un ejemplo del uso de estas funciones.

A. ART_EXAMPLE

```
        input = [0.5, 0.3, 0.2, 0.91, 1.0;
                0.7, 0.4, 0.3, 0.55, 0.2];

        ccInput = ART_Complement_Code(input);
```

Esto produce una matriz como la siguiente

ccInput =

0.5000	0.3000	0.2000	0.9100	1.0000
0.5000	0.7000	0.8000	0.0900	0
0.7000	0.4000	0.3000	0.5500	0.2000
0.3000	0.6000	0.7000	0.4500	0.8000

```
        net = ART_Create_Network(4);
```

Esto produce una red como la siguiente

net =

```
         numFeatures: 4
        numCategories: 1
```

```
        maxNumCategories: 100
        weight: [4x1 double]
        vigilance: 0.7500
        bias: 1.0000e-006
        numEpochs: 100
        learningRate: 1
```

```
[newNet, cat] = ART_Learn(net, ccInput);
```

Esto produce una salida como la siguiente

```
newNet =

        numFeatures: 4
        numCategories: 3
        maxNumCategories: 100
         weight: [4x3 double]
         vigilance: 0.7500
         bias: 1.0000e-006
         numEpochs: 100
         learningRate: 1

 cat =
         1     1     2     3     3
```

Ahora que la red ha sido entrenada, se puede usarlo para categorizar una nueva entrada

```
newInput = [0.2; 0.4];
ccNewInput = ART_Complement_Code(newInput);
newCat = ART_Categorize(newNet, ccNewInput);
```

Esto produce una salida de

```
    newCat =
             2
```

3.2 EJEMPLO DEL TOOLBOX DE MATLAB: ALGORITMO ART1[31]

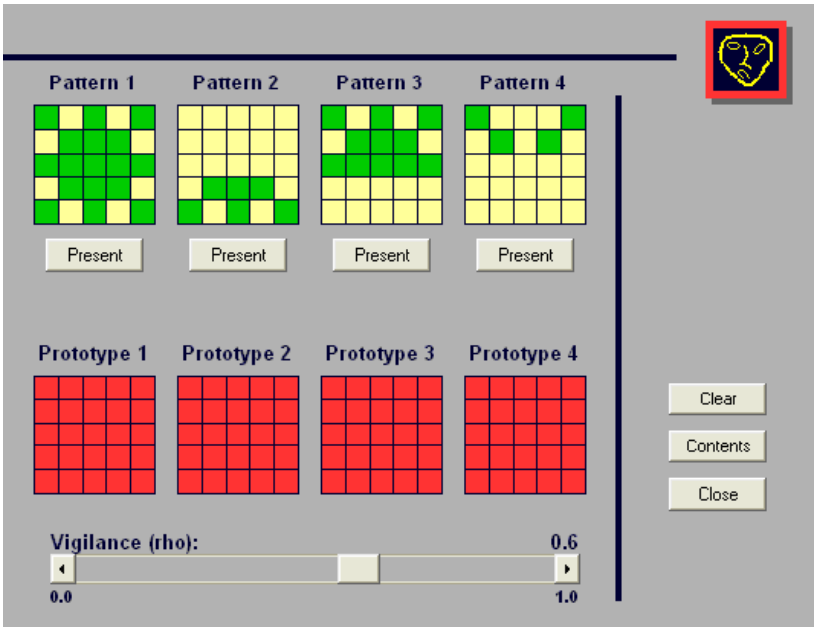


Figura 9. Ejemplo algoritmo ART1

3.2.1 Instrucciones

Este ejemplo el cual se encuentra en el toolbox de Matlab 5.3, consta de cuatro patrones de entrada que se pueden ir presentado a medida que se oprime clic en la tecla "present" debajo de cada uno respectivamente; Tenemos la opción de cambiar el parámetro de vigilancia de la red por medio de la barra que se encuentra debajo de la ventana del programa; Para volver a iniciar el estado de la red, se hace clic en la tecla "clear" al costado de la ventana (*Figura 9*).

3.2.2 Descripción

Este ejemplo nos muestra el comportamiento de una red ART1. El objetivo

[31] Ver Help Desk de Matlab V 5.3, algoritmo de ART1

de este, es observar como la red aprende y se comporta de acuerdo al parámetro de vigilancia escogido, y a medida que se van presentado los diferentes patrones, con el fin de observar el resultado final en los prototipos de la red.

3.2.3 Comentarios

En el ejemplo a medida que se presenta los patrones, la red va aprendiendo. Con una vigilancia dentro del rango de 0.0 y 0.3 (*Figura 10*), observamos que la red, de los cuatro patrones de entrada presentado, solo escoge 2 de estos como los prototipos finales, lo que quiere decir, que la red solamente reconocerá estos dos prototipos y los otros patrones serán asociados a alguno de estos.

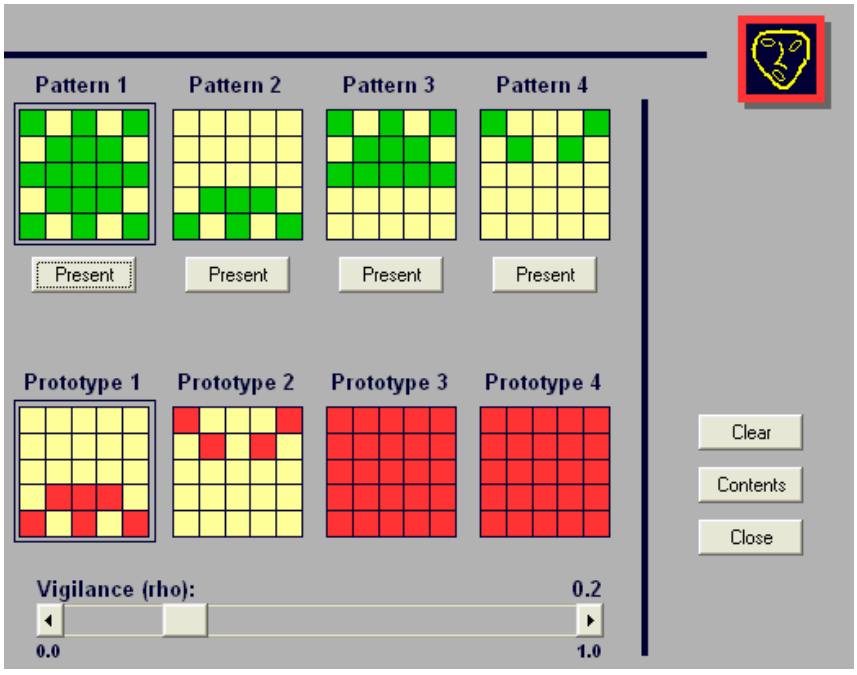


Figura 10. Parámetro de vigilancia entre los rangos de 0.0 y 0.3

Utilizando un parámetro de vigilancia dentro del rango de 0.4 a 0.6 (*Figura 11*) observamos que la red, de los cuatro patrones de entrada presentados, solo escoge 3 de estos como los prototipos finales, lo que quiere decir, que la red solamente reconocerá tres de los patrones de entrada.

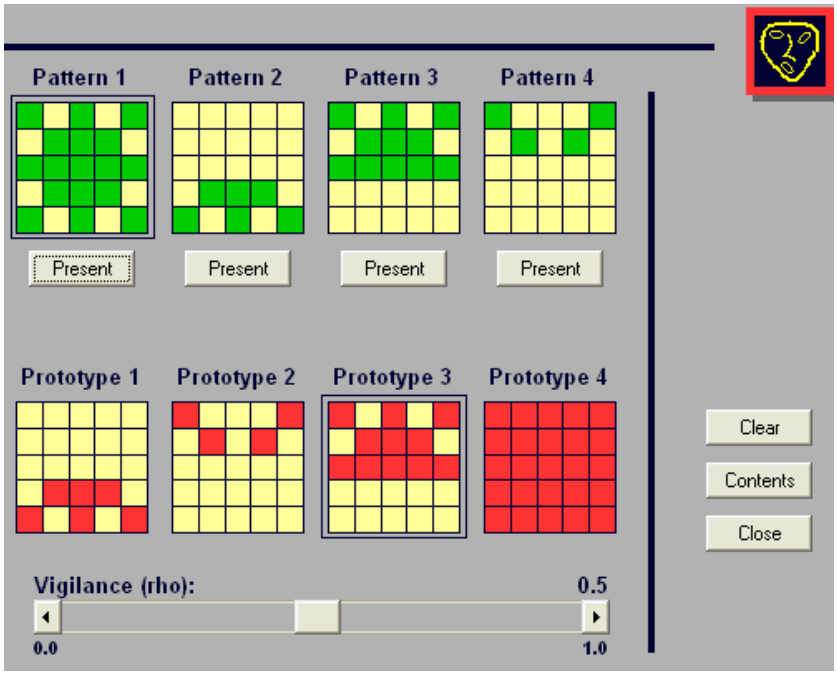


Figura 11. Parámetro de vigilancia entre los rangos de 0.4 y 0.6

Utilizando un parámetro de vigilancia dentro del rango de 0.7a 1 (*Figura 12*), observamos que la red escoge los cuatro patrones de entrada presentados como los prototipos finales, lo que quiere decir, que la red reconocerá todos los patrones de entrada

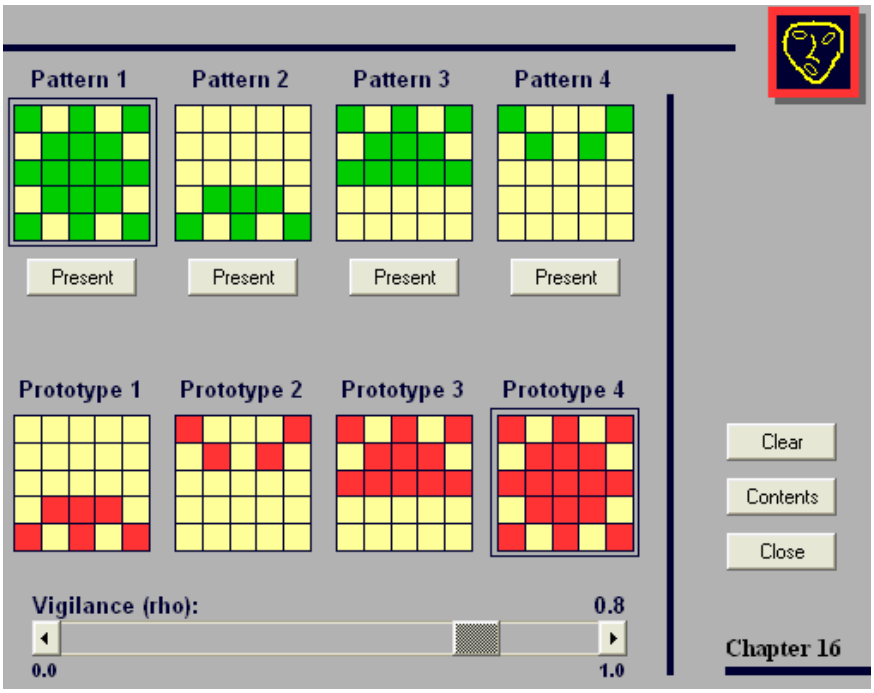
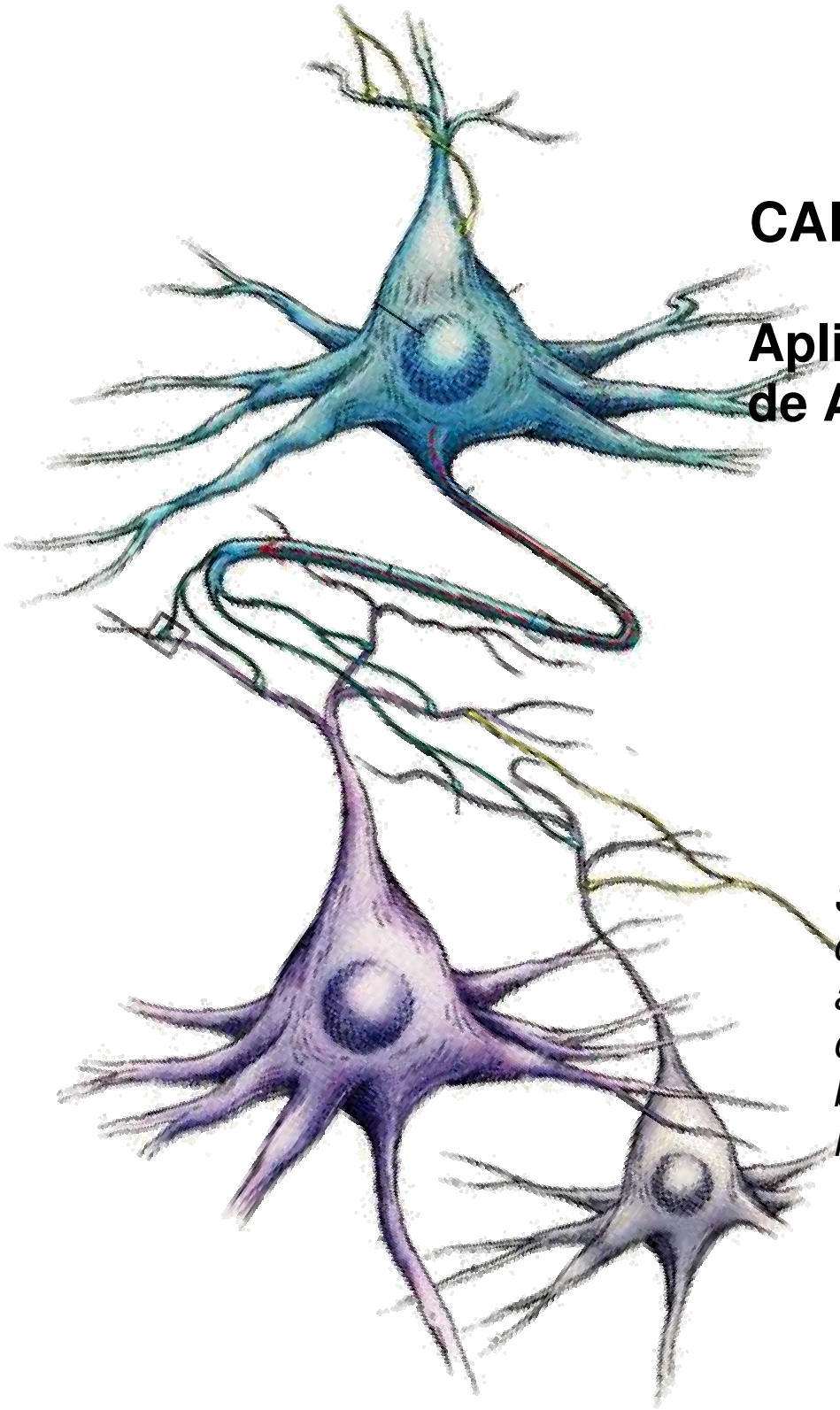


Figura 12. Parámetro de vigilancia entre los rangos de 0.7 a 1

Se puede concluir que en este sistema para que la red reconozca todos los patrones de entrada correctamente, el parámetro de vigilancia debe estar en el rango de 0.7 a 1.

Entonces se puede observar la importancia que tiene el patrón de vigilancia en las redes ART el cual nos va a discriminar las diferentes clases de patrones de entrada, y nos va a influir la salida de nuestro sistema, pero hay que tener en cuenta que a medida que este parámetro de vigilancia aumente, la red va a requerir mas neuronas; Por lo cual se debe tener una relación entre el parámetro de vigilancia y los patrones de entrada, con el fin de que la red reconozca correctamente todas las entadas sin crear demasiadas prototipos o categorías (categorías sobrantes por ruidos).



CAPITULO 4

Aplicaciones de ART

Se presenta diferentes aplicaciones, así como applets basados en el lenguaje JAVA

4. APLICACIONES DE ART [32]

Los sistemas ART son parte de una familia en crecimiento de redes auto-organizativas que tiene como características retroalimentación y aprendizaje de códigos. Áreas de aplicación tecnológicas incluyen diseño industrial, el control de robots móviles, reconocimiento de rostro, reconocimiento de objetivos, diagnósticos médicos, análisis de electrocardiograma, verificación de firmas, monitoreo de fallas de herramientas, análisis químicos, diseño de circuitos, análisis DNS/Proteínas, reconocimiento visual de objetos 3D, análisis musical y sísmico, reconocimiento sonar y radar. Un libro de Serrano-Gotarredona, Linares-Barranco, y Andreou (1998) discute la implementación de los sistemas ART como VLSI microchips. Las memorias ART también traducen a un sistema transparente de reglas IF-THEN el cual caracteriza el proceso de toma de decisiones y el cual debe ser usado para selecciones de características.

Debido al carácter de funcionamiento autónomo, sin supervisor, esta red ha sido también utilizada para modelar procesos biológicos y así poder estudiar y predecir su comportamiento. Estos principios han ayudado a explicar el comportamiento paramédico y los datos del cerebro en el área de percepción

[32] Comparar con el articulo http://cns.bu.edu/Profiles/Grossberg/CarGro2003HBTNN2.pdf

visual, reconocimiento de objetos, identificación de fuente auditiva, reconocimiento de palabras y voz, y control adaptativo de motor-sensor.

En la investigación realizada por *Carpenter*[33] se describe un sistema autónomo para percibir y producir voz de forma parecida a como lo haría un humano. El subsistema de percepción recibe la voz en forma de sonido, estableciendo, con la ayuda de una red ART2, diferentes códigos o categorías fonéticas. El subsistema de producción, mediante un proceso de imitación, transforma los códigos de voz oída en códigos de control de un motor para la generación de la voz.

Se presentarán algunos ejemplos de aplicaciones del modelo ART para el reconocimiento de patrones discretos y continuos.

4.1 RECONOCIMENTO DE IMÁGENES

En el capitulo 3 en la sección 3.2 se presentó el mecanismo de aprendizaje de la red ART discreta, se utilizó un ejemplo muy simple de clasificación de figuras compuestas por 30 puntos (píxeles). En este capitulo se analizará un ejemplo similar, pero de mayor complejidad. También se analizará la posibilidad de utilizar el modelo continuo (ART2) para el reconocimiento invariante de imágenes que han sufrido algún tipo de deformación, rotación,

[33]**Carpenter**, G. Carpenter y S, Grossberg. "The ART of Adaptive Pattern Recognition by a Self-Organizing Neural Network. IEEE Computer, Págs 77-88, Marzo, 1988

ampliación o desplazamiento.

En Carpenter[21], los autores del modelo presentan un ejemplo de aprendizaje de las letras del alfabeto por parte de una red ART discreta. El ejemplo que se presenta a continuación es similar, aunque en este caso se va a trabajar con números, en lugar de letras, y servirá para comprobar el efecto del parámetro de vigilancia en el establecimiento de categorías.

Se va a diseñar una red para clasificar figuras de dígitos representados mediante 7x6= 42 píxeles. Esta red consta de 42 neuronas de entrada a través de las cuales se recibe la figura que debe clasificar. La red debe responder indicando a qué clase, de las establecidas hasta ese momento, pertenece la letra. Si no pertenece a ninguna, deberá crear una nueva. Se fija un valor de 0.9 para el parámetro de vigilancia (*p*); es decir, se exige una semejanza mayor del 90% para considerar que dos figuras son de la misma clase.

El primer dígito que se aplica a la red es el 2. Como se trata de la primera información, se realiza el aprendizaje completo de la misma, actualizándose los pesos de las conexiones para almacenar este primer ejemplar, que será el representante o prototipo de la primera categoría establecida por la red (figura 13(a)).

Después se presenta el dígito 3. En este caso, la red comprobaría el grado de semejanza con respecto al único prototipo almacenado en la red (2). Al ser informaciones binarias, y suponiendo que un píxel negro se representa mediante un bit *1* en el vector de entrada *(E_k),* Y uno blanco con un bit 0. Se

puede comprobar el grado de semejanza aplicando la fórmula *Ek*- X / *Ek* . En el denominador se indica en número de píxeles negros, en este caso es de 17. En el numerador se indica el número de píxeles negros coincidentes al superponer el dígito de entrada con el prototipo con el que se compara (operación AND entre ambas figuras). En este caso, puede comprobarse que hay 11 píxeles coincidentes, por tanto la relación de semejanza es 11/17=0.65, menor que el *parámetro de vigilancia,* que se había fijado a 0.9, por tanto la red determinaría que el dígito 3 no pertenece a la categoría del 2, por lo que crearía en los pesos un nuevo prototipo (exactamente igual al numero **3**) para esta nueva categoría (Fig. 13(b)).

Cuando se introduce el dígito 8, se compara con los dos prototipos almacenados (2 y 3) obteniéndose una relación de semejanza respecto a 2 de 11/20=0.55 < 0.9, Y respecto a 3, de 17/20=0.85 < 0.9. Por tanto, se considera como prototipo de una nueva categoría (Fig. 13(c)).

Cuando se presenta la cuarta figura, un 8 distorsionado, al comparar con el prototipo 8 se obtiene una relación 19/20=0.95, por lo que se integra en la categoría del 8, ajustando entonces la red los pesos para adaptar el prototipo de esa categoría, incorporando alguna de las características' de la nueva figura. En realidad, el nuevo prototipo se obtiene aplicando una operación AND entre los dos números.

(fig. 13(d)).

Finalmente, cuando se presenta el 8 con otra distorsión (fig. 13(e)) la

semejanza respecto al prototipo anterior es de 17/19=0.89 < 0.9, con lo que no se considera de la misma categoría, creándose una nueva de la que esta figura pasa a ser su representante.

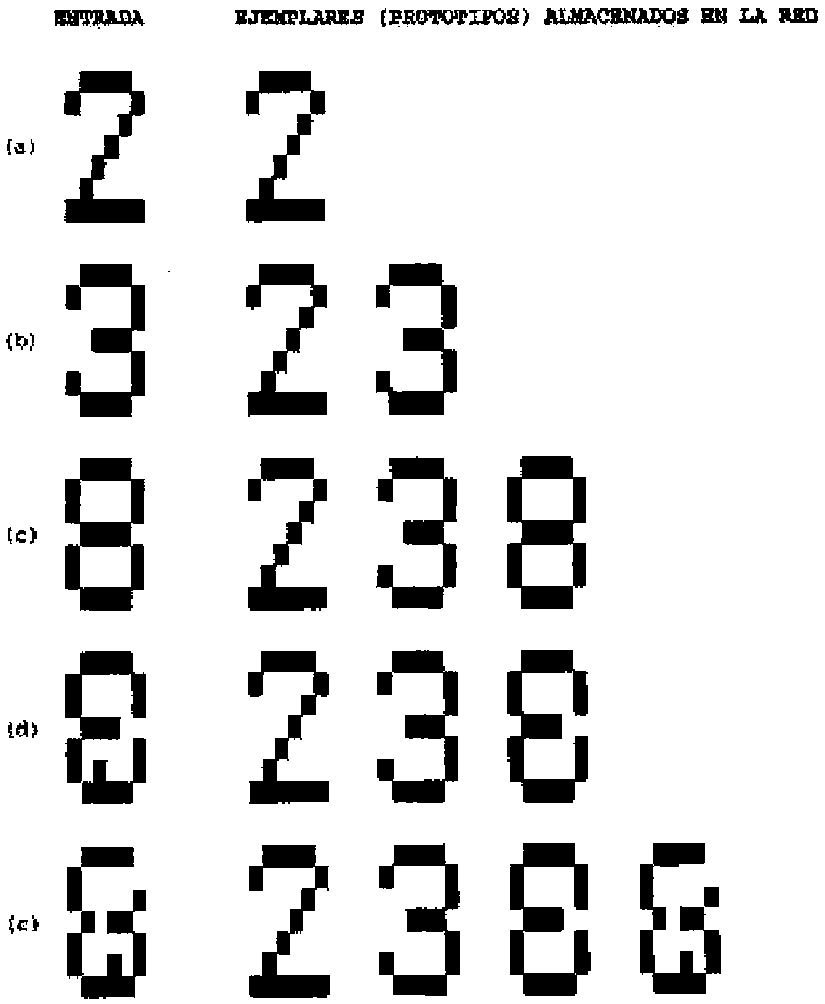


Figura 13. Categorías establecidas por la red cuando *p*=0.9[34]

Para comprobar el efecto del parámetro de vigilancia, puede analizarse lo que ocurriría si su valor fuese menor. En ese caso, se estaría relajando la condición de vigilancia, de tal forma que formarán parte de la misma

[34] Tomado de Bibliografía [1]. Capitulo 6, ART

categoría figuras que difieran en una mayor proporción que en el caso anterior. Si, por ejemplo, p=0.8, cuando se presenta la imagen del 8, la relación de semejanza con el 3 es 17/20=0.85, que es mayor que p, con lo que se consideran de la misma clase, perdiéndose, por tanto, el dígito 8 al establecer el nuevo prototipo con la operación AND entre ambos patrones (Fig. 14(e)). Lo que ocurre al presentar el resto de las imágenes puede observarse en la misma figura 7.

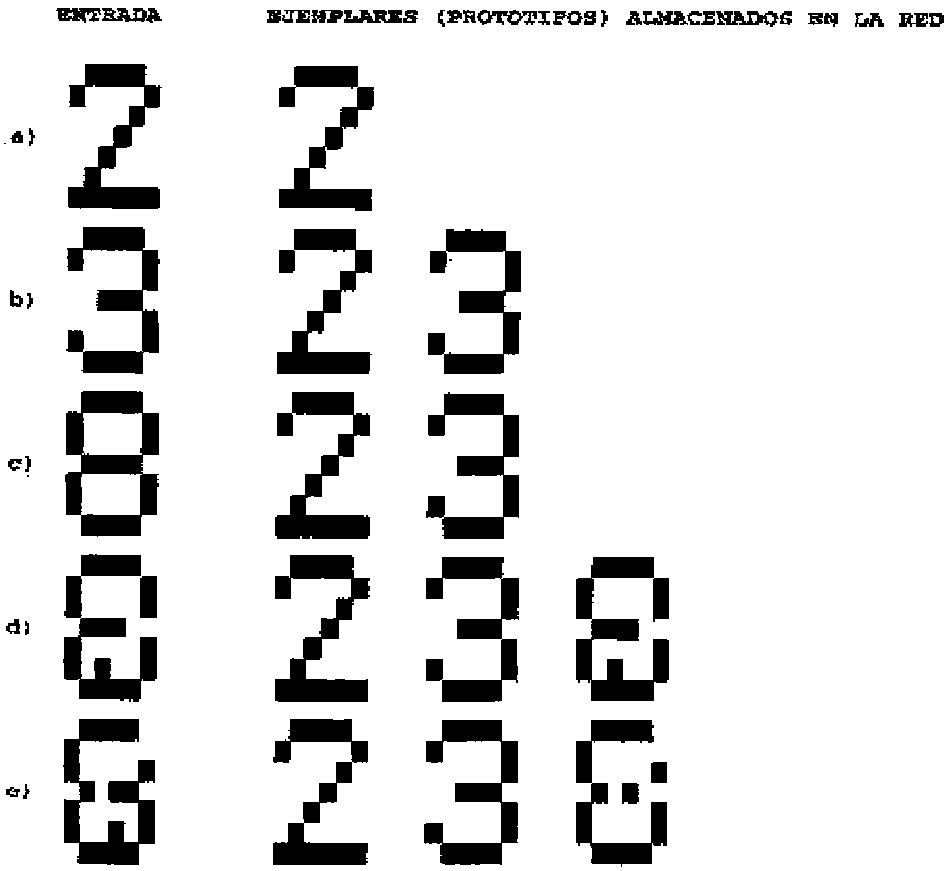


Figura 14. Categorías establecidas por la red cuando *p*=0.8[35]

[35] Tomado de Bibliografía [1]. Capitulo 6, ART

4.2 RECONOCIMIENTO DE SEÑALES ANALÓGICAS

Un ejemplo más complejo que el anterior se puede encontrar en *Carpenter*[36], donde se muestra como aplicación un sistema de reconocimiento invariante de imágenes obtenidas mediante sensores de radar láser. El sistema consta de una red ART2 y una etapa de preprocesamiento. En esta etapa previa se realizan las funciones: a) La recepción de la imagen radar b) Detección, marcado y completado del contorno de la imágenes mediante una red neuronal denominada *Boundary Contour System* Grossberg[37]; c) filtrado de la imagen aplicando la transformada de Fourier-Mellin, que obtiene una representación (espectro) de la imagen independiente de su posición, rotación o tamaño. La red ART2 recibe la imagen transformada y establece la categoría a la que pertenece.

En la *figura 15* se muestra la correcta clasificación de 24 imágenes de 4 tipos de vehículos en cuatro categorías. En este ejemplo se trabaja con un ruido en las imágenes de un 10%.

[36] G. Carpenter y S. Grossberg. "ART2: Self-Organization of Stable Category Recognition Codes for analog input Patterns". Proceedings of the IEEE first international conference on neural network. Vol II. 1987

[37] 1Grossberg. S. Grossberg y E. Mingolla. "Neural Dynamics of Forum Perception: Boundary Completion, Illusory Figures and Neon Color Spreading.. Psychological Review, 92 1985

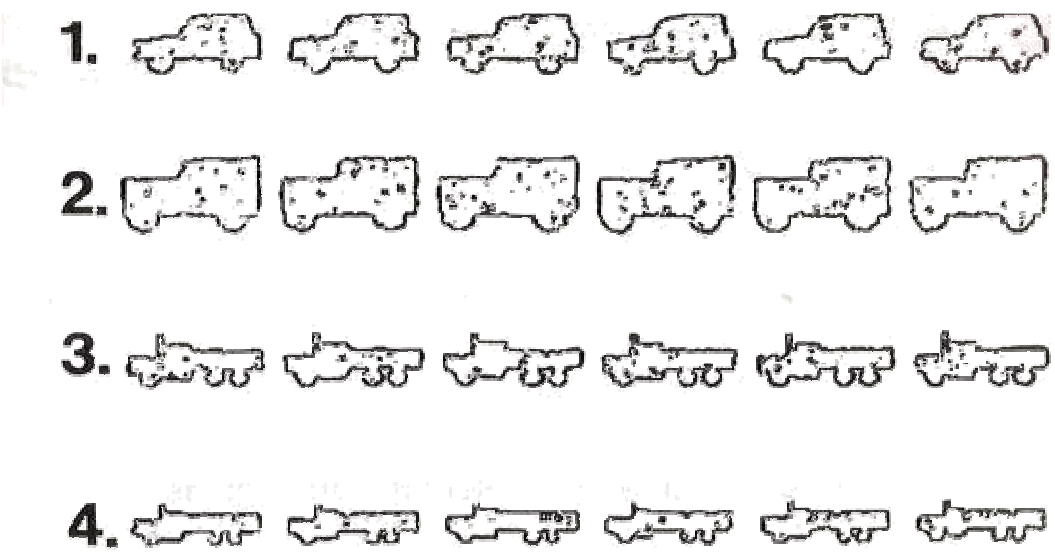


Figura 15. Clasificación de 24 imágenes en 4 categorías, realizada por una red ART2.[38]

Se pueden realizar modificaciones espaciales de las figuras (rotación, traslación y cambio de tamaño) y repetir el proceso anterior con las nuevas imágenes. En la *figura 16* se muestra el resultado de la clasificación de 32 figuras, con un 5% de ruido, en 5 categorías. Se ha producido una división en 2 clases (3,5) de las imágenes que originariamente eran del tipo 3. Este problema puede solucionarse mediante una etapa de postprocesamiento basada en otra red ART 2

[38] Tomado de Bibliografía [1]. Capitulo 6, ART

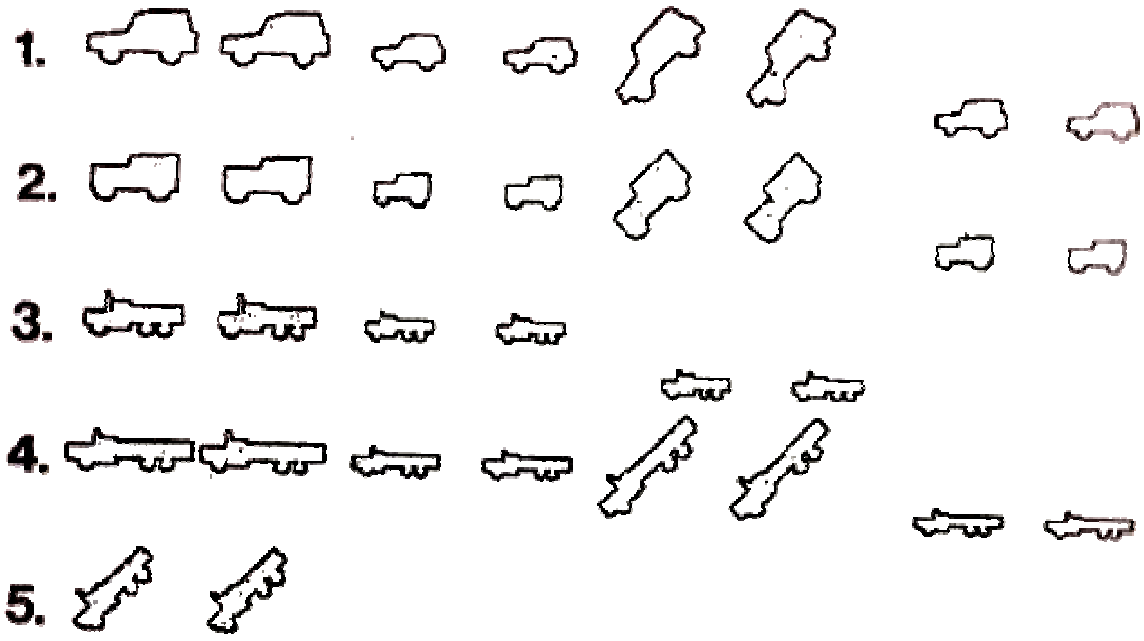


Figura 16. Clasificación invariante de 32 imágenes con una red ART2.[39]

Igual que en el caso de informaciones binarias, una red ART2 puede utilizarse para establecer categorías de patrones analógicos, como pueden ser señales en el tiempo o en el dominio de la frecuencia, que podrían corresponder, por ejemplo, a señales de voz, de un transductor médico, etc. En la *figura 17* se muestra un ejemplo de cómo una red ART2 de *N* neuronas de entrada establece 34 categorías a partir de 50 patrones (señales) analógicos compuestos de *N* muestras de valores reales. En este caso de utiliza un valor grande para el parámetro de vigilancia.

Si el valor del parámetro p se reduce, el número de categorías también

[39] Tomado de Bibliografía [1]. Capitulo 6, ART

disminuye, ya que se relaja la condición de pertenencia a una clase u otra. Con las 50 señales del ejemplo anterior, se obtienen entonces 20 grupos, en lugar de 32 (*Figura 18*). Lo que ha ocurrido es que las categorías 32 de la figura 17 se han fundido en la categoría 4, entre otras.

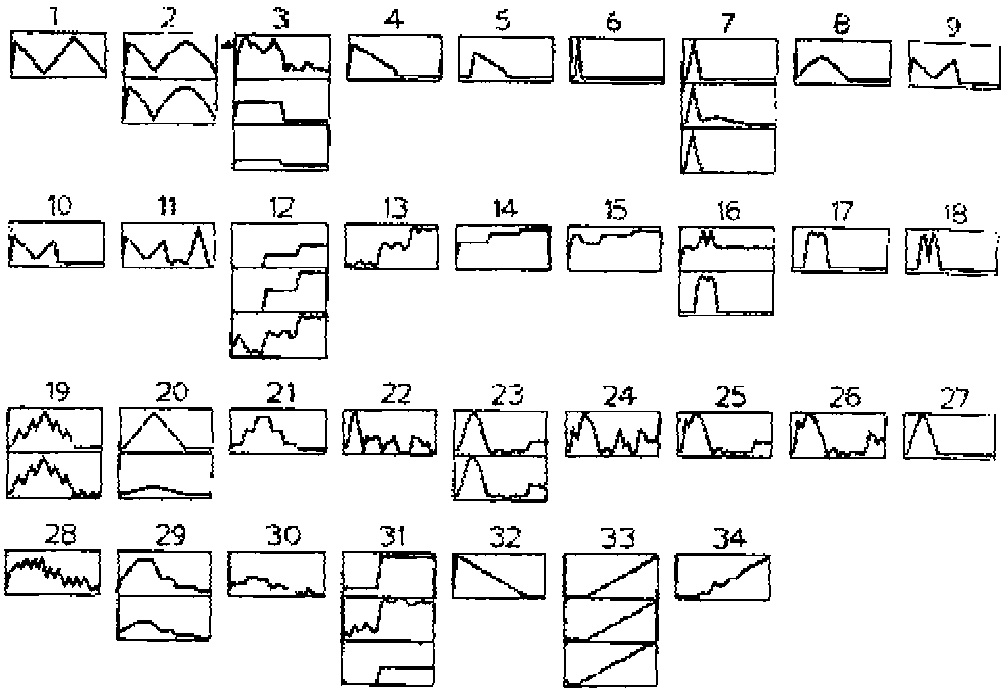


Figura 17. Clasificación de patrones analógicos con una red ART2 (con un valor grande de *p*).[40]

[40] Tomado de Bibliografía [1]. Capitulo 6, ART

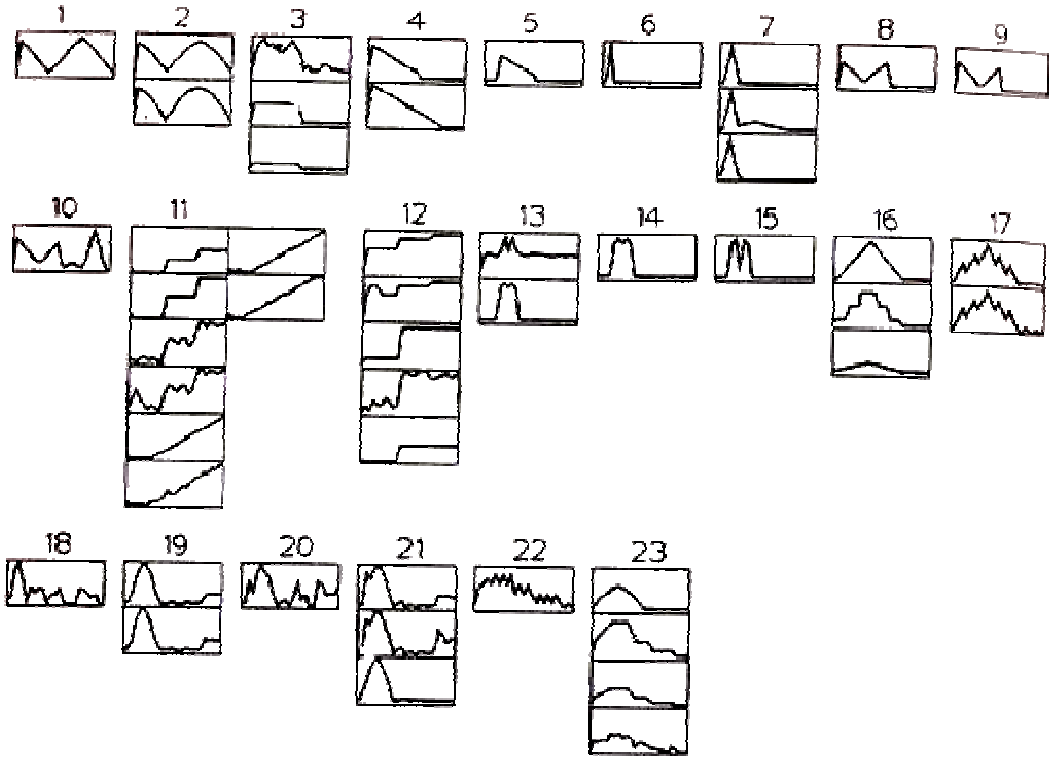


Figura 18. Variación del numero de categorías al reducir el valor de *p* (2Carpenter).[41]

Otro ejemplo del modelo ART2 en tareas de clasificación de informaciones analógicas lo constituye la siguiente aplicación médica. Se trata de reconocer la forma de los complejos QRS (*Figura 19*) de los latidos (ciclos) cardiacos en señales electrocardiográficas (ECG) para ayudar en un posterior diagnóstico clínico.

[41] Tomado de Bibliografía [1]. Capitulo 6, ART

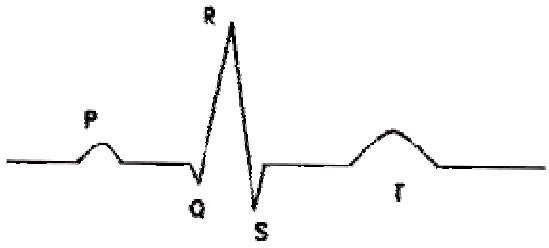


Figura 19. Puntos representativos de un ciclo cardiaco.[42]

El sistema propuesto (*Figura 20*) consiste en: *un preprocesador,* para dividir la señal ECG en ciclos cardiacos; *dos redes neuronales ART2,* que reciben como entrada un ciclo cardiaco e indican las localizaciones aproximadas de los puntos significativos para reconocer el complejo QRS, y *un reconocedor,* para buscar las localizaciones exactas de los puntos significativos alrededor de la posición indicada por las redes.

El preprocesador detecta los puntos *R* en el ECG y divide esta señal en ciclos cardiacos. Un ciclo a su vez, se considera formado por dos tramos. Un tramo del ciclo, 100 ms desde el punto *R* hacia la onda *T,* se introduce en una red neuronal para reconocer el punto S. De forma similar, una porción de ECG desde el punto *R* hacia la onda *P,* anterior se introduce en la otra red para reconocer el punto Q.

[42] Tomado de Bibliografía [1]. Capitulo 6, ART

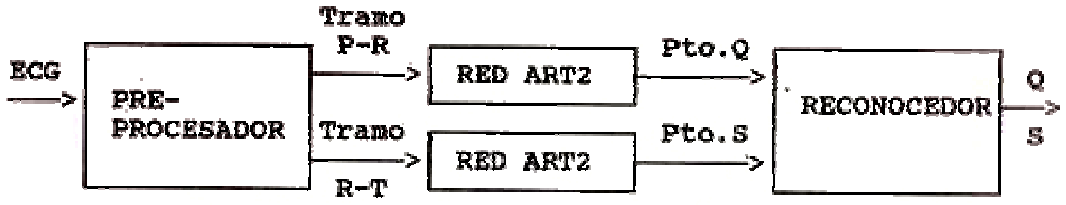


Figura 20. Sistemas de reconocimiento de complejos QRS.[43]

Las *redes ART2* utilizadas son capaces de indicar las posiciones aproximadas de los puntos significativos de la señal. Esto se hace de la siguiente forma (*Figura 21*)

- **Localización del punto S**: Se asume previamente la aproximación de que los puntos R y S están unidos mediante una línea recta, formando junto con la base de tiempos un triángulo rectángulo. Inicialmente se presentan a la red tríangulos de este tipo que se irán almacenando en su memoria a largo plazo (LTM: *Long Term Memory).* Cuando a la red se le presenta un nuevo tramo de ECG, internamente buscará el triángulo (patrón) almacenado que más se parece al fragmento de ECG de entrada. El punto inferior derecho del patrón localizado indicará la posición aproximada del punto S. La red se diseña para que genere a la salida esta posición.

- **Localización del punto** Q: Se procede de la misma manera que con el punto S, utilizando otra red ART2.

[43] Tomado de Bibliografía [1]. Capitulo 6, ART

Cada vez que se presenta a las redes un ciclo cardiaco para reconocer el complejo QRS, se modificarán los pesos de las conexiones entre neuronas para adaptar los prototipos (triángulos) almacenados a las particulares características de la señal ECG de entrada, que puede corresponder a pacientes diferentes.

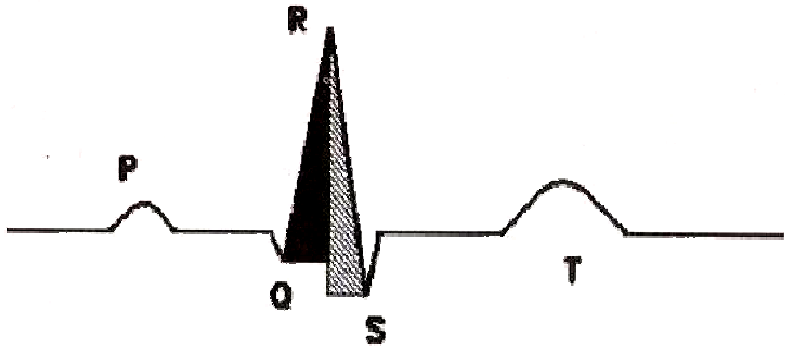


Figura 21. Triángulos aprendidos por las redes.[44]

Por Ultimo, el reconocedor del sistema realiza un ajuste fino, encontrando las posiciones exactas de los puntos *S* y *Q* a partir de las suministradas por las redes.

Según los autores, procesando un total de 300 ciclos cardiacos con ECG de tres personas diferentes, el sistema reconocía el complejo QRS en el 97% de los casos .

[44] Tomado de Bibliografía [1]. Capitulo 6, ART

4.3 EJEMPLOS DE APLICACIONES BASADAS EN APPLETS DE JAVA

4.3.1. ART De Categorías[45]

A. INSTRUCCIONES

Éste es un applet muy simple sin comandos interactivos(*Figura 22*). Para poder empezar una nueva demostración actualiza la aplicación.

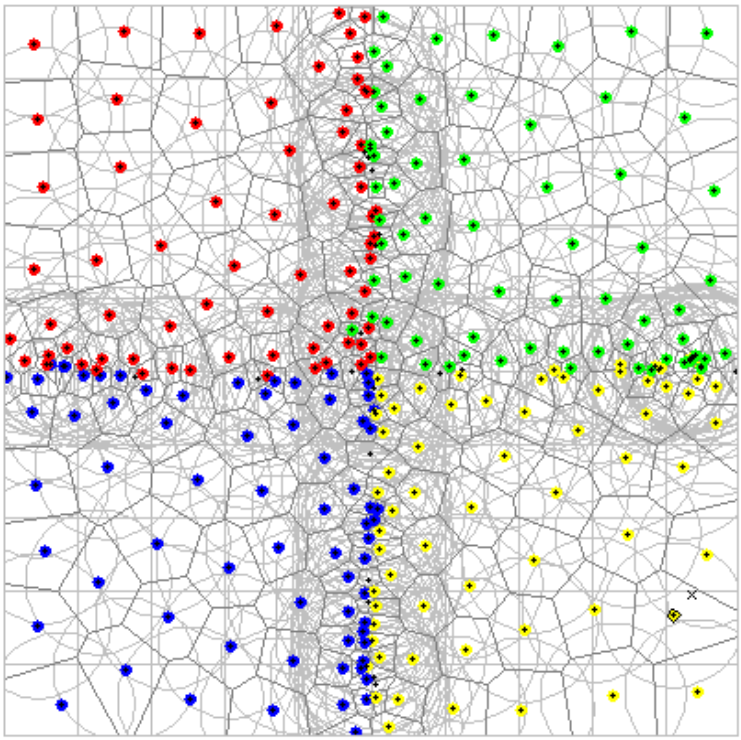


Figura 22. Applet ART de categorías

[45] Para ver en ejecuciones el applet http://people.arch.usyd.edu.au/~rob/applets/neuro/categoryARTdemo.html

B. DESCRIPCIÓN

Este applet demuestra el comportamiento de un algoritmo ART de Categorías. Este applet se parece mucho a la demostración del ART Simplificado, y es porque usa una red muy similar. Sin embargo, el objeto adicional en este applet es que pueden ser asignadas neuronas a las categorías. Las categorías se representan como los aros coloreados alrededor de cada neurona.

C. COMENTARIOS

La *Figura 23* muestra como se debe ver después de un corto tiempo. Las cuatro categorías son bien definidas por las neuronas localizadas en los cuatro cuadrantes del display y muchas neuronas son clusterizadas a lo largo de los bordes de las categorías que definen el display.

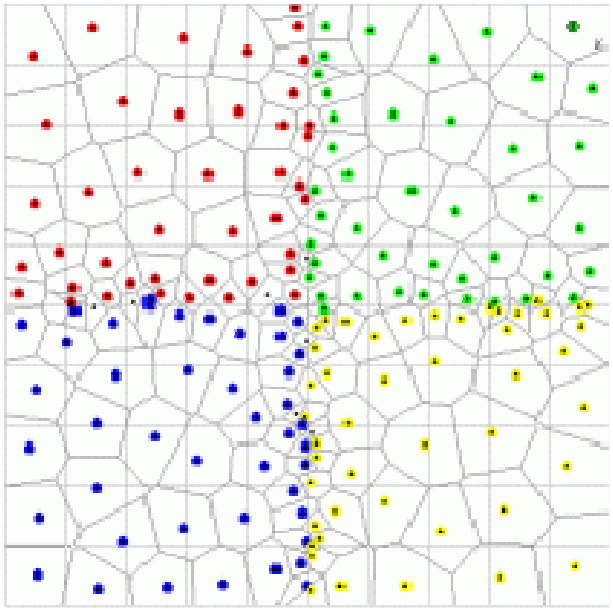


Figura 23. Una red ART de Categorías que ha desarrollado una buena categorización de los espacios de entrada con muchas neuronas cercanas a los bordes de cada categoría

El algoritmo ART de Categorías es una versión mucho más simple que el algoritmo de ARTMAP desarrollada por Carpenter y Grossberg (Inventores de la Teoría de Resonancia Adaptable y la original red ART). El algoritmo ART de Categorías se propuso recientemente por David Weenink y anula en gran parte con los códigos complejos en el algoritmo original de ARTMAP. El resultado es más fácil de implementar y más rápido para correr.

D. CÓDIGO DE LA FUENTE

El código de la fuente para este applet:

CategoryARTDemo-java

CategoryARTView-java

CategoryART-java

4.3.2. Art Simplificado[46]

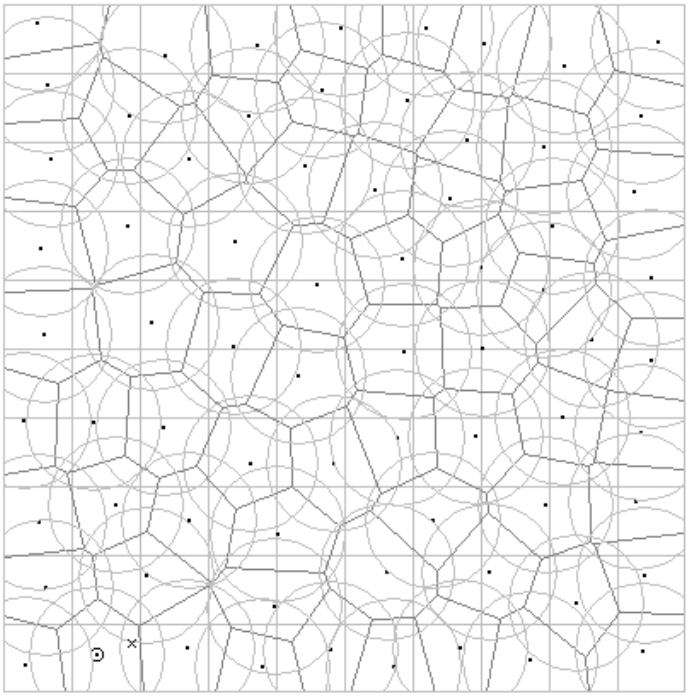


Figura 24. Applet ART Simplificado

A. INSTRUCCIONES

Éste es un applet muy simple sin comandos interactivos (*Figura 24*). Para poder empezar una nueva demostración actualiza la aplicación.

B. DESCRIPCION

Este applet demuestra el comportamiento de redes ART simplificadas. ART simplificado es una variación de la teoría de la resonancia adaptativa (ART) de Carpenter y Grosberg. El display muestra la localización de las neuronas como unos pequeños puntos. Los círculos alrededor de cada neurona

[46] Para ver en ejecuciones el applet http://people.arch.usyd.edu.au/~rob/applets/neuro/SimplifiedARTDemo.html

representa la “tolerancia” de cada neurona, el área del espacio de entrada por el cual la neurona esta intentando competir. Se debería ver neuronas que tienen áreas en común tratando de apartarse para minimizar la coincidencia. Cada entrada esta representada por una cruz pequeña. Las líneas de gris entre las neuronas muestran las células del diagrama de Voronoi[47] para la red ART. Cada célula incluye solamente una neurona; cualquiera entrada que caiga dentro de la célula va a ser mapeada a la neurona contenida. Cuando una entrada cae dentro de la célula Voronoi para una neurona pero cae afuera del radio de tolerancia, la neurona debe flashearse de color rojo dependiendo si la neurona esta determinada a ser disponible para completar la porción del espacio de entrada que contiene la entrada. Finalmente, si no se encuentra un mapeo satisfactorio de la entrada actual entonces todas las neuronas se flasheará rojo y una nueva neurona se añade a la red ART.

C. COMENTARIOS

Igual que un mapa organizativo (SOM), una red ART es un red de aprendizaje competitivo; define un grupo de neuronas que compiten por representar porciones del espacio de entrada, en este caso un plano 2D. A

[47] Un diagrama de Voronoi es una estructura geométrica que representa información de proximidad acerca de un conjunto de puntos u objetos, es decir, es una partición del espacio en celdas cada una de la cuales contiene una colindancia con sus puntos más cercanos. También es usado en solución de un problema de comunicación sin hilos

diferencia de SOM, las redes ART no tiene un número arreglado de neuronas organizadas en celdas. En cambio, ART añade nuevas neuronas cuando se requieran. También, ART no trata de mantener una topología preservando el mapa entre el espacio de entrada y la localización de las neuronas en alguna estructura enrejada. En vez la red ART simplemente trata de cubrir el espacio de entrada uniformemente. Después de un momento se debe ver en el applet algo así (*Figura 25*):

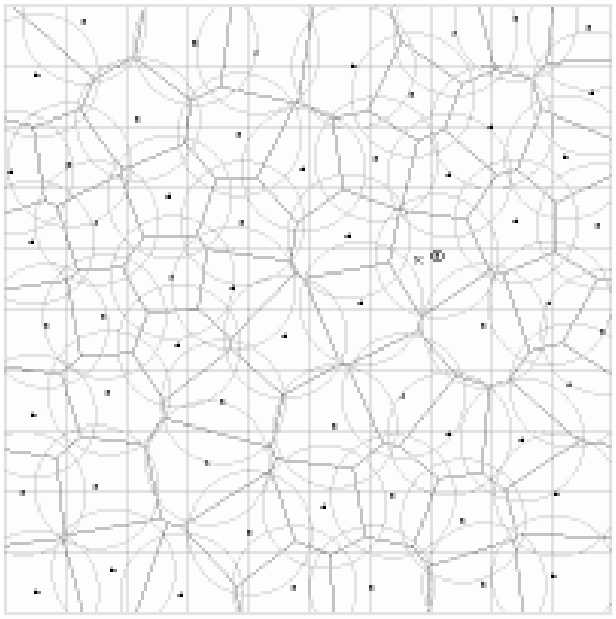


Figura 25. Una red ART que ha desarrollado un buen mapeo del espacio de entrada con espacio de neuronas uniforme (vigilancia = 0.9, tolerancia = 0.1).

La tolerancia en cada neurona es determinada por el parámetro de vigilancia. Mas alta la vigilancia, mas pequeña la tolerancia de cada neurona y el área del espacio de entrada que cada neurona compite mas pequeña. Una red ART con una alta vigilancia va a requerir mas neuronas para representar le

mismo espacio de entrada que uno con baja vigilancia., como se muestra en la *Figura 26* donde la vigilancia a sido incrementada de 0.9 a 0.95, además de bajar la tolerancia de cada neurona de 0.1 a 0.05.

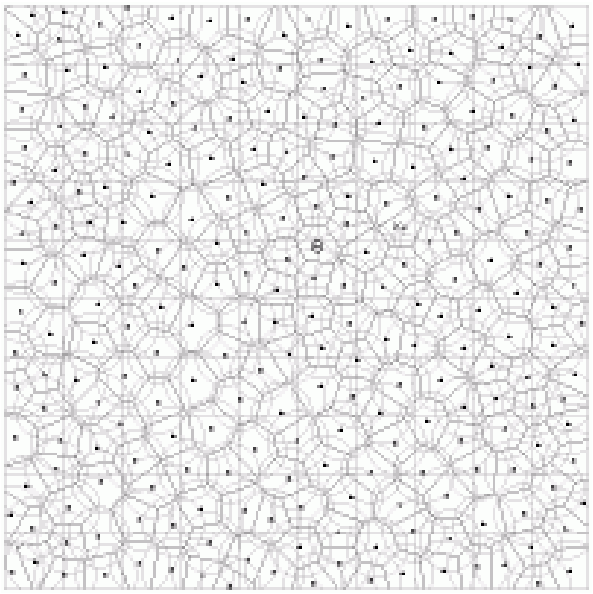


Figura 26. Una red ART que ha desarrollado un buen mapeo del espacio de entrada con neuronas espaciadas uniformemente (vigilancia = 0.95, tolerancia = 0.05).

Determinar si una neurona concuerda satisfactoriamente a la entrada actual depende de dos cosas. Primero, la distancia entre la entrada y el (prototipo) posición representado por la neurona en el espacio de entrada. Si la distancia entre la entrada y la neurona es mayor que las tolerancias de las neuronas entonces es potencialmente una concordancia no satisfactoria. Sin embargo, El estatus de la neurona también juega un rol importante. Una concordancia es no satisfactoria solamente si una neurona a sido "comprometida" a una área de espacio de entrada. Las neuronas

comprometidas han sido asignadas una posición en el espacio de entrada y trataran de mantener esa posición, mientras hacen pequeños ajustes para una mayor aproximación a la posición de todas las entradas que caen sin su jurisdicción. Las neuronas sin compromiso no han sido colocada arriba de una área de espacio todavía y por eso son más libre de “vagar” a través del espacio de entrada. Neuronas sin compromiso son presentados en el applet como neurona sin valor de tolerancia. En el presente no veras neuronas sin compromiso porque el applet implementa un aprendizaje rápido el cual significa que cuando una neurona se añade a la red, esta inmediatamente se asigna a la localización de entrada que causa que la red falle y se comprometa a la parte del espacio de entrada. Este es un modelo de aprendizaje rápido y efectivo para datos que no contengan.

D. CODIGO DE FUENTE

El código de este applet se encuentra en

SimplifiedARTDemo.java

SimplifiedART.java

CONCLUSIONES Y OBSERVACIONES

En algunos métodos de aprendizajes populares y modelos de procesamiento de información las arquitecturas pueden ser inadecuadas porque no usan los principios donde se puedan establemente auto-organizar. Modelos de aprendizaje el cual no pueden adaptativamente enfrentarse con cambios impredecibles en un ambiente complejo tiene un futuro no muy prometedor con modelos de mente y cerebro y tienen poca esperanza de solucionar los problemas de aprendizajes cognitivos el cual no son todavía bien manejados por métodos tradicionales de inteligencia artificial e ingeniería.

Las redes ART de observo que son un tipo de redes autoorganizativas, no supervisadas, estable, la cual es capaz automáticamente de encontrar categorías y crear nuevas cuando se necesitan, solucionando el problema de elasticidad y plasticidad haciendo los pesos flexibles para acomodar el nuevo conocimiento y no mucho como para perder el viejo.

De las aplicaciones en Matlab se puede observar que la red ART es una red con una gran sensibilidad ante el ruido o distorsión, también ante el desplazamiento de los datos de entrada, que pueden dar lugar a una gradual

degradación de los prototipos correspondientes, debido al carácter destructivo de la regla de aprendizaje que realiza la adaptación de los pesos.

Otro factor a tener en cuenta es que el modelo presenta el problema de la ineficiencia en cuanto a las necesidades requeridas para el almacenamiento de los pesos de las conexiones entre neuronas.

Se debe también tener en cuenta el parámetro de vigilancia, con el fin de que la red reconozca todas las entadas sin crear demasiadas prototipos o categorías por ruidos.

Las redes ART se pueden aplicar en proyectos industriales como son en el control de robots móviles, reconocimiento de rostro, reconocimiento de objetivos, diagnósticos médicos, análisis de electrocardiograma, verificación de firmas, monitoreo de fallas de herramientas, reconocimiento visual de objetos 3D, análisis musical y sísmico, reconocimiento sonar y radar

BIBLIOGRAFIA

1. Hilera González, José Ramón, Redes neuronales artificiales: fundamentos, modelos y aplicaciones, México D. F : Alfaomega : Ra-ma, 2000.

2. Martin del Brio, Bonifacio, Redes neuronales y sistemas difusos, México D. F : Alfaomega, 2002, 2 Ed.

3. Freeman,James A. Skapura, David, Neural Network, Algorithms, Application and Programming Techniques. Series Editor, 1991.

4. Carpenter, G.A. & Grossberg, S. (2003), Adaptive Resonance Theory, In M.A. Arbib (Ed.), The Handbook of Brain Theory and Neural Networks, Second Edition (pp. 87-90). Cambridge, MA: MIT Press

5. Grossberg, S. (1987), Competitive learning: From interactive activation to adaptive resonance, Cognitive Science (Publication), 11, 23-63

6. Carpenter, G.A. & Grossberg, S. (1987), ART 2: Self-organization of stable category recognition codes for analog input patterns, Applied Optics, 26(23), 4919-4930

7. Carpenter, G.A. & Grossberg, S. (1990), ART 3: Hierarchical search using chemical transmitters in self-organizing pattern recognition architectures, Neural Networks (Publication), 3, 129-152

8. Carpenter, G.A., Grossberg, S., & Rosen, D.B. (1991b), Fuzzy ART: Fast stable learning and categorization of analog patterns by an adaptive resonance system, Neural Networks (Publication), 4, 759-771
9. Carpenter, G.A., Grossberg, S., & Reynolds, J.H. (1991), ARTMAP: Supervised real-time learning and classification of nonstationary data by a self-organizing neural network, Neural Networks (Publication), 4, 565-588.
10. http://people.arch.usyd.edu.au/~rob/applets/neuro/SimplifiedARTDemo.html
11. http://people.arch.usyd.edu.au/~rob/applets/neuro/categoryARTdemo.html
12. http://en.wikipedia.org/wiki/Adaptive_resonance_theory.
13. http://cns.bu.edu/Profiles/Grossberg/CarGro2003HBTNN2.pdf
14. http://web.umr.edu/~tauritzd/art/
15. http://cannes.itam.mx/Alfredo/English/Publications/Nslbook/MitPress/157_170.CH08.pdf
16. http://www.ifi.unizh.ch/~krafft/papers/2001/wayfinding/html/node97.html
17. http://scalab.uc3m.es/~docweb/rn-inf/documentacion/ArtTC.pdf
18. www.monografias.com/trabajos12/redneuro/redneuro2.shtml
19. html.rincondelvago.com/redes-neuronales_1.html
20. www.gui.uva.es/login/login/13/redesn.html
21. et.evannai.inf.uc3m.es/docencia/rn-inf/documentación /NoSupervisado0506.pdf
22. www.monografias.com/trabajos12/redneuro/redneuro.shtml
23. ohm.utp.edu.co/neuronales/main.htm