Árboles de redes neuronales autoorganizativas

Diego H. Milone, José C. Sáez, Gonzalo Simón, Hugo L. Rufiner.

Laboratorio de Cibernética, Departamento de Bioingeniería, Facultad de Ingeniería, Universidad Nacional de Entre Ríos, Ruta 11 Km. 10, Oro Verde (CP: 3101), Entre Ríos, Argentina. E-Mail: cyberlab@fi.uner.edu.ar

Resumen

Un campo muy importante de la inteligencia artificial es el de la clasificación automática de patrones. Para este tipo de tareas se han utilizado diferentes técnicas. En este trabajo se presenta una combinación de árboles de decisión y redes neuronales autoorganizativas como una alternativa para atacar el problema. Para la construcción de estos árboles se recurre a procesos de crecimiento. En estos procesos es necesario evaluar el rendimiento en la clasificación de uno o varios nodos ante distintas configuraciones para tomar decisiones que optimicen la estructura y el desempeño del árbol de redes neuronales autoorganizativas. Para ello se define un grupo de coeficientes que cuantifican el rendimiento y se desarrolla un algoritmo de crecimiento basado en estos coeficientes. En las pruebas se muestra como los coeficientes miden de manera objetiva el rendimiento en la clasificación para estructuras simples. Finalmente se realiza una comparación con otros métodos de clasificación utilizando métodos de validación cruzada y bases de datos reales y artificiales.

Abstract

Automatic pattern classification is a very important field of artificial intelligence. For this type of tasks several techniques have been used. In this work a combination of decision trees and self-organizing neural networks is presented as an alternative to attack the problem. For the construction of these trees growth processes were applied. In these processes, evaluation of the classification efficiency of one or several nodes in different configurations is necessary in order to take decisions to optimize the structure and performance of the self-organizing neural tree net. For this task a group of coefficients that quantify the efficiency is defined and a growth algorithm based on these coefficients is developed. The tests show that the coefficients give an objective measure of the classification performance for simple structures. Finally, a comparison with other classification methods, using cross-validation methods and real and artificial databases, is carried out.

Palabras clave

Árboles de decisión. Redes neuronales autoorganizativas. Clasificación automática. Comparación de clasificadores. Reconocimiento de fonemas.

Keywords

Decision Trees. Self-Organizing Neural Networks. Automatic Classification. Clasificators comparison. Phoneme recognition.

1 Introducción y antecedentes

Actualmente, la clasificación de patrones es una importante herramienta utilizada en el análisis de datos experimentales, reconocimiento automático, visión por computadora y otras disciplinas ingenieriles y científicas. Los algoritmos de aprendizaje maquinal forman parte de lo que se denomina clasificación automática y están encargados de extraer la información relevante de un conjunto de patrones que permita su clasificación. Estos algoritmos pueden dividirse en dos categorías: supervisados y no supervisados. Los algoritmos supervisados requieren un conocimiento previo acerca de las clases a las que pertenecen los patrones, mientras que los no supervisados se basan en agrupar a los patrones de acuerdo a sus similitudes, por lo que también se denominan métodos de agrupamiento. Desde otro punto de vista, los algoritmos de clasificación pueden reunirse en dos categorías: simples y jerárquicos [1]. La clasificación jerarquizada se lleva a cabo, al contrario que la simple, en cierta cantidad de etapas o subclasificaciones sucesivas. El caso jerárquico normalmente implica una reducción en la carga computacional [2]. Las particiones que se generan en un proceso de clasificación pueden ser duras o difusas [3]. Por ejemplo, dentro de los algoritmos de clasificación simple, el algoritmo *k-means* [4] genera particiones duras y el *fuzzy c-means* [5] genera particiones difusas. Las clasificaciones jerarquizadas serán vistas con mayor detalle en la siguiente sección.

Los *árboles de decisión* (AD) y las *redes neuronales artificiales* (RNA) son dos técnicas ampliamente utilizadas para la implementación de clasificadores. Los AD generan un conjunto de particiones de los datos basándose en una estructura jerárquica de nodos en los que se realizan comparaciones sobre alguna componente del vector de características. Las redes neuronales están formadas por un conjunto de unidades de procesamiento no lineal altamente interconectadas, que procesan en paralelo un conjunto de datos para

extraer información. Existen diferentes modelos neuronales para la implementación de clasificadores supervisados y no supervisados. El *perceptrón multicapa* (PMC) es un ejemplo clásico de clasificador simple supervisado [6] [4], mientras que los *mapas autoorganizativos* (MAO) [6] son ejemplos de clasificadores simples no supervisados.

Este artículo está organizado de la siguiente forma. A continuación se presenta el paradigma de AD y los conceptos básicos de los MAO, para continuar con las alternativas híbridas como la que se presenta en este trabajo. Después de esta introducción se encuentra una sección dedicada a un conjunto de coeficientes necesarios en el proceso de crecimiento y entrenamiento de los *árboles de redes neuronales autoorganizativas* (ARNA) propuestos en este trabajo. En esta sección se presentan sus definiciones y propiedades, y algunos ejemplos que aportan a la comprensión de su funcionamiento. En la sección de métodos se encuentra todo lo relacionado con el algoritmo de crecimiento y entrenamiento de los ARNA conjuntamente con la información acerca de los métodos de validación y las bases de datos utilizadas en las pruebas. Finalmente, en las últimas secciones se muestran y discuten los resultados y se presentan las conclusiones del trabajo.

1.1 Inducción de reglas mediante árboles de decisión

En este paradigma el algoritmo de aprendizaje busca una colección de reglas que clasifican "mejor" los ejemplos de entrenamiento. Dichas reglas pueden ser representadas como un AD. Éste puede pensarse como un diagrama de flujo en donde cada nodo representa una prueba y cada rama que sale del nodo representa un resultado posible de dicha prueba. Para una revisión más detallada ver [7], [8].

Existen AD binarios y *n*-arios, de acuerdo a la cantidad de particiones realizadas en cada nodo. Dependiendo de las características de la función del nodo y del tamaño del árbol, la frontera final de decisión puede ser muy compleja. Una de las funciones más empleadas es la prueba mediante un cierto umbral para cada atributo, teniendo como resultado la partición del espacio de atributos por medio de hiperplanos paralelos u ortogonales a los ejes coordenados del espacio de atributos.

Dos de los algoritmos más utilizados son ID3 y CART y sus derivados [9]. El algoritmo ID3 genera AD *n*-arios debido a que particiona el conjunto de ejemplos de entrenamiento en función del mejor atributo. La función heurística que utiliza ID3 para determinar el mejor atributo es una medida de la entropía para cada atributo. CART genera AD binarios ya que para particionar el conjunto de ejemplos en un nodo elige el mejor par atributo-valor de acuerdo con un criterio denominado *criterio de Gini* [10].

Aunque los AD son intuitivamente atractivos y han tenido aplicaciones exitosas [9], [8] existen algunos problemas que pueden obstaculizar su empleo en casos reales. Entre estos problemas se encuentran la presencia de datos inconclusos, incompletos o ruidosos y el empleo simultáneo de todos los vectores de atributos. Para solucionar algunos de estos problemas se debe pensar en algún criterio de terminación adecuado, es decir, hasta que punto se debe hacer crecer el árbol de tal forma que no se obtenga un árbol demasiado complejo, difícil de entender y poco eficiente. Existe otro problema inherente a los clasificadores jerárquicos que tiene que ver con la cantidad de datos de entrenamiento disponibles en cada nivel por lo que constituye otra razón para elegir un criterio de terminación adecuado.

1.2 Mapas autoorganizativos

Diversas áreas del cerebro, especialmente de la corteza cerebral, se hallan organizadas según diferentes modalidades sensoriales. Esta organización de la actividad cortical del cerebro puede describirse mediante mapas ordenados. Por ejemplo se encuentran los mapas retinoscópicos de la corteza visual, los mapas tonotópicos de la corteza auditiva [11], los mapas somatotópicos de la corteza somatosensorial y los mapas de retardo interaural [2], [12].

Inspirado en el mapeo ordenado del cerebro, Kohonen introdujo en 1982 [13] un algoritmo de autoorganización que produce mapas ordenados que simulan cortezas biológicas simplificadas con el objeto de resolver problemas prácticos de clasificación y reconocimiento de patrones. Estos mapas fueron denominados mapas autoorganizativos. Los MAO presentan la propiedad de preservación de la vecindad, que los distingue de otros paradigmas de redes neuronales.

Los MAO son redes neuronales entrenadas mediante aprendizaje competitivo. En este tipo de aprendizaje, las neuronas de salida compiten entre ellas para ser activadas, dando como resultado la activación de una sola a la vez. Esta neurona es llamada "neurona ganadora". A diferencia de otras redes neuronales donde sólo se permite que aprenda la unidad ganadora, en los MAO todas las unidades vecinas a la ganadora reciben una realimentación procedente de la misma, participando de esta manera en el proceso de aprendizaje. Esta importante característica es también denominada realimentación lateral y puede ser excitatoria, inhibitoria o una combinación de ambas [14].

En la figura 1 se puede ver la configuración básica de un MAO. Se observan las neuronas de entrada e_i y una red bidimensional de neuronas de salida s_j. Un peso sináptico $w_{i,j}$ conecta a la neurona e_i con la s_j. A cada neurona de entrada e_i se le presenta el *i*-ésimo elemento de cada patrón de entrada $\boldsymbol{x}(n) \in \Re^D$, siendo n la ocurrencia temporal de este patrón. El arreglo bidimensional de neuronas de salida incluye conexiones entre las neuronas vecinas simulando la realimentación lateral.

Si G es una neurona ganadora durante el entrenamiento de un MAO, las neuronas vecinas que también serán actualizadas quedan en una región determinada por una función de vecindad

sinc(*i*) Research Center for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
D. H. Milone, J. C. Sáez, G. Simón & H. L. Rufiner; "Arboles de redes neuronales autoorganizativas"
Revista Mexicana de Ingeniería Biomédica, Vol. 19, No. 4, pp. 13-26, Dic, 1998.

$\Lambda_G(n)$. Esta región puede tener diferentes formas y es variable con el tiempo. El área cubierta comienza siendo máxima y se reduce a medida que avanza el entrenamiento hasta no incluir ninguna neurona vecina a la ganadora. En la figura 2 se puede observar una región de vecindad cuadrada que disminuye su área en función del tiempo. Para un análisis más completo de esta función puede consultarse [14] y [2].

En la figura 3 se puede ver el algoritmo de entrenamiento de un MAO. Tanto la velocidad de aprendizaje $\eta(n)$ como $\Lambda_G(n)$, deben variar dinámicamente durante el entrenamiento, aunque no existe una base teórica para la selección de estos parámetros. Durante el aprendizaje se consideran dos etapas: la etapa de ordenamiento y la de convergencia. Las variaciones de $\eta(n)$ y de $\Lambda_G(n)$ están directamente relacionadas con estas etapas [2], [14]. Una vez que se ha entrenado un MAO, los vectores de pesos $\boldsymbol{w}_j$, que van desde la salida s_j a todas las entradas, determinan los denominados centroides de cada clase.

Los MAO han sido utilizados con éxito en el reconocimiento de fonemas en discurso continuo (finlandés y japonés) [2], control de brazos robotizados [15] [16], diseño de circuitos integrados [17] y muchas otras aplicaciones. En este trabajo se utilizan como parte del algoritmo de clasificación propuesto.

1.3 Soluciones híbridas

Para solucionar algunos de los problemas que se presentan con los AD, una alternativa consiste en una implementación híbrida de AD y RNA, como la presentada por Sankar y Mammone en 1991 [18] denominada *árboles de redes neuronales* (ARN). Este tipo de enfoque permite aprovechar las ventajas de la clasificación jerárquica y crear fronteras de decisión más complejas con menos nodos, minimizando los problemas de ruido y de estructuras intrincadas. Los ARN son AD que implementan la tarea de decisión en los nodos mediante una red neuronal. De esta manera la decisión que se toma en cada nodo se basa en reglas más complejas, lo que permite aproximar mejor las fronteras a costa de perder claridad en la interpretación de las reglas (ya no son reglas lógicas sencillas). Se han utilizado diversas arquitecturas en los nodos, entre ellas se pueden citar: perceptrones simples [18], PMC [19][20] y MAOs [21]. La cantidad de particiones que se producen en cada nodo puede ser fija [18],[19] o variable [20]. Cuando la cantidad de clases generadas puede variar para cada nodo, el ARN tiene la posibilidad de adoptar una configuración más adecuada para el problema a resolver. Otras alternativas que combinan estructuras jerárquicas con redes neuronales son las redes neuronales modulares [22] y los MAO con neuronas en estructura jerárquica [2].

Los ARN realizan una clasificación basada en una combinación de los métodos de clasificación simples y jerárquicos. En este trabajo, el ARNA propuesto consiste en un árbol de MAOs. Estas redes no supervisadas separan a los patrones de acuerdo a la distribución natural de los mismos y permiten aprovechar las características jerárquicas de los árboles. El algoritmo permite que en las primeras capas o nodos se separen los grupos de patrones más alejados entre sí (o más fácilmente separables) y en las capas finales se separen los patrones de manera más fina (es decir, los más dificilmente separables). Esto puede funcionar muy bien si los patrones poseen cierta jerarquización natural en clases y subclases, como en el caso de las características espectrales de los fonemas. Un problema importante es como decidir acerca de la cantidad de particiones a realizar en cada nodo en el caso de árboles n-arios. Para atacar este problema se establecieron criterios basados en los coeficientes de clasificación que se describen en la sección siguiente.

2 Coeficientes de clasificación

Dado un clasificador general se define $X = \{\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_P\}$ donde $\boldsymbol{x}_i \in \Re^D$, como el conjunto de patrones de entrada. Estos patrones de entrada corresponden a *M* clases que se denominan clases de entrada C_i^I. Para el conjunto de clases de entrada $C^I = \{C_1^I, C_2^I, ..., C_M^I\}$ se cumplen las siguientes hipótesis:

hi1) $X = C_1^I \cup C_2^I \cup \cdots \cup C_M^I$

hi2) $C_i^I \cap C_j^I = \varnothing \quad \forall\, i \neq j$

hi3) $C_i^I \neq \varnothing \quad \forall\, i$

De la misma forma en que los patrones de entrada se agrupan según las clases a las que pertenecen realmente, también se pueden agrupar de acuerdo a las clases C_j^O en que son separados por el clasificador. Éstas últimas forman el conjunto de clases de salida $C^O = \{C_1^O, C_2^O, ..., C_N^O\}$ y cumplen con las siguientes hipótesis:

ho1) $X = C_1^O \cup C_2^O \cup \cdots \cup C_N^O$

ho2) $C_i^O \cap C_j^O = \varnothing \quad \forall\, i \neq j$

(no se requiere $C_j^O \neq \varnothing \quad \forall\, j$).

Se observa que, en el caso más general, la cantidad de clases de entrada *M* no necesariamente debe ser igual a la cantidad de clases de salida *N*. Esta generalización resulta muy útil cuando el proceso de clasificación se realiza mediante clasificaciones sucesivas. En estas etapas intermedias en general $M \geq N$. No obstante, en el clasificador visto como un solo conjunto generalmente se tiene $M \leq N$.

Un elemento importante para el desarrollo posterior es la matriz de intersección de entrada - salida definida como:

$$N_{i,j}^{IO} = \mathrm{n}\left(C_i^I \cap C_j^O\right) \text{ con } 1 \leq i \leq M \text{ y } 1 \leq j \leq N \qquad (1)$$

donde n es el operador de cardinalidad. Esta matriz contiene en su *i,j*-ésima celda la cantidad de patrones de la clase de entrada C_i^I clasificados como pertenecientes a la clase de salida C_j^O.

sinc(*i*) Research Center for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
D. H. Milone, J. C. Sáez, G. Simón & H. L. Rufiner; "Arboles de redes neuronales autoorganizativas"
Revista Mexicana de Ingeniería Biomédica, Vol. 19, No. 4, pp. 13-26, Dic, 1998.

A continuación se analizan las limitaciones en la utilización del coeficiente de reconocimiento clásico como criterio para el desarrollo de la topología de un ARNA.

2.1 Coeficiente de reconocimiento clásico

El coeficiente de reconocimiento que se utiliza generalmente para medir el desempeño de un clasificador en el reconocimiento de patrones se puede definir según:

$$cr = \frac{\sum_{i=1}^{M} \max_{j=1}^{N}\left(N_{i,j}^{IO}\right)}{\mathrm{n}(X)} \qquad (2)$$

siempre que se cumplan las hipótesis:

hcr1) $M = N$

sea $j_i = \arg\left[\max_{j=1}^{N}\left(N_{i,j}^{IO}\right)\right]$

hcr2) $j_{i_1} \neq j_{i_2} \quad \forall i_1 \neq i_2$ para $1 \leq i \leq M$

Este coeficiente tiene algunas propiedades que suelen hacer confusa su interpretación. Sin embargo todavía continua utilizándose en muchos trabajos de clasificación para reportar resultados. Si se cumplen las hipótesis hi, ho y hcr, el máximo que puede alcanzar *cr* es 1 cuando $\forall i \; \exists \, j / N_{i,j}^{IO} = \mathrm{n}\left(C_i^I\right)$. Vale aclarar que para cada *i* existe un único *j* por la restricción impuesta en la hcr2.

El mínimo que puede alcanzar *cr* no es cero ya que éste depende del número de clases de salida *M*. Cuando el clasificador se encuentra en un máximo de confusión, distribuye igualmente cada clase de entrada en las clases de salida. Por lo tanto el mínimo para *cr* es 1/*M*, ya que el máximo en cualquier clase de salida es $\mathrm{n}\left(C_i^I\right) / M$. Esto es particularmente confuso ya que un clasificador con dos clases de salida no podría tener nunca un desempeño menor al 50% ($cr \geq 0.5$).

Por otro lado, este coeficiente de reconocimiento no es aplicable cuando $M \neq N$. Además, la hipótesis hcr2 restringe su aplicabilidad cuando se hacen agrupaciones intermedias de varias clases de entrada en una clase de salida para ser luego separadas por otro clasificador. Cuando se relaja hcr2 el coeficiente no permite discernir en que medida patrones de las misma clase de entrada son concentrados en la misma clase de salida y patrones de distintas clases de entrada son distribuidos en distintas clases de salida. Para poder eliminar hcr1 y hcr2, se definen dos coeficientes que miden estas concentraciones y dispersiones por separado.

2.2 Coeficiente de concentración interclase

Para medir en que grado un clasificador agrupa patrones pertenecientes a una clase de entrada en una misma clase de salida se define, en primer lugar, el coeficiente de concentración interclase para la clase de entrada C_i^I en las *N* clases de salida C_j^O como:

$$cc_i = \frac{N \max_{j=1}^{N}\left(N_{i,j}^{IO}\right) - \sum_{j=1}^{N} N_{i,j}^{IO}}{(N-1)\sum_{j=1}^{N} N_{i,j}^{IO}} \qquad (3)$$

Se observa que $\sum_j N_{i,j}^{IO} = \mathrm{n}\left(C_i^I\right)$ y además por la hi3 se cumple $\sum_j N_{i,j}^{IO} \neq 0 \quad \forall i$.

Este coeficiente posee las siguientes propiedades:

pcci1) $cc_i = 1$ si $\exists \, j^* / N_{i,j^*}^{IO} = n\left(C_i^I\right)$

(si existe j^* entonces es único por ho2)

pcci2) $cc_i = 0$ si $N_{i,j_1}^{IO} = N_{i,j_2}^{IO} \quad \forall 1 \leq j_1, j_2 \leq N$

pcci3) cc_i está siempre en el intervalo [0,1].

pcci4) cc_i es monótono decreciente con $\max_j\left(N_{i,j}^{IO}\right)$.

Se define el coeficiente de concentración interclase para un clasificador como el promedio de los cc_i ponderados por la cantidad de patrones de la clase de entrada correspondiente:

$$cc = \frac{\sum_{i=1}^{M} n\left(C_i^I\right) \cdot cc_i}{\sum_{i=1}^{M} n\left(C_i^I\right)} \qquad (4)$$

sustituyendo según la ecuación (3) y simplificando se obtiene:

$$cc = \frac{\sum_{i=1}^{M} N \max_{j=1}^{N}\left(N_{i,j}^{IO}\right) - \sum_{i,j=1}^{i=M,\,j=N} N_{i,j}^{IO}}{(N-1)\sum_{i,j=1}^{i=M,\,j=N} N_{i,j}^{IO}} \qquad (5)$$

donde se observa que $\sum_{i,j=1}^{i=M,\,j=N} N_{i,j}^{IO} = \mathrm{n}(X)$ y $\mathrm{n}(X) \neq 0$ por hi3.

El coeficiente de concentración para un clasificador posee propiedades que hereda directamente del coeficiente de concentración para cada clase de entrada.

Asumiendo las hipótesis hcr1 y hcr2, y observando las ecuaciones (2) y (5) se deduce la siguiente relación:

$$cr = \frac{(M-1)}{M} cc + \frac{1}{M} \qquad (6)$$

Así, se puede ver que cuando los máximos de cada clase de entrada se encuentran en distintas clases de salida y la cantidad de clases de salida es igual a la de clases de entrada, el coeficiente de reconocimiento puede expresarse como una versión escalada y desplazada del coeficiente de concentración intraclase.

Hay que destacar que, si bien el coeficiente de concentración mide la capacidad con que un clasificador agrupa patrones de una misma clase de entrada en una única clase de salida, no es capaz de detectar cuando todos los patrones de entrada son llevados a una misma clase de salida. Para esto se

sinc(*i*) Research Center for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
D. H. Milone, J. C. Sáez, G. Simón & H. L. Rufiner; "Arboles de redes neuronales autoorganizativas"
Revista Mexicana de Ingeniería Biomédica, Vol. 19, No. 4, pp. 13-26, Dic, 1998.

define a continuación el coeficiente de dispersión intraclase.

2.3 Coeficiente de dispersión intraclase

Para medir la capacidad que posee un clasificador para llevar patrones de distintas clases de entrada a distintas clases de salida se define, en primer lugar, el coeficiente de dispersión intraclase para la clase de salida C_j^O en las M clases de entrada C_i^I como:

$$cd_j = \begin{cases} \dfrac{M \max_{i=1}^{M}\left(N_{i,j}^{IO}\right) - \sum_{i=1}^{M} N_{i,j}^{IO}}{(M-1)\sum_{i=1}^{M} N_{i,j}^{IO}} & \text{si} \quad \mathrm{n}\left(C_j^O\right) \neq 0 \\ 0 & \text{si} \quad \mathrm{n}\left(C_j^O\right) = 0 \end{cases} \qquad (7)$$

Se observa que $\sum_i N_{i,j}^{IO} = \mathrm{n}\left(C_j^O\right)$.

Este coeficiente posee las siguientes propiedades:

pcdj1) $cd_j = 1$ si $\exists i^* / N_{i^*,j}^{IO} = 0 \ \forall i \neq i^*$
(si existe i^* entonces es único por hi2)

pcdj2) $cd_j = 0$ si $N_{i_1,j}^{IO} = N_{i_2,j}^{IO} \ \forall 1 \leq i_1, i_2 \leq M$

pcdj3) cd_j está siempre en el intervalo [0,1].

pcdj4) cd_j es monótono creciente con $\max_i\left(N_{i,j}^{IO}\right)$.

De forma similar que para el coeficiente de concentración, se define el coeficiente de dispersión intraclase para un clasificador como el promedio de los cd_j ponderados por la cantidad de patrones en cada clase de salida:

$$cd = \frac{\sum_{j=1}^{N} n\left(C_j^O\right) \cdot cd_j}{\sum_{j=1}^{N} n\left(C_j^O\right)} \qquad (8)$$

Sustituyendo según (7) y simplificando se obtiene:

$$cd = \frac{\sum_{j=1}^{N} M \max_{i=1}^{M}\left(N_{i,j}^{IO}\right) - \sum_{j,i=1}^{j=N,i=M} N_{i,j}^{IO}}{(M-1)\sum_{j,i=1}^{j=N,i=M} N_{i,j}^{IO}} \qquad (9)$$

donde nuevamente se tiene $\sum_{j,i=1}^{j=N,i=M} N_{i,j}^{IO} = \mathrm{n}(X)$.

Las propiedades de *cd* se heredan directamente de las propiedades del coeficiente de dispersión para cada clase de salida.

El coeficiente de dispersión intraclase no mide el grado en que patrones de entrada de una misma clase son derivados a diferentes clases de salida. Esto es cuantificado por el coeficiente de concentración interclase.

Finalmente se muestran algunos ejemplos de la aplicación de los coeficientes de clasificación. Se puede observar en la figura 4.a) la matriz N^{IO} correspondiente a un clasificador ideal. En este caso los tres coeficientes de clasificación llegan a su valor máximo indicando una clasificación perfecta. La figura 4.b) muestra el comportamiento de un clasificador totalmente confundido. El ejemplo muestra como *cc* y *cd* marcan más fehacientemente que *cr* la deficiencia del clasificador. En la figura 4.c), la matriz N^{IO} corresponde a un clasificador que clasificó a todos los patrones de entrada como pertenecientes a una misma clase de salida. Este es un ejemplo típico de máxima concentración y mínima dispersión, como los coeficientes lo indican. En la matriz de la figura 4.d) se encuentra el ejemplo opuesto donde la dispersión es máxima.

3 Materiales y Métodos

En esta sección se describe el nuevo algoritmo de clasificación propuesto y la forma de entrenarlo tomando como base los coeficientes descriptos en la sección anterior. También se describen las bases de datos utilizadas en los experimentos, el método de validación con el cual se obtuvieron los resultados y los detalles de implementación computacional.

3.1 Algoritmo de entrenamiento del ARNA

La totalidad de los patrones de entrenamiento es presentada inicialmente al nodo raíz. A los nodos de los niveles siguientes les llega un subconjunto de patrones que ha sido derivado jerárquicamente desde los niveles anteriores del árbol.

Considerando un nodo en particular se debe decidir, en primer lugar, si se justifica o no realizar una tarea de clasificación. Así se distingue entre dos tipos de nodos: nodos clasificadores y nodos terminales. Para declarar que un nodo es terminal o clasificador se deben tener en cuenta dos características de su conjunto de patrones de entrada: el grado de homogeneidad en clases y el número de patrones que posee. Si bien esta última característica no presenta ninguna dificultad en cuanto a su medición objetiva la medida de la homogeneidad en clases no es trivial. Por esta razón se define el coeficiente de concentración para el conjunto de patrones de entrada como:

$$pc = \frac{M \max_{i=1}^{M}\left(\mathrm{n}(C_i)\right) - \mathrm{n}(X)}{(M-1)\,\mathrm{n}(X)} \qquad (10)$$

del cual, en forma similar a los coeficientes *cc* y *cd*, pueden enunciarse las propiedades:

ppc1) $pc = 1$ si $\exists i^* / n\left(C_i\right) = 0 \ \forall \ 1 \leq i \neq i^* \leq M$
(si existe j^* entonces es único por ho2)

ppc2) $pc = 0$ si $\mathrm{n}(C_{i_1}) = \mathrm{n}(C_{i_2}) \ \forall 1 \leq i_1, i_2 \leq M$

ppc3) $0 < pc < 1$ en cualquier otro caso

ppc4) pc es monótono creciente con $\max_j\left(\mathrm{n}(C_i)\right)$

Para determinar el tipo de nodo en base a las características mencionadas se comparan sus medidas con dos umbrales: el umbral de concentración mínima de patrones de entrada (*upc*) y el umbral de cantidad mínima de patrones de entrada (*unX*). De esta forma se dice que un nodo es

sinc(*i*) Research Center for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
D. H. Milone, J. C. Sáez, G. Simón & H. L. Rufiner; "Arboles de redes neuronales autoorganizativas"
Revista Mexicana de Ingeniería Biomédica, Vol. 19, No. 4, pp. 13-26, Dic, 1998.

terminal cuando la concentración de patrones de entrada supera el umbral *upc* o cuando la cantidad de patrones es menor que el umbral *unX*.
Si se encuentra un nodo clasificador entonces éste debe entrenarse. La red neuronal que debe entrenarse para implementar la tarea de clasificación en el nodo es un MAO y se entrena por el método explicado anteriormente. La dimensión de entrada en esta red está determinada por la dimensión de los patrones y es la misma para todo el árbol. La dimensión o cantidad de clases de salida junto con los nodos terminales definen la topología final del árbol.
Para determinar la cantidad apropiada de clases de salida se utiliza un proceso de crecimiento de nodo basado en los coeficientes *cc* y *cd* y dos umbrales de capacidad de clasificación mínima *ucc* y *ucd* respectivamente. Se adopta inicialmente una configuración con dos clases de salida (N=2), se entrena la red y se evalúa su desempeño en la clasificación. En el caso en que no se supere alguno de los umbrales se incrementa N en uno y se repite el entrenamiento y prueba. Este proceso culmina cuando ambos coeficientes superan sus correspondientes umbrales o cuando N alcanza el máximo permitido *maxN*. En este último caso, se elige la mejor de todas las configuraciones entre 2 y *maxN* y se considera concluido el entrenamiento de ese nodo. Este algoritmo de crecimiento de nodo se repite para todos los nodos de cada nivel del árbol y puede verse en la figura 7.
Las exigencias en cuanto a concentración y dispersión varían de acuerdo al nivel de profundidad en el proceso de clasificación. En las primeras etapas de la clasificación se propone una mayor exigencia en cuanto a la concentración, mientras que la separación basada en detalles más finos se realiza progresivamente en niveles posteriores, en los que se exige mejor dispersión en la clasificación. Así se pasa gradualmente desde la no supervisión a la supervisión y se logra progresivamente la concordancia entre las clases de salida y las de entrada.
Cuando el árbol ha sido entrenado se procede al etiquetado de los nodos terminales. La elección de la etiqueta asignada a cada nodo terminal se realiza en base a la siguiente ecuación:

$$J_i = \arg\left[\max_{j=1}^{N}\left\{P(x_i \in C_j^O | x_i)\right\}\right] \qquad (11)$$

Los nodos terminales se unen, de acuerdo a su etiqueta, en otro nivel de nodos artificiales que poseen las etiquetas de todas las clases y de esta forma en el ARNA completo se cumple $M = N$.

3.2 Funcionamiento del ARNA entrenado

Para realizar la clasificación de un patrón una vez que el ARNA ha sido entrenado, se necesita propagarlo a través de él. La propagación del patrón puede realizarse en forma secuencial o en forma paralela.
Cuando se propaga un patrón en forma secuencial se describe un camino a través del árbol mediante un simple algoritmo como el siguiente:

1) Se comienza haciendo pasar el patrón por el nodo raíz
2) Se miden las distancias del patrón a cada uno de los centroides del MAO correspondiente.
3) Se elige como nodo siguiente aquel cuyo respectivo centroide está más cerca del patrón.
4) Se vuelve a 2 hasta que se llega a un nodo terminal y se clasifica al patrón según la etiqueta de este nodo.

En la propagación paralela se miden las distancias entre el patrón y todos los centroides simultáneamente y luego se sigue el camino formado por los nodos activados a partir del nodo raíz, hasta llegar a un nodo terminal. De esta manera el funcionamiento del ARNA ya armado puede paralelizarse en una implementación hardware que, para el caso de los MAO, no requiere la evaluación de funciones de nodo no lineales y puede basarse en simples comparadores. Esto hace que la implementación pueda ser simple y eficiente [20].

3.3 Patrones de entrenamiento y pruebas de validación

3.3.1 Descripción de las bases de datos

Las bases de datos empleadas en las pruebas del algoritmo fueron tomadas en su mayoría de [23]. En el trabajo citado se presenta una comparación de varios paradigmas incluido un *árbol de perceptrones simples* (APS). Estos resultados se utilizarán para contrastar con los obtenidos en el presente trabajo. Se emplearon dos tipos de datos: artificiales y reales. Entre las bases reales se eligieron dos relacionadas con la clasificación de fonemas debido a que esta aplicación es de interés central para nuestro grupo. A continuación se describen brevemente las bases de datos utilizadas.
Clouds: esta base de datos artificial fue creada con el objetivo de estudiar el comportamiento de los clasificadores para una distribución de clases con gran intersección y alto grado de no linealidad en las fronteras [23]. Está formada por 5000 patrones bidimensionales separados en dos clases. La primera clase está constituida por la suma de tres distribuciones gaussianas diferentes y la segunda por una distribución normal única.
IRIS: esta base de datos real fue tomada de [23], [24]. La fuente original fue [25], aunque se hizo famosa a través de los trabajos de R. A. Fisher [26], [27]. Esta base de datos ha sido ampliamente utilizada en la bibliografía y es quizás la más conocida en la literatura de reconocimiento de patrones. Para ejemplos referirse a [28], [29], [30] y recientemente [23]. El conjunto de datos está formado por 150 ejemplos (50 de cada clase), cada uno de los cuales consta de 4 atributos. El número de clases es 3 y cada una corresponde a un tipo de planta IRIS. Una de las clases es linealmente separable de las otras dos, las cuales no son

linealmente separables una de otra. En la figura 5 se observa la distribución en clases de las patrones proyectados en sus componentes principales.

Phoneme: esta base de datos se desarrollo para su aplicación a un sistema para reconocimiento del habla en francés y español en tiempo real [23]. Los datos iniciales fueron elaborados obteniendo un espectro coclear (del tipo escala de Mel [31]). La salida de los filtros se tomó cada 2 u 8 [mseg] dependiendo del tipo de fonema observado (estacionario o transitorio). El objetivo de la base de datos es distinguir entre vocales nasales y orales por lo que hay dos clases. Esta base contiene vocales tomadas de 1809 sílabas aisladas. Para caracterizar cada vocal se escogieron cinco atributos diferentes. Estos atributos corresponden a las amplitudes de los cinco primeros armónicos, normalizados por la energía total (integrada en todas las frecuencias). Se guardaron tres instantes diferentes por cada vocal para obtener los 5404 patrones de la base actual luego de remover los patrones inconsistentes.

Peterson: esta es una base de datos clásica en audiología. Consiste en las primeras cuatro formantes de las vocales del inglés [F1-F4]. La compilación de la base se debe a Peterson y Barney [32]. La base consta de dos repeticiones de diez vocales inglesas emitidas por un total de 76 hablantes (15 niños, 33 hombres y 28 mujeres), lo que produce 1520 emisiones. Los datos se identificaron luego por 26 oyentes y se marcaron los que tuvieron problemas de reconocimiento. La distribución de patrones para las clases de esta base de datos se pueden ver proyectadas sobre F1 y F2 en la figura 6. Recientemente se han tomado datos nuevos y se han hecho revisiones sobre los datos originales [33].

3.3.2 Descripción del método de validación

Es ampliamente conocido que cuando se prueba la exactitud de clasificación de un algoritmo la tasa de error tiende a estar sesgada si se la estima a partir del mismo conjunto de datos que fueron utilizados para entrenar el método. Lo que en realidad debe medirse es el poder de generalización del algoritmo con datos distintos a los utilizados durante el entrenamiento. Sin embargo en las aplicaciones reales se cuenta con un numero finito de ejemplos de cada clase. El método de validación cruzada conocido como *reservar aparte* (RA) (en inglés *hold out*) consiste en realizar una partición del conjunto de patrones para obtener los subconjutos de entrenamiento y prueba. De esta manera los patrones de prueba no se utilizan durante el entrenamiento. Como usualmente la distribución de patrones por clase en cada conjuto no es uniforme, es posible que la partición particular sesgue la estimación del error. Es por ello que se han propuesto varias alternativas que permiten mejorar la estimación del error [34], [23]. La mejor estimación la provee el método *dejar-uno-aparte* (DUA). Si existen N patrones, este método consiste en realizar N entrenamientos separando en cada uno un patrón distinto para usarlo en la prueba. El resultado final es el porcentaje de aciertos en las pruebas. En algunos casos este método no es implementable debido a su alto costo computacional.

Existen alternativas que permiten establecer un compromiso entre el costo computacional y la probabilidad de sesgar los resultados. Una de estas alternativas es el método denominado *dejar-k-aparte* (DKA) [23] que consiste en tomar m particiones de los N datos: $m = N /k$; dejando k patrones para prueba y usando los restantes $N – k$ para entrenamiento. Los m resultados de desempeño así obtenidos son promediados para obtener la estimación final del error. Los patrones no se repiten en los conjuntos de prueba, es decir que cada conjunto de prueba posee patrones diferentes a los de los otros conjuntos. Otra alternativa es el método *reservar aparte promediado* (RAP), que consiste en realizar varias estimaciones RA con diferentes particiones y promediar los resultados. En este caso no se restringe la cantidad de patrones por partición de prueba y se reduce el costo computacional manteniendo una buena estimación del desempeño.

3.4 Detalles de implementación

Todas las pruebas se realizaron en computadoras PC tipo Pentium, y el desarrollo del nuevo algoritmo (ARNA) fue realizado bajo el paradigma de programación orientada a objetos en ambiente Delphi 3, utilizándose para los MAOs librerias dinámicas precompiladas provenientes del paquete comercial NeuroWindows [35]. También se desarrollaron rutinas para implementar los métodos de validación cruzada.

4 Resultados

Con el objeto de aclarar el comportamiento de los coeficientes se muestra un ejemplo sencillo en el que se entrenó un clasificador real. El clasificador utilizado es un MAO aislado como los descriptos anteriormente. Como patrones de entrenamiento se utilizan los de la base de datos de **Peterson**. En una experimentación controlada basada en el método DKA se obtuvieron los resultados que se ven en la figura 8. Aquí se puede ver la evolución de los coeficientes de concentración y dispersión en función de la cantidad de clases de salida.

El coeficiente de concentración comienza con valores altos (0.98 para 2 clases de salida) y decrece a medida que aumenta la cantidad de clases de salida (hasta 0.78 para 15 clases de salida). El coeficiente de dispersión comienza con valores bajos (0.21 para 2 clases de salida) y se incrementa a medida que aumenta la cantidad de clases de salida (0.67 para 15 clases de salida). Ambos coeficientes se cruzan cuando la cantidad de clases de entrada es igual a la cantidad de nodos de salida. También se consiguieron gráficas similares a ésta para las bases de datos **IRIS**, **Clouds** y **Phoneme**.

Los siguientes resultados conciernen al entrenamiento del ARNA. En primer lugar se presentan resultados relacionados con su crecimiento. La figuras 9 muestran dos diagramas de flujo de clases para un entrenamiento con los patrones de **Peterson** y los fonemas /ao/ y /iy/ respectivamente. En estos diagramas cada línea

sinc(*i*) Research Center for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
D. H. Milone, J. C. Sáez, G. Simón & H. L. Rufiner; "Arboles de redes neuronales autoorganizativas"
Revista Mexicana de Ingeniería Biomédica, Vol. 19, No. 4, pp. 13-26, Dic, 1998.

horizontal separa un nivel del siguiente y cada rectángulo representa a un nodo. Los rectángulos que llegan hasta el último nivel son nodos terminales. En tonos de gris se muestra la cantidad de patrones de la clase en cuestión, en cada nodo y en relación al total de patrones de la clase.
A continuación se muestra las relación que existe entre las características de la base de datos de entrenamiento y la topología del ARNA construido por el algoritmo de crecimiento. Para este ejemplo se utilizó la base de datos **IRIS**. El árbol que se genera con esta distribución de patrones puede observarse en la figura 10. Esta figura muestra la topología del ARNA e indica además la cantidad de patrones de cada una de las clases que ha llegado a cada nodo. Cada nodo terminal posee una etiqueta que lo identifica según su clase mayoritaria.
Para terminar esta sección se presentan los resultados finales de las pruebas de validación para un conjunto de problemas de clasificación bien conocidos. Para contrastar los resultados obtenidos con ARNA se muestran los resultados obtenidos con otras técnicas de clasificación aplicadas al mismo problema y una estimación del máximo desempeño de Bayes mediante el método de los *k-vecinos más cercanos* [23]. En las tablas 1 a 3 se compara el coeficiente de reconocimiento del ARNA con el de PMC, el de *aprendizaje por cuantización vectorial* (ACV, o conocido por su sigla en inglés LVQ) y el de APS extraídos del trabajo comparativo de [23]. La tabla 4 compara los resultados con PMC y ACV a partir de los patrones de **Peterson**. En las figuras 11 a 14 se muestran estos resultados en forma gráfica para una comparación cualitativa. El método de validación utilizado para evaluar el rendimiento del ARNA fue el RAP. Se eligió este método con el objeto de que las comparaciones con los resultados reportados previamente en [23] fueran consistentes. Además, a pesar de los inconvenientes ya señalados para el coeficiente de reconocimiento, el mismo tuvo que utilizarse en las comparaciones ya que los autores de [23] lo utilizaron para reportar sus resultados.
Para tener una referencia de la velocidad de entrenamiento se entrenaron un PMC y un ARNA con la base de datos **Peterson**. Para que la comparación sea válida se utilizó RAP en el caso del PMC, requiriendo de éste un desempeño similar al del ARNA. El resultado final indica que el ARNA es 8.4 veces más rápido que el PMC.

5 Discusión

En las experiencias realizadas con el MAO aislado el *cd* aumentó en forma notoria a medida que la cantidad de nodos de salida se aproximaba a la cantidad de clases de entrada. Esto concuerda con un aumento en la eficiencia de la red para separar las clases de entrada. El *cd* continuó incrementándose, aunque en menor medida, cuando la cantidad de nodos de salida fue superior a la cantidad de clases de entrada. Por otro lado se observa que el *cc* se comporta de forma opuesta. Cuando el clasificador posee pocos nodos de salida, se produce una mayor concentración de patrones en las clases de salida y por ende una mayor concentración de patrones de una misma clase de entrada en una misma clase de salida. A medida que la cantidad de nodos de salida aumenta, existen más posibilidades de que la red separe en distintas clases de salida patrones de la misma clase de entrada. Esta situación implica una disminución en la capacidad de la red para concentrar patrones de una misma clase de entrada en una misma clase de salida. En las experiencias realizadas el *cc* comenzó con valores altos para pocos nodos de salida y disminuyó con el aumento de los nodos de salida, lo cual concuerda con la disminución en la capacidad de concentración de la red. Se puede observar con esta simple experiencia que *cd* y *cc* son indicadores de la capacidad de la red para dispersar patrones de distintas clases de entrada en distintas clases de salida y para concentrar patrones de la misma clase de entrada en una misma clase de salida respectivamente. Esto es lo esperado de acuerdo a la forma en la que se plantearon los coeficientes.
Las gráficas de flujo de patrones muestran el comportamiento de un ARNA ya entrenado para el caso de **Peterson** (figura 9). Los fonemas fácilmente diferenciables entre sí como los conjuntos de fonemas /ao/ e /iy/ (figura 6) son separados en el primer nivel de decisión. Por otro lado cuando dos clases de patrones son más confundibles son separadas en niveles de decisión posteriores. Esto deja de manifiesto el comportamiento jerárquico del ARNA, el cual realiza una separación más fina de los fonemas en los últimos niveles de decisión.

Es interesante comparar la topología del ARNA generada de la figura 10 con la distribución de patrones de la base de datos **IRIS** en la figura 5. Se puede observar como la clase Setosa, que es linealmente separable, queda aislada en una única clase de salida a partir del nodo 2. Al nodo 3 llegan la mayor parte de los patrones correspondientes a la clase Versicolor y la totalidad de los patrones de la clase Virginica. Se puede observar en la figura 5 que la complejidad de este subproblema de clasificación es mayor, lo que genera una estructura más compleja en la topología a partir del nodo 3. Todas las ramas a partir del nodo 6 se generan para resolver el problema de separar un patrón Versicolor rodeado por 16 patrones cercanos de la clase Virginica. Nótese que el árbol generado posee error cero debido a que el umbral de *pc* fue fijado en 1 (con lo que se obtiene un árbol sobreentrenado). Ésto también es importante ya que indica que el ARNA es una estructura de clasificación que en forma práctica puede alcanzar un error de clasificación cero durante el entrenamiento. Otras estructuras, como el PMC, también presentan esta propiedad en forma teórica pero difícilmente la alcanzan en la práctica.

La jerarquización en la clasificación adquiere especial importancia cuando se resuelven problemas en el campo del reconocimiento automático del habla. En estos problemas existe una jerarquía inherente a la generación misma de los datos de entrenamiento. En el caso de la

sinc(*i*) Research Center for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
D. H. Milone, J. C. Sáez, G. Simón & H. L. Rufiner; "Arboles de redes neuronales autoorganizativas"
Revista Mexicana de Ingeniería Biomédica, Vol. 19, No. 4, pp. 13-26, Dic, 1998.

clasificación de fonemas se encuentran sub clasificaciones según la energía, el grado de tensión de las cuerdas vocales, el sexo, la posición de las constricciones en el tracto vocal, etc. Las sucesivas particiones generadas por los MAO permiten separar los patrones según sus características más sobresalientes en los primeros niveles del árbol e incrementar el nivel de detalle para lograr una clasificación definitiva en los nodos terminales.

Si se comparan los resultados obtenidos con el ARNA y otros clasificadores se observa que la performance comparativa varía para las diferentes bases de datos. El ARNA tuvo un desempeño superior a todas las otras arquitecturas cuando se utilizó la base **IRIS** para el entrenamiento. El desempeño del ARNA para la base **Phoneme** fue prácticamente el mismo que el desempeño obtenido por el resto de las arquitecturas. Sin embargo, para la base **Clouds** el rendimiento del ARNA fue 3.72% inferior al obtenido por un PMC, que fue el clasificador que obtuvo el mejor resultado. En el caso de los patrones de **Peterson** los resultados favorecen al ARNA frente a ACV y PMC.

La razón de la disminución comparativa de los resultados del ARNA se debe a una limitación técnica de la implementación del algoritmo. La máxima cantidad de MAOs que se pueden generar está limitada a 128. Esta cantidad parece ser suficiente para bases simples como **IRIS** y **Peterson**, sin embargo no lo es para bases más complejas como **Clouds** y **Phoneme**. De esta manera, debido a la limitación del número máximo de MAOs que se pueden generar, las configuraciones alcanzadas por el ARNA no son las óptimas para la solución del problema.

Una consideración muy importante a la hora de realizar comparaciones entre diferentes arquitecturas es el hecho de que mientras el ARNA adapta su topología al problema en cuestión, otros métodos como ACV y PMC necesitan que se especifique una configuración inicial, generalmente basada en la experiencia del usuario y refinada mediante prueba y error. De hecho los resultados para ACV, PMC y APS que se encuentran en las tablas 1, 2, 3 corresponden a resultados alcanzados con la mejor configuración hallada después de numerosas pruebas. Aquí se debe aclarar que ARNA adapta su topología a través de múltiples ensayos, como se desprende del método de crecimiento de nodos (la figura 7), aunque lo hace de manera jerárquica y automática en cada nodo lo que resulta en un ahorro computacional sustancial con respecto a los métodos de prueba y error citados.

Dado que los cómputos realizados para la generación de un ARNA son sencillos, esta arquitectura es considerablemente más veloz que otras estructuras con algoritmos más complejos. Sin embargo puede requerir más operaciones para solucionar un mismo problema que los métodos clásicos de generación de árboles como ID3 o C4. Su principal ventaja frente a éstos últimos la posibilidad de generar fronteras mucho más complejas.

6 Conclusiones y trabajos actuales

En el presente trabajo se presenta un algoritmo para el crecimiento de árboles de redes neuronales autoorganizativas y se compara su desempeño frente a otros métodos de clasificación con distintas bases de datos. La principal fuente de las ventajas de este método está en la combinación de diferentes paradigmas de clasificación, lo que permite aprovechar las ventajas de cada uno. El algoritmo planteado combina las ventajas del aprendizaje supervisado con las del aprendizaje no supervisado. Para el crecimiento y definición de la topología del árbol se utiliza información acerca de la identidad de los patrones. En cambio, para la tarea de clasificación en cada nodo se utiliza un extractor de características basado en un MAO que no usa información acerca de identidad de los patrones de entrenamiento. Otra de las combinaciones de paradigmas de clasificación que se encuentran en este algoritmo es la de los clasificadores simples y los jerarquizados. Mientras que la estructura general responde a los métodos de clasificación jerarquizada, en cada nodo se utiliza un típico clasificador simple como lo es una red neuronal. El ARNA es un clasificador muy flexible que no necesita la definición a priori de la topología (como se requiere en el caso del PMC). La velocidad del método es otra característica que hace del ARNA una configuración adecuada para la implementación de tareas de clasificación.
Si bien en algunos de los experimentos ARNA clasificó mejor, se puede decir que los resultados no favorecen a ningún método en particular. Con el objeto de resolver el problema generado por la limitación en el número de redes neuronales actualmente se realizan modificaciones de la implementación, para poder generar un número indeterminado de redes neuronales. De esta forma se podrán atacar problemas complejos. Como hay problemas que pueden ser muy difíciles de resolver para un MAO y no para otros paradigmas de redes neuronales, actualmente se implementa la sustitución de los nodos MAO de bajo desempeño con PMC, ACV o redes neuronales de base radial. Existen problemas donde el sistema de clasificación debe tomar decisiones que dependen del patrón de entrada actual y de T patrones anteriores, siendo T variable. Este es el caso del reconocimiento automático de habla y se debe proveer de memoria al sistema de clasificación. Se estudian actualmente diferentes alternativas para convertir al ARNA en un sistema dinámico de clasificación.

7 Agradecimientos

Agradecemos a Daniel A. Zapata por su participación en las discusiones llevadas a cabo durante las etapas iniciales de la elaboración es este trabajo. A Aldo D. Sigura, y Damián Tochetto por

sinc(*i*) Research Center for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
D. H. Milone, J. C. Sáez, G. Simón & H. L. Rufiner; "Arboles de redes neuronales autoorganizativas"
Revista Mexicana de Ingeniería Biomédica, Vol. 19, No. 4, pp. 13-26, Dic, 1998.

sus aportes en la implementación y depuración de los programas.
Este trabajo ha sido soportado por la Universidad Nacional de Entre Ríos.

8 Referencias

[1] J. Mao, A.C. Jain, *"A Self-Organizing Network for Hyper-Ellipsoidal Clustering"*, IEEE Trans. On Neural Networks, vol.7, no.1, pp16, Jan. 1996.
[2] T. Kohonen, *"The Self-Organizing Map"*, New York: Springer-Verlag, 1995.
[3] B. Kosko, *"Fuzzy Engineering"*, Prentice Hall, 1997.
[4] D. R. Hush, B. G. Horne, *"Progress in Supervised Neural Netwoks. What's New Since Lippmann"*, IEEE ASSP Mag., vol. 10, no.1, pp 8-39, 1993.
[5] M. P. Windham, *"Cluster Validity for the Fuzzy c-Means Clustering Algorithm"*, IEEE Trans. Pattern Anal. Machine Intel., vol. PAMI-4, no.4, pp 357-363, July 1982.
[6] R. P. Lippmann, *"An introduction to computing with neural nets"*, IEEE ASSP Mag., vol.4, no.2, pp 4-22, 1987.
[7] Goddard J.C., Martinez F.M.; Martinez A.E., Cornejo J.M., Rufiner H.L., Acevedo R.C., *"Aprendizaje Maquinal en Medicina: Diabetes, un caso de Estudio";* Revista Mexicana de Ingeniería Biomédica Volumen 16, no.2, Octubre de 1995.
[8] L. Breiman, J. H. Friedman, R. A. Olshen, C. J. Stone, *"Classification and Regression Trees"*, Wadsworth. Int., 1984.
[9] J. Ross Quinlan, *"C4.5: Programs for Machine Learning"*, The Morgan Kaufmann Series in Machine Learning, Pat Langley, Series Editor, 1993.
[10] Sestito, Dillon, *"Automated Knowledge Acquisition"*, Prentice Hall 1994.
[11] Rubén Acevedo, *"Representación tonotópica de la corteza auditiva"*, Tesis de Maestría en Ingeniería Biomédica UAMI, México, Marzo 1997.
[12] T. Kohonen, *"The Self-Organizing Map"*, Proc. IEEE, vol.78, no.9, pp 1464-1480, Sept. 1990.
[13] T. Kohonen, *"Self-Organized Formation of Topologically Correct Feature Maps"*, Biological Cybernetics, vol. 44, pp. 59-69, 1982.
[14] S. Haykin, *"Neural Networks. A Comprehensive Foundation"*, Macmillan College Publishing Company, 1994.
[15] J. A. Walter, K.Schulten, *"Implementation of Self-Organizing Neural Networks for Visuo-Motor Control of an Industrial Robot"*, IEEE Trans. on Neural Networks, vol.4, no.1, pp 86, 1993.
[16] J. Freeman, D. Skapura, *"Neural Networks. Algorithms, Aplications and Programming Techniques"*, Addison-Wesley Publishing Company, 1991.
[17] Chen-Xiong Zhang, D.A.Mlynski, *"Mapping and Hierarchical Self-Organizing Neural Networks for VLSI Placement"*, IEEE Trans. On Neural Networks, vol.8, no.2, pp 299-314, 1997.
[18] A. Sankar, R. J. Mammone, *"Neural Tree Networks"*, en *"Neural Networks: Theory and Applications"*, R. Mammone and Y. Zeevi Eds., New York: Academic Press, pp 281-302, 1991.
[19] I. K. Sethi, *"Decision tree performance enhancement using an artificial neural network implementation"*, Artificial Neural Networks and Statitical Pattern Recognition. Old and New Conections, Ed. I. K. Sethi, A. K. Jain, Elsevier Science Publishers B.V., 1991.
[20] M. G. Rahim, *"A self-learning neural tree network for phone recognition"*, Artificial Neural Networks for Speech and Vision, Ed.R. J. Mammone, Chapman & Hall, 1993.
[21] C. Ke, Y. Xiang, CH. Huisheng, Y. Liping, *"Modular-Tree: A Self-architecture Neural Network Architecture"*, Report of National Lab of Machine Perception and Center for Information Science Peking University, vol. 32, no.1, pp 110-119, 1996.
[22] E. Ronco, P. Gawthrop, *"Modular Neural Networks: a state of the art"*, Tech. Report: CSC-95026. Centre for System and Control, Univ. of Glasgow, Glasgow, UK, May 1995.
[23] Guerin-Dugue, A. and others, *"Deliverable R3-B4-P - Task B4: Benchmarks"*, Technical Report of Elena-Nerves II "Enhanced Learning for Evolutive Neural Architecture", ESPRIT-Basic Research Project Number 6891, June 1995.
[24] Blake, C., Keogh, E. & Merz, C.J., *"UCI Repository of machine learning databases"* [http://www.ics.uci.edu/~mlearn/MLRepository.html]. Irvine, CA: University of California, Department of Information and Computer Science, 1998.
[25] Anderson, E. *"The Irises of the Gaspe Peninsula"*, Bulletin of the American Iris Society, 59, pp 2-5, 1935.
[26] Fisher, R.A. *"The use of multiple measurements in taxonomic problems"*, Annual Eugenics, 7, Part II, pp 179-188, 1936.
[27] Fisher, R.A. *"Contributions to Mathematical Statistics"*, John Wiley, NY, 1950.
[28] Duda, R.O., & Hart, P.E. *"Pattern Classification and Scene Analysis"*. John Wiley & Sons., pp 218, 1973.
[29] Dasarathy, B.V. *"Nosing Around the Neighborhood: A New System Structure and Classification Rule for Recognition in Partially Exposed Environments"*. IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. PAMI-2, no.1, pp 67-71, 1980.
[30] Gates, G.W. *"The Reduced Nearest Neighbor Rule"*. IEEE Transactions on Information Theory, pp 431-433, May 1972.
[31] J. Deller, J. Proakis, J. Hansen, *"Discrete Time Processing of Speech Signals"*. Macmillan Publishing, NewYork, 1993.
[32] G. E. Peterson, H. L. Barney, *"Control methods used in a study of the vowels"*, J. Acoust. Soc. Am. 24, pp 175-184, 1952.
[33] J. Hillenbrand, L. A. Getty, M. J. Clark, K. Wheeler, *"Acoustic characteristics of American English vowels"* J. Acoust. Soc. Am. 97, pp 3099-3111, 1995.

[34] D. Michie, D.J. Spiegelhalter, C.C. Taylor, *"Machine Learning, Neural and Statistical Classification"*, Ellis Horwood, University College, London, 1994.

[35] Ward Systems Group, Inc.; *NeuroWindows*; [http://www.wardsystems.com] Executive Park West; 5 Hillcrest Drive; Frederick, MD 21702; USA.

9 Figuras

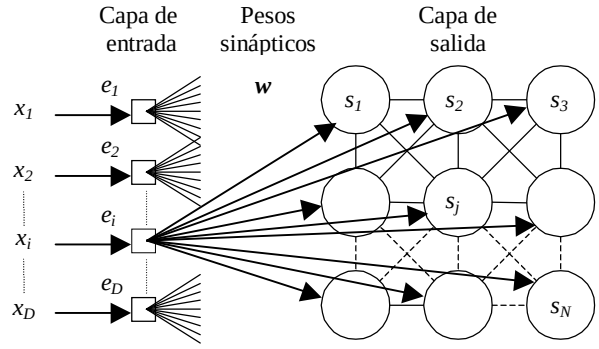


Fig. 1: Configuración de un MAO.

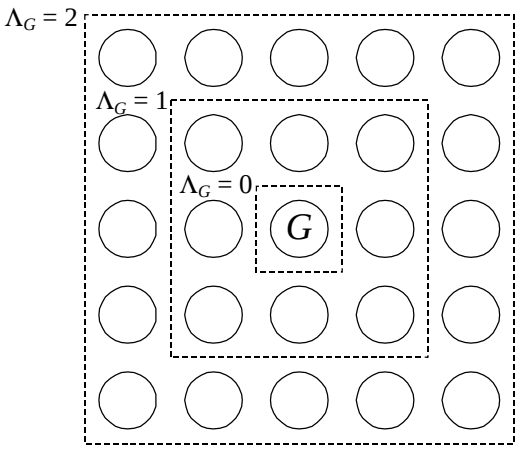


Fig.2: Reducción de vecindades en torno a la neurona ganadora *G*.

sinc(*i*) Research Center for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
D. H. Milone, J. C. Sáez, G. Simón & H. L. Rufiner; "Arboles de redes neuronales autoorganizativas"
Revista Mexicana de Ingeniería Biomédica, Vol. 19, No. 4, pp. 13-26, Dic, 1998.

Inicialización: Se asignan pequeños valores aleatorios para los vectores de peso.
Repetir

Muestreo: Se presenta un patrón de entrada $\boldsymbol{x}(n)$ elegido de forma aleatoria.
Prueba de similitud: se encuentra la neurona ganadora G para el tiempo n usando un criterio de mínima distancia euclídea. Si $\boldsymbol{w}_j$ es el vector de pesos que relaciona a la neurona de salida s_j con todas las neuronas de entrada, entonces la neurona ganadora G será aquella tal que

$$G[\boldsymbol{x}(n)] = \arg\left[\min_{j=1}^{N} \left\|\boldsymbol{x}(n) - \boldsymbol{w}_j(n)\right\|\right]$$

Adaptación: se ajustan los vectores de pesos sinápticos para la neurona ganadora y sus vecinas usando la fórmula:

$$\boldsymbol{w}_j(n+1) = \begin{cases} \boldsymbol{w}_j(n) + \eta(n)[\boldsymbol{x}(n) - \boldsymbol{w}_j(n)], & si\ s_j \in \Lambda_{G(x)}(n) \\ \boldsymbol{w}_j(n), & \text{en otro caso} \end{cases}$$

donde $\eta(n)$ es la velocidad de aprendizaje y $\Lambda_{G(x)}(n)$ es la función de entorno o vecindad centrada alrededor de la neurona ganadora.

Hasta no observar cambios en el mapa de características.

Figura 3: Algoritmo de entrenamiento para un MAO.

$$N^{IO} = \begin{bmatrix} 10 & 0 & 0 \\ 0 & 10 & 0 \\ 0 & 0 & 10 \end{bmatrix} \rightarrow \begin{cases} cc = 1 \\ cd = 1 \\ cr = 1 \end{cases} \qquad N^{IO} = \begin{bmatrix} 10 & 10 & 10 \\ 10 & 10 & 10 \\ 10 & 10 & 10 \end{bmatrix} \rightarrow \begin{cases} cc = 0 \\ cd = 0 \\ cr = 1/3 \end{cases}$$

(a) (b)

$$N^{IO} = \begin{bmatrix} 0 & 0 & 10 \\ 0 & 0 & 10 \\ 0 & 0 & 10 \end{bmatrix} \rightarrow \begin{cases} cc = 1 \\ cd = 0 \\ cr = NA \end{cases} \qquad N^{IO} = \begin{bmatrix} 10 & 0 & 0 & 0 & 10 \\ 0 & 10 & 0 & 10 & 0 \\ 0 & 0 & 10 & 0 & 0 \end{bmatrix} \rightarrow \begin{cases} cc = 1/2 \\ cd = 1 \\ cr = NA \end{cases}$$

(c) (d)

Figura 4: Ejemplos de aplicación de los coeficientes de clasificación: a) clasificador ideal; b) clasificador totalmente confundido; c) hay concentración pero nada de dispersión; d) máxima dispersión pero poca concentración.

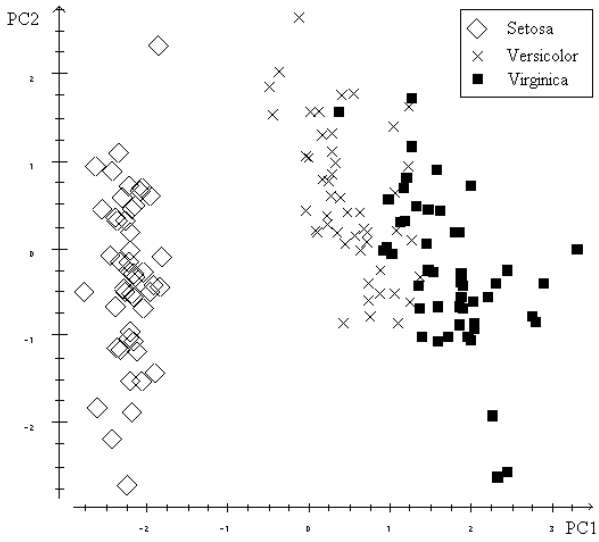


Figura 5: distribución de los patrones de **IRIS** proyectados es dos de sus componentes principales

sinc(*i*) Research Center for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
D. H. Milone, J. C. Sáez, G. Simón & H. L. Rufiner; "Arboles de redes neuronales autoorganizativas"
Revista Mexicana de Ingeniería Biomédica, Vol. 19, No. 4, pp. 13-26, Dic, 1998.

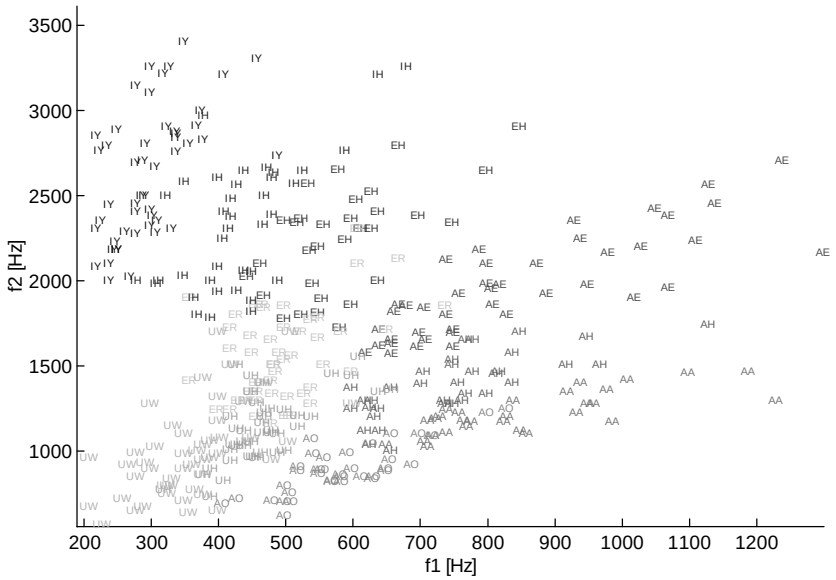


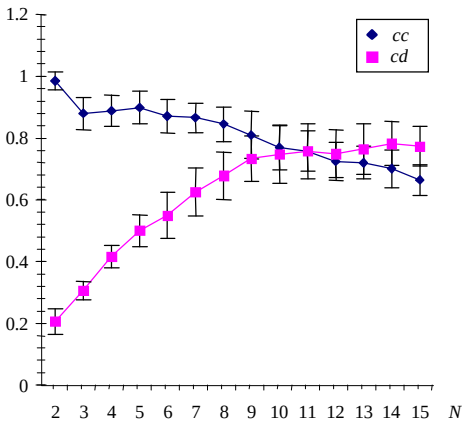
Figura 6: Distribución de algunas de las vocales del Inglés según Peterson.

```
Para cada nivel del árbol
  Para cada nodo del nivel
    Nodo Terminal = (pc>upc) OR (n(X)<unX)
    Si no(Nodo Terminal) entonces
      N=minN
      Mientras no(Nodo Entrenado) hacer
        Crear Nodo
        Entrenar Nodo
        Probar Nodo

        Si N = maxN entonces
          Nodo Entrenado = verdadero
          Buscar Mejor N
          Si Mejor N <> maxN entonces
            Destruir Nodo
            N = Mejor N
            Crear Nodo
            Entrenar Nodo
          Si No
            Nodo Entrenado = (cc>ucc) AND (cd>ucd)

        Si no(Nodo Entrenado) entonces
          Destruir nodo
          N=N+1
  Actualizar los umbrales
```

Figura 7: Algoritmo para el crecimiento de ARNA.



sinc(*i*) Research Center for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
D. H. Milone, J. C. Sáez, G. Simón & H. L. Rufiner; "Arboles de redes neuronales autoorganizativas"
Revista Mexicana de Ingeniería Biomédica, Vol. 19, No. 4, pp. 13-26, Dic, 1998.

Figura 8: Evolución de los coeficientes de clasificación en función de la cantidad de nodos en un clasificador MAO. La gráfica se obtuvo mediante el método de leave-*k*-out aplicado sobre la base de datos de **Peterson**. Las barras indican la desviación estándar y los puntos el promedio para un k = 10 (para mayores detalles ver la sección Patrones de entrenamiento y pruebas de validación).

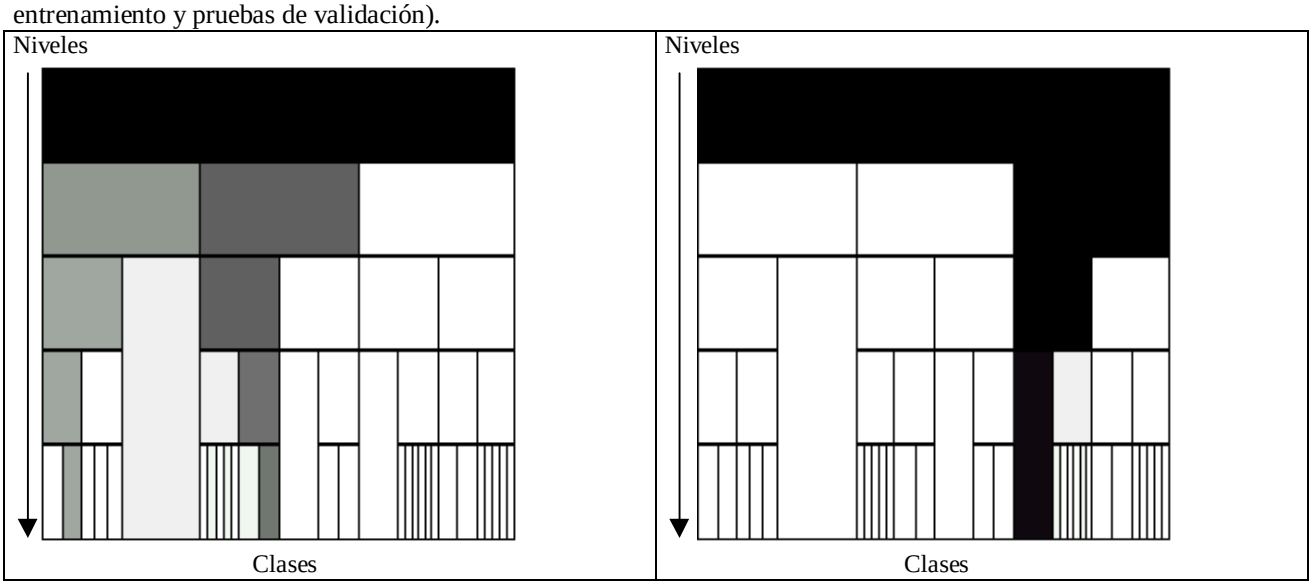
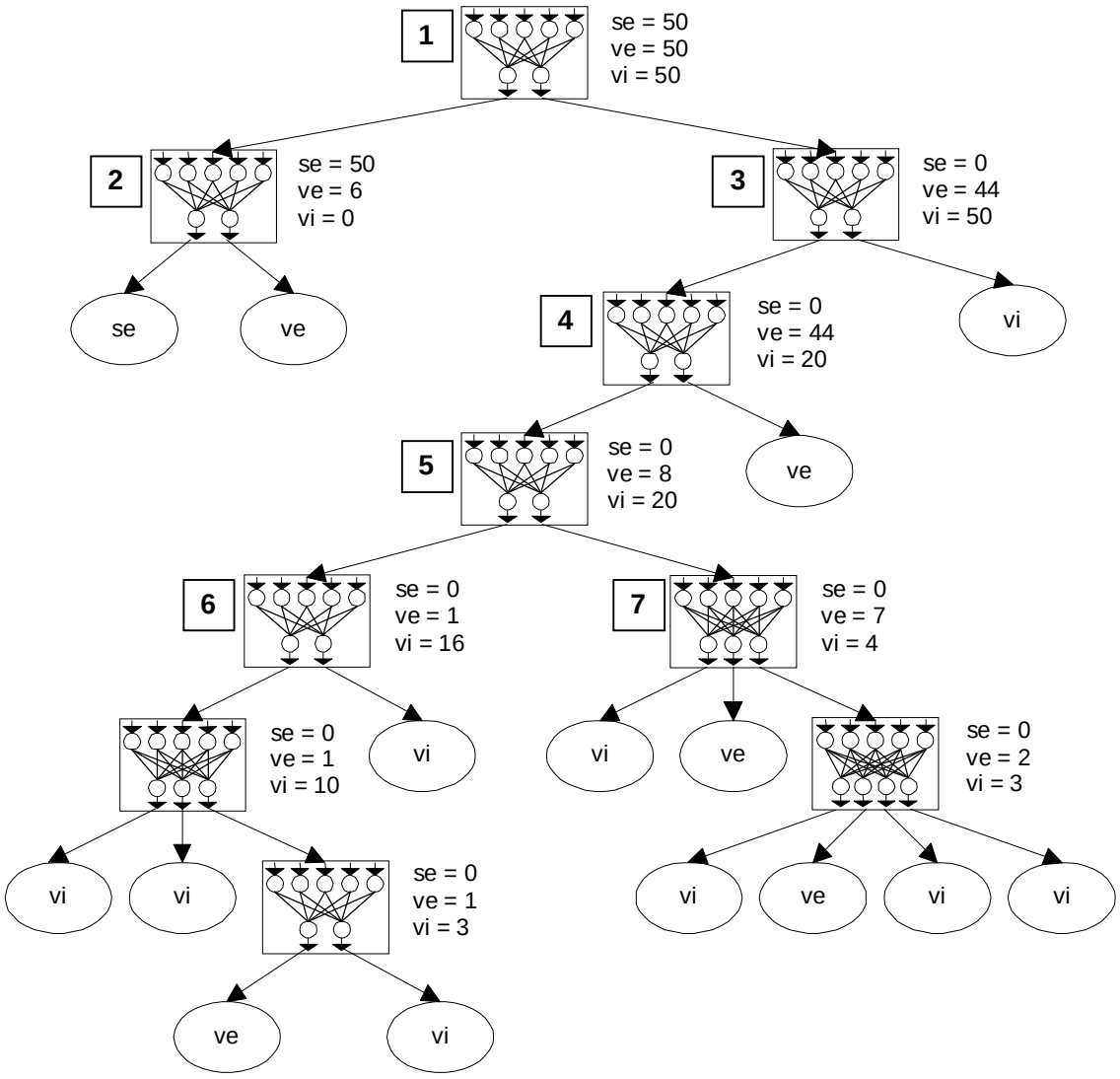


Figura 9: Diagramas de flujo de clase para el un entrenamiento basado en la base de datos de **Peterson**. Se observa el flujo de patrones para las clases /ao/ (izda) y /iy/ (dcha).



sinc(*i*) Research Center for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
D. H. Milone, J. C. Sáez, G. Simón & H. L. Rufiner; "Arboles de redes neuronales autoorganizativas"
Revista Mexicana de Ingeniería Biomédica, Vol. 19, No. 4, pp. 13-26, Dic, 1998.

Figura 10: árbol generado durante el entrenamiento con la base de datos **IRIS**. Los nodos clasificadores son los simbolizados con una red neuronal. Los nodos terminales poseen todos un *pc* = 1 y están simbolizados con una elipse en la que se indica la clase representada en el nodo. Referencias: se = Iris-setosa; ve = Iris-versicolor; vi = Iris-virginica.

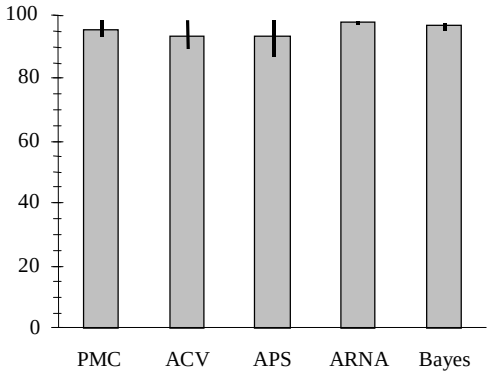


Figura 11: Resultados comparativos para la base de datos **IRIS**.

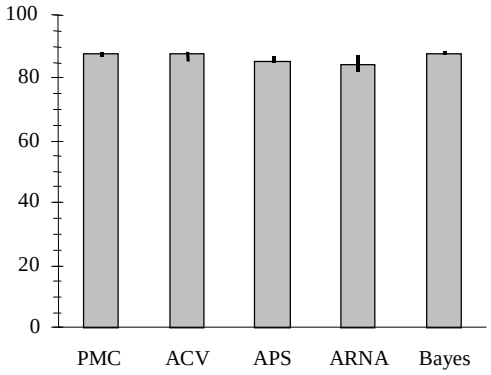


Figura 12: Resultados comparativos para la base de datos **Clouds**.

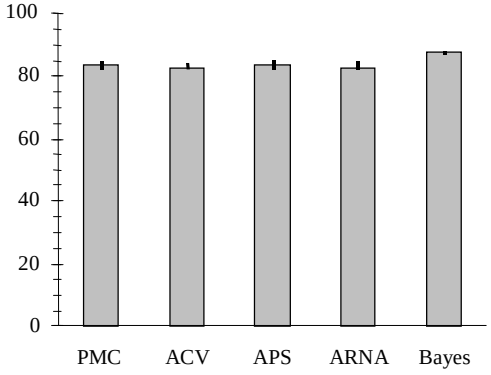


Figura 13: Resultados comparativos para la base de datos **Phoneme**.

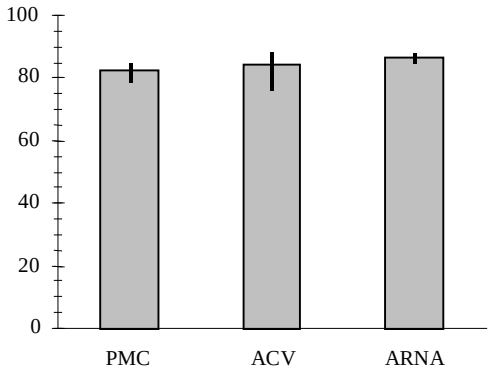


Figura 14: Resultados comparativos para la base de datos **Peterson**.

10 Tablas

cr %	μ	Min	Max
PMC	95.78	93.33	98.61
ACV	93.83	89.44	98.67
APS	93.33	86.61	98.61
ARNA	97.78	96.67	98.33
Bayes	96.66	94.72	97.27

Tabla 1: Resultados comparativos para la base de datos **IRIS**.

cr %	μ	Min	Max
PMC	87.66	86.77	88.33
ACV	87.66	85.66	88.55
APS	85.66	85.00	86.33
ARNA	83.93	82.20	87.20
Bayes	88.11	87.44	88.77

Tabla 2: Resultados comparativos para la base de datos **Clouds**.

cr %	μ	Min	Max
PMC	83.63	82.36	84.69
ACV	83.00	82.20	83.76
APS	83.47	82.04	85.06
ARNA	82.99	81.87	84.84
Bayes	87.7	86.81	88.02

Tabla 3: Resultados comparativos para la base de datos **Phoneme**.

cr %	μ	Min	Max
PMC	83.08	79.55	84.85
ACV	84.36	74.81	87.57
ARNA	86.36	84.85	87.88

Tabla 4: Resultados comparativos para la base de datos **Peterson**.

11 GLOSARIO DE SIGLAS UTILIZADAS

ACV	Aprendizaje por Cuantización Vectorial (LVQ)
AD	Árboles de Decisión
APS	Árboles de Perceptrones Simples
ARN	Árboles de Redes Neuronales
ARNA	Árboles de Redes Neuronales Autoorganizativas
DKA	Dejar-K-Aparte
DUA	Dejar-Uno-Aparte
MAO	Mapas Autoorganizativos
PMC	Perceptrón Multicapa
RA	Reservar Aparte
RAP	Reservar Aparte Promediado
RNA	Redes Neuronales Artificiales