Tutorial sobre Redes Neuronales Artificiales: Los Mapas Autoorganizados de Kohonen

Article · January 2002

CITATION
1

READS
2,059

3 authors:

Alfonso Palmer
University of the Balearic Islands
106 PUBLICATIONS **1,487** CITATIONS
SEE PROFILE

Juan José Montaño
University of the Balearic Islands
49 PUBLICATIONS **1,016** CITATIONS
SEE PROFILE

Rafael Jiménez
University of the Balearic Islands
48 PUBLICATIONS **324** CITATIONS
SEE PROFILE

REVISTA ELECTRÓNICA DE PSICOLOGÍA
Vol. 6, No. 1, Enero 2002
ISSN 1137-8492

Tutorial sobre Redes Neuronales Artificiales: Los Mapas Autoorganizados de Kohonen

Palmer, A., Montaño, J.J. y Jiménez, R.
Área de Metodología de las Ciencias del Comportamiento.
Facultad de Psicología.
Universitat de les Illes Balears.
e-Mail: alfonso.palmer@uib.es

1.- Introducción

En los últimos quince años, las redes neuronales artificiales (RNA) han emergido como una potente herramienta para el modelado estadístico orientada principalmente al reconocimiento de patrones –tanto en la vertiente de clasificación como de predicción. Las RNA poseen una serie de características admirables, tales como la habilidad para procesar datos con ruido o incompletos, la alta tolerancia a fallos que permite a la red operar satisfactoriamente con neuronas o conexiones dañadas y la capacidad de responder en tiempo real debido a su paralelismo inherente.

Actualmente, existen unos 40 paradigmas de RNA que son usados en diversos campos de aplicación (Taylor, 1996; Arbib, Erdi y Szentagothai, 1997; Sarle, 1998). Entre estos paradigmas, podemos destacar la red *backpropagation* (Rumelhart, Hinton y Williams, 1986) y los mapas autoorganizados de Kohonen (Kohonen, 1982a, 1982b).

La red *backpropagation*, mediante un esquema de aprendizaje supervisado, ha sido utilizada satisfactoriamente en la clasificación de patrones y la estimación de funciones. La descripción de este tipo de red se puede encontrar en un documento anterior (Palmer, Montaño y Jiménez, en prensa).

En el presente documento, nos proponemos describir otro de los sistemas neuronales más conocidos y empleados, los mapas autoorganizados de Kohonen. Este tipo de red neuronal, mediante un aprendizaje no supervisado, puede ser de gran utilidad en el campo del análisis exploratorio de datos, debido a que son sistemas capaces de realizar análisis de clusters, representar densidades de probabilidad y proyectar un espacio de alta dimensión sobre otro de dimensión mucho menor. A fin de asegurar la comprensión de los conceptos expuestos por parte del lector y, al mismo tiempo, explotar al máximo los recursos que nos ofrece la Web, tal descripción irá acompañada de applets y software ilustrativos, los cuales estarán a disposición del lector via internet.

2.- Los mapas autoorganizados de Kohonen

En 1982 Teuvo Kohonen presentó un modelo de red denominado mapas autoorganizados o SOM (*Self-Organizing Maps*), basado en ciertas evidencias descubiertas a nivel cerebral y con un gran potencial de aplicabilidad práctica. Este tipo

de red se caracteriza por poseer un aprendizaje no supervisado competitivo. Vamos a ver en qué consiste este tipo de aprendizaje.

A diferencia de lo que sucede en el aprendizaje supervisado, en el no supervisado (o autoorganizado) no existe ningún maestro externo que indique si la red neuronal está operando correcta o incorrectamente, pues no se dispone de ninguna salida objetivo hacia la cual la red neuronal deba tender. Así, durante el proceso de aprendizaje la red autoorganizada debe descubrir por sí misma rasgos comunes, regularidades, correlaciones o categorías en los datos de entrada, e incorporarlos a su estructura interna de conexiones. Se dice, por tanto, que las neuronas deben autoorganizarse en función de los estímulos (datos) procedentes del exterior.

Dentro del aprendizaje no supervisado existe un grupo de modelos de red caracterizados por poseer un aprendizaje competitivo. En el aprendizaje competitivo las neuronas compiten unas con otras con el fin de llevar a cabo una tarea dada. Con este tipo de aprendizaje, se pretende que cuando se presente a la red un patrón de entrada, sólo una de las neuronas de salida (o un grupo de vecinas) se active. Por tanto, las neuronas compiten por activarse, quedando finalmente una como neurona vencedora y anuladas el resto, que son forzadas a sus valores de respuesta mínimos.

El objetivo de este aprendizaje es categorizar (clusterizar) los datos que se introducen en la red. De esta forma, las informaciones similares son clasificadas formando parte de la misma categoría y, por tanto, deben activar la misma neurona de salida. Las clases o categorías deben ser creadas por la propia red, puesto que se trata de un aprendizaje no supervisado, a través de las correlaciones entre los datos de entrada.

2.1.- Fundamentos biológicos

Se ha observado que en el córtex de los animales superiores aparecen zonas donde las neuronas detectoras de rasgos se encuentran topológicamente ordenadas (Kohonen, 1989, 1990); de forma que las informaciones captadas del entorno a través de los órganos sensoriales, se representan internamente en forma de mapas bidimensionales. Por ejemplo, en el área somatosensorial, las neuronas que reciben señales de sensores que se encuentran próximos en la piel se sitúan también próximas en el córtex, de manera que reproducen --de forma aproximada--, el mapa de la superficie de la piel en

una zona de la corteza cerebral. En el sistema visual se han detectado mapas del espacio visual en zonas del cerebro. Por lo que respecta al sentido del oído, existen en el cerebro áreas que representan mapas tonotópicos, donde los detectores de determinados rasgos relacionados con el tono de un sonido se encuentran ordenados en dos dimensiones (Martín del Brío y Sanz, 1997).

Aunque en gran medida esta organización neuronal está predeterminada genéticamente, es probable que parte de ella se origine mediante el aprendizaje. Esto sugiere, por tanto, que el cerebro podría poseer la capacidad inherente de formar mapas topológicos de las informaciones recibidas del exterior (Kohonen, 1982a).

Por otra parte, también se ha observado que la influencia que una neurona ejerce sobre las demás es función de la distancia entre ellas, siendo muy pequeña cuando están muy alejadas. Así, se ha comprobado que en determinados primates se producen interacciones laterales de tipo excitatorio entre neuronas próximas en un radio de 50 a 100 micras, de tipo inhibitorio en una corona circular de 150 a 400 micras de anchura alrededor del círculo anterior, y de tipo excitatorio muy débil, prácticamente nulo, desde ese punto hasta una distancia de varios centímetros. Este tipo de interacción tiene la forma típica de un sombrero mejicano como veremos más adelante.

En base a este conjunto de evidencias, el modelo de red autoorganizado presentado por Kohonen pretende mimetizar de forma simplificada la capacidad del cerebro de formar mapas topológicos a partir de las señales recibidas del exterior.

2.2.- Arquitectura

Un modelo SOM está compuesto por dos capas de neuronas. La capa de entrada (formada por N neuronas, una por cada variable de entrada) se encarga de recibir y transmitir a la capa de salida la información procedente del exterior. La capa de salida (formada por M neuronas) es la encargada de procesar la información y formar el mapa de rasgos. Normalmente, las neuronas de la capa de salida se organizan en forma de mapa bidimensional como se muestra en la figura 1, aunque a veces también se utilizan capas de una sola dimensión (cadena lineal de neuronas) o de tres dimensiones (paralelepípedo).

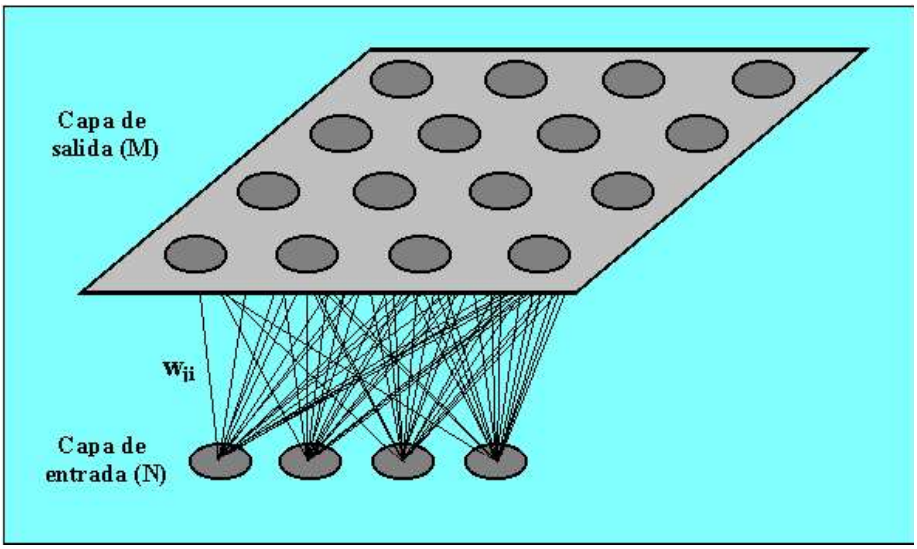

Figura 1: Arquitectura del SOM.

Las conexiones entre las dos capas que forman la red son siempre hacia delante, es decir, la información se propaga desde la capa de entrada hacia la capa de salida. Cada neurona de entrada i está conectada con cada una de las neuronas de salida j mediante un peso w_{ji}. De esta forma, las neuronas de salida tienen asociado un vector de pesos W_j llamado vector de referencia (o *codebook*), debido a que constituye el vector prototipo (o promedio) de la categoría representada por la neurona de salida j.

Entre las neuronas de la capa de salida, puede decirse que existen conexiones laterales de excitación e inhibición implícitas, pues aunque no estén conectadas, cada una de estas neuronas va a tener cierta influencia sobre sus vecinas. Esto se consigue a través de un proceso de competición entre las neuronas y de la aplicación de una función denominada de vecindad como veremos más adelante.

2.3.- Algoritmo

En el algoritmo asociado al modelo SOM podemos considerar, por un lado, una etapa de funcionamiento donde se presenta, ante la red entrenada, un patrón de entrada y éste se asocia a la neurona o categoría cuyo vector de referencia es el más parecido y, por otro lado, una etapa de entrenamiento o aprendizaje donde se organizan las categorías que forman el mapa mediante un proceso no supervisado a partir de las relaciones descubiertas en el conjunto de los datos de entrenamiento.

2.3.1.- Etapa de funcionamiento

Cuando se presenta un patrón p de entrada X_p: $x_{p1},..., x_{pi},...,x_{pN}$, éste se transmite directamente desde la capa de entrada hacia la capa de salida. En esta capa, cada neurona calcula la similitud entre el vector de entrada X_p y su propio vector de pesos W_j o vector de referencia según una cierta medida de distancia o criterio de similitud establecido. A continuación, simulando un proceso competitivo, se declara vencedora la neurona cuyo vector de pesos es el más similar al de entrada.

La siguiente expresión matemática representa cuál de las M neuronas se activará al presentar el patrón de entrada X_p:

$$y_{pj} = \begin{cases} 1 & \min \left\| X_p - W_j \right\| \\ 0 & \text{resto} \end{cases}$$

donde y_{pj} representa la salida o el grado de activación de las neuronas de salida en función del resultado de la competición (1 = neurona vencedora, 0 = neurona no vencedora), $||X_p-W_j||$ representa una medida de similitud entre el vector o patrón de entrada X_p: $x_{p1},..., x_{pi},...,x_{pN}$ y el vector de pesos W_j: $w_{j1},..., w_{ji},...w_{jN}$, de las conexiones entre cada una de las neuronas de entrada y la neurona de salida j. En el siguiente apartado veremos las medidas de similitud más comúnmente utilizadas. En cualquier caso, la neurona vencedora es la que presenta la diferencia mínima.

En esta etapa de funcionamiento, lo que se pretende es encontrar el vector de referencia más parecido al vector de entrada para averiguar qué neurona es la vencedora y, sobre todo, en virtud de las interacciones excitatorias e inhibitorias que existen entre las neuronas, para averiguar en qué zona del espacio bidimensional de salida se encuentra tal neurona. Por tanto, lo que hace la red SOM es realizar una tarea de clasificación, ya que la neurona de salida activada ante una entrada representa la clase a la que pertenece dicha información de entrada. Además, como ante otra entrada parecida se activa la misma neurona de salida, u otra cercana a la anterior, debido a la semejanza entre las clases, se garantiza que las neuronas topológicamente próximas sean sensibles a entradas físicamente similares. Por este motivo, la red es especialmente útil para

establecer relaciones, desconocidas previamente, entre conjuntos de datos (Hilera y Martínez, 1995).

2.3.2.- Etapa de aprendizaje

Se debe advertir, en primer lugar, que no existe un algoritmo de aprendizaje totalmente estándar para la red SOM. Sin embargo, se trata de un procedimiento bastante robusto ya que el resultado final es en gran medida independiente de los detalles de su realización concreta. En consecuencia, trataremos de exponer el algoritmo más habitual asociado a este modelo (Kohonen, 1982a, 1982b, 1989, 1995).

El algoritmo de aprendizaje trata de establecer, mediante la presentación de un conjunto de patrones de entrenamiento, las diferentes categorías (una por neurona de salida) que servirán durante la etapa de funcionamiento para realizar clasificaciones de nuevos patrones de entrada.

De forma simplificada, el proceso de aprendizaje se desarrolla de la siguiente manera. Una vez presentado y procesado un vector de entrada, se establece a partir de una medida de similitud, la neurona vencedora, esto es, la neurona de salida cuyo vector de pesos es el más parecido respecto al vector de entrada. A continuación, el vector de pesos asociado a la neurona vencedora se modifica de manera que se parezca un poco más al vector de entrada. De este modo, ante el mismo patrón de entrada, dicha neurona responderá en el futuro todavía con más intensidad. El proceso se repite para un conjunto de patrones de entrada los cuales son presentados repetidamente a la red, de forma que al final los diferentes vectores de pesos sintonizan con uno o varios patrones de entrada y, por tanto, con dominios específicos del espacio de entrada. Si dicho espacio está dividido en grupos, cada neurona se especializará en uno de ellos, y la operación esencial de la red se podrá interpretar como un análisis de clusters.

La siguiente interpretación geométrica (Masters, 1993) del proceso de aprendizaje puede resultar interesante para comprender la operación de la red SOM. El efecto de la regla de aprendizaje no es otro que acercar de forma iterativa el vector de pesos de la neurona de mayor actividad (ganadora) al vector de entrada. Así, en cada iteración el vector de pesos de la neurona vencedora rota hacia el de entrada, y se aproxima a él en una cantidad que depende del tamaño de una tasa de aprendizaje.

En la figura 2 se muestra cómo opera la regla de aprendizaje para el caso de varios patrones pertenecientes a un espacio de entrada de dos dimensiones, representados en la figura por los vectores de color negro. Supongamos que los vectores del espacio de entrada se agrupan en tres clusters, y supongamos que el número de neuronas de la red es también tres. Al principio del entrenamiento los vectores de pesos de las tres neuronas (representados por vectores de color rojo) son aleatorios y se distribuyen por la circunferencia. Conforme avanza el aprendizaje, éstos se van acercando progresivamente a las muestras procedentes del espacio de entrada, para quedar finalmente estabilizados como centroides de los tres clusters.

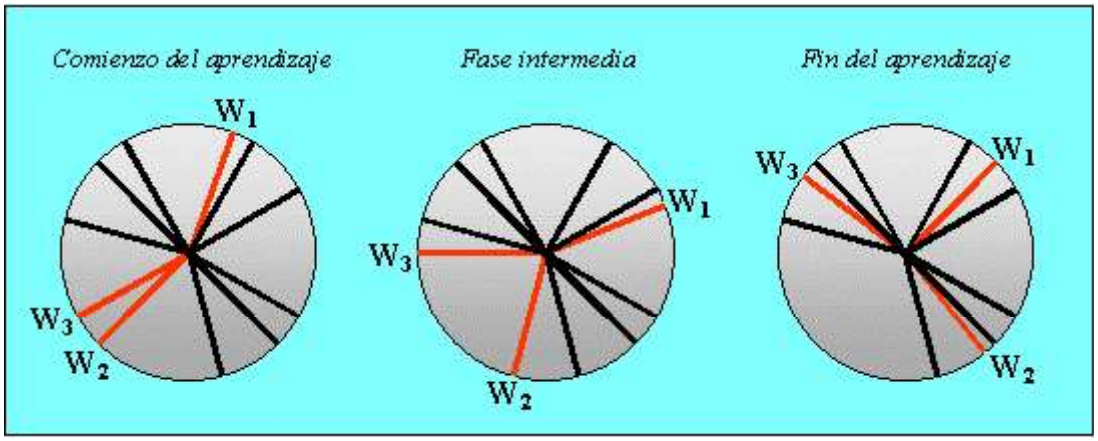

Figura 2: Proceso de aprendizaje en dos dimensiones.

Al finalizar el aprendizaje, el vector de referencia de cada neurona de salida se corresponderá con el vector de entrada que consigue activar la neurona correspondiente. En el caso de existir más patrones de entrenamiento que neuronas de salida, como en el ejemplo expuesto, más de un patrón deberá asociarse con la misma neurona, es decir, pertenecerán a la misma clase. En tal caso, los pesos que componen el vector de referencia se obtienen como un promedio (centroide) de dichos patrones.

Además de este esquema de aprendizaje competitivo, el modelo SOM aporta una importante novedad, pues incorpora relaciones entre las neuronas próximas en el mapa. Para ello, introduce una función denominada zona de vecindad que define un entorno alrededor de la neurona ganadora actual (vecindad); su efecto es que durante el aprendizaje se actualizan tanto los pesos de la vencedora como los de las neuronas pertenecientes a su vecindad. De esta manera, en el modelo SOM se logra que neuronas

próximas sintonicen con patrones similares, quedando de esta manera reflejada sobre el mapa una cierta imagen del orden topológico presente en el espacio de entrada.

Una vez entendida la forma general de aprendizaje del modelo SOM, vamos a expresar este proceso de forma matemática. Recordemos que cuando se presenta un patrón de entrenamiento, se debe identificar la neurona de salida vencedora, esto es, la neurona cuyo vector de pesos sea el más parecido al patrón presentado. Un criterio de similitud muy utilizado es la distancia euclídea que viene dado por la siguiente expresión:

$$\min \left\| X_p - W_j \right\| = \min \sum_{i=1}^{N} (x_{pi} - w_{ji})^2$$

De acuerdo con este criterio, dos vectores serán más similares cuanto menor sea su distancia.

Una medida de similitud alternativa más simple que la euclídea, es la correlación o producto escalar:

$$\min \left\| X_p - W_j \right\| = \max \sum_{i=1}^{N} x_{pi} \cdot w_{ji}$$

según la cual, dos vectores serán más similares cuanto mayor sea su correlación.

Identificada la neurona vencedora mediante el criterio de similitud, podemos pasar a modificar su vector de pesos asociado y el de sus neuronas vecinas, según la regla de aprendizaje:

$$\Delta w_{ji}(n+1) = \alpha(n)\left(x_{pi} - w_{j*i}(n)\right) \quad \text{para } j \in \text{Zona}_{j*}(n)$$

donde n hace referencia al número de ciclos o iteraciones, esto es, el número de veces que ha sido presentado y procesado todo el juego de patrones de entrenamiento. $\alpha(n)$ es la tasa de aprendizaje que, con un valor inicial entre 0 y 1, decrece con el número de iteraciones (n) del proceso de aprendizaje. $\text{Zona}_{j*}(n)$ es la zona de vecindad alrededor de la neurona vencedora j* en la que se encuentran las neuronas cuyos pesos son

actualizados. Al igual que la tasa de aprendizaje, el tamaño de esta zona normalmente se va reduciendo paulatinamente en cada iteración, con lo que el conjunto de neuronas que pueden considerarse vecinas cada vez es menor.

Tradicionalmente el ajuste de los pesos se realiza después de presentar cada vez un patrón de entrenamiento, como se muestra en la regla de aprendizaje expuesta. Sin embargo, hay autores (Masters, 1993) que recomiendan acumular los incrementos calculados para cada patrón de entrenamiento y, una vez presentados todos los patrones, actualizar los pesos a partir del promedio de incrementos acumulados. Mediante este procedimiento se evita que la dirección del vector de pesos vaya oscilando de un patrón a otro y acelera la convergencia de los pesos de la red.

En el proceso general de aprendizaje suelen considerarse dos fases. En la primera fase, se pretende organizar los vectores de pesos en el mapa. Para ello, se comienza con una tasa de aprendizaje y un tamaño de vecindad grandes, para luego ir reduciendo su valor a medida que avanza el aprendizaje. En la segunda fase, se persigue el ajuste fino del mapa, de modo que los vectores de pesos se ajusten más a los vectores de entrenamiento. El proceso es similar al anterior aunque suele ser más largo, tomando la tasa de aprendizaje constante e igual a un pequeño valor (por ejemplo, 0.01) y un radio de vecindad constante e igual a 1.

No existe un criterio objetivo acerca del número total de iteraciones necesarias para realizar un buen entrenamiento del modelo. Sin embargo, el número de iteraciones debería ser proporcional al número de neuronas del mapa (a más neuronas, son necesarias más iteraciones) e independiente del número de variables de entrada. Aunque 500 iteraciones por neurona es una cifra adecuada, de 50 a 100 suelen ser suficientes para la mayor parte de los problemas (Kohonen, 1990).

3.- Fases en la aplicación de los mapas autoorganizados

En el presente apartado, pasamos a describir las diferentes fases necesarias para la aplicación de los mapas autoorganizados a un problema típico de agrupamiento de patrones.

3.1.- Inicialización de los pesos

Cuando un mapa autoorganizado es diseñado por primera vez, se deben asignar valores a los pesos a partir de los cuales comenzar la etapa de entrenamiento. En general, no existe discusión en este punto y los pesos se inicializan con pequeños valores aleatorios, por ejemplo, entre -1 y 1 ó entre 0 y 1 (Kohonen, 1990), aunque también se pueden inicializar con valores nulos (Martín del Brío y Serrano, 1993) o a partir de una selección aleatoria de patrones de entrenamiento (SPSS Inc., 1997).

3.2.- Entrenamiento de la red

Vista la manera de modificar los vectores de pesos de las neuronas a partir del conjunto de entrenamiento, se van a proporcionar una serie de consejos prácticos acerca de tres parámetros relacionados con el aprendizaje cuyos valores óptimos no pueden conocerse *a priori* dado un problema.

3.2.1.- Medida de similitud

En el apartado 2.3.2. hemos visto las dos medidas de similitud más ampliamente utilizadas a la hora de establecer la neurona vencedora ante la presentación de un patrón de entrada, tanto en la etapa de funcionamiento como en la etapa de aprendizaje de la red. Sin embargo, se debe advertir que el criterio de similitud y la regla de aprendizaje que se utilicen en el algoritmo deben ser métricamente compatibles. Si esto no es así, estaríamos utilizando diferentes métricas para la identificación de la neurona vencedora y para la modificación del vector de pesos asociado, lo que podría causar problemas en el desarrollo del mapa (Demartines y Blayo, 1992).

La distancia euclídea y la regla de aprendizaje presentada son métricamente compatibles y, por tanto, no hay problema. Sin embargo, la correlación o producto escalar y la regla de aprendizaje presentada no son compatibles, ya que dicha regla procede de la métrica euclídea, y la correlación solamente es compatible con esta métrica si se utilizan vectores normalizados (en cuyo caso distancia euclídea y correlación coinciden). Por tanto, si utilizamos la correlación como criterio de similitud, deberíamos utilizar vectores normalizados; mientras que si utilizamos la distancia euclídea, ésto no será

necesario (Martín del Brío y Sanz, 1997). Finalmente, independientemente del criterio de similitud utilizado, se recomienda que el rango de posibles valores de las variables de entrada sea el mismo, por ejemplo, entre -1 y 1 ó entre 0 y 1 (Masters, 1993).

3.2.2.- Tasa de aprendizaje

Como ya se ha comentado, $\alpha(n)$ es la tasa de aprendizaje que determina la magnitud del cambio en los pesos ante la presentación de un patrón de entrada. La tasa de aprendizaje, con un valor inicial entre 0 y 1, por ejemplo, 0.6, decrece con el número de iteraciones (n), de forma que cuando se ha presentado un gran número de veces todo el juego de patrones de aprendizaje, su valor es prácticamente nulo, con lo que la modificación de los pesos es insignificante. Normalmente, la actualización de este parámetro se realiza mediante una de las siguientes funciones (Hilera y Martínez, 1995):

$$\alpha(n) = \frac{1}{n} \qquad \alpha(n) = \alpha_1\left(1 - \frac{n}{\alpha_2}\right)$$

Siendo α_1 un valor de 0.1 ó 0.2 y α_2 un valor próximo al número total de iteraciones del aprendizaje. Suele tomarse un valor $\alpha_2 = 10000$.

El empleo de una u otra función no influye en exceso en el resultado final.

3.2.3.- Zona de vecindad

La zona de vecindad ($Zona_{j^*}(n)$) es una función que define en cada iteración n si una neurona de salida pertenece o no a la vecindad de la vencedora j*. La vecindad es simétrica y centrada en j*, pudiendo adoptar una forma circular, cuadrada, hexagonal o cualquier otro polígono regular.

En general, $Zona_{j^*}(n)$ decrece a medida que avanza el aprendizaje y depende de un parámetro denominado radio de vecindad R(n), que representa el tamaño de la vecindad actual.

La función de vecindad más simple y utilizada es la de tipo escalón. En este caso, una

neurona j pertenece a la vecindad de la ganadora j* solamente si su distancia es inferior o igual a R(n). Con este tipo de función, las vecindades adquieren una forma (cuadrada, circular, hexagonal, etc.) de bordes nítidos, en torno a la vencedora (figura 3); por lo que en cada iteración únicamente se actualizan las neuronas que distan de la vencedora menos o igual a R(n).

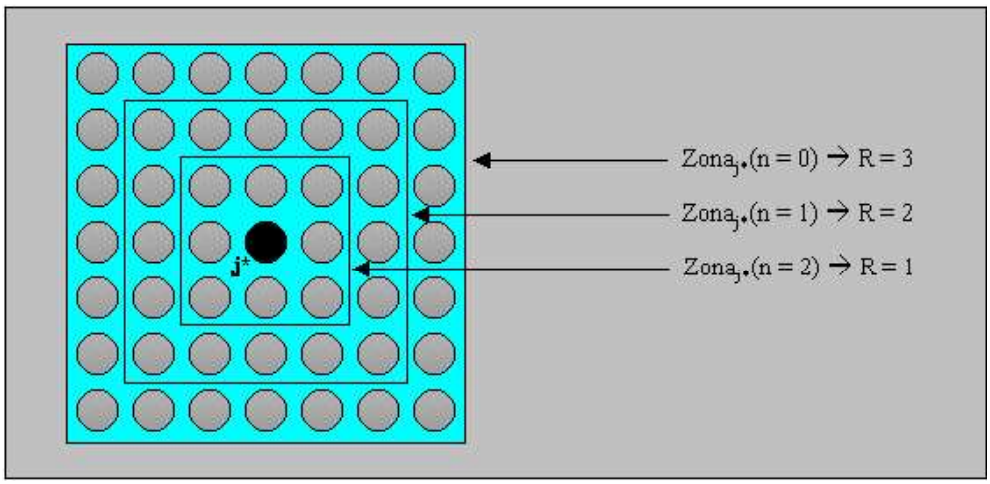

Figura 3: Posible evolución de la zona de vecindad.

También se utilizan a veces funciones gaussianas o en forma de sombrero mejicano (figura 4), continuas y derivables en todos sus puntos, que al delimitar vecindades decrecientes en el dominio espacial establecen niveles de pertenencia en lugar de fronteras nítidas.

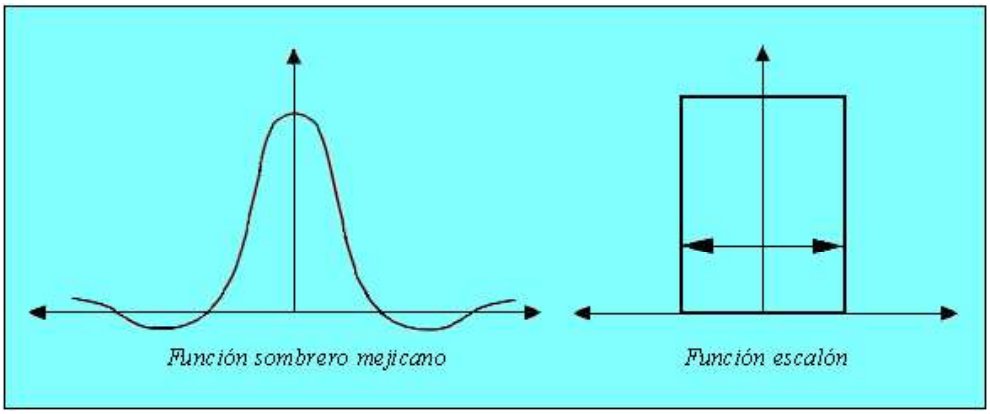

Figura 4: Formas de la función de vecindad.

La función en forma de sombrero mejicano se basa en el tipo de interacción que se produce entre ciertas neuronas del córtex comentado al inicio del documento. Con esta

función, una neurona central emite señales excitatorias a una pequeña vecindad situada a su alrededor. A medida que aumenta la distancia lateral desde la neurona central, el grado de excitación disminuye hasta convertirse en una señal inhibitoria. Finalmente, cuando la distancia es considerablemente grande la neurona central emite una débil señal excitatoria. Por su parte, la función escalón supone una simplificación de la función en forma de sombrero mejicano y, como hemos visto, define de forma discreta la vecindad de neuronas que participan en el aprendizaje.

La zona de vecindad posee una forma definida, pero como hemos visto, su radio varía con el tiempo. Se parte de un valor inicial R_0 grande, por ejemplo, igual al diametro total del mapa (SOM_PAK, 1996; Koski, Alanen, Komu et al., 1996), que determina vecindades amplias, con el fin de lograr la ordenación global del mapa. R(n) disminuye monótonamente con el tiempo, hasta alcanzar un valor final de $R_f = 1$, por el que solamente se actualizan los pesos de la neurona vencedora y las adyacentes. Una posible función de actualización de R(n) es la siguiente (Martín del Brío y Sanz, 1997):

$$R(n) = R_0 + (R_f - R_0)\frac{n}{n_R}$$

donde n es la iteración y n_R el número de iteraciones para alcanzar R_f.

3.3.- Evaluación del ajuste del mapa

En los mapas autoorganizados, el conjunto de vectores de pesos finales va a depender entre otros factores, del valor de los pesos aleatorios iniciales, el valor de la tasa de aprendizaje, el tipo de función de vecindad utilizado y la tasa de reducción de estos dos últimos parámetros. Como es obvio, debe existir un mapa óptimo que represente de forma fiel las relaciones existentes entre el conjunto de patrones de entrenamiento. El mapa más adecuado será aquel cuyos vectores de pesos se ajusten más al conjunto de vectores de entrenamiento. Esto se puede operativizar mediante el cálculo del error cuanficador promedio a partir de la media de $|| X_p - W_{j*}||$, esto es, la media de la diferencia (por ejemplo, la distancia euclídea) entre cada vector de entrenamiento y el vector de pesos asociado a su neurona vencedora (SOM_PAK, 1996). La expresión del error cuantificador promedio utilizada en nuestras simulaciones es la siguiente:

$$\text{Error}_{\text{medio}} = \frac{\sum_{p=1}^{P}\sum_{i=1}^{N}(x_{pi} - w_{j*i})^2}{P}$$

Por tanto, con el objeto de obtener un mapa lo más adecuado posible, deberíamos comenzar el entrenamiento en múltiples ocasiones, cada vez utilizando una configuración de parámetros de aprendizaje diferentes. Así, el mapa que obtenga el error cuantificador promedio más bajo será el seleccionado para pasar a la fase de funcionamiento normal de la red.

3.4.- Visualización y funcionamiento del mapa

Una vez seleccionado el mapa óptimo, podemos pasar a la fase de visualización observando en qué coordenadas del mapa se encuentra la neurona asociada a cada patrón de entrenamiento. Esto nos permite proyectar el espacio multidimensional de entrada en un mapa bidimensional y, en virtud de la similitud entre las neuronas vecinas, observar los clusters o agrupaciones de datos organizados por la propia red. Por este motivo, el modelo de mapa autoorganizado es especialmente útil para establecer relaciones, desconocidas previamente, entre conjuntos de datos.

En la fase de funcionamiento, la red puede actuar como un clasificador de patrones ya que la neurona de salida activada ante una entrada nueva representa la clase a la que pertenece dicha información de entrada. Además, como ante otra entrada parecida se activa la misma neurona de salida, u otra cercana a la anterior, debido a la semejanza entre las clases, se garantiza que las neuronas topológicamente próximas sean sensibles a entradas físicamente similares.

3.5.- Análisis de sensibilidad

Una de las críticas más importantes que se han lanzado contra el uso de RNA trata sobre lo difícil que es comprender la naturaleza de las representaciones internas generadas por la red para responder ante un determinado patrón de entrada. A diferencia de los modelos estadísticos clásicos, no es tan evidente conocer en una red la importancia (o relación) que tiene cada variable de entrada sobre la salida del modelo. Sin embargo,

esta percepción acerca de las RNA como una compleja "caja negra", no es del todo cierta. De hecho, han surgido diferentes intentos por interpretar los pesos de la red neuronal (Garson, 1991; Zurada, Malinowski y Cloete, 1994; Rambhia, Glenny y Hwang, 1999; Hunter, Kennedy, Henry et al., 2000), de los cuales el más ampliamente utilizado es el denominado análisis de sensibilidad.

El análisis de sensibilidad está basado en la medición del efecto que se observa en una salida y_j debido al cambio que se produce en una entrada x_i. Así, cuanto mayor efecto se observe sobre la salida, mayor sensibilidad podemos deducir que presenta respecto a la entrada. En la mayoría de casos, este tipo de análisis ha sido aplicado a la red *backpropagation* (Palmer, Montaño y Calafat, 2000; Palmer, Montaño y Jiménez, en prensa), sin embargo, apenas se han realizado estudios que apliquen el análisis de sensibilidad a los modelos SOM.

En este sentido, Hollmén y Simula (1996) investigaron el efecto de pequeños cambios realizados en una de las variables de entrada sobre la salida de un modelo SOM. Siguiendo un proceso parecido al que se suele aplicar a la red *backpropagation*, estos autores iban realizando pequeños cambios a lo largo de uno de los ejes definido por una variable de entrada (las demás variables se fijaban a un valor promedio) y observando cómo la neurona vencedora iba cambiando de ubicación a lo largo del mapa. Este procedimiento permitió analizar el grado de relación o importancia que tenía cada variable de entrada sobre la salida de la red.

4.- Un ejemplo: Clasificación de la planta del Iris

Como ejemplo ilustrativo vamos a utilizar la matriz de datos sobre el conocido problema de la clasificación de la planta del Iris. Esta matriz proporciona los datos referentes a una muestra de 150 plantas. Cada ejemplar consta de cuatro características y la tarea consiste en determinar el tipo de planta del Iris en base a esas características. Las caracterísitcas son: longitud del sépalo, ancho del sépalo, longitud del pétalo y ancho del pétalo. Hay una submuestra de 50 ejemplares para cada tipo de planta del Iris: Setosa, Versicolor y Virgínica.

El lector interesado puede bajarse esta matriz de datos junto a otras matrices muy conocidas en el ámbito del reconocimiento de patrones (discriminación del cáncer de

mama, detección de cardiopatías, problema OR-Exclusiva, reconocimiento de imágenes via satélite, etc.) en la dirección de Universal Problem Sorvers.

A partir de esta matriz de datos, nos propusimos averiguar si un modelo SOM era capaz de agrupar en el mapa los tres tipos de planta, proporcionándole únicamente los datos sobre las cuatro características citadas. Por tanto, las categorías deberían ser creadas de forma no supervisada por la propia red a través de las correlaciones descubiertas entre los datos de entrada, puesto que no se proporciona la categoría de pertenencia del ejemplar.

Comenzamos con el preprocesamiento de los datos reescalando los cuatro parámetros que iban a servir como variables de entrada a la red. Para ello, se acotó el rango de las variables a valores comprendidos entre 0 y 1. No fue necesario normalizar los vectores de entrada debido a que se utilizaría como criterio de similitud, la distancia euclídea en la etapa de entrenamiento. Como salida de la red se determinó un mapa bidimensional 10x10, por tanto, el mapa estaría compuesto por 100 neuronas de salida.

Se entrenó un total de 10 modelos, cada uno con una configuración inicial de pesos diferente (con rango comprendido entre 0 y 1), pero siguiendo todos el mismo esquema de aprendizaje. Así, el entrenamiento de cada mapa se organizó en dos fases. En la primera fase, cuyo objetivo consiste en organizar el mapa, se utilizó una tasa de aprendizaje alta igual a 1 y un radio de vecindad grande igual al diámetro del mapa, es decir, igual a 10. A medida que avanzaba el aprendizaje, tanto la tasa de aprendizaje como el radio de vecindad iban reduciéndose de forma lineal hasta alcanzar unos valores mínimos, 0.05 y 1, respectivamente. En la segunda fase, cuyo objetivo es el ajuste fino del mapa, se utilizó una tasa de aprendizaje pequeña y constante igual a 0.05, y un radio de vecindad constante y mínimo igual a 1. La primera fase constó de 1000 iteraciones, mientras que la segunda fase constó de 2000 iteraciones.

Una vez entrenados los 10 modelos, se calculó para cada uno de ellos, el error cuantificador promedio. Quedó seleccionado el modelo cuyo error fue el más pequeño --con un valor igual a 0.0156.

A continuación, se muestra el mapa del modelo finalmente seleccionado (figura 5):

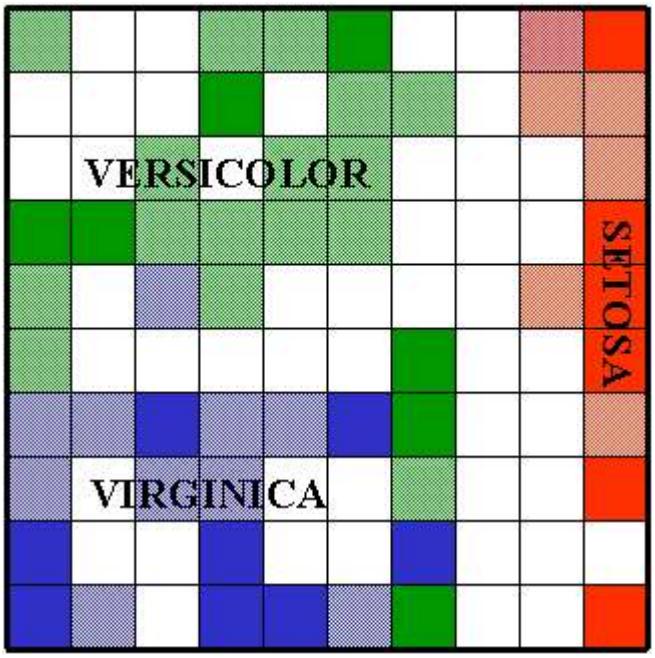

Figura 5: Mapa del modelo seleccionado.

En la figura, las neuronas de salida que representan ejemplares de planta Versicolor aparecen de color verde, las neuronas que representan ejemplares de planta Virgínica aparecen de color azul y, finalmente, las neuronas que representan ejemplares de planta Setosa aparecen de color rojo. Las neuronas de color blanco no representan patrón alguno. Con el objeto de poder analizar la distribución de frecuencias de cada clase de planta en el mapa, cada color puede aparecer con dos posibles intensidades según la frecuencia de patrones asociados a una determinada neurona de salida. Así, cuando el color es poco intenso, hay pocos patrones (1 ó 2) asociados a la neurona; cuando el color es intenso, hay muchos patrones (de 3 a 10) asociados a la neurona.

Se puede observar que los ejemplares de planta Setosa han sido perfectamente discriminados respecto a las otras dos categorías, quedando agrupados todos en la parte derecha del mapa. Por su parte, los ejemplares de Versicolor y Virgínica se reparten la parte superior e inferior izquierda del mapa, respectivamente; compartiendo zonas adyacentes. Esto parece indicar que la planta Setosa se diferencia perfectamente de los otros dos tipos de planta, mientras que la planta Versicolor y Virgínica mantienen características más similares, aunque bien diferenciables.

A continuación, podríamos pasar a la etapa de funcionamiento de la red donde se presentarían nuevos ejemplares y, mediante el proceso de competición, podríamos

observar en que zona del mapa está ubicada la neurona de salida vencedora asociada a cada nuevo ejemplar.

5.- Recursos gratuitos en internet sobre los mapas autoorganizados

En la Web podemos encontrar multitud de recursos relacionados con el campo de las RNA. A modo de ejemplos ilustrativos, describiremos el funcionamiento de dos applets y un programa totalmente gratuitos que permiten simular el comportamiento de un mapa autoorganizado de Kohonen tal como ha sido expuesto a lo largo de este documento.

5.1.- Applets

Se han seleccionado dos ejemplos de applets, creados por Jussi Hynninen del Laboratorio Tecnológico de Acústica y Procesamiento de Señales de Audio de la Universidad de Helsinki (Finlandia) y colaborador de Kohonen. Uno simula un mapa autoorganizado unidimensional, mientras que el otro simula un mapa autoorganizado bidimensional. El lector interesado en este tipo de recursos puede visitar la página del Instituto Nacional de Biociencia y Tecnología Humana (M.I.T.I.) de Japón, donde podrá encontrar un numeroso listado de applets demostrativos sobre RNA.

5.1.1.- Ejemplo de mapa autoorganizado unidimensional

En el primer applet (figura 6) se representa un mapa unidimensional mediante una fila de cinco barras verticales en forma de termómetro. Estas barras representan el valor (un valor escalar) de las neuronas de salida del mapa y están controladas por el algoritmo del modelo SOM. Cuando se presentan patrones de entrada aleatorios consistentes en un valor escalar, los valores de las barras se van organizando gradualmente. Durante el entrenamiento, la diferencia entre el nuevo y el anterior valor de una barra se muestra de color amarillo después de la presentación de un patrón. La barra cuyo valor es el más cercano al patrón de entrada (es decir, la barra que representa la neurona vencedora), aparece de color rojo. El tamaño de la zona de vecindad es constante y con un radio igual a 1, es decir, solo se modifica el valor de la neurona vencedora y el de sus adyacentes. La barra de color rosado situada en el extremo izquierdo es la que controla

el patrón de entrada, ya que muestra el valor de la entrada presentada al mapa en cada paso del entrenamiento.

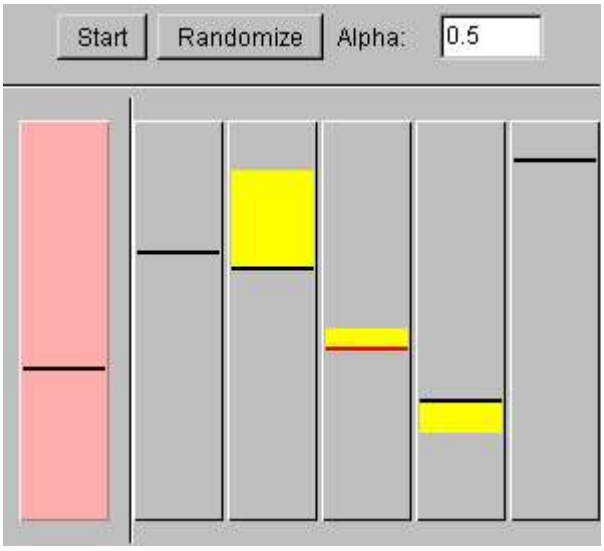

Figura 6: Applet demostrativo de un mapa unidimensional.

El mapa puede ser entrenado de forma automática o manual. Durante el entrenamiento automático, se presentan patrones de entrada aleatorios. En el entrenamiento manual el usuario puede especificar el patrón de entrada que será utilizado en el aprendizaje, simplemente haciendo click con el ratón en el controlador de entrada (la barra de color rosado). Presionando la tecla ESPACIO, se realizará un ciclo de aprendizaje con un patrón de entrada aleatorio.

Usando el botón START/STOP situado en el applet, comenzará o parará el entrenamiento automático. Por su parte, el botón RANDOMIZE inicializará los valores de las neuronas con valores aleatorios. Se puede usar el campo de texto *Alpha* para determinar el valor de la tasa de aprendizaje, escribiendo el valor y luego pulsando la tecla ENTER. Finalmente, el valor de las neuronas del mapa también se pueden modificar haciendo click con el ratón sobre las barras verticales.

5.1.2.- Ejemplo de mapa autoorganizado bidimensional

En el segundo applet se representa un mapa bidimensional (figura 7) formado por un rectángulo de 4x5 neuronas.

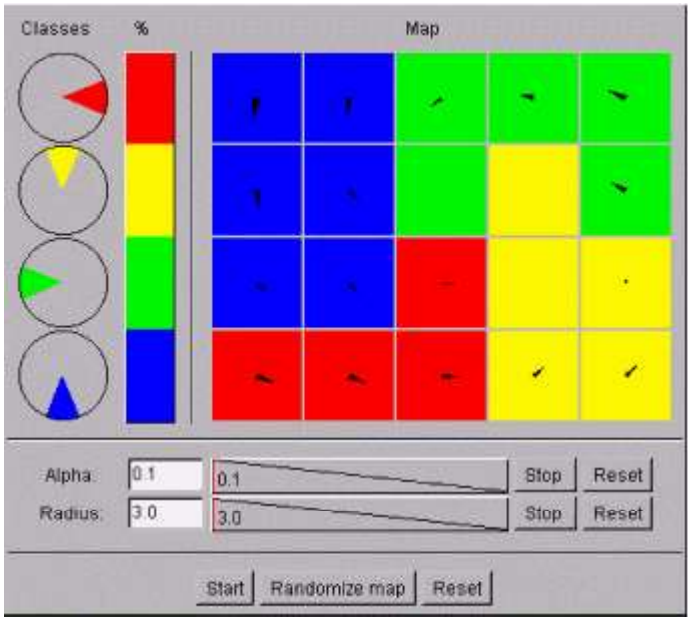

Figura 7: Applet demostrativo de un mapa bidimensional.

En este applet, los cuatro gráficos de "pastel" situados en la parte izquierda definen las clases o categorías de entrada. Hay cuatro clases representadas por los colores rojo, amarillo, verde y azul. En cada gráfico, el sector dibujado de color representa la distribución de vectores (formados por dos valores: x y) que la clase produce. Los cuatro gráficos proporcionan vectores aleatorios que se distribuyen en el sector mostrado en el gráfico. Los gráficos puede ser editados con el ratón de dos maneras diferentes: haciendo click cerca del centro del gráfico podemos cambiar la ubicación del sector, haciendo click cerca del borde del gráfico podemos cambiar la anchura del sector.

La barra vertical situada a la derecha de los gráficos de "pastel" determina las proporciones relativas de las cuatro clases respecto a los datos de entrada. Inicialmente los vectores de entrada se distribuyen de forma igualitaria entre las cuatro clases. El usuario puede cambiar las proporciones con el ratón: haciendo click en el borde que delimita dos clases y, a continuación, subiendo o bajando el cursor.

El área de la derecha del applet corresponde al mapa autoorganizado. El mapa consiste en 20 neuronas organizadas en un rectángulo de 4x5 neuronas. Durante el entrenamiento, el color de fondo de las neuronas corresponde al color de la clase del patrón de entrada más próximo.

A cada neurona del mapa le corresponde un vector de referencia de dos dimensiones cuyos componentes se interpretan como dos coordenadas, x e y. Cada unidad del mapa se representa mediante una flecha que apunta desde las coordenadas [0, 0] a las coordenadas [x, y] almacenadas en el vector de referencia. El rango de las coordenadas oscila entre -1 y 1.

Debajo de los gráficos de "pastel" y del mapa está el panel de control que determina el valor de la tasa de aprendizaje (alfa) y el radio de vecindad. Ambos parámetros funcionan de la misma forma: en la parte izquierda se sitúa el campo de texto donde el usuario puede introducir el valor que desee del parámetro y, a continuación, apretar la tecla ENTER. Por defecto, el valor de la tasa de aprendizaje alfa se reduce durante el aprendizaje desde el valor inicial hasta el valor 0 y el valor del radio de vecindad decrece desde el valor inicial hasta el valor 1. El usuario puede determinar un valor constante para estos parámetros, pulsando la tecla STOP situada a la derecha del panel que controla el parámetro correspondiente. Para que sigua decreciendo, pulse otra vez el botón (leerá START). Pulsando el botón RESET reinicia el valor del parámetro a su valor inicial.

Finalmente, el panel de control situado en la parte inferior del applet se utiliza para iniciar y parar el aprendizaje, y también para iniciar con valores aleatorios los vectores de pesos del mapa. Pulse el botón START para comenzar el aprendizaje del mapa, pulse otra vez para parar. Pulsando el botón RANDOMIZE MAP inicializa el mapa con valores aleatorios. El botón RESET permite reiniciar los parámetros tasa de aprendizaje alfa y radio de vecindad.

5.2.- Software

En la Web podemos encontrar multitud de programas simuladores de redes neuronales de libre distribución (gratuitos o *sharewares*). El lector interesado puede visitar dos listados completos sobre este tipo de programas: Por un lado, tenemos el listado ofrecido por el grupo de noticias sobre redes neuronales comp.ai.neural-nets, por otro lado, tenemos el listado ofrecido por el Pacific Northwest National Laboratory.

5.2.1.- SOM_PAK

De entre los simuladores de libre distribución que podemos encontrar en internet, cabe destacar el programa SOM_PAK (para UNIX y MS-DOS), software de dominio público desarrollado en la Universidad de Tecnología de Helsinki (Finlandia) por el grupo de Kohonen para el trabajo y experimentación con el modelo SOM.

Para bajarse el programa SOM_PAK (versión 3.1), pulse el siguiente botón:

En este mismo enlace, se puede obtener el manual de usuario del programa (SOM_PAK, 1996), por tanto, remitimos al lector la lectura del manual para una descripción del funcionamiento del programa. Este sencillo manual proporciona numerosos consejos prácticos y, mediante un sencillo ejemplo, el usuario puede seguir la secuencia de pasos necesarios (inicialización, entrenamiento, evaluación y visualización del mapa) para la construcción de un modelo SOM.

Referencias bibliográficas

Arbib, M.A., Erdi, P. y Szentagothai, J. (1997). *Neural organization: structure, function and dynamics*. Cambridge, Mass.: MIT Press.

Demartines, P. y Blayo, F. (1992). Kohonen self-organizing maps: Is the normalization necessary?. *Complex Systems, 6*, 105-123.

Garson, G.D. (1991). Interpreting neural-network connection weights. *AI Expert*, April, 47-51.

Hilera, J.R. y Martínez, V.J. (1995). *Redes neuronales artificiales: Fundamentos, modelos y aplicaciones*. Madrid: Ra-Ma.

Hollmén, J. y Simula, O. (1996). Prediction models and sensitivity analysis of industrial process parameters by using the self-organizing map. *Proceedings of IEEE Nordic Signal Processing Symposium (NORSIG'96)*, 79-82.

Hunter, A., Kennedy, L., Henry, J. y Ferguson, I. (2000). Application of neural networks and sensitivity analysis to improved prediction of trauma survival. *Computer Methods and Programs in Biomedicine, 62*, 11-19.

Kohonen, T. (1982a). Self-organized formation of topologically correct feature maps. *Biological Cybernetics, 43*, 59-69.

Kohonen, T. (1982b). Analysis of a simple self-organizing process. *Biological Cybernetics, 44*, 135-140.

Kohonen, T. (1989). *Self-organization and associative memory*. New York: Springer-Verlag.

Kohonen, T. (1990). The self-organizing map. *Proceedings of the IEEE, 78*(9), 1464-1480.

Kohonen, T. (1995). *Self-organizing maps*. Berlin: Springer-Verlag.

Koski, A., Alanen, A., Komu, M., Nyman, S., Heinänen, K. y Forsström, J. (1996). Visualizing nuclear magnetic resonance spectra with self-organizing neural networks. En: Taylor, J.G. (Ed.). *Neural networks and their applications* (pp. 83-92). Chichester: John Wiley and Sons.

Martín del Brío, B. y Sanz, A. (1997). *Redes neuronales y sistemas borrosos*. Madrid: Ra-Ma.

Martín del Brío, B. y Serrano, C. (1993). Self-organizing neural networks for analysis and representation of data: some financial cases. *Neural Computing and Applications, 1*, 193-206.

Masters, T. (1993). *Practical neural networks recipes in C++*. London: Academic Press.

Palmer, A., Montaño, J.J. y Calafat, A. (2000). Predicción del consumo de éxtasis a partir de redes neuronales artificiales. *Adicciones, 12*(1), 29-41.

Palmer, A., Montaño, J.J. y Jiménez, R. (en prensa). Tutorial sobre redes neuronales artificiales: el perceptrón multicapa. *Internet y Salud*, URL: http://www.intersalud.net

Rambhia, A.H., Glenny, R. y Hwang, J. (1999). Critical input data channels selection for progressive work exercise test by neural network sensitivity analysis. *IEEE International Conference on Acoustics, Speech, and Signal Processing*, 1097-1100.

Rumelhart, D.E., Hinton, G.E. y Williams, R.J. (1986). Learning internal representations by error propagation. En: D.E. Rumelhart y J.L. McClelland (Eds.). *Parallel distributed processing* (pp. 318-362). Cambridge, MA: MIT Press.

Sarle, W.S. (Ed.) (1998). *Neural network FAQ*. Periodic posting to the Usenet newsgroup comp.ai.neural-nets, URL: ftp://ftp.sas.com/pub/neural/FAQ.html.

SOM_PAK (1996). *The Self-Organizing Map Program Package: User's Guide* [Manual de programa para ordenadores]. Helsinki University of Technology, Finland.

SPSS Inc. (1997). *Neural Connection 2.0: User´s Guide* [Manual de programa para ordenadores]. Chicago: SPSS Inc.

Taylor, J.G. (1996). *Neural networks and their applications*. Chichester: John Wiley and Sons.

Zurada, J.M., Malinowski, A. y Cloete, I. (1994). Sensitivity analysis for minimization of input data dimension for feedforward neural network. *Proceedings of IEEE International Symposium on Circuits and Systems*, 447-450.