UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO

U.A.C.P. y P. DEL C.C.H.
I.I.M.A.S.

RECONOCIMIENTO ADAPTABLE DE PALABRAS AISLADAS UTILIZANDO REDES NEURONALES

T E S I S

QUE PARA OBTENER EL GRADO DE:

MAESTRO EN CIENCIAS DE LA COMPUTACION

P R E S E N T A :

RAFAEL SANCHEZ AROCHE

DIRECTOR DE TESIS: M. EN I. ABEL HERRERA CAMACHO

FEBRERO 1997

A Tania
mi esposa
por su apoyo, comprensión y ayuda

AGRADECIMIENTOS

A mis padres y hermanos por su ayuda, comprensión y formación

A la Dra. Hanna Oktaba por su apoyo incondicional y su amistad sincera

Al M. en I. Abel Herrera por su orientación en la realización de este trabajo

A la familia Calvo de la Peña por su apoyo incondicional

A Edgar y a Pedro por sus oportunos comentarios

A Edgar y a Lupita por la entrañable amistad que nos une.

A mis amigos Pedro, Chema y Lazo por soportarme

A Juanita, Lulú, Vero, Mardonio y Alfredo por su apoyo material para la edición de este trabajo y por su ayuda durante los 2 años transcurridos en la maestría

Al CONACYT por haberme brindado la oportunidad de estudiar en este país

A la UNAM por brindarme los conocimientos deseados

A todos los profesores que contribuyeron en mi formación

RESUMEN

El presente trabajo propone una nueva alternativa de solución al problema de reconocimiento de palabras aisladas independiente del usuario. La solución consiste en utilizar un método novedoso de preprocesamiento de la señal de voz y una nueva red neuronal para las etapas de aprendizaje y clasificación. El trabajo presenta también el comportamiento de la conocida red FuzzyARTMAP para el problema propuesto. A grandes rasgos el método de preprocesamiento consiste en subdividir la señal de voz en tramas (ventana de Hanning) y segmentar la palabra en subpalabras acústicas (SACs) que no son más que una cantidad variable de tramas consecutivas. Esta segmentación se lleva a cabo mediante el método de razón de máxima verosimilitud (MLR). Cada trama se representa mediante 18 bandas críticas y finalmente se aplica el método KLT a los vectores de bandas críticas de una misma subpalabra. De los 18 vectores propios que regresa el método KLT se utiliza solamente los asociados con los 6 valores propios más grandes. con lo cual se tiene una significativa reducción de los parámetros para describir una señal de voz.

Se diseñó un sistema que cuenta de dos partes, una de preprocesamiento de la señal de voz y otra de clasificación mediante varias redes neuronales (una por cada tipo diferente de SAC), con lo cual se logra una gran velocidad de respuesta.

Se presenta un nuevo modelo de redes neuronales llamado AMONSTA para clasificar objetos pertenecientes a conjuntos de datos no separables linealmente y generados por distribuciones que cambian con el tiempo. Durante el aprendizaje se determina un conjunto de prototipos (hiperesferas) para cubrir el conjunto de datos de entrenamiento. Tales prototipos son los encargados de clasificar un nuevo objeto de entrada. El algoritmo de aprendizaje cuenta con mecanismos de fusión de prototipos y un mecanismo de olvido, los cuales permiten a la red adaptarse eficientemente a nuevos conjuntos de palabras correspondientes a nuevos usuarios. Las redes tipo AMOMSTA determinan automáticamente el número de prototipos necesarios para aprender a clasificar un conjunto de datos dado.

Se implementaron dos variantes del sistema correspondientes a las redes AMONSTA y a las redes FuzzyARTMAP. En el primer caso se logró un porciento de errores de tan sólo 5.68, mientras que en el segundo de 31.82 %.

INTRODUCCIÓN

El problema de Reconocimiento de Palabras Aisladas Independiente del Usuario (RPAIU) consiste en clasificar automáticamente palabras individuales pronunciadas por cualquier persona. Se trata entonces de crear un sistema de cómputo que sea capaz de clasificar cada palabra pronunciada en un conjunto finito predeterminado de palabras. El sistema debe además funcionar adecuadamente independientemente del interlocutor. Una alternativa interesante de solución a este problema, consiste en diseñar un sistema que se adapte en tiempo real a cualquier grupo predeterminado de usuarios (en particular a un sólo usuario) y además a cualquier conjunto predeterminado de palabras. Esto significa que el sistema no depende del usuario ni del vocabulario. En el presente trabajo se aborda esta alternativa de solución mediante el uso de redes neuronales adaptivas [Carpenter91, Carpenter92a, Sánchez96] que utilizan como entradas, descripciones de las señales de voz en términos de coeficientes de la transformada de Karhunen Lòeve (KLT) [Ahmed]. Un sistema de este tipo puede adaptarse muy rápido a cualquier grupo de usuarios y a cualquier vocabulario, sin necesidad de almacenar una gran base de datos de palabras, minimizando así el tiempo de respuesta.

Todo sistema de Reconocimiento de Palabras Aisladas se compone en general, de las siguientes partes: extracción de características, cálculo de similitud, algoritmos de compresión de parámetros y algoritmos de clasificación. En cada una han sido empleadas diversas técnicas, que sin pretender hacer una revisión detallada, se referencia algunas a fin de que el lector pueda profundizar si es su deseo.

En la etapa de extracción de características se procesa la señal de voz ya sea en el dominio del tiempo o en el dominio de la frecuencia y se calculan características o parámetros para describir la señal. Uno de los parámetros más utilizados son los coeficientes de predicción lineal (LPC del Inglés) o variaciones de estos [Makhoul75, Gersho91]. Otros parámetros son la energía, amplitud media, número de cruces por cero [Casacubierta87] y función de autocorrelación [Gersho91, Casacubierta87]. En este trabajo se subdividen las palabras en subpalabras acústicas las cuales se representan mediante coeficientes KLT [Herrera94].

Existen diferentes métodos para el cálculo de la similitud entre dos señales de voz, entre ellos los métodos de alineamiento temporal (DTW del Inglés) [Casacubierta87], la distancia de Itakura-Saito [Rabiner81], y en general, varias

de las distancias conocidas, como el producto escalar normalizado, la distancia euclidiana, etc. En este trabajo se utiliza la distancia euclidiana y otras medidas de similitud empleadas en las redes neuronales de Carpenter y cols. [Carpenter91, Carpenter92a].

Los métodos de cuantización vectorial se han aplicado al área de Reconocimiento de Voz para codificar y comprimir la señal de voz, así también para reducir el número de elementos del diccionario de referencia con el que se compara una señal dada a clasificar. Entre los métodos más conocidos están el algoritmo de Lloyd [Gersho91], y algunas variaciones como el método de las K medias [MacQueen87], el método de las K medias-difusas [Bezdek92] y el algoritmo LBG [Linde80]. También, las redes neuronales LVQ [Kohonen89], UABC [Nissani91], ART1, ART2 y FuzzyARTMAP [Carpenter87a, Carpenter87b, Carpenter92a]. En este trabajo se presenta una nueva red neuronal que durante su etapa de aprendizaje obtiene un conjunto de prototipos representantes de todo el diccionario inicial [Sánchez96].

Los métodos de clasificación son los encargados de llevar a cabo el aprendizaje y en general combinan varios de los elementos anteriores. Existen una gran variedad destacándose fundamentalmente modelos ocultos de Markov [Rabiner89, Rabiner84] y la regla de los K vecinos más cercanos (K-NN) [Duda73] entre otros. En este trabajo se utiliza el enfoque de redes neuronales [Pao89].

Las redes neuronales constituyen uno de los nuevos enfoques que comienzan a utilizarse en el área de Reconocimiento de Voz. Se han empleado tanto para procesar las señales de voz y obtener parámetros para describirla [Mazin93, Kato93] como para clasificar las señales de voz ya sea en forma cruda o a partir de parámetros previamente calculados [Mellouk93, Konig93].

Las redes neuronales son métodos de procesamiento de la información que imitan en algún sentido la arquitectura y el funcionamiento del cerebro humano. Estos sistemas artificiales son concebidos para emular una serie de funciones del cerebro tales como: la memoria, el aprendizaje y la generalización o abstracción del conocimiento, entre otras.

Existen numerosos trabajos tanto de tipo introductorio [Pao89, Schalkoff92, Riquenes94] como avanzados, incluyendo, desarrollos teóricos y presentación de aplicaciones interesantes [Ripley94, Carpenter92c, Carpenter95]. En este trabajo, sólo se presentarán algunas ideas en torno a las redes neuronales adaptivas, dejando al lector no familiarizado la tarea de revisar la bibliografía citada.

Han habido substanciales avances en la formalización de diferentes métodos del área, mediante modelos y métodos matemáticos [Ripley94, Hornik93, Funahashi89], llegando incluso a demostrarse que varios de los métodos utilizados en redes neuronales son nuevas formulaciones de problemas y métodos conocidos en otras áreas de las matemáticas, como por ejemplo, estadística multivariada y aproximación de funciones [Hornik93, Ito92]. Sin embargo debe señalarse que la formulación de estos métodos en el área de redes neuronales fue inspirada en modelaciones realizadas sobre el cerebro en áreas como fisiología, psicología, neuropsicología, etc., lo cual en muchos casos rompió con esquemas tradicionales implantados por las matemáticas dando una mayor flexibilidad a los métodos, y combinando áreas de las matemáticas comúnmente alejadas. Actualmente se considera que las matemáticas del futuro, van a ser aquellas inspiradas por sistemas biológicos.

En el presente trabajo se propuso como objetivos:

1.- Proponer un nuevo modelo de red neuronal adaptiva como una nueva alternativa de solución al problema de reconocimiento de palabras aisladas independiente del usuario.

2.- Evaluar la eficiencia de describir palabras mediante subpalabras acústicas unido al empleo de redes neuronales adaptivas.

CAPÍTULO I: Datos, procesamientos y objetos

Los datos que se utilizan en este trabajo corresponden a la conocida base de datos TI-46 de Texas Instruments Inc.. Los datos corresponden a señales de voz de dígitos aislados pronunciados en Inglés por 10 parlantes (5 masculinos y 5 femeninos). Cada parlante pronunció 26 veces cada dígito. El total de datos fue entonces 2600. La base de datos está originalmente muestreada a 12500 Hz y fue remuestrada a una tasa de 10000 Hz. Se utilizó un filtro preénfasis para igualar el espectro. Las tramas se compusieron de 128 muestras y se utilizó una ventana de Hanning con un traslape de 20 muestras con la trama anterior.

Los datos fueron procesados mediante un método para obtener subpalabras acústicas [Herrera94]. Este método es una combinación de criterios matemáticos y lingüísticos de segmentación de palabras. La idea es segmentar cada palabra en regiones cuasiestacionarias de la señal. Una vez segmentada la palabra, a cada subpalabra se le aplica la transformada de Karhunen-Loéve (KLT) para obtener finalmente una descripción en términos de vectores y valores propios.

Al aplicar el método KLT se obtiene, para cada señal de voz, un conjunto de vectores propios (18-componentes) y sus respectivos valores propios, de los cuales basta utilizar solamente 6, según Herrera [Herrera94], para obtener una descripción adecuada de la palabra.

A continuación se explica el proceso de segmentación de palabras en subpalabras acústicas:

1- Primero se aplica a toda la señal un filtro paso baja de cuarto orden para eliminar cualquier frecuencia por encima de 4k

2- Se aplica una ventana de Hanning a tramas de 128 muestras con un solapamiento de 20 muestras. Se utiliza este tipo de ventana pues es simétrica y por tanto tal segmentación puede ser utilizada también con fines de codificación.

3- A cada trama se le aplica la FFT y se calculan 18 bandas críticas [Tobias70, Zwicker90] con el objetivo de reducir el número de parámetros para caracterizar la trama. Tal forma de comprimir está inspirada en la manera en que el ser humano escucha, o sea descomponiendo la señal en bandas críticas [Zwicker90].
De esta manera cada trama queda representada por un vector de 18 componentes que denotaremos BC-vector.

4- Se aplica el método de razón de máxima verosimilitud (MLR) para detectar el inicio y fin de la palabra, además de segmentarla en subpalabras acústicas. Siempre se hace la segmentación en frontera de tramas y puede ocurrir que las palabras sean segementadas en diferentes número de subpalabras. En el apéndice IV se muestra un ejemplo.

5- Se aplica a cada subpalabra la transformada de Karhunen Lòeve (KLT), que consiste a grandes rasgos en:

a) Calcular la matriz de correlación para todas las tramas de una subpalabra, o sea:

$$R = \frac{1}{N_i} \sum_{n=1}^{N_i} [C(n)C(n)^T]$$

donde:

N_i es el número de subpalabras en la trama i
$C(n)$ es el BC-vector de la trama n-ésima.
R es una matriz de 18x18

b) Se calculan los valores y vectores propios de R, o sea se obtienen 18 valores propios y 18 vectores propios por cada subpalabra.
Se sabe que los valores propios miden la concentración de la energía espectral, por lo que mientras menor sea el valor propio menos información aporta.
Por otro lado se sabe que el método KLT tiene la propiedad que cualquier truncado en sus parámetros (o sea ignorar los valores menores y vectores propios asociados) minimiza el error mínimo cuadrado de la aproximación de los BC-vectores originales. Por ello no se necesita considerar todos estos vectores, sino que usualmente basta tomar los primeros 6 o 7 valores propios y vectores propios.

Debe notarse que como los vectores propios forman una base entonces cada BC-vector se puede expresar mediante una combinación lineal de ellos, y es en este sentido que se habló anteriormente de un error de aproximación. También en este mismo sentido se habla de optimalidad, puesto que si se utiliza cualquier otra base para expresar los BC-vectores, entonces cualquier truncado en la sumatoria (de la combinación lineal) provocará un error mayor o igual que el error cuadrático medio.

En el apéndice V se muestra un ejemplo de este punto.

Conviene señalar que la programación de los métodos anteriores no se llevó a cabo en este trabajo, sino se utilizaron programas previamente realizados.

Los objetos a utilizar se formaron concatenando los primeros seis vectores propios de cada subpalabra y por tanto cada una estuvo representada por 108 componentes. Por ejemplo, para una palabra segmentada en 5 subpalabras, el objeto correspondiente tiene 540 componentes. Con los valores propios se realizó un proceso análogo; para garantizar tener la misma cantidad de componentes que el objeto (concatenación de vectores propios) se repitió cada valor propio 18 veces.

Resumiendo, por cada señal de voz se construyeron dos vectores de igual dimensión uno con la concatenación de los vectores propios y otro de los valores propios, repitiendo el valor propio para el mismo vector propio.

Capítulo II. Redes tipo ART

Las redes tipo ART (del inglés 'Adaptive Resonance Theory') se caracterizan por determinar un conjunto de prototipos durante su etapa de aprendizaje, mediante el cual llevan a cabo la clasificación de cualquier conjunto de objetos de entrada a la red.

Inicialmente no se cuenta con categoría o prototipo alguno. A medida que se van presentando entradas a la red, esta comienza a crear prototipos, garantizando siempre que cada entrada tenga asociada un prototipo que lo clasifique. En general varias entradas pueden tener asociado un mismo prototipo.

En esta familia hay redes que aprenden de manera supervisada [Carpenter92a] y redes que aprenden no supervisadamente [Carpenter87a, Carpenter87b, Carpenter91], asimismo hay redes cuyas entradas pueden ser continuas [Carpenter91, Carpenter92a] y otras con entradas discretas [Carpenter87a]..

En este trabajo se presentan dos redes neuronales de dicha familia, FuzzyART [Carpenter91] y FuzzyARTMAP [Carpenter92a], las cuales aprenden de manera no supervisada y supervisada, respectivamente. Estas son las dos redes más recientes realizadas por S. Grossberg y G. Carpenter, las cuales han mostrado un comportamiento adecuado en problemas como reconocimiento de caracteres escritos [Carpenter92c], estimación no paramétrica para problemas de reconocimiento de patrones no estacionario [Carpenter95], clasificación de potenciales evocados auditivos [Sánchez95a], y otros [Carpenters92b].

Red FuzzyART

Las redes tipo FuzzyART [Carpenter91] constituyen una generalización a la conocida red ART1 [Carpenter87a], que sirve para resolver problemas de clasificación con entradas continuas basadas en lógica difusa. Básicamente las modificaciones que se hacen a ART1 son extender las operaciones AND y OR para el caso difuso mediante las conocidas operaciones MIN y MAX, respectivamente. En este trabajo se presenta de manera breve las ideas fundamentales de la red, haciendo énfasis en elementos que no quedan bien detallados en la presentación original [Carpenter91].

FuzzyART permite resolver problemas de clasificación no supervisada, es decir, se presentan objetos de entrada a la red y ésta por sí sola aprende a clasificarlos. FuzzyART determina automáticamente el número de categorías necesarias para clasificar los objetos de entrada, a diferencia de otros métodos no supervisados, incluidos en las áreas de redes neuronales y de agrupamientos [Kohonen89, MacQueen87, Bezdek92], para los que es necesario especificar el número de categorías a priori.

Arquitectura

La arquitectura la red consta de 4 capas interconectadas con retroalimentación hacia adelante. En la capa de entrada hay la misma cantidad de neuronas como componentes tengan los objetos de entrada a la red. Esta capa es la encargada de recibir los estímulos del medio ambiente, un vector de coeficientes KLT en este problema. La segunda capa contiene el doble de neuronas que la primera; y su función es llevar a cabo un preprocesamiento de tipo complemento (que se explica más adelante), de las neuronas de la primera capa. Cada neurona de la primera capa se conecta con dos neuronas de la segunda y cada neurona de la segunda sólo recibe información de una neurona en la primera capa. La tercera capa contiene el mismo número de neuronas que la segunda capa, con la cual mantiene conexión (abajo-arriba) 1-1. Esta tercera capa recibe además conexiones (arriba-abajo) de la última capa del tipo n-n, o sea cada neurona de la última envía conexiones a la tercera capa y viceversa. En la cuarta capa o capa de salida hay un número prefijado de neuronas correspondiente al número máximo de prototipos que podrá utilizar la red. Ver figura 1.

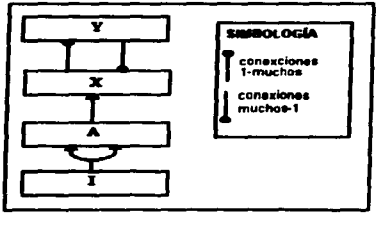


Figura 1. Arquitectura general de una red FuzzyART

Para mayor claridad se utilizará la notación vectorial, asociando a todas las neuronas de una capa, un vector cuyos valores representan las activaciones de estas. Además los pesos de las conexiones que recibe cada neurona también se representarán mediante un vector. Así, las entradas se asocian con el vector de M dimensiones I ($I_i \in [0,1]$), la segunda y tercera capas con vectores de 2M dimensiones A y X, respectivamente y la última con el vector de N dimensiones Y. Los pesos de cada neurona j de la última capa se denotan por el vector de 2M dimensiones W_j.

El ciclo básico de procesamiento de esta red, consiste:

1.- Presentar una entrada
2.- Evaluar una medida de similaridad de la entrada a cada categoría existente
3.- Determinar el máximo de esta medida sujeta a la restricción de que se satisfaga una desigualdad para la categoría en cuestión y el vector de entrada.
4.- Actualizar los pesos de la categoría ganadora.

La activación de neuronas de la segunda capa es:

$$A_i = \begin{cases} I_i & 1 \le i \le M \\ 1 - I_i & M+1 \le i \le 2M \end{cases} \qquad (1)$$

Debe notarse que con esta activación se están normalizando todas las entradas a vectores de longitud 2M, pues:

$$|A| = \sum_{j=1}^{2M} A_j = \sum_{j=1}^{M} I_j + \sum_{J=M+1}^{2M} 1 - \sum_{j=1} I_j = M$$

Tal normalización evita la proliferación de categorías.

La activación de las neuronas de la tercera capa es:

$$X = \begin{cases} A & \text{si no hay neurona activa en la última capa} \\ A \wedge W_j & \text{si la } j-\text{ésima neurona de la última capa está activa} \end{cases} \qquad (2)$$

donde:

$$a \wedge W_j = (a_1 \wedge W_{j1},\ a_2 \wedge W_{j2},\ \ldots,\ a_{2M} \wedge W_{j2M})$$

el operador $\wedge$ se generaliza mediante MIN, o sea

$$a_i \wedge w_{ji} = \text{MIN}(a_i, w_{ji})$$

La medida de similaridad entre la entrada actual y una categoría constituye la activación de la neurona asociada en la última capa y se computa:

$$Y_j = \frac{|A \wedge W_j|}{\alpha + |W_j|} \qquad (3)$$

donde: α es un parámetro, $\alpha > 0$

$\wedge$ es generalizado mediante MIN, o sea $a \wedge b = \text{MIN}(a,b)$

La restricción que debe satisfacer la categoría seleccionada como posible ganadora es:

$$\frac{|A \wedge W_j|}{|A|} < \rho \qquad (4)$$

donde: ρ es un parámetro, $\rho \in [0,1]$

Inicialmente se prefija un número máximo de categorías, es decir, la dimensión de Y y se inicializan los vectores W_j con el vector 1, o sea:

$$W_{ji} = 1 \quad \forall 1 \le j \le N, \ 1 \le i \le 2M$$

Esta inicialización de todos los W_j indica que ningún prototipo está acometido, o sea que no hay prototipo alguno asociado con las entradas. En este punto conviene destacar que la red puede ser fácilmente modificada para en lugar de tener un número máximo de prototipos inicializados con el vector 1, partir de no tener prototipo alguno, e ir creándolos a medida que se necesite. Los pesos W_j van decreciendo con el tiempo debido a la extensión del AND mediante MIN; pero además están acotados inferiormente por el vector nulo (vea (5)); de ahí que esta red siempre converge.

Desde el punto de vista geométrico los vectores W_j pueden considerarse ubicados en el mismo espacio 2M dimensional de las entradas y por tanto una categoría no es más que un vector prototipo que se encuentra ubicado en una

región en la que se encuentran uno o más objetos del problema, a los cuales clasifica correctamente. El tamaño de dicha región está determinado por los parámetros α y ρ.

Algoritmo de aprendizaje y predicción

0.- Fijar valores para los parámetros α, β, ρ

1.- Presentar una entrada, o sea inicializar I.

2.- Calcular A mediante el complemento de I.

3.- Activar las neuronas de la última capa:

$$Y_j = \frac{|A \wedge W_j|}{\alpha + |W_j|} \qquad 1 \le j \le N$$

4.- $$J = MAX^*(Y) = \begin{cases} J & si \quad \exists J,\ 1 \le j \le N\ (Y_j > 0 \quad y \quad Y_j \ge Y_k\ ,\forall k \ne j\ 1 \le k \le N) \\ -1 & s.o.c. \end{cases}$$

Si J<0 entonces TERMINAR

Activar las neuronas de la tercera capa $X = |A \wedge W_J|$

5.- Si $\frac{|X|}{|A|} < \rho$ entonces hacer $Y_J = 0$; ir al paso 4-

Actualizar los pesos de la neurona ganadora:

$$W_j^{(nuevo)} = \beta \cdot X + (1-\beta) W_j^{(viejo)} \qquad (5)$$

donde: β es un parámetro, $\beta \in [0,1]$

Comentarios

1- En el paso 3, Y_j está acotado en [o,a]; a<1. Además mientras mayor es α, más pequeño es a.

El planteamiento de la manera de calcular X según (2) es útil fundamentalmente cuando se lleva a cabo una implementación paralela de esta red. Por ello es que en el algoritmo anterior se simplificó el cálculo de X.

2- En el paso 4, si se cumple que J<0 entonces significa que ninguna de las categorías satisface (4) y además que las N categorías ya fueron acometidas. Nótese que para una categoría K no acometida, siempre se satisface (4), pues:

$$\frac{|A \wedge W_k|}{|A|} = 1 >= \rho$$

Carpenter y cols. [Carpenter91] proponen utilizar β=1 para un aprendizaje rápido. En este caso se "memoriza" lo más posible del objeto actual, por lo que la tendencia es a crear un mayor número de categorías. También proponen para conjuntos de datos ruidosos, utilizar β=1 para categorías no acometidas y β<1 para categorías acometidas. Nótese que en el segundo caso, cuando se acomete una categoría j se inicializa W_j=A, o sea, la categoría inicialmente coincide con la entrada actual.

Debe observarse que a medida que β se acerque a cero, menos se memoriza del objeto actual y por tanto se requiere un número de iteraciones mucho mayor para aprenderlo.

Adaptaciones de la red para el problema de Reconocimiento de Palabras Aisladas

Para poder utilizar la red FuzzyART para resolver el problema de reconocimiento de palabras aisladas independiente del usuario se propone modificar la medida de similaridad (3), la restricción (4) y la fórmula de actualización (5), incluyendo los valores propios asociados a los vectores de parámetros KLT que se utilizan como entrada a la red. En el capítulo IV se evalúan varias maneras de utilizar los valores propios.

Las modificaciones a (3), (4) y (5) quedan entonces:

$$Y_j = \frac{|\lambda \cdot (A \wedge W_j)|}{\alpha + |\lambda \cdot W_j|} \qquad (3')$$

$$\frac{|\lambda \cdot (A \wedge W_j)|}{|\lambda \cdot A|} < \rho \qquad (4')$$

$$W_{jk}^{(nuevo)} = \beta \cdot \lambda_k \cdot X_k + (1-\beta) W_{jk}^{(viejo)} \qquad \forall 1 \le k \le 2M \qquad (5')$$

La idea esencial de esta modificación surge del hecho de que algunas pruebas preliminares, presentadas algunas en el capítulo V, mostraron la importancia de considerar los valores propios en las medidas de similaridad que utilizaran.

Nota: Al vector λ también se le aplica el complemento (1).

El papel del parámetro α es el de resolver el conflicto que se produce si varias categorías coinciden el en valor de la expresión:

$$\frac{|A \wedge W_j|}{|W_j|}$$

Nótese que puede ocurrir esto sin necesidad de que sean las mismas categorías. Al sumar $\alpha>0$ en el denominador, el conflicto se resuelve a favor de la categoría con mayor longitud $|W_j|$ con lo cual se garantiza seleccionar la categoría más general de todas las posibles, para clasificar el actual objeto I. Por ejemplo, para dos categorías W_1 y W_2 tal que:

$$\frac{|I \wedge W_1|}{|W_1|} = \frac{|I \wedge W_2|}{|W_2|}$$

entonces se toma W_1, mediante la expresión (3), si $|W_1| \geq |W_2|$ (ver demostración en el apéndice I).

Red FuzzyARTMAP

Esta red permite resolver problemas de clasificación supervisada que caen en general, en el caso difuso, aunque en el presente trabajo se considera el problema a resolver como un problema de clasificación duro.

Arquitectura

FuzzyARTMAP [Carpenter92a] integra dos redes FuzzyART (llamémosle ARTa y ARTb respectivamente), las cuales conecta mediante un mapa que es el encargado de asociar cada objeto con una clase de salida conocida. La red

ARTa se alimenta con los objetos a clasificar, mientras que ARTb con las respectivas clases conocidas asociadas a estos objetos. El mapa no es más que un conjunto de neuronas con conexiones hacia las capas de salida de ambas redes FuzzyART. Las conexiones expresan el grado de pertenencia de cada prototipo de ARTa a una clase de ARTb, codificada por un prototipo de esta red.

Durante el aprendizaje, cada subred ARTa y ARTb crea adaptivamente los prototipos necesarios, y las conexiones del mapa se van actualizando, haciendo cada vez más selectiva la asociación de ambas redes.

Cada subred (ARTa y ARTb) actúa de manera autónoma, modificando sus vectores de pesos según leyes no supervisadas que se expresan a través de un mecanismo de competencia. La parte supervisada de la red FuzzyARTMAP se da a través del mapa.

El mapa cuenta con el mismo número de neuronas que la capa de salida de ARTb y cada neurona del mapa, recibe conexiones de todas las neuronas de la capa de salida de ARTa. Además cada neurona del mapa tiene conexiones bidireccionales 1-1 con las neuronas de salida de ARTb. En la figura 2 se muestra la arquitectura general de esta red .

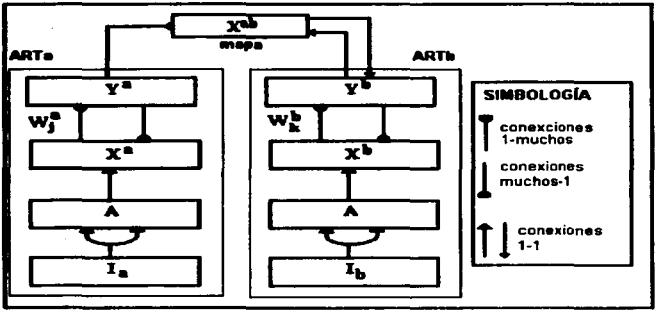


Figura 2. Esquema general de una red FuzzyARTMAP

Utilizando una notación vectorial, las activaciones de las neuronas de las capas de entrada de ARTa y ARTb se asocian con los vectores I_a y I_b de M_a y M_b dimensiones, respectivamente. Las activaciones de las neuronas de la segunda y tercera capas de ARTa, se asocian con A y X^a, respectivamente; asimismo B y X^b para ARTb. Las últimas capas de ARTa y ARTb se asocian con Y^a (N_a dimensiones) y Y^b (N_b dimensiones) respectivamente. Los pesos de las neuronas j-ésima y k-ésima de las últimas capas de ARTa y ARTb se denotan por W_j^a y W_k^b, respectivamente.

Las activaciones de las neuronas del mapa se asocian con el vector de N_b dimensiones X^{ab} y los pesos de entrada a cada una de estas neuronas, con los vectores W_j^{ab} ($1 \le j \le N_a$) de N_b dimensiones.

Las neuronas del mapa pueden activarse porque se activó una neurona de la capa de salida de ARTa y/o una neurona de la capa de salida de ARTb. En general X^{ab} se computa como sigue:

$$X^{ab} = \begin{cases} Y^b \wedge W_j^{ab} & \text{Si la neurona J-ésima de salida de ARTa está activa y alguna neurona de salida de ARTb está activa} \\ W_j^{ab} & \text{Si la neurona J-ésima de salida de ARTa está activa y ninguna de las neuronas de ARTb está activa} \\ Y^b & \text{Si ninguna neurona de salida de ARTa está activa y alguna neurona de salida de ARTb está activa} \end{cases} \qquad (6)$$

Valga hacer la aclaración, como se verá más adelante, que Y_b es un vector binario que posee un único 1 en la posición asociada con la neurona ganadora de ARTb.

Inicialmente los pesos W_j^{ab} se igualan al vector 1, o sea :

$$W_j^{ab} = 1 \qquad 1 \le j \le N_a$$

Se dice que la predicción de ARTa es confirmada por ARTb cuando se satisface:

$$|X^{ab}| < \rho_{ab}|Y^b| \qquad (7)$$

donde: ρ_{ab} es un parámetro, $\rho_{ab} \in [0,1]$

Esta desigualdad expresa que es suficientemente grande el peso de la conexión de la neurona ganadora de ARTa a la neurona ganadora ARTb. Nótese que como se planteó anteriormente estos pesos expresan el grado de pertenencia de los prototipos de ARTa a las clases codificadas por ARTb.

La idea central del funcionamiento de la red, consiste en activar cada subred ARTa y ARTb, entonces verificar, mediante (7), si la predicción que hace ARTa es confirmada por ARTb. En caso negativo se incrementa ρ_a y se vuelve a activar ARTa. Se repite la verificación de la confirmación y la actualización de ρ_{ab} hasta que la predicción de ARTa sea confirmada por ARTb. En cuyo caso se procede a actualizar los pesos W_j^{ab} de la neurona ganadora en ARTa.

Algoritmo de aprendizaje y predicción

0.- Fijar valores para los parámetros $\overline{\rho_a}$, ρ_{ab} y β_{ab}. Crear las subredes ARTa y ARTb.

1.- Presentar un objeto de entrada A a ARTa y su objeto asociado de salida B (clase) a ARTb.
Hacer $\rho_a = \overline{\rho_a}$

2.- Activar ARTb, determinar el ganador y actualizar sus pesos. Sea K el índice de la categoría ganadora.

3.- Activar ARTa, determinar el ganador y actualizar sus pesos. Sea J el índice de la categoría ganadora.

4.- Computar X^{ab}

5.- Si $|X^{ab}| < \rho_{ab}|Y^b|$ entonces
hacer: $\rho_a = \frac{|A \wedge W_J|}{|A|} + \varepsilon$

donde: $\varepsilon > 0$ y suficientemente pequeño.

ir al paso 3.

$$(W_{Jk}^{ab})^{(nuevo)} = \begin{cases} (1-\beta_{ab})(W_{Jk}^{ab})^{(viejo)} + \beta_{ab} Z_k^{ab} & si \quad j = J \\ (W_{jk})^{(viejo)} & si \quad j \neq J \end{cases} \qquad (8)$$

donde: $\beta_{ab} \in [0,1]$

$$Z_k^{ab} = \begin{cases} 1 & k = K \\ 0 & k \neq K \end{cases}$$

Comentarios

1- En el paso 0, $\overline{\rho_a}$ es un parámetro que se conoce como línea base para ρ_a, $0 \leq \overline{\rho_a} \leq 1$. Dicho parámetro es el valor que tuviera ρ_a si se estuviera utilizando ARTa independientemente. El objetivo del empleo de $\overline{\rho_a}$ es el de inicializar ρ_a, ante cada nueva presentación de una pareja de objetos (A,B). Nótese que en el paso 5, ρ_a puede ser modificado.

2- El cómputo de $|X^{ab}|$ en el paso 4, en la etapa de aprendizaje, en esencia consiste en determinar el peso de la conexión de la categoría ganadora de ARTa a la neurona del mapa que tiene conexión 1-1 con la neurona ganadora de ARTb. Después en el paso 5 se verifica cuando "grande" es este peso.

En los pasos 2 y 3 ocurre que por lo general no coinciden los índices de las neuronas ganadoras de ARTa y ARTb, lo cual no significa que ARTb desconfirma la predicción de ARTa. Esto es, pues precisamente hay un mapa que se encarga de asociar tales índices.

3- En el paso 5, se actualiza ρ_a de esa manera para garantizar que en la próxima activación de ARTa no se vuelva a seleccionar la categoría J de ARTa como ganadora. Nótese que el primer término es el mismo que la expresión utilizada para verificar la restricción (4) de la red FuzzyART.

Respecto a la expresión de actualización (8), se debe observar que cuando se acomete una nueva categoría, el peso de su conexión a la neurona del mapa de índice K, se mantiene igual a uno, después de aplicar la expresión; mientras que los pesos de las conexiones hacia las demás neuronas decrecen en una cantidad proporcional a $(1-\beta_{ab})$. Por otro lado, los pesos de las conexiones cuyo origen es una neurona de índice diferente a J, permanecen igual.

Cuando se satisface la desigualdad se modifica ρ_a y se vuelve al paso 3, lo cual implica que ante la presentación de una pareja (A,B) a la red FuzzyARTMAP, puede ocurrir que varios vectores de pesos de neuronas de ARTa sean actualizados.

Debe notarse que si se satisface la desigualdad, se vuelve al paso 3 para intentar buscar otra categoría ganadora en ARTa. En el apéndice II se demuestra que siempre existe al menos una, a no ser que ya todas las categorías de ARTa estén acometidas.

4- En el paso 2 de la formulación original de esta red [Carpenter92a] no se dice como seleccionar la categoría ganadora de ARTb, en caso de que la red sólo sea utilizada para clasificar o predecir. Sin embargo, a partir del hecho de que los pesos W_j^{ab} expresan el grado de pertenencia de cada categoría de ARTa a cada clase de ARTb, se deriva, que buscando el máximo de las componentes de W_j^{ab} se obtiene la categoría deseada de ARTb. De ahí que la modificación a este paso consistiría en no activar ARTb e incorporar la siguiente búsqueda al paso 3:

$K = MAX\{ W_{Jk}^{ab} , 0 \le k \le N_b \}$
donde J es la categoría ganadora de ARTa

CAPÍTULO III: AMONSTA: Una nueva Red Neuronal

Antecedentes

Considere el problema de aprendizaje en un medio ambiente no estacionario. A diferencia de medios ambientes estáticos, no es posible estimar un conjunto de parámetros para construir una función de clasificación, la cual para siempre proporcione la clase correcta de datos desconocidos, sino que por el contrario debe contarse con un método de aprendizaje suficientemente adaptivo que sea capaz de detectar los cambios en el medio ambiente y adaptar sus parámetros a las nuevas condiciones. De este modo clasificadores que son construidos basados en hipótesis estacionarias no funcionan, por razones obvias.

A primera vista parece que los métodos adaptivos del área de Reconocimiento de Patrones son capaces de resolver el problema planteado, sin embargo pueden mostrarse contraejemplos interesantes en los que se pone de manifiesto el conocido problema de saturación de la memoria. En general estos métodos parecen funcionar mejor cuando la distribución de las clases de nuevos objetos (etapa de uso) es una ligera extensión de la distribución de las clases utilizadas para el entrenamiento.

Un contraejemplo es el siguiente, considere un problema de dos clases generadas por una distribución unimodal, unidimensional y sin mezclas, la cual se mueve con el tiempo de manera tal que puntos que de la clase uno que eran generados en determinada región en el instante $t=t_0$, son generados en el instante $t=t_1$, en la región donde se encontraban los puntos de la clase dos y viceversa. En el caso extremo la pertenencia a las clases ha sido completamente intercambiada y los puntos que pertenecían a la clase uno estarán en la región de los puntos que eran de la clase dos. Si se considera un sistema con un número prefijado de parámetros a aprender, entonces un intento de hacer que este sistema se adapte a los cambios del medio ambiente anterior, provocará que se sature su capacidad y eventualmente no predecirá o aprenderá nuevos datos.

Bajo esta motivación el objetivo fue entonces proponer una nueva red neuronal [Sánchez95b, Sánchez96] capaz de resolver el problema planteado. Esta nueva red evita el problema de saturación de la memoria mediante un mecanismo de olvido con el que cuenta, y es así como se adapta a los cambios del medio ambiente.

En el problema de Reconocimiento de Palabras Aisladas Independiente del Usuario (RPAIU), tradicionalmente [Rabiner81, Rabiner89, Casacubierta87, Herrera94], el sistema es entrenado utilizando un conjunto finito prefijado de entrenamiento compuesto de señales de voz que fueron pronunciadas por un conjunto finito prefijado de parlantes. Usualmente, el conjunto de entrenamiento es grande para garantizar la representatividad de las diferentes clases y así obtener un comportamiento aceptable del sistema, aún si éste será usado por un pequeño grupo de usuarios desconocidos. Esta aproximación presenta algunas desventajas, primero el sistema suele ser lento no sólo en la etapa de aprendizaje sino también en la etapa de uso. Una segunda y muy remarcada desventaja es que el sistema no podrá ser reentrenado en línea aún si su comportamiento es pobre para nuevos usuarios.

Adaptar el sistema para nuevos usuarios puede ser considerado como cambios en los parámetros de la distribución de las clases; así que se espera que la red neuronal AMONSTA [Sánchez96] se comporte aceptable en la solución del problema RPAIU.

Ideas preliminares

Este nuevo modelo de red neuronal determina un cubrimiento supervisado de los datos de entrenamiento dados en un determinado momento. Cuando el modelo se utiliza sólo para predecir, entonces los objetos de entrada se clasifican según la hiperesfera en que caigan. En caso de que un determinado objeto no caiga en una hiperesfera, este es asignado a la clase cuyo vecino más cercano sea el centro de una hiperesfera de la misma clase. Para asegurar coherencia, si hay más de una hiperesfera cubriendo el objeto, se utilizan reglas adicionales las cuales se presentan más adelante.

AMONSTA (del inglés Adaptive Model for Non-Stationary environments) es una red neuronal de entradas continuas con aprendizaje supervisado, en-línea y adaptivo. Los problemas de proliferación de prototipos y saturación de memoria son evitados por medio de un mecanismo de olvido, la fusión de prototipos y la actualización de los radios de las hiperesferas.

Para esta red, un prototipo es una combinación lineal de objetos cercanos entre sí, aunque debe señalarse que cada prototipo se forma de manera adaptiva a medida que el aprendizaje procede.

Arquitectura

La red consta de tres capas conectadas con alimentación hacia adelante. El número de neuronas en la capa de entrada está determinado por la dimensión de los objetos de entrada. Cada neurona de la segunda capa o capa oculta se asocia con un prototipo que no es más que el vector de pesos de sus conexiones de entrada, mientras que su umbral se asocia con el radio de la hiperesfera. Inicialmente esta capa no tiene neuronas y durante el aprendizaje se van añadiendo. La tercera y última capas tiene tantas neuronas como clases.

Cada neurona de la capa oculta, recibe conexiones de todas las neuronas de la capa de entrada. Cada neurona de la capa de salida recibe conexiones de un subconjunto de neuronas de la capa oculta. Todas las neuronas de un subconjunto, conectado a una misma neurona en la capa de salida, están asociadas a la misma clase de esta neurona. Ver la figura 3.

Cada neurona de la capa oculta tiene además asociado un coeficiente de vitalidad, que está directamente relacionado con el número de activaciones de la neurona. Mediante este coeficiente se toma la decisión de cuándo debe ser borrada la neurona.

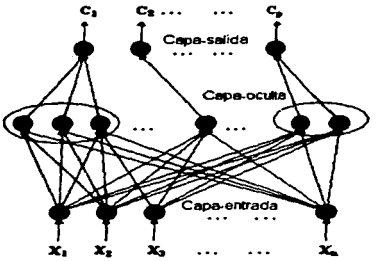


Figura 3. Esquema general de una red tipo AMONSTA.

Utilizando una notación vectorial, a las neuronas de entrada se asocia el vector de M dimensiones X. A las neuronas de la capa oculta se les asocia el vector Y, cuya dimensión varía según la cantidad de prototipos que existen. Por último, a las neuronas de la capa de salida, se les asocia el vector C de p dimensiones (p= número de clases). A cada neurona j de la capa oculta, se le asocia un vector Wj de M dimensiones.

AMONSTA inicialmente no tiene memoria; es decir, no hay neurona representando prototipos en la capa intermedia. Ante un error de clasificación la red crea un nuevo prototipo asociándolo con la entrada actual. Al presentar el primer objeto a la red, éste no puede ser clasificado por no existir prototipos, por lo que se crea el primer prototipo a partir de este primer objeto. Posteriormente, cuando se presenta el segundo objeto, si éste es clasificado correctamente por el único prototipo que hay, entonces utilizando el objeto actual que actualiza el prototipo, lo cual provoca un ligero desplazamiento de éste en la dirección del objeto actual. En caso contrario si se produce una clasificación incorrecta, un segundo prototipos creado. El proceso anterior es repetido con todos los objetos de entrada. A continuación se explican los procesos fundamentales que tienen lugar en una red tipo AMONSTA.

Creación de un nuevo prototipo

La creación de un nuevo prototipo tiene lugar cuando no hay prototipos, o los que existen son incapaces de clasificar correctamente el objeto de entrada X.

Supongamos que el nuevo prototipo a ser creado tiene índice N, entonces:

$N^{(nuevo)} = N^{(viejo)} + 1$, N es el número de prototipos en un momento dado.
$W_N = X$
$R_N = r_\varepsilon$, donde: R_N es el radio del prototipo N-ésimo, $R_N \in \Re$,
r_ε es un valor suficientemente cercano a cero, $r_\varepsilon \in \Re$

Desde el punto de vista geométrico de creación de un prototipo se traduce en crear una hiperesfera cuyo centro es X y radio r_ε.

Selección del prototipo ganador

Para determinar el prototipo ganador primero se calcula la distancia euclidiana del objeto X a cada prototipo W_i ($1 \le i \le N$). Si X no está incluido en ninguna hiperesfera entonces el prototipo ganador W_i es el más cercano a X, exigiendo para la etapa de aprendizaje, que W_i sea de la misma clase que X. Si X está incluido en una o más hiperesferas entonces el ganador W_J es aquel que se encuentre más cercano a X y además lo contenga en su región. Si hay más de un prototipo que cumpla lo anterior se toma el de menor índice.

Verificación de la intersección de dos hiperesferas

La cuantificación de la intersección de dos regiones asociadas con prototipos, es decir, de dos hiperesferas, se determina por la siguiente expresión:

$$S = R_1 + R_2 - d(P_1, P_2)$$

donde: W_1, R_1 y W_2, R_2 son dos prototipos y sus respectivos radios.

De esta expresión se deriva el hecho de que si $S<0$ entonces no hay intersección entre las hiperesferas; mientras que si $S \ge 0$ entonces sí hay intersección y además S es una medida de cuán grande es la misma.

Cada vez que un prototipo y/o su radio se modifica, la expresión anterior es utilizada para determinar si hay o no intersección entre la nueva hiperesfera y cualquier otra de las demás clases. Si S>0 entonces se dice que la *condición de intersección no se satisface*.

Actualización de un prototipo y su radio

Cuando el prototipo que resulta ganador clasifica correctamente al actual objeto de entrada a la red; tiene lugar una actualización al prototipo y/o a su radio, siempre que se satisfaga la condición de intersección para el nuevo prototipo y su radio. Esto se hace para garantizar que no exista intersección entre la nueva hiperesfera y cualquier otra de las demás clases. La actualización del prototipo y su radio se hace:

$$W_J^{(nuevo)} = W_J^{(viejo)} + \delta^*(X - W_J^{(viejo)}) \qquad (9)$$

donde: W_J es el prototipo ganador
δ es una constante conocida como velocidad de aprendizaje, $0<\delta \leq 1$

$$R_J^{(nuevo)} = R_J^{(viejo)} + d(X, W_J^{(nuevo)})$$

donde: R_J es el radio del prototipo W_J
d es la distancia euclidiana

Estas actualizaciones solo tienen lugar si la condición de la intersección es satisfecha para $(W_J^{(nuevo)}, R_J^{(nuevo)})$.

Fusión de un prototipo con un objeto

La fusión entre un prototipo y un objeto significa actualizar el prototipo y su radio, de forma tal que tanto el viejo prototipo como el objeto, queden contenido en la hiperesfera del nuevo prototipo. Esta fusión, tiene lugar cada vez que para un objeto X, resulta ganador un prototipo que no contiene a X en su región. Claro está, similar al proceso de actualización visto previamente, esta fusión sólo es válida si se satisface la condición de intersección para el prototipo resultante. La fusión es un caso particular del proceso de actualización, utilizando δ=1/2.

Fusión de prototipos

La fusión de dos o más prototipos significa crear un nuevo prototipo cuyo radio los contenga. Con este tipo de fusión se busca reducir el número de prototipos que la red crea y además evitar en cierta medida la influencia del orden en que los objetos fueron presentados a la red. La fusión de prototipos puede consumir algún tiempo de cómputo por lo cual debe decidirse si se utiliza al finalizar la presentación de cada objeto o cada cierto número de presentaciones. A continuación se presentan varias maneras de fusionar prototipos:

a) fusión general de M prototipos: se crea un nuevo prototipo que es la media de los otros M, ($M \geq 2$) y se toma el radio de forma tal que queden contenido en la nueva hiperesfera. En este trabajo se implementó el caso particular de M=2.
b) verificación de prototipos cubiertos por otros: se verifica si la hiperesfera de un prototipo es cubierta completamente (o casi) por M ($M \geq 1$) prototipos. En este trabajo se implementó el caso particular de M=1.

Debe observarse que la variante a), considerada en general, es un problema combinatorio por lo que deben buscarse heurísticas adecuadas.

Reducción del Radio

De acuerdo al proceso de selección del prototipo ganador, presentado anteriormente, primero se busca el más cercano a X, entre aquellos prototipos que contienen al objeto X (en su hiperesfera). De esta manera, tales prototipos tienen mayor preferencia, respecto a aquellos que no contienen a X, ocurriendo de hecho, que aunque X se encuentre más cercano a un prototipo W_1, si se encuentra dentro de la región de otro prototipo W_2, entonces X es clasificado por W_2. Este comportamiento es ventajoso, fundamentalmente, para clasificar objetos que se encuentran cercanos a la frontera de alguna subregión de su clase.

En esta red, la frontera de la hiperesfera no se considera como parte de la región que un prototipo pueda clasificar bajo el esquema anterior, o sea X puede ser clasificado "primeramente" por W_J si $d(X,W_J)<R_J$.

Como consecuencia de todo lo anterior, una manera eficiente de reducir el radio de un prototipo W_J cuando se comete un error de clasificación con W_J es:

$R_J = d(X,W_J)$

Esta actualización tiene lugar, como se verá más adelante, cuando X es clasificado por un prototipo W_J, $d(X,W_J)<R_J$ y la clase de X es diferente a la clase de W_J.

Mecanismo de Olvido

Una red tipo AMONSTA cuenta con un mecanismo de olvido, mediante el cual se borran neuronas de la red. El mecanismo de olvido se implementa básicamente a través de un coeficiente V, que llamaremos coeficiente de vitalidad. Todas las neuronas de la capa de intermedia tienen asociado un coeficiente de vitalidad mediante el cual se modela el número de veces que dicha neurona ha resultado ganadora.

El mecanismo que se presenta fue elaborado atendiendo a los siguientes principios:

1.- Prototipos creados recientemente no deben ser borrados. Esto dota a la red de una mayor estabilidad.

2.- No debe ser considerado el concepto de iteración en dicho mecanismo, con lo cual, el mecanismo resulta de mayor uniformidad respecto al tipo de red en que se está utilizando, o sea, dado que esta red es de tipo adaptivo, entonces no tiene sentido hablar de iteraciones.

3.- Evitar complejidad computacional en la implementación del mecanismo, evitando que la red se vuelva excesivamente más lenta.

El mecanismo de olvido, se define a través de los coeficientes de vitalidad V_i ($1 \le i \le N$) y las constantes α y β, como sigue:

-Cuando se crea una nueva neurona en la segunda capa, su coeficiente de vitalidad es inicializado con la constante α:

$V_J = \alpha, \quad \alpha > 1$

-Cuando un objeto es presentado a la red, el coeficiente de vitalidad de todas las neuronas es actualizado de acuerdo a la siguiente expresión:

$$V_j = \begin{cases} V_j * \alpha & si\ j = J\ índice\ de\ la\ neurona\ ganadora \\ \frac{V_j}{\alpha} & si\ j \neq J \end{cases}$$

-Si $V_j < \beta_v$ entonces la neurona j-ésima se borra
donde: β_v es una constante que representa la cota inferior para el coeficiente de vitalidad.
$0 \leq \beta_v \leq 1$

Algoritmo de Aprendizaje y Predicción

0.- Activar la capa de entrada con X. Se utilizará la notación funcional C(X) para obtener la clase de X.
1.- Activar la capa oculta mediante el cálculo de las distancias y actualizar los coeficientes de vitalidad asociados:

$$Y_j = \sum_{i=1}^{M} (X_i - W_{ji})^2 \qquad \forall\ 1 \leq j \leq N \qquad (10)$$

$$V_j^{nuevo} = \frac{V_j^{viejo}}{\alpha} \qquad \forall\ 1 \leq j \leq N$$

2.- Determinar la neurona ganadora:
2.1 Sea b un vector binario de N componentes:

$$b_j = \begin{cases} 1 & si\ \ Y_j < R_j \\ 0 & e.o.c. \end{cases} \qquad \forall\ 1 \leq j \leq N$$

2.2 Si $\sum_{j=1}^{N} b_j = 0$ entonces

Si C(X)= "no conocida" entonces

$$J = MIN(Y) = \begin{cases} j & si\ \ \exists j, 1 \leq j \leq N (Y_j \leq Y_l, \forall 1 \leq l \leq N) \\ -1 & e.o.c. \end{cases}$$

ir a 5-

Si C(X)≠ "no conocida" entonces

$$J = MIN^*(Y) = \begin{cases} j & si \quad \exists j, 1 \le j \le N (C(Y_j) = C(X) \wedge Y_j \le Y_i, \forall 1 \le i \le N) \\ -1 & e.o.c. \end{cases}$$

ir a 3-

2.3 Si $\sum_{j=1}^{N} b_j > 0$ entonces

$$J = MIN^+(Y) = \begin{cases} j & si \quad \exists 1 \le j \le N (b_j = 1 \wedge Y_j \le Y_i, \forall 1 \le i \le N) \\ -1 & e.o.c. \end{cases}$$

3.- Si J<0 entonces crear un nuevo prototipo, hacer J=N e ir a 5-

4.- Si $C(Y_J)$= C(X) y b_J=1 entonces actualizar el prototipo W_J y su coeficiente de vitalidad:

$W_J^{(nuevo)} = W_J^{(viejo)} + \delta^*(X - W_J^{(viejo)})$

$V_J^{(nuevo)} = V_J^{(viejo)} * \alpha^2$, ir a 5-

4.1 Si b_J=0 entonces hacer la fusión de W_J y X si es posible; en cuyo caso además actualizar el coeficiente de vitalidad: $V_J^{(nuevo)} = V_J^{(viejo)} * \alpha^2$.

Si no es posible la fusión de W_J y X entonces crear un nuevo prototipo, hacer J=N e ir a 5-

4.2 Si b_J=1 entonces reducir el radio R_J e ir a 5-

5. El objeto X se clasifica en la clase $C(Y_J)$ siempre que J>0, en caso contrario, significa una clasificación de abstención.

5.1 Fusionar prototipos

5.2 Eliminar los prototipos para los que se satisfaga: $V_j < \beta_V$, $1 \le j \le N$

Comentarios

1.-En el paso 4.1 debe observarse que si b_J=0 entonces significa que el prototipo ganador es de la misma clase, pero no contiene a X.

2.- En el paso 4.2 si b_J=1 entonces $C(Y_J) \neq C(X)$, pero además X está contenida en la región de W_J por tanto, debe reducirse el radio de W_J.

3.- En el paso 5, sólo ocurre que J<0 cuando la clase de X no es conocida y la red no tiene ninguna neurona oculta, lo cual sólo se presenta al inicio.

4.- Puede realizarse una formulación del método, con respecto a la creación de prototipos, en el mismo sentido que las redes FuzzyART y FuzzyARTMAP, así también con respecto al proceso de búsqueda de la neurona ganadora, pero variando las condiciones que debe satisfacer esta.

5.- Este algoritmo es fácilmente modificable para incorporar nuevas reglas heurísticas al estilo de los pasos 2.2, 2.3, 4.1, 4.2.

Adaptaciones de la red para el problema de Reconocimiento de Palabras Aisladas

Para aprovechar el vector de valores propios λ asociado con cada objeto de entrada se propone modificar las expresiones (10) y (9) como sigue:

$$Y_j = \sum_{i=1}^{M} \lambda_i (X_i - W_{ji})^2 \qquad \forall\ 1 \le j \le N \qquad (10')$$

$$W_{jk}^{(nuevo)} = W_{jk}^{(viejo)} + \delta \cdot \lambda_k (X_k - W_{jk}^{(viejo)}) \qquad \forall\ 1 \le k \le M \qquad (9')$$

Estimación de los parámetros de la red para un conjunto finito de objetos

El ajuste de los parámetros de una red neuronal, en general, suele ser un problema difícil. En una red tipo AMONSTA los parámetros a estimar son δ (velocidad de aprendizaje), α (vitalidad inicial) y β_V (cota inferior de vitalidad). Considerando el problema de RPAIU, en lo que sigue se proponen vías de estimar tales parámetros.

Aunque en el capítulo IV se explican detalladamente los resultados, conviene en este momento hacer una breve mención de dos tipos de resultados que se presentarán, con el objetivo de ayudar a esclarecer la manera en que se ajustaron los parámetros.

El resultado fundamental que se pretende mostrar es que una red tipo AMONSTA es capaz de adaptarse a un nuevo usuario, a partir de un conjunto finito y pequeño de señales de voz pronunciadas por éste.

Primeramente se presentarán los resultados tradicionales que consisten en dividir todos los datos en dos muestras una de aprendizaje y otra de verificación, para llevar a cabo el aprendizaje y la verificación, respectivamente. El objetivo de presentar estos resultados es, por un lado ver el papel que juegan los valores propios en la descripción de los objetos mediante subpalabras acústicas y por otro lado, explorar criterios para ajustar los parámetros de las redes utilizadas.

Posteriormente se presentarán los resultados adaptivos, que consisten en entrenar primeramente la red para un conjunto de datos iniciales, luego reentrenarla utilizando los datos de entrenamiento de un usuario y mostrar los resultados de clasificación para sus datos de verificación. Después, se utilizan los datos de otro nuevo usuario para reentrenar la red (previamente entrenada) y se clasifican los datos de verificación de este nuevo usuario. Este proceso se repetirá para cada nuevo usuario con que se desea probar la red.

Parámetro δ

El parámetro δ muy conocido de otras redes como las de la familia ART y los perceptrones (en las que se fija ad hoc), representa cuánto se toma del actual objeto de entrada X para modificar un determinado vector de pesos W. Si δ es grande entonces se toma "mucho" de X y "poco" de W, y viceversa. Después de algunas pruebas sobre los datos utilizados en los resultados tradicionales, se llegó a la conclusión de que el comportamiento de la red no varía mucho para diferentes valores de δ y se decidió fijarlo a 0.4 para garantizar buenos resultados adaptivos. Nótese que en principio puede ocurrir que los datos que se presentan por cada nuevo usuario no tengan que ver con los aprendidos previamente por la red, o sea puede existir un cambio en los parámetros de las distribuciones de los nuevos datos, provocando que existen objetos de determinada clase C_1 cercanos a prototipos de una clase distinta C_2.

Parámetros α y β_V

El parámetro β_V representa la cota inferior para el coeficiente de vitalidad de cada neurona de la red, de forma tal que si $V_j<\beta_V$ entonces la neurona j-ésima es borrada.

Una exigencia lógica para garantizar estabilidad en la red es que en las primeras épocas (en particular en la primera) no se borre ninguna neurona, pues la red está aprendiendo por primera vez los datos. Supongamos que se desea no borrar neurona alguna en las primeras k iteraciones. El peor caso se presenta cuando la

red cuenta de dos o más neuronas y la primera neurona que se crea nunca resulta ganadora, pues al cabo de k iteraciones (ver Apéndice III) se tiene:

$$V_1 = \frac{1}{\alpha^{k-2}}$$

Del peor caso se obtiene entonces que β_V puede ser tomado como:

$$\beta_v = \frac{1}{\alpha^{k-2}} \quad \text{con } \alpha > 1,\ k \geq N \geq 2$$

De esta manera los parámetros α y β se estiman a través del parámetro k, con lo cual no sólo se disminuye un parámetro a estimar, sino también que es mucho más fácil de estimar k. Se decide tomar k como el número de objetos en la muestra de entrenamiento, garantizando así que la red no borre neuronas durante el entrenamiento inicial.

Al concluir el entrenamiento inicial ocurre, en general, que existen varios $V_j<\beta_V$. Si se utiliza la misma estimación de β_V para la muestra de entrenamiento del nuevo usuario con que se desee reentrenar la red, ocurrirá que se borrarán neuronas (aprendidas en el entrenamiento anterior); pero de una manera no controlada. Si no se deseará borrar neurona alguna bastaría tomar:

$$\beta_v = \frac{V_{min}}{\alpha^k}, \quad V_{min} = \min_{1 \leq j \leq N} V_j, \quad \text{N: número de neuronas de la red entrenada}$$

Dado que nuestro interés es que si se borren neuronas, entonces se decide tomar:

$$\beta_v = \frac{\overline{V}}{\alpha^k}, \quad \overline{V} = \frac{\sum_{j=1}^{N} V_j}{N}$$

k: número de objetos en la muestra de entrenamiento
V_j, $1 \leq j \leq N$: coeficientes de vitalidad aprendidos previamente

Utilizando la estimación anterior para β_V, la tendencia de la red será a borrar primero las neuronas con coeficientes de vitalidad $V_j < \overline{V}$, aunque no necesariamente pues depende de k y de si tales neuronas comienzan a clasificar correctamente a los nuevos objetos.

Debe observarse que dependiendo del problema pueden utilizarse otras maneras de estimar β_V como función de las V_j, por ejemplo utilizando la mediana o la moda de las V_j. Todas estas variantes pueden se fácilmente verificadas mediante un sencillo programa de simulación en el cual se puedan hacer supuestos sobre las muestras de datos.

Simulaciones

En esta sección se muestra el comportamiento de una red AMONSTA para un problema simulado. Para ganar una mayor claridad de cómo trabajan este tipo de redes, se decidió utilizar un problema en el que los objetos son de dos componentes, y así poder graficar fácilmente los resultados.

En la simulación hay dos clases A y B generadas aleatoriamente a través de una distribución uniforme. Los objetos de la clase A pertenecen a la región limitada por la circunferencia de centro en el origen y radio uno. Los objetos de la clase B pertenecen al anillo limitado por dos circunferencias de centro en el origen, con radio uno y dos respectivamente; ver figura 4. En este problema las clases no se solapan, no son linealmente separables y no cambian con el tiempo.

Figura 4. Representación gráfica de los objetos y sus clases. Se utiliza cuadros para la clase A y puntos para la clase B.

Se generaron aleatoriamente dos muestras de datos, una de aprendizaje y otra de entrenamiento, cada una de 200 objetos. En las tablas 1-a y 1-b se resume la información básica de cada una de las etapas y en la figura 5 se muestran los prototipos de cada clase creados durante la etapa de entrenamiento.

Se observa que la red creó el número de prototipos óptimo (1) para la clase A. Además el número total de prototipos creados (13) está cercano a la cantidad óptima de prototipos para este problema (9).

TABLA 1-a. Datos de la etapa de entrenamiento

Total de objetos	200
Total en la clase A	100
Total en la clase B	100
Total de épocas	1
Total de prototipos	13
Total en la clase A	1
Total en la clase B	12
α=1.001 β=0.8204 δ=0.2	

TABLA 1-b. Datos de la etapa de verificación

Total de objetos	200
Total en la clase A	100
Total en la clase B	100
Total de errores	4 (2%)
Total de la clase A	3
Total de la clase B	1

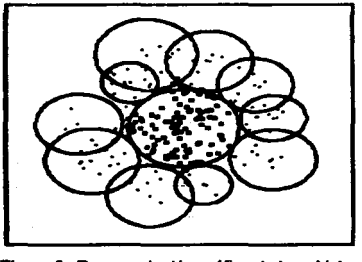


Figura 5. Representación gráfica de los objetos y los prototipos. Se utiliza cuadros para la clase A y puntos para la clase B.

CAPITULO IV: Adaptaciones generales

Cantidad variable de subpalabras acústicas

Para todas las redes revisadas en los capítulos II y III se presenta el mismo problema y es que la cantidad variable de subpalabras acústicas (SAC) determina diferencias en cuanto a las longitudes de objetos asociados a distintas SAC. Aunque pudiera pensarse en modificar las distancias y/o medidas de similaridad utilizadas por estas redes, para utilizar algún algoritmo de tipo DTW, debe notarse que esto resultaría extremadamente costoso en tiempo.

En este trabajo se propone utilizar varias redes asociadas con el número de SAC, es decir tantas redes como diferentes números de SAC se tenga. Con lo cual se resuelve no sólo el problema anterior, sino también resulta extremadamente rápido. Debe notarse que todas las redes revisadas emiten una clasificación a partir de comparaciones con un conjunto de prototipos calculados previamente, por lo que mientras menor sea la cardinalidad de este conjunto más rápido clasifica la red. Así dada una palabra, se selecciona cual red utilizar según el número de SAC y se reduce considerablemente el número de prototipos con los cuales comparar. A partir de los resultados experimentales se obtuvo que aproximadamente más del 80 % de los datos de TI-46 se subdividen entre 5,6,7, y 8 subpalabras acústicas, por lo que los resultados que se presentarán se limitaron a estos números. Para las otras cantidades de SAC se presenta la limitante de que hay dígitos que no quedan adecuadamente representados en las muestras de entrenamiento y/o verificación, pues no hay objetos o son extremadamente pocos.

Empleo de los valores propios

Asociado a cada objeto existe un vector de valores propios λ. En los capítulos II y III se propusieron modificaciones a las redes presentadas para tomar en consideración los valores propios en la evaluación de las medidas de similaridad, así como en la regla de actualización de los pesos de la red.

Una primera idea acerca de cómo utilizar los valores propios pudiera pensarse que es emplear el vector λ asociado al actual objeto de entrada, tanto en el aprendizaje como en la etapa de verificación. En este caso se tiene que el papel preponderante lo desempeña el objeto de entrada actual y no el prototipo, pues la similaridad entre ambos se llevará a cabo acorde con las restricciones que

impone el λ del objeto actual. En este sentido pudiera pensarse también en otras maneras de combinar los λ de cada objeto. Por razones de eficiencia en memoria y tiempo de ejecución se decidió estimar un único vector λ_*. Este vector se construye dinámicamente a partir de todos los vectores λ asociados a los objetos que se han presentado a la red y clasificados por ésta de manera correcta. Cuando se va a evaluar la similaridad entre cada prototipo y el objeto actual **X**, se utiliza el λ_* construido hasta el momento y si **X** resulta bien clasificado entonces λ_* es actualizado utilizando el λ actual. A continuación se proponen algunas variantes de actualización de λ_*:

a) $\lambda_* = (\lambda_* + \lambda)/2$

En este caso la idea es obtener un vector λ_* cercano al promedio de todos los λ presentados cuyo **X** es bien clasificado. Durante la etapa de verificación se utiliza λ_* en lugar de λ de cada objeto de entrada.

b) $\lambda_* = MIN(\lambda_*, \lambda)$

Tomando en cuenta que $\lambda_*, \lambda \in [0,1]$ entonces aplicar MIN es como una especie de AND entre ambos vectores, por lo que pudiera decirse que esta actualización es "pesimista" pues da el menor peso posible a cada componente de λ_* y por tanto basta que para algún objeto determinadas partes no tengan importancia para que entonces en los demás objetos dichas partes se consideren de manera similar. En este caso durante la etapa de verificación se utiliza la misma expresión.

c) $\lambda_* = MAX(\lambda_*, \lambda)$

Al contrario del caso anterior, aplicar el MAX es como una especie de OR entre λ_* y λ, por lo que pudiera decirse que es una actualización "optimista" ya que basta que para algún objeto sea importante determinadas partes y así lo será también para los demás. Durante la etapa de verificación se utiliza la misma expresión.

CAPITULO V: Resultados

En lo que sigue se presentan varios resultados con el objetivo de explorar los siguientes aspectos:
- eficiencia de distintas maneras de utilizar los valores propios en la ponderación de las medidas de similitud
- capacidad de las redes presentadas para adaptarse a cada nuevo usuario, entrenando solamente con algunas repeticiones del usuario en cuestión. En este caso se evalúa además el papel que juega haber entrenado previamente a la red para un grupo de usuarios distintos del nuevo.

Eficiencia de la ponderación con valores propios

En el capítulo IV se presentaron varias maneras de utilizar los valores propios. En la presente sección se muestran los resultados de varias de ellas para las dos redes neuronales que se estudian en este trabajo. Primeramente se muestra el comportamiento de ambas redes para resultados tradicionales, es decir se separaron los datos en dos muestras, una de aprendizaje y otra de verificación y se realizaron el aprendizaje y la verificación con ellas, respectivamente.

Los resultados que se muestran fueron los mejores obtenidos durante el ajuste empírico de los parámetros. En estos resultados no se presentan las matrices de confusión pues el objetivo es simplemente notar como varía la eficiencia con cada manera de utilizar los valores propios.

Las tablas de esta sección constan de las siguientes columnas:
- SAC: número de subpalabras acústicas
- Objet: número de objetos en la muestra de entrenamiento o verificación según sea
- Parámetros: en esta columna se especifican los valores utilizados para los parámetros según el método de que se trate. Para FuzzyARTMAP se muestran valores para (β_a,ρ_a), mientras que para AMONSTA (δ).
- Epoc: número de veces que se presenta la muestra de entrenamiento completa
- Protot: número de prototipos o neuronas creadas por la red y entre paréntesis el porciento con respecto al total de objetos de la muestra de entrenamiento.
- Errores: número de objetos mal clasificados y entre paréntesis el porciento con respecto al total de objetos de la muestra de verificación.

La última fila de cada tabla resume los resultados más importantes.

FUZZYARTMAP ENTRENAMIENTO					VERIFICACION		AMONSTA ENTRENAMIENTO			VERIFIC
SAC	objet	parámetros	época	protot	objet	errores	parámet	época	protot	errores
5	268	0.01, 0.985	1	268	175	152	0.4	1	183	166
6	411	0.01, 0.985	1	411	291	268	0.4	1	386	269
7	370	0.01, 0.985	1	370	217	188	0.4	1	355	190
8	225	0.01, 0.985	1	225	144	122	0.4	1	209	124
TOT	1274	?	?	1274 (100.00)	827	730 (88.27)	?	?	1133 (88.93)	749 (90.56)

TABLA 2. Resultados Tradicionales sin utilizar ponderación. El símbolo ? significa que no tiene sentido calcular ese dato.

FUZZYARTMAP ENTRENAMIENTO					VERIFICACION		AMONSTA ENTRENAMIENTO			VERIFIC
SAC	objet	parámetros	época	protot	objet	errores	parámet	época	protot	errores
5	268	1.0, 0.98	2	213	175	127	0.4	1	181	24
6	411	1.0, 0.98	1	284	291	217	0.4	1	404	29
7	370	1.0, 0.98	2	293	217	162	0.4	1	251	25
8	225	0.01, 0.985	1	165	144	86	0.4	1	174	25
TOT	1274	?	?	955 (74.96)	827	592 (71.58)	?	?	1010 (79.28)	103 (12.4)

TABLA 3. Resultados Tradicionales. $\lambda_{-}= (\lambda_{-}+\lambda)/2$. El símbolo ? significa que no tiene sentido calcular ese dato.

FUZZYARTMAP ENTRENAMIENTO					VERIFICACION		AMONSTA ENTRENAMIENTO			VERIFIC
SAC	objet	parámetros	época	protot	objet	errores	parámet	época	protot	errores
5	268	0.011, 0.985	1	187	175	106	0.4	1	227	25
6	411	0.01, 0.985	1	262	291	187	0.4	1	402	36
7	370	0.01, 0.985	1	245	217	106	0.4	1	298	26
8	225	0.01, 0.985	1	212	144	116	0.4	1	168	27
TOT	1274	?	?	906 (71.11)	827	515 (62.27)	?	?	1095 (85.95)	114 (13.78)

TABLA 4. Resultados Tradicionales. $\lambda_{-}=MIN(\lambda_{-},\lambda)$. El símbolo ? significa que no tiene sentido calcular ese dato.

FUZZYARTMAP ENTRENAMIENTO					VERIFICACION		AMONSTA ENTRENAMIENTO			VERIFIC
SAC	objet	parámetros	época	protot	objet	errores	parámet	época	protot	errores
5	268	0.011, 0.985	1	266	175	152	0.4	1	155	29
6	411	0.01, 0.985	1	409	291	265	0.4	1	347	30
7	370	0.01, 0.985	1	370	217	186	0.4	1	233	27
8	225	0.01, 0.985	1	223	144	122	0.4	1	160	27
TOT	1274	?	?	1268 (99,53)	827	725 (87.66)	?	?	895 (70,25)	113 (13.66)

TABLA 5. Resultados Tradicionales. $\lambda_{\bullet}$=MAX($\lambda_{\bullet}$,λ). El símbolo ? significa que no tiene sentido calcular ese dato.

Se observa, para ambas redes, que cuando la descripción de las señalarse de voz se hace mediante subpalabras acústicas, no tiene sentido pretender resolver el problema planteado si no se utilizan los valores propios. Aunque en la tabla 2 la red FuzzyARTMAP se comporta ligeramente mejor que AMONSTA, de todas maneras ambas se comportan peor que haber clasificado al azar.

Cuando se utilizan valores propios (tablas 3-5) la red AMONSTA tiene un porciento de errores entre 12.45 y 13.78, mientras que FuzzyARTMAP entre 62.27 y 87.66. Debe señalarse que resultó muy sorprendente el comportamiento de FuzzyARTMAP, pues es una red que ha sido muy utilizada y ha mostrado buenos comportamientos en otros problemas [Carpenter92b, Carpenter92c, Carpenter95, Sánchez95a]. Conviene señalar que dado que los autores no han proporcionado un método para el ajuste de los parámetros de esta red, se tuvo que hacer un número elevado de pruebas para ajustarlos, aunque no se demostró que la búsqueda realizada fue exhaustiva.

La cantidad de prototipos creados durante la etapa de aprendizaje es un indicador de la variabilidad de los objetos en la muestra de entrenamiento y del grado de similaridad de objetos de diferentes clases. Cuando se utilizaron valores propios (tablas 3-5), para AMONSTA el porciento de prototipos varió entre 70.25 y 85.95 del total de objetos de la muestra de entrenamiento, mientras que para FuzzyARTMAP entre 71.11 y 99.53. Se observa que AMONSTA tiene su menor y mayor porciento de prototipos cuando se construyen $\lambda_{\bullet}$ a partir del MAX y $\lambda_{\bullet}$ a partir del MIN, respectivamente, mientras que ocurre completamente lo contrario para la red FuzzyARTMAP. El hecho de que hayan obtenido porcientos tan alto de prototipos refleja el grado de dificultad del problema.

Capacidad de las redes para adaptarse a un nuevo usuario

En esta sección se presentan, lo que llamaremos resultados adaptivos, que se obtienen mostrando a la red los datos de entrenamiento de un nuevo usuario (para entrenarla) y luego los datos de verificación. Los datos de entrenamiento sólo se presentan una vez, es decir la red realiza una sola época, a partir de la cual debe ser capaz de aprender suficiente. Inicialmente se utiliza un conjunto de datos para entrenar la red y posteriormente se realiza repetidamente el esquema anterior. El entrenamiento para un nuevo usuario parte de la red previamente reentrenada para el usuario anterior. El conjunto de datos iniciales corresponde a usuarios diferentes a aquellos utilizados más tarde para verificar la capacidad adaptiva.

En las tablas 6 y 7 se presentan los resultados resumidos, mientras que en las tablas 8-12 se muestran las matrices de confusión por cada nuevo usuario. Los resultados corresponden a la manera de calcular λ- a partir del MIN. Las tablas constan de dos partes, la primera subtitulada como ANTES muestra los errores de clasificación cometidos por la red antes de ser reentrenada con los datos de entrenamiento del usuario en cuestión. La segunda parte subtitulada como DESPUES muestra los errores de clasificación una vez que ha sido reentrenada la red. En ambas partes la información por columna es la siguiente:

- Usuario: es el índice que identifica a uno de los cinco usuarios.
- Objet : total de objetos utilizados para la verificación
- SAC5, SAC6, SAC7, SAC8: número de errores cometido por la red para cada número de subpalabras acústicas
- Total: suma de los errores cometidos por cada número de subpalabras acústicas para un usuario y entre paréntesis el porciento respecto al total de objetos utilizados en la verificación.

FuzzyARTMAP	ANTES						FuzzyARTMAP	DESPUES			
usuario	sac5	sac6	sac7	sac8	objet	TOTAL	sac5	sac6	sac7	sac8	TOTAL
VI	10	26	18	15	86	69 (80.23)	3	26	18	15	62 (72.09)
VII	9	22	18	4	89	53 (59.55)	8	19	7	3	37 (41.57)
VIII	17	22	15	11	83	65 (78.31)	12	15	13	7	47 (56.63)
IX	6	23	15	11	84	55 (65.48)	5	18	11	9	43 (51.19)
X	12	15	22	19	88	68 (77.27)	12	12	14	14	52 (59.09)

TABLA 6. Resultados Adaptivos para FuzzyARTMAP. Ponderación con λ-=MIN.

AMONSTA	ANTES						AMONSTA	DESPUES			
usuario	sac5	sac6	sac7	sac8	objet	TOTAL	sac5	sac6	sac7	sac8	TOTAL
VI	6	11	13	8	86	38 (44.86)	1	7	9	10	27 (31.40)
VII	9	11	9	8	89	37 (41.57)	4	10	5	3	22 (24.72)
VIII	15	7	10	10	83	42 (50.60)	1	3	6	6	16 (19.28)
IX	0	10	9	3	84	22 (26.19)	4	6	2	3	15 (17.86)
X	6	10	6	6	88	28 (31.82)	1	1	2	1	5 (5.68)

TABLA 7. Resultados Adaptivos para AMONSTA.. Ponderación con λ-=MIN

El comportamiento de la red FuzzyARTMAP fue pésimo, tuvo porcientos de errores que varían entre 44.04 y 74.7. Por su parte AMONSTA tuvo mucho mejores resultados, variando entre 5.62 y 31.4. En la tabla 7 se observa que a medida que la red AMONSTA ha "visto" más usuarios los resultados mejoran, hasta el punto de cometer tan sólo un 5.62 % de errores.

En las tablas 8-12 se presentan los errores cometidos por AMONSTA para cada usuario por cada dígito. Las celdas vacías significan cero, o sea que no hubo confusión entre los dígitos correspondiente a fila y columna. El número romano en la primera casilla designa el usuario. Cada fila de la tabla indica las diferentes confusiones cometidas al clasificar el dígito, es decir con que otro dígito se ha confundido el asociado con la fila en cuestión. Cada columna indica cuántas veces ha sido confundido otro dígito con el asociado a la columna en cuestión.

VI	0	1	2	3	4	5	6	7	8	9	TOTAL
0			1	.				1	1	1	4
1	2							1			3
2				1				3	1		5
3	2							1			3
4										1	1
5					1					2	3
6								2			2
7											0
8	1			1							2
9	1	3									4
TOTAL	6	3	1	2	1	0	0	8	2	4	[illegible]

TABLA 8. Matriz de confusión para el usuario VI, utilizando el método AMONSTA. Ponderación con λ-=MIN

VII	0	1	2	3	4	5	6	7	8	9	TOTAL
0		1			1						2
1				1	2					1	4
2				2				1			3
3			2								2
4		1									1
5											0
6											0
7			1				1				2
8				2				1			3
9		2						3			5
TOTAL	0	4	3	5	3	0	1	5	0	1	[redacted]

TABLA 9. Matriz de confusión para el usuario VII, utilizando el método AMONSTA. Ponderación con λ-=MIN

VIII	0	1	2	3	4	5	6	7	8	9	TOTAL
0		1						1			2
1				1						2	3
2	1			1							2
3			2								2
4											0
5											0
6			1	1					1		3
7											0
8											0
9	2			1				1			4
TOTAL	3	1	3	4	0	0	0	2	1	2	[redacted]

TABLA 10. Matriz de confusión para el usuario VIII, utilizando el método AMONSTA. Ponderación con λ-=MIN

IX	0	1	2	3	4	5	6	7	8	9	TOTAL
0		1		1							2
1										1	1
2				3							3
3			1				1	1	1		4
4			1					1			2
5											0
6											0
7											0
8		1	2								3
9											0
TOTAL	0	2	4	4	0	0	1	2	1	1	[redacted]

TABLA 11. Matriz de confusión para el usuario IX, utilizando el método AMONSTA. Ponderación con λ-=MIN

X	0	1	2	3	4	5	6	7	8	9	TOTAL
0											0
1								1			1
2											0
3											0
4	1										1
5											0
6											0
7											0
8											0
9		2		1							3
TOTAL	1	2	0	1	0	0	0	1	0	0	

TABLA 12. Matriz de confusión para el usuario X, utilizando el método AMONSTA. Ponderación con λ-=MIN

Finalmente se presenta un resumen de los mejores resultados tradicionales y adaptivos para facilitar la comparación con otros trabajos.

Dígitos	Resultados Tradicionales	Resultados Adaptivos
0	13/82 (15.85)	0/07 (0.00)
1	10/83 (12.05)	1/09 (11.11)
2	12/82 (14.63)	0/09 (0.00)
3	14/84 (16.66)	0/09 (0.00)
4	00/85 (0.00)	1/10 (0.10)
5	05/80 (6.25)	0/09 (0.00)
6	06/87 (6.19)	0/09 (0.00)
7	11/81 (12.08)	0/10 (0.00)
8	12/62 (19.35)	0/07 (0.00)
9	18/88 (20.45)	3/09 (33.33)
TOTAL	175/827 (21.16)	5/88 (5.68)

TABLA 13. Resumen de los resultados tradicionales y adaptivos.
Se presentan el total de errores/repeticiones (y porciento) por cada dígito.
Ejemplo 13/82 (15.85) significa 13 errores de 82 objetos para un porciento de error de 15.85%.

En la tabla 13 se observa que los resultados adaptivos son mejores que los tradicionales lo cual se atribuye al hecho de que el método AMONSTA está orientado a ser más eficiente para los resultados adaptivos, que fue el propósito original de este trabajo.

Conviene señalar que los criterios y métodos utilizados para estimar los parámetros (δ,α,β_v) de las redes tipo AMONSTA también resultan válidos si se utilizaran conjuntos de datos correspondientes a otros parlantes y aunque se debe continuar en la búsqueda de metodologías generales para la estimación de tales parámetros, los criterios y métodos utilizados parecen ser válidos para la aplicación de redes AMONSTA al problema de RPAIU.

Los datos empleados en este trabajo fueron también utilizados por Herrera [Herrera94] y Mondragón [Mondragon96]. Ambos trabajos evaluaron la eficiencia de describir las señales de voz mediante subpalabras acústicas, pero para diferentes métodos y bajo un esquema dependiente-del-usuario. En el primer caso se utilizó la regla del vecino más cercano junto con un algoritmo de alineamiento temporal (DTW) y se obtuvo 98.4 % de clasificaciones correctas. En el segundo caso se exploraron los métodos de agrupamiento k-medias y k-medias difuso, combinados con coeficientes LPC obtenidos a partir de las subpalabras acústicas y/o combinados con distancias euclidiana y de Itakura-Saito. Se obtuvo un 92.7% de clasificaciones correctas. Los resultados de las tablas 3-5 son significativamente peores que los de Herrera y ligeramente peores que los de Mondragón, lo cual se atribuye a que las redes aquí presentadas son de tipo adaptivas por lo que no explotan la ventaja de ver todos los datos a la vez. Sin embargo los resultados de Mondragón fueron a su vez ligeramente peores que los de la tabla 7 de este trabajo, que fueron obtenidos para un problema más difícil. Una pregunta interesante es cómo se comportaría la solución propuesta por Herrera para el problema de Reconocimiento de Palabras Aisladas Independiente del Usuario y del Vocabulario abordado en este trabajo y resuelto con eficiencia.

CONCLUSIONES Y COMENTARIOS FINALES

1- Se presentó un nuevo modelo de redes neuronales adaptivas (AMONSTA) el cual está dotado de un mecanismo de olvido que permite adaptar a la red a variaciones que ocurren en el medio ambiente de donde se toman las entradas. Estas redes cuentan además con mecanismos para fusionar prototipos que son muy útiles en la reducción del total de prototipos que se crean y evitan en gran medida la dependencia del orden en que se presentan los objetos a la red.

2- La solución propuesta al problema RPAIU consiste de un preprocesamiento inicial de la señal de voz que alimenta a un conjunto de redes neuronales AMONSTA, cada una especializada en determinado número de subpalabras acústicas. Esta arquitectura provee al sistema de una gran velocidad tanto de aprendizaje como de predicción pues por un lado el sistema cuenta con un método muy eficiente de procesamiento el cual reduce en alto grado el número de parámetros para describir la señal de voz y por otro lado se utiliza una sola red neuronal por cada palabra de entrada al sistema.

3- La red neuronal AMONSTA se comportó de manera eficiente para el problema planteado obteniendo un 94.32 % de clasificaciones correctas.

4- La red FuzzyARTMAP se comportó pésimamente para el problema planteado, no pudiendo sobrepasar el 73.81 % de clasificaciones correctas. Al parecer, el elemento clave que influyó en esto fueron las medidas de similaridad utilizadas por esta red, las cuales no pudieron discriminar correctamente.

5- Una red neuronal tipo AMONSTA que utilice descripciones de palabras aisladas en términos de subpalabras acústicas representadas por coeficientes KLT, es una alternativa válida para el problema de Reconocimiento de Palabras Aisladas Independiente del Usuario y del Vocabulario.

6- Futuros trabajos a realizar deberán:
a) explorar el comportamiento de AMONSTA para otras distancias y otras maneras de utilizar los valores propios.
b) hacer un análisis riguroso de la convergencia del método.
c) abordar heurísticas y/o algoritmos para implementar las variantes generales de fusión de prototipos propuestas y/o introducir nuevas variantes.

APÉNDICE I

Sea I un vector de entrada y W_1, W_2 dos categorías.

Suposición I: $\frac{|I \wedge W_1|}{|W_1|} = \frac{|I \wedge W_2|}{|W_2|} = C$ tal que $C = MAX\left\{\frac{|I \wedge W_j|}{|W_j|}, \forall 1 \le j \le N\right\}$

Suposición II: $|W_1| > |W_2|$

entonces se demostrará que $\frac{|I \wedge W_1|}{\alpha + |W_1|} > \frac{|I \wedge W_2|}{\alpha + |W_2|}$ y por lo tanto la categoría ganadora es la de mayor longitud.

De la suposición I sigue que

$$|I \wedge W_1||W_2| = |I \wedge W_2||W_1| \qquad \text{(I-1)}$$

y

$$|I \wedge W_1| = C|W_1|, \quad |I \wedge W_2| = C|W_2| \qquad \text{(I-2)}$$

De la suposición II, (I-2) y del hecho que $\alpha > 0$ y $C > 0$ se tiene:

$$\frac{|I \wedge W_1|}{C} = \frac{|I \wedge W_2|}{C} \quad y \quad \alpha + |I \wedge W_1| > \alpha + |I \wedge W_2|$$

Sumando en ambos miembros: $|W_1||I \wedge W_2|$

se tiene

$$\alpha|I \wedge W_1| + |W_1||I \wedge W_2| > \alpha|I \wedge W_2| + |W_1||I \wedge W_2| \qquad \text{(I-3)}$$

Por otro lado de (I-1) se tiene

$$\alpha|I \wedge W_1| + |W_2||I \wedge W_1| > \alpha|I \wedge W_2| + |W_1||I \wedge W_2|$$

equivalente a

$$(\alpha + |W_2|)|I \wedge W_1| > (\alpha + |W_1|)|I \wedge W_2|$$

y se tiene

$$\frac{|I \wedge W_1|}{\alpha + |W_1|} > \frac{|I \wedge W_2|}{\alpha + |W_2|}$$

APÉNDICE II

Sea J_1 índice de la actual categoría ganadora de ARTa, la cual cumple:

$$|X^{ab}| = |Y^{b} \wedge W_{J1}^{ab}| < \rho_{ab}|Y^{b}|$$

Demostrar que entonces ARTa ya no tiene categorías no acometidas o existe J_2 tal que:

$$|X^{ab}| = |Y^{b} \wedge W_{J2}^{ab}| \geq \rho_{ab}|Y^{b}| \qquad \text{(II-1)}$$

Supongamos que J_2 es el índice de una categoría no acometida de ARTa entonces:

$$|X^{ab}| = |Y^{b} \wedge W_{J2}^{ab}| = |Y^{b}|$$

demostrándose así lo deseado.

Debe notarse que si no existe alguna categoría no acometida de ARTa, no significa que no exista de todas maneras otra categoría de índice J_2 que satisfaga (II-1). Además que incluso existiendo categorías no acometidas en ARTa puede encontrarse alguna categoría acometida que satisfaga (II-1). En tal sentido esta breve demostración asegura que siempre se encuentra una categoría que satisfaga (II-1) o no hay categorías no acometidas de ARTa.

APÉNDICE III

Sea $V_1, V_2, \ldots, V_N$ los coeficientes de vitalidad de N neuronas, k: número de iteraciones y $\alpha>1$ tal que $K \geq N \geq 1$, demostrar que entonces se cumple:

$$\min_{1 \leq j \leq N} V_j \geq \frac{1}{\alpha^{k-2}}$$

EL caso N=1 es trivial pues como sólo hay una neurona entonces siempre su coeficiente de vitalidad se incrementa en proporción α y entonces al cabo de k iteraciones $_1 = \alpha^k$. Analicemos entonces el caso de k>1.

Al inicio del entrenamiento la red no tiene neuronas. Al presentar el primer objeto se crea una neurona y su coeficiente de vitalidad $V_1=\alpha$. Al presentar un segundo puede ocurrir que la neurona actual lo clasifique correctamente en cuyo caso $V_1=\alpha^2$ o que se cree una nueva neurona, teniéndose entonces $V_1=1$ y $V_2=\alpha$. En general cada vez que se presenta un objeto puede ocurrir que:
a) éste sea clasificado por alguna de las neuronas existentes, en cuyo caso su coeficiente de vitalidad se incrementa en múltiplo de α.
b) se cree una nueva neurona con coeficiente de vitalidad α y se decrementa el coeficiente de vitalidad de todas las demás neuronas en un múltiplo de $1/\alpha$

De b) se obtiene que el menor coeficiente de vitalidad es de la forma $1/\alpha^p$, $p \in N^*$, el cual se hace mínimo cuando p es máximo. En k iteraciones el valor máximo que puede tomar p es k-2 que es cuando la neurona fue la primera en se creada y nunca resultó ganadora. Nótese que se le resta 2 a k pues en la primera iteración $V_1=\alpha$ y en la segunda $V_1=1$. Queda entonces demostrado lo que se quería.

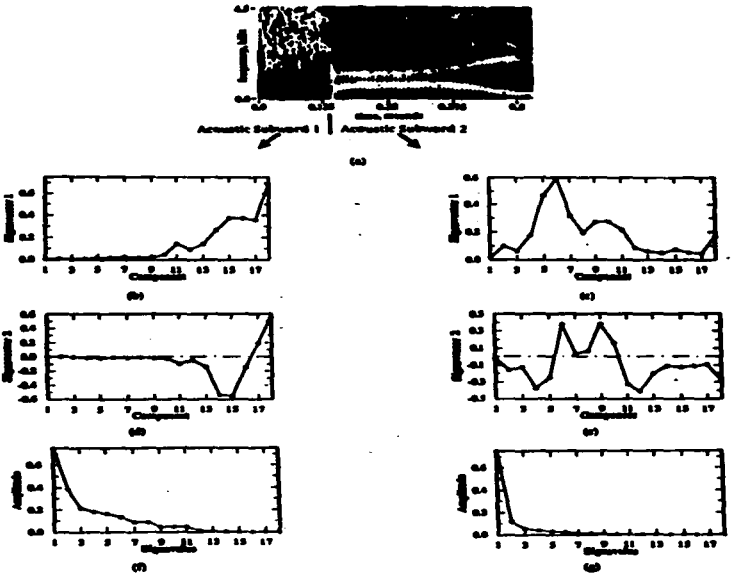
APÉNDICE IV

En la figura siguiente se muestra la segmentación de la señal correspondiente a la palabra siete (seven en inglés). Debajo de la figura aparece la segmentación en fonemas y en subpalabras acústicas en la que se observa que una subpalabra acústica puede corresponder a una o más fonemas.

Phoneme	/s/	/E/	/v/	/A/	/n/
Acoustic Subword	#1	#2			#3

APÉNDICE V

A continuación se muestra un ejemplo de la aplicación del método KLT a la palabra estrella (*star* en inglés), la cual fue descompuesta en dos subpalabras acústicas. De cada subpalabra se muestra gráficamente los dos primeros vectores propios y los 18 valores propios.



REFERENCIAS

Ahmed — Ahmed N, Rao R, "Orthogonal transforms for digital signal processing", cap. 8.

Bezdek92 — Bezdek J, Pal S, "Fuzzy models for pattern recognition, methods that search for structures in data", IEEE Press (1992).

Carpenter87a — Carpenter GA, Grossberg S: "A massively parallel architecture for self-organizing neural pattern recognition machine", *Computer Vision, Graphics, and Image Processing*, 37 (1987) 34-115.

Carpenter87b — Carpenter GA, Grossberg S: ART2: Stable self-organization of pattern recognition codes for analog input patterns, *Applied Optics*, 26 (1987) 4919-30.

Carpenter91 — Carpenter G, Grossberg S, Rosen D, "FuzzyART: fast stable learning and categorization of analog patterns by an adaptive resonance system", Neural Network, v.4 (1991) 759-771.

Carpenter92a — Carpenter G, Grossberg S, Markuzon N, Reynolds J, Rosen D, "FuzzyARTMAP: A neural network architecture for incremental supervised learning of analog multidimensional maps", IEEE Trans. Neural Networks, v.3 (1992) 698-713.

Carpenter92b — Carpenter G,. Grossberg S, "A self organizing neural network for supervised learning, recognition, and prediction", IEEE Communications Magazine 9 (1992) 38-49.

Carpenter92c — Carpenter G, Grossberg S, Iizuka K, "Comparative performance measures of FuzzyARTMAP, learned vector quantization and back propagation for handwritten character recognition", Proc. IJCNN-92, v.1 (1992) 794-799.

Carpenter95 — Carpenter G, Grossberg S, Reynolds J, "A FuzzyARTMAP nonparametric probability estimator for nonstationary pattern recognition problems", IEEE Trans. on neural networks, v.6, n.6 (1995) 1330-1336.

Casacubierta87 Casacubierta F, Vidal E: "Reconocimiento Automático del Habla", *Ed. Marcombo*, ISBN:84-267-0656-8, (1987).

Duda73 Duda RO, Hart PE: Pattern Classification and Scene Analysis, *John Wiley & Sons*, New York, 1973.

Funahashi89 Funahashi K, "On the approximate realization of continuous mappings by neural network", Neural Networks, v.2 (1989) 183-192.

Gersho91 Gersho A., Gray R., "Vector quantization and signal compression", Kluwer academic publishers, (1991).

Herrera94 Herrera A, Algazi VR, Irvine D, "An acoustic approach for isolated word recognition", Proc. of the International Conference on Signal Processing Applications and Technology ICSPAT'94, v.2 (1994) 1677-1681.

Hornik93 Hornik K, "Some new results on neural network approximation", Neural Networks, v.6 (1993) 1069-1072.

Ito92 Ito Y, "Approximation of continuous functions on R^d by linear combinations of shifted rotations of a sigmoid function with and without scaling", Neural Networks, 3 (1992) 551-560.

Kato93 Kato Y. Sugiyama M.,"Speaker-independent features extracted by a neural network", ICASSP-93, V.I (1993) 553-556.

Kohonen89 Kohonen T: Self-Organization and associate memory, Springer-Verlag, (1989).

Konig93 Konig Y., Morgan N., "Supervised and unsupervised clustering of the speaker space for connectionist speech recognition", ICASSP-93, V.I (1993) 545-548.

Linde80 Linde Y, Buzo A, Gray R, "An algorithm for vector quantizer design", IEEE Trans. on Communications, Vol. COM-28, (1980) 84-95.

MacQueen67 MacQueen J: Some methods for classification and analysis of multivariate observations. En *Proc. of the Fifth Berkeley Symposium on Math. Stat and Prob*, 1 (1967) 281-296.

Makhoul75 Makhoul J, "Linear prediction: a tutorial review", Proc. of IEEE, v.63, n.4 (1975) 561-580

Mazin93 Mazin G.R., "A self-learning neural network for recognition of speech features", ICASSP-93, V.I (1993) 517-520.

Mellouk93 Mellouk A., Gallinari P.,"A discriminative neural prediction system for speech recognition", ICASSP-93, V.I (1993) 533-536.

Mondragon96 Mondragon A, "Técnicas de reconocimiento automático de voz utilizando segmentación acústica", Tesis de Maestría, Facultad de Ingeniería, Universidad Nacional Autónoma de México, Junio (1996).

Nissani91 Nissani DN: "An unsupervised hyperspheric multi-layer feedforward neural network model", *Biol. Cybern.*, 65 (1991) 441-450.

Pao89 Pao YH, "Adaptive pattern recognition and neural networks", Addison-Wesley, (1989).

Rabiner81 Rabiner L, Levinson S, "Isolated and connected word recognition - Theory and selected applications", IEEE Trans on Communications, v. COM-29, n.5 (1981) 621-659.

Rabiner89 Rabiner L, "A tutorial on hidden markov models and selected applications in speech recognition", Proc. of the IEEE, v.77, n.2 (1989) 257-286.

Rabiner84 Rabiner L, Levinson SE, Sondhi MM, "On the use of hidden markov model for speaker-independent recognition of isolated-words from medium-size vocabulary", AT&T Tech., j., vol.63 n.4 (1984) 627-642.

Ripley94 Ripley B, "Neural networks and related methods for classification", J.R. Statist. Soc. B., v.56, n.3 (1994) 409-456.

Riquenes94 Riquenes A, Sánchez R, "Introducción a la computación neuronal", Ingeniería Industrial, v.XV, n.1 (1994) 21-40.

Sánchez95a Sánchez R, Riquenes A, Pérez MC, "Automatic Detection of Auditory Brainstem Responses using Feature Vectors", *Int. J. of Biomed Computing*, 39 n.3 (1995) 287-297.

Sánchez95b — Sánchez R, González P, Riquenes R, Chávez E, "A neural network to learn patterns from non-stationary environments", The First Joint Mexico-US International Workshop on Neural Networks and Neurocontrol, (1995) 310-325.

Sánchez96 — Sánchez R, Chávez E, "A learning neural network algorithm that learns time-varying classes", Proc. of IX Symposium on the Artificial Intelligence, (1996) 228-234.

Schalkoff92 — Schalkoff R, "Pattern recognition, statistical, structural and neural approaches", John Wiley & Sons, Inc. (1992).

Tobias70 — Tobias J.V., "Foundations of modern auditory theory", Academic Press (1970).

Zwicker90 — Zwicker E, Fastk H, "Psicoacoustics", Springer-Verlag (1990).