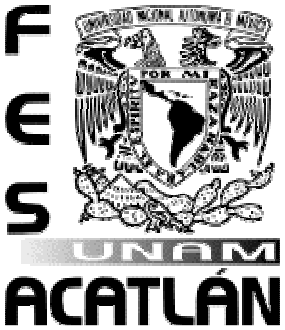

UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE ESTUDIOS SUPERIORES ACATLÁN

LAS REDES NEURONALES ARTIFICIALES COMO UNA HERRAMIENTA ALTERNATIVA O COMPLEMENTARIA A LOS MÉTODOS DE CLASIFICACIÓN TRADICIONALES

T E S I N A

QUE PARA OBTENER EL TÍTULO DE

A C T U A R I A

P R E S E N T A

ALMA ROSA AGUILAR SÁNCHEZ

ASESOR: VÍCTOR MANUEL ULLOA ARELLANO

AGOSTO, 2008

Contenido

Prólogo

Hasta ahora, los métodos de clasificación más ampliamente utilizados tanto con fines didácticos como para la solución de problemas reales son, entre otros, el análisis *cluster* en cualquiera de sus variantes, los árboles de decisión y el análisis discriminante. A diferencia del análisis discriminante, los dos primeros métodos (análisis *cluster* y árboles de decisión) funcionan tanto para fines de agrupación como de clasificación; mientras que el análisis discriminante tiene básicamente fines clasificatorios.

Cuando se desea encontrar una propuesta de solución a un problema de clasificación se recomienda evaluar diferentes alternativas, es decir, obtener soluciones mediante diferentes técnicas y con diversos escenarios (por ejemplo en el análisis *cluster* puede optarse por los métodos jerárquicos o no jerárquicos) con la finalidad de probar la consistencia de las soluciones o bien, elegir la que mejor se adecue al problema planteado.

El objetivo de este trabajo es mostrar que las redes neuronales artificiales (específicamente, los mapas autoorganizados) pueden ser utilizadas como un método alternativo o complementario a los métodos de clasificación tradicionales.

Si se puede mostrar que para un mismo problema la solución obtenida mediante redes neuronales artificiales es semejante o incluso superior a la solución generada por los métodos tradicionales se podrá entonces contar con la opción de utilizar la solución de este método (mapas autoorganizados) como un resultado válido o bien, si se prefiere, complementarla con otros métodos si se desea una solución más robusta.

Para mostrar una posible semejanza entre métodos, se realizará la comparación de los resultados arrojados por cada uno identificando las ventajas y desventajas así como sus principales similitudes y diferencias desde la etapa de implementación hasta la interpretación de los resultados generados.

En el caso de los métodos tradicionales se contemplará únicamente el análisis *cluster* y los árboles de decisión debido a que se realiza primero una agrupación (la cual el análisis discriminante no realiza) seguida de una clasificación para determinar el margen de error de cada método.

La teoría presentada se enfocará en las redes neuronales artificiales, en el caso del análisis *cluster* y los árboles de decisión se mostrará un anexo estadístico indicando también las fuentes en donde puede profundizarse la teoría sobre estos métodos.

El trabajo consta de 3 capítulos; en el primero se presentan los fundamentos de las redes neuronales artificiales así como un breve repaso a la historia del desarrollo de sistemas y máquinas dotadas de cierta inteligencia y su paralelismo con los sistemas neuronales biológicos.

Se exponen también las características generales de los modelos neuronales artificiales y las motivaciones que desembocan en la introducción de éstas como alternativa o complemento a los sistemas de procesamiento más utilizados en la actualidad.

Una vez expuestos los conceptos básicos relacionados con las redes neuronales artificiales, se da paso al capítulo 2 centrado en el modelo de mapas autoorganizados, uno de los sistemas neuronales no supervisados más conocidos y utilizados. Se presenta el modelo general de mapas autoorganizados así como algunos de sus algoritmos de aprendizaje.

Finalmente, en el capítulo 3 se muestran las soluciones generadas por cada uno de los tres métodos expuestos, se resaltan las ventajas y desventajas de cada uno de tal forma que el lector pueda decidirse sobre uno de ellos o bien, optar por una combinación para robustecer la solución generada.

Introducción

El cerebro es un procesador de información con características muy notables. Es capaz de procesar inmediatamente grandes cantidades de información procedentes de los sentidos, combinarla o compararla con la información almacenada y dar respuestas adecuadas incluso en situaciones nuevas. Logra discernir un susurro en una sala ruidosa, distinguir una cara en una calle mal iluminada o leer entre líneas en una conversación; pero lo más impresionante de todo es su capacidad de aprender a representar la información necesaria para desarrollar tales habilidades sin instrucciones explícitas para ello.

Aunque todavía se ignora mucho sobre la forma en que el cerebro aprende a procesar la información, se han desarrollado modelos que tratan de mimetizar tales habilidades; a este tipo de modelos se les conoce como Redes Neuronales Artificiales. La elaboración de estos modelos supone en primer lugar la deducción de los rasgos o características esenciales de las neuronas y sus conexiones, y en segundo lugar, la implementación del modelo computacional de forma que se pueda simular.

Existen varios tipos de modelos de Redes Neuronales Artificiales, cada uno con un algoritmo y fin específico. Los mapas autoorganizados o mapas de Kohonen (nombre de su desarrollador) se basan en el hecho de que en algunas partes del cerebro, las neuronas, están interconectadas siguiendo una misma estructura, el lugar en donde las conexiones de las neuronas forman una estructura u organización es el reflejo del entorno sensitivo.

En el cerebro podemos encontrarnos mapas topológicos de los órganos sensoriales de nuestro cuerpo. En determinadas zonas del cerebro humano se ha encontrado experimentalmente que las neuronas detectoras de rasgos se encuentran topológicamente ordenadas. Ante un estímulo proveniente de sensores de la piel próximos entre sí, se estimulan neuronas del cerebro pertenecientes a una misma zona.

Estos mapas se organizan de manera autónoma sin una referencia por medio de la cual se puedan corregir errores; pues el cerebro tiene la capacidad suficiente para clasificar la información nueva sin tener una referencia anterior en la cual pueda apoyarse. Así, el cerebro se organiza de manera automática, o dicho de otra manera, se autoorganiza.

La vida nos proporciona abundantes ejemplos de la autoorganización; cuando los alumnos asisten a un curso, el primer día se sientan en las sillas de forma aleatoria, conforme pasan los días se recolocan en el aula, de forma que conforme pasa el

tiempo se sientan juntos según sus afinidades. A menudo hay grupos exclusivamente formados por chicas o chicos, el grupo de alumnos que se sientan en las últimas filas, etc.

Algunos problemas reales en los que los mapas autoorganizados han demostrado su eficacia incluyen tareas de clasificación, reducción de dimensiones y extracción de rasgos. Su utilidad más importante se relaciona con la clasificación de información o el agrupamiento de patrones. Este modelo neuronal utiliza una estrategia de aprendizaje que los humanos utilizamos frecuentemente, el llamado aprendizaje no supervisado.

Si el aprendizaje supervisado se asemeja al profesor que enseña y corrige al alumno, el aprendizaje no supervisado o autoorganizado es semejante al alumno que aprende por sí mismo, sin la ayuda de un profesor, pero disponiendo de material docente, libros, etc.

Para generar un mejor entendimiento en cuanto a los fines que la autoorganización persigue, conviene revisar los conceptos de agrupación y clasificación que en algunas ocasiones son interpretados (erróneamente) como lo mismo.

El agrupamiento o *clustering* trabaja a partir de una serie de observaciones y determina si existen clases en la que dichas observaciones puedan ser agrupadas. Es decir, determina la existencia de clases en las cuales poder agrupar. El número y características de las clases son desconocidos *a priori* (aprendizaje no supervisado).

Por su parte, la clasificación trabaja a partir del conocimiento de la existencia de un conjunto de clases y determina la regla para asignar cada nueva observación (o ejemplo) a la clase que pertenece. Es decir, determina reglas de asignación a clases conocidas (aprendizaje supervisado).

Al realizar un agrupamiento, al término de éste podemos generar una regla de clasificación que permita continuar categorizando la nueva información en las clases ahora ya existentes.

Capítulo 1

1.- Fundamentos de las Redes Neuronales Artificiales

En este capítulo se realizará un breve repaso a la historia del desarrollo de sistemas y máquinas dotadas de cierta inteligencia así como su paralelismo con los sistemas neuronales biológicos. Se expondrán las características generales de los modelos neuronales artificiales y las motivaciones que desembocan en la introducción de éstas como alternativa o complemento a los sistemas de procesamiento más utilizados en la actualidad.

1.1 Introducción

Muchos de los desarrollos del hombre se deben a su capacidad para explicar y emular funciones que son realizadas por seres vivos. Por ejemplo, se puede citar el radar, que surge como imitación de la forma en la que un murciélago es capaz de detectar los objetos que están en su camino sin necesidad de verlos, por medio de la emisión de una onda ultrasónica, la posterior recepción de la señal de eco y su procesamiento.

Aunque el hombre ha sido capaz de reproducir funciones de los animales, aún se enfrenta con el reto de poder imitar, la llamada por muchos la máquina perfecta: el cerebro humano.

Cuando la neurociencia pudo explicar de forma un poco convincente el funcionamiento de la unidad principal de procesamiento de información que posee el cerebro, la neurona, surge casi de manera automática la idea de poder imitar dicho funcionamiento en un elemento artificial, "la neurona artificial".

Una de las metodologías con mayor auge en la última década son los modelos de Redes Neuronales Artificiales (RNA)[1], que en esencia son estructuras formales de

[1] En Hilera, J. y Martínez, V. (1995). *Redes Neuronales Artificiales: Fundamentos, Modelos y aplicaciones.* Rama. Madrid, se define una red neuronal como una nueva forma de computación, inspirada en modelos biológicos.

carácter matemático y estadístico con la propiedad de aprendizaje, es decir, la adquisición de conocimientos que en la mayoría de los casos es a partir de ejemplos.

Este aprendizaje se produce mediante un estilo de computación que intenta simular algunas de las capacidades que posee nuestro cerebro: la capacidad de memorizar y asociar hechos. Si examinamos con atención aquellos problemas que no pueden expresarse a través de un algoritmo, nos daremos cuenta de que todos ellos tienen una característica común: la experiencia.

En definitiva, las redes neuronales artificiales no son más que un modelo artificial y simplificado del cerebro humano, es decir, un sistema para el tratamiento de la información, que es capaz de adquirir conocimiento a través de la experiencia y cuya unidad básica de procesamiento está inspirada en la célula fundamental del sistema nervioso humano, la neurona.

Los modelos de redes neuronales son variados, al menos 50 diferentes tipos han sido explorados en investigación o han sido desarrollados para aplicaciones. Se muestran los principales de uso común en la **Tabla 1.1.**

Red	Año	Aplicaciones más importantes	Comentarios	Inventada/ desarrollada por
Perceptrón	1957	Reconocimiento de caracteres impresos.	La red más antigua.	Frank Rosenblatt.
Adaline / Madaline	1960	Filtrado de señales. Ecualizador adaptativo. Módems.	Rápida, fácil de implementar.	Bernard Widrow.
Avalancha	1967	Reconocimiento de habla. Control de brazos de robot.	Ninguna red sencilla puede hacer todo esto.	Stephen Grossberg.
Cerebellatron	1969	Control de movimiento de los brazos de un robot.	Semejante a Avalancha.	David Marr, James Albus, Andres Pellionez.
Back Propagation	1974-85	Síntesis de voz desde texto. Control de robots. Predicción. Reconocimiento de patrones.	Red más popular. Numerosas aplicaciones con éxito. Facilidad de aprendizaje. Potente.	Paul Werbos, David Parker, David Rumelhart.
Brain-Estate-in-a-Box	1977	Extracción de conocimiento de bases de datos.	Posiblemente mejor realización que las redes de Hopfield.	James Anderson.
Neocognitrón	1978-84	Reconocimiento de caracteres manuscritos.	Insensible a la translación, rotación y escala.	K. Fukushima.
Self-Organizing-Map (SOM). T*opology-Preserving- Map* (TPM)	1980-84	Reconocimiento de patrones, codificación de datos, optimización.	Realiza mapas de características comunes de los datos aprendidos.	Teuvo Kohonen.
Hopfield	1982	Reconstrucción de patrones y optimización.	Fácil de conceptualizar.	John Hopfield.
Memoria Asociativa Bidireccional	1985	Memoria heteroasociativa de acceso por contenido.	Aprendizaje y arquitectura simples.	Bart Kosko.
Máquinas de Boltzmann y Cauchy	1985-86	Reconocimiento de patrones (imágenes, sonar y radar). Optimización.	Redes simples. Capacidad de representación óptima de patrones.	Jeffrey Hinton, Terry Sejnowski, Harold Szu.
Teoría de la Resonancia Adaptativa (ART)	1986	Reconocimiento de patrones (radar, sonar, etc.)	Sofisticada. Poco utilizada.	Gail Carpenter, Stephen Grossberg.
Counter- propagation	1986	Tratamiento de imágenes.	Combinación de Perceptrón y TPM.	Robert Hecht-Nielsen.

Tabla 1.1. Modelos de RNA más conocidos (Fuente: Adaptada de Hetch- Nielsen, 1998)

1.2 Breve introducción biológica

Antes de abordar el estudio de las RNA es conveniente exponer algunos conceptos básicos de los sistemas neuronales biológicos, para poder establecer más fácilmente el paralelismo entre ambos[2].

La historia de las redes neuronales artificiales comienza con el científico Santiago Ramón y Cajal, un gran neuroanatomista español descubridor de la estructura neuronal del sistema nervioso. A finales del siglo XIX la teoría reticularista[3], que sostenía que el sistema nervioso estaba formado por una red continua de fibras nerviosas, era la creencia extendida.

Sin embargo en 1988, tras años de trabajo, Ramón y Cajal demostró que el sistema nervioso en realidad estaba compuesto por una red de células individuales, las neuronas, ampliamente interconectadas entre sí. Pero no sólo observó al microscopio los pequeños espacios vacíos que separaban unas neuronas de otras, sino que también estableció que la información fluye de una neurona a otra desde las dendritas hasta el axón, atravesando el soma, descubrimiento básico para el desarrollo de las neurociencias en el siglo XX.

Se estima que el sistema nervioso contiene alrededor de cien mil millones de neuronas, este tipo de células puede presentarse en múltiples formas, aunque muchas de ellas tienen un aspecto similar muy peculiar **(Figura 1.1.)**, con un cuerpo celular o soma (de entre 10 y 80 micras de longitud), del que surge un denso árbol de ramificaciones (árbol dendrítico) compuesto por las dendritas, y del cual parte una fibra tubular denominada axón que también se ramifica en su extremo final para conectarse con otras neuronas.

---

[2] Se trata de una visión simplificada (válida para nuestro propósito) del funcionamiento del cerebro, sin embargo se puede mostrar cómo el sencillo modelo de neurona artificial que se emplea en las RNA puede derivarse de modelos más complejos, que reflejan más fielmente la realidad biológica. Para un mejor entendimiento del funcionamiento del cerebro se recomienda leer la obra de Eccles, J.C. (1973). *The understanding of the Brain*. McGraw-Hill.

[3] Varios investigadores defendieron la concepción reticularista del sistema nervioso. Comparaban la estructura de la sustancia gris cerebral y de otros territorios nerviosos a un complejo retículo formado por la fusión de las prolongaciones de las células nerviosas. Esta teoría implicaba negar la individualidad de estas células. Cajal desmintió la teoría reticularista gracias a sus investigaciones, demostró que las relaciones entre las células nerviosas o neuronas eran de contigüidad y no de continuidad, dejando firmemente establecido que la neurona es la unidad histológica y fisiológica del sistema nervioso. Esta es la llamada Teoría Neuronal, por la que se le concedió el premio Nobel en 1906.

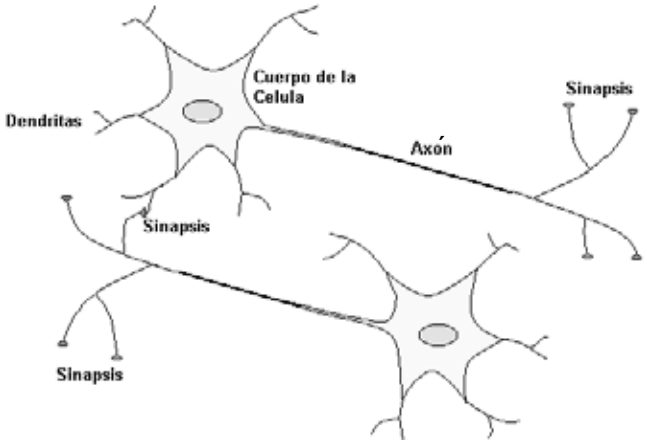

Figura 1.1. Estructura de una neurona biológica típica.

Desde un punto de vista funcional, las neuronas constituyen procesadores de información sencillos. Como todo sistema de este tipo, poseen un canal de entrada de información, las dendritas, un órgano de cómputo, el soma, y un canal de salida, el axón[4] . En el espacio inter neuronal el axón envía la información a otras neuronas, mientras que en las neuronas motoras lo hace directamente al músculo.

Existe un tercer tipo de neuronas, las receptoras o sensoras, que en vez de recibir la información de otras neuronas, la reciben directamente del exterior. Se calcula que una neurona del córtex cerebral recibe información, por término medio, de unas $10,000$ neuronas (convergencia), y envía impulsos a varios cientos de ellas (divergencias).

En el córtex cerebral se aprecia la existencia de una organización horizontal en capas (se suelen señalar unas seis capas), coexistiendo una organización vertical en forma de columnas de neuronas. Hay grupos neuronales, compuestos por millones de neuronas pertenecientes a una determinada región del cerebro, que constituyen unidades funcionales especializadas en ciertas tareas (por ejemplo, existe un área visual, un área auditiva, etc.). Se tiene evidencia de que el procesamiento en el sistema nervioso involucra la actuación de muchos subsistemas, que intercambian continuamente información.

[4] En realidad, en el árbol dendrítico también se lleva a cabo un cierto procesamiento; por otra parte, el soma también puede recibir información directamente de otros axones, sin la mediación de las dendritas.

1.3 Generación y transmisión de la señal nerviosa

La unión entre dos neuronas se denomina sinapsis[5]. En el tipo de sinapsis más común no existe un contacto físico entre las neuronas, sino que éstas permanecen separadas por un pequeño vacío de unas 0.2 micras. En relación a la sinapsis, se habla de neuronas presinápticas (que envían las señales) y postsinápticas (que las reciben).

Las señales nerviosas se pueden transmitir eléctrica o químicamente. La transmisión química prevalece fuera de la neurona y se basa en el intercambio de neurotransmisores, mientras que la eléctrica lo hace en el interior mediante descargas que se producen en el cuerpo celular, y que se propagan por el axón.

La forma de comunicación más habitual entre dos neuronas es del tipo químico. La neurona presináptica libera unas complejas sustancias químicas denominadas neurotransmisores (como la adrenalina), que atraviesan el vació sináptico. Si la neurona postsináptica posee en las dendritas o en el soma canales sensibles a los neurotransmisores liberados, los fijarán, y como consecuencia de ello permitirán el paso de determinados iones a través de la membrana. Las corrientes iónicas que de esta manera se crean provocan pequeños potenciales postsinápticos, excitadores o inhibidores, que se integrarán en el soma; éste es el origen de la existencia de sinapsis excitatorias y de sinapsis inhibitorias[6].

Ante un estímulo mayor la frecuencia de respuesta aumenta, hasta que se alcanza una saturación conforme nos acercamos a la frecuencia máxima (umbral). La señal que es recibida por la neurona posee diferentes grados de ponderación; cuando la ponderación es alta, el potencial de membrana de la neurona se vuelve positivo, rebasa el umbral y envía la correspondiente señal a la siguiente neurona del circuito; contrariamente cuando la ponderación es baja, el umbral no se rebasa y no se transmite señal a las otras neuronas.

Para establecer una similitud directa entre la actividad sináptica y la analogía con las RNA, vamos a fijar los siguientes aspectos: las señales que llegan a la sinapsis son las entradas a la neurona; éstas son ponderadas (atenuadas o amplificadas) a través de un parámetro, denominado peso sináptico, asociado a la sinapsis correspondiente.

[5] La palabra sinapsis viene del griego "synapto" que significa unión o conexión estrecha.
[6] Existen evidencias experimentales que indican que un axón sólo puede generar sinapsis excitatorias o inhibitorias, pero no de ambos tipos (Ley de Dale).

Estas señales de entrada pueden excitar a la neurona (sinapsis con peso positivo) o inhibirla (peso negativo). El efecto es la suma de las entradas ponderadas. Si la suma es igual o mayor que el umbral de la neurona, entonces la neurona se activa (da salida). Esta es una situación todo o nada, es decir, cada neurona se activa o no se activa.

1.4 Procesos plásticos en la sinapsis

Se ha observado que la conectividad entre dos células puede modificarse en función de la experiencia. A este tipo de procesos se les denomina procesos plásticos en la sinapsis o simplemente plasticidad sináptica.

Para nuestro estudio de los procesos plásticos vamos a clasificarlos en procesos presinápticos, si los cambios ocurren en la presinapsis; postsinápticos, cuando los cambios ocurren en las postsinapsis; transinápticos, cuando los cambios ocurren por la acción concentrada entre la pre y la postsinapsis. Asimismo, haremos la división entre los procesos homosinápticos, que sólo ocurren por la acción de una sola sinapsis, y los heterosinápticos, cuando los cambios plásticos ocurren como resultado de la interacción de varias sinapsis. A los procesos heterosinápticos podríamos a su vez clasificarlos como procesos *cooperativos* y de *competencia*[7]. Por último, definiremos que los cambios plásticos pueden ser a corto y a largo plazo.

Los procesos plásticos más ampliamente estudiados son:

1) **Habituación.** Consiste en una disminución en la cantidad de transmisor liberado como resultado de la experiencia. Es un proceso homosináptico-presináptico, que tiene componentes de corto y largo plazo.

2) **Potenciación postetánica.** Es un incremento en la cantidad de transmisor liberado como resultado de la actividad repetitiva de una vía, es un proceso homosináptico-presináptico, con componentes de corto y largo plazo.

3) **Sensibilización.** Es el incremento de la cantidad de transmisor de una vía por la acción de otra neurona, es un proceso presináptico-heterosináptico cooperativo con componentes de corto y largo plazo.

[7] Este tipo de procesos son los que se utilizan en los mapas autoorganizados.

4) **Inhibición heterosináptica.** Es la disminución de la cantidad de transmisor de una vía por la acción de otra neurona, es un proceso presináptico-heterosináptico competitivo, con componentes de corto y largo plazo.

5) **Condicionamiento.** Es la capacidad que se tiene para modificar la conducta en base a la asociación de dos estímulos, es un proceso presináptico-heterosináptico cooperativo de corto y largo plazo.

1.5 Sistemas paralelos, distribuidos y adaptativos

Los tres conceptos clave de los sistemas nerviosos que se pretende emular en los artificiales, son: paralelismo de cálculo, memoria distribuida y adaptabilidad al entorno. De esta manera, podemos hablar de las RNA como sistemas paralelos, distribuidos y adaptativos.

1.5.1 Procesamiento paralelo

El cerebro tarda aproximadamente $20\,ms$ en preprocesar una imagen compuesta por millones de píxeles, extraer sus características, analizarla e interpretarla. Ningún sistema creado por el hombre es capaz de realizar algo semejante. La clave reside en que en este último caso los miles de millones de neuronas que intervienen en el proceso de visión están operando en paralelo sobre la totalidad de la imagen.

1.5.2 Memoria distribuida

Mientras que en un procesador la información ocupa posiciones de memoria bien definidas, en las RNA se encuentra distribuida por las sinapsis de la red, de modo que si una sinapsis resulta dañada, se pierde sólo una parte muy pequeña de la información.

Además, los sistemas neuronales biológicos son redundantes, de modo que muchas neuronas y sinapsis pueden realizar un papel similar; en definitiva, el sistema resulta tolerante a fallos. Por ejemplo, cada día mueren miles de neuronas en nuestro cerebro, y sin embargo tienen que pasar muchos años para que se resientan nuestras

capacidades. La razón por la que las RNA son tolerantes a fallos es que tienen su información distribuida.

Las RNA son los primeros métodos computacionales con la capacidad inherente de tolerancia a fallos. Hay dos aspectos distintos respecto a la tolerancia a fallos: primero, las redes pueden aprender a reconocer patrones con ruido, distorsionados o incompletos, ésta es una tolerancia a fallos respecto a los datos. Segundo, pueden seguir realizando su función (con cierta degradación) aunque se destruya parte de la red.

1.5.3 Aprendizaje adaptativo

Es una de las características más atractivas de las RNA. Esto es, aprenden a llevar a cabo ciertas tareas mediante un entrenamiento con ejemplos ilustrativos, es decir, no es necesario elaborar modelos *a priori* ni especificar funciones de distribución de probabilidad. Una RNA no necesita un algoritmo para resolver un problema, ya que ella puede generar su propia distribución de los pesos de los enlaces mediante aprendizaje. Las RNA resuelven problemas mediante autoaprendizaje y autoorganización.

1.6 Características generales de los modelos neuronales artificiales

Los modelos neuronales asumen muchas simplificaciones del modelo biológico para poder plantear su desarrollo matemático, en esta línea, el primer modelo artificial fue diseñado por McCulloch-Pitts (1943) **(Figura 1.2.)**, el cual utilizaba unidades de procesamiento denominadas neuronas que poseían dos estados discretos. Asociados a cada uno de ellos, se conseguía una salida que se transmitía a lo largo de la estructura vinculada a la red neuronal, pero con la limitación que sólo permitían computar funciones booleanas.

El modelo de McCulloch-Pitts se basa en las siguientes hipótesis. En primer lugar, el estado de una neurona en el tiempo, "$t+1$", depende solamente del estado que poseía en el período anterior, "t".

En segundo lugar, una neurona estará activada o no si supera un umbral θ, y en último lugar, se asume la sincronía[8] entre las entradas y las salidas.

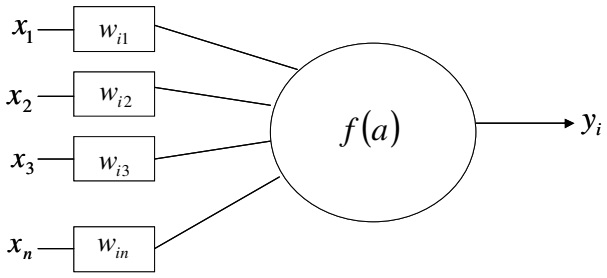

Figura 1.2. Modelo de neurona McCulloch-Pitts

La formalización del diseño del modelo de McCulloch-Pitts consiste, en primer lugar, en definir el estado de la entrada, "x_t" y en segundo lugar, la salida en el momento t, "y_t". La expresión que describe su funcionamiento es,

$$y_i = f\left(\sum_{j=1}^{n} w_{ij}\, x_j - \theta_i\right) = f(a)$$

$$f(a) = \begin{Bmatrix} 1\ si\ a \geq 0 \\ 0 \quad e.o.c \end{Bmatrix}$$

En los modelos neuronales la información se genera a partir del aprendizaje de la estructura interna de los datos, de forma que son las propias conexiones o pesos donde se retiene el conocimiento. Es de gran importancia notar que no existe *a priori* una definición explícita de la forma del conocimiento, el propio algoritmo iterativo de estimación de los parámetros (pesos) desconocidos, se encarga de extraer la presencia de regularidades en los datos.

[8] La dinámica que rige la actualización de los estados de las neuronas (evolución de la red neuronal) puede ser de dos tipos: asincrónico y sincrónico. En el primer caso, las neuronas evalúan su estado continuamente, según les va llegando información, y lo hacen de forma independiente. En el caso sincrónico aunque la información llega de forma continua, los cambios se realizan simultáneamente. Los sistemas neuronales biológicos muy probablemente actúan de forma mixta.

Los aspectos de mayor relevancia en los modelos neuronales son, primeramente, su arquitectura o topología[9], en segundo lugar, el tipo de sus unidades de procesamiento, en tercer lugar, el tipo de conexiones de estas unidades o neuronas, y en cuarto lugar, los tipos de aprendizaje.

El primer aspecto, la arquitectura de una red neuronal, se refiere a la forma de las conexiones entre las unidades neuronales. Su forma genera toda una familia de posibles modelos, cuya gran variedad obliga a la vertebración de los mismos mediante clasificaciones o taxonomías.

En una primera aproximación, podemos encontrar una clasificación en función a los tipos de las salidas que genera el modelo, divididos en: modelos deterministas y modelos estocásticos. Para el caso determinista tenemos que cada neurona sigue una ley del tipo,

$$y = f\left(\sum_{i=1}^{n} w_i \, x_i\right)$$

donde $f(\cdot)$ es una función de activación[10], en cambio para las redes con neuronas estocásticas, la activación de la red se interpreta como una probabilidad de un estado lógico tal y como se expresa en las siguientes ecuaciones,

$$P(y=1) = f\left(\sum_{i=1}^{n} w_i \, x_i\right)$$

y

$$P(y=0) = 1 - P(y=1) = 1 - f\left(\sum_{i=1}^{n} w_i \, x_i\right)$$

donde la salida es un valor continuo entre $[0,1]$ que se interpreta como una probabilidad.

[9] La topología de la red (forma de la red) muestra cómo los diferentes nodos están conectados entre sí, y la forma en cómo se comunican.
[10] Se definirá más a detalle en la siguiente sección.

El segundo aspecto es la tipología existente en las unidades de procesamiento o neuronas. Existen neuronas visibles y neuronas ocultas. Por neuronas visibles se entienden tanto las entradas (variables exógenas) como las salidas (variables endógenas), en cambio las neuronas ocultas, poseen la función de capturar la representación interna de los datos. Éstas pueden no estar conectadas directamente con las neuronas visibles **(Figura 1.3.)**.

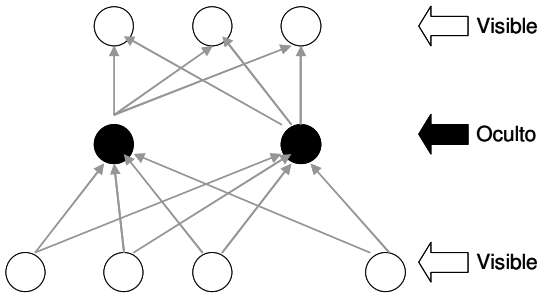

Figura 1.3. Tipología de las unidades de procesamiento de neuronas

El tercer aspecto descansa en el tipo de conexiones que se establecen entre las unidades de procesamiento o neuronas. Así tenemos, en primer lugar, los modelos que se propagan en una sola dirección, denominados *feed-forward* y en segundo lugar, los modelos recurrentes, cuyas conexiones se establecen en todas las direcciones incluso con procesos de realimentación, es decir, las propias neuronas consigo mismas.

El cuarto aspecto hace referencia a los tipos de aprendizaje. Existen dos tipos de aprendizaje; supervisado y no supervisado. La diferencia fundamental entre ambos tipos estriba en la existencia o no de un agente externo (supervisor o maestro) que controle el proceso de aprendizaje de la red.

1.7 Modelo general de neurona artificial

Los elementos básicos de un modelo de neurona artificial son **(Figura 1.4.)**:

1. **Un conjunto de entradas,** $x_j(t)$**.**
2. **Los pesos sinápticos de la neurona** i**,** w_{ij}**.**
3. **La regla de propagación,** $h_i(t)=\sigma(w_{ij}, x_j(t))$**.**
4. **La función de activación o transferencia,** $g_i(h_i(t))$**.**
5. Y finalmente la **función de salida,** $f_i(g_i(h_i(t)))$**.**

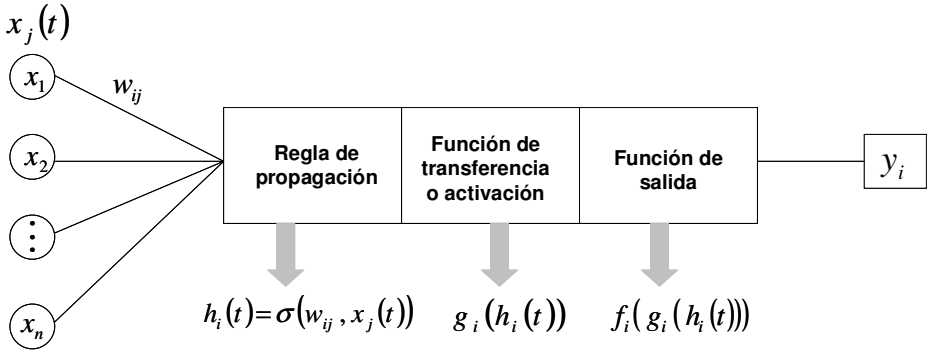

Figura 1.4. Modelo genérico de neurona artificial

Los elementos descritos posibilitarán una generalización formal, que permitirá aglutinar tanto los modelos supervisados como los no supervisados. A continuación se definen cada uno de estos elementos:

1.7.1 Conjunto de entradas

El conjunto de entradas $x_j(t)$ se refiere a un vector de entradas procedentes del exterior o de otras neuronas, es decir, son las señales que llegan a la sinapsis.

1.7.2 Pesos sinápticos

Los pesos sinápticos de la neurona i, w_{ij} representan la intensidad de interacción entre cada neurona presináptica j y la neurona postsináptica i, al igual que en una neurona biológica se establecen sinapsis entre las dendritas de una neurona y el axón de otra. Si el peso es positivo tenderá a excitar a la neurona postsináptica, si es negativo tenderá a inhibirla.

1.7.3 La regla de propagación

La regla de propagación $h_i(t)=\sigma(w_{ij}, x_j(t))$, también conocida como función de ponderación o de excitación, proporciona el valor del potencial postsináptico de la neurona i en función de sus pesos y entradas. La regla de propagación es un elemento relevante que puede poseer diferentes formas, en la **Figura 1.5.** se muestran algunas de ellas.

La regla de propagación habitual, especialmente en los modelos basados en el cálculo de distancias entre vectores (como en los mapas autoorganizados) es la distancia euclídea que representa la distancia (al cuadrado) existente entre el vector de entradas y el de pesos. Cuando ambos vectores son muy similares, la distancia es muy pequeña; cuando son muy diferentes, la distancia crece. Se pueden utilizar también otros tipos de distancias, como la Manhattan o la de Mahalanobis.

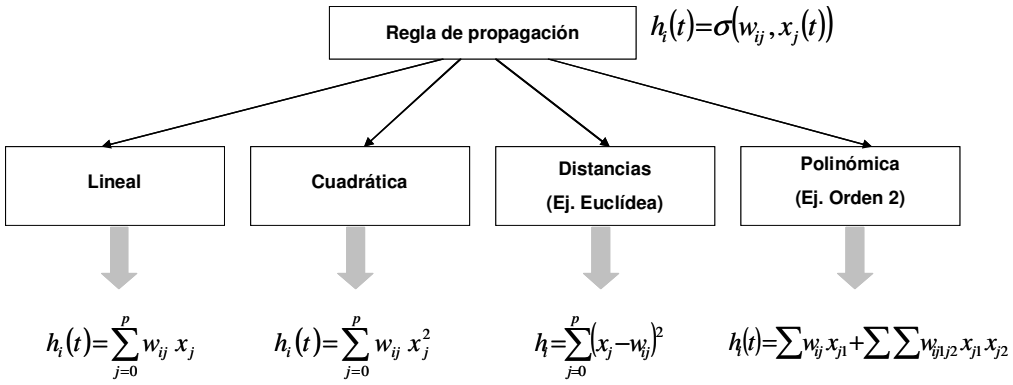

Figura 1.5. Ejemplo de reglas de propagación

1.7.4 La función de transferencia o activación

La función de activación o transferencia, $g_i(h_i(t))$ filtra el valor de la regla de propagación para comparase con algún valor umbral para determinar la salida final de la neurona. Si la suma es mayor que el valor umbral, la neurona generará una señal. Si la suma es menor que el valor umbral, ninguna señal será generada. La función de activación puede o no existir, siendo en este caso la salida la misma función de propagación.

En ocasiones los algoritmos de aprendizaje requieren que la función de activación cumpla con la condición de ser derivable. Las más empleadas en este sentido son las del tipo sigmoideo, como la del *back propagation*. Otra función clásica es la gaussiana, que se utiliza junto con reglas de propagación que involucran el cálculo de cuadrados de distancias (por ejemplo, la euclídea) entre los vectores de entradas y pesos. Por último, en ocasiones se emplean funciones sinusoidales, como en aquellos casos en los que se requiere expresar explícitamente una periodicidad temporal. La **Figura 1.6.** muestra las principales funciones de activación.

Función identidad o lineal $y_i = g\left(\sum_{j=0}^{p} w_{ij}\, x_j\right)$ $g(a) \equiv a$	Función escalón $y_i = g\left(\sum_{j=0}^{p} w_{ij}\, x_j\right)$ $g(a) = \begin{Bmatrix} 1 & a \geq 0 \\ 0 & a < 0 \end{Bmatrix}$
Función lineal a tramos $y_i = g\left(\sum_{j=0}^{p} w_{ij}\, x_j\right)$ $g(a) = \begin{Bmatrix} 0 & a < 0 \\ a & 0 \leq a < 1 \\ 1 & a > 0 \end{Bmatrix}$	Función escalón simétrica $y_i = g\left(\sum_{j=0}^{p} w_{ij}\, x_j\right)$ $g(a) = \begin{Bmatrix} 1 & a \geq 0 \\ -1 & a < 0 \end{Bmatrix}$
Función lineal a tramos simétrica $y_i = g\left(\sum_{j=0}^{p} w_{ij}\, x_j\right)$ $g(a) = \begin{Bmatrix} -1 & a < -1 \\ a & -1 \leq a < 1 \\ 1 & a > 1 \end{Bmatrix}$	Función sinusoidal $y_i = g\left(\sum_{j=0}^{p} w_{ij}\, x_j\right)$ $g(a) = sen(a)$
Función Logística o Log sigmoidea $y_i = g\left(\sum_{j=0}^{p} w_{ij}\, x_j\right)$ $g(a) = \frac{1}{1+\exp(-a)}$	Función Tangente hiperbólica sigmoidea $y_i = g\left(\sum_{j=0}^{p} w_{ij}\, x_j\right)$ $g(a) = \frac{\exp(a) - \exp(-a)}{\exp(a) + \exp(-a)}$
Función Gaussiana $y_i = g\left(\sum_{j=0}^{p} w_{ij}\, x_j\right)$ $g(a) = \exp\left(-a^2\right)$	

Figura 1.6. Funciones de transferencia o activación (Nota: se han considerado en todos los casos regla de propagación lineal y función de salida identidad).

1.7.5 La función de salida

La función de salida, $f_i(g_i(h_i(t)))$ proporciona la salida global de la neurona y_i en función de su estado de activación actual. Muy frecuentemente la función de salida es simplemente la función identidad, de modo que el estado de activación de la neurona se considera la propia salida.

1.8 Modelos de Redes Neuronales (Taxonomía)

La gran variedad de modelos de redes neuronales existentes en la actualidad obliga en cierta medida a la realización de clasificaciones o taxonomías. De esta forma, los modelos neuronales se pueden clasificar desde una triple óptica: en función de la arquitectura ("*network architecture*"), en función del tipo de aprendizaje ("*learning paradigm*"), y de acuerdo a sus aplicaciones **(Figura 1.7.)**.

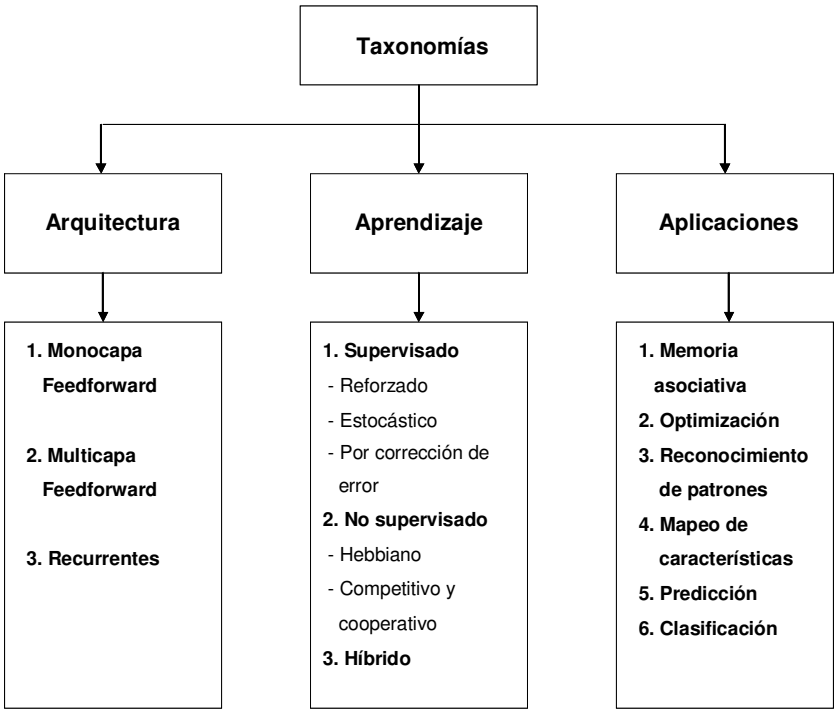

Figura 1.7. Taxonomías de acuerdo al tipo de arquitectura, aprendizaje y aplicaciones.

1.8.1 Taxonomía de acuerdo con el tipo de arquitectura

La arquitectura o topología de las RNA consiste en la organización y disposición de las neuronas en la red formando capas o agrupaciones de neuronas más o menos alejadas de la entrada y la salida de la red. En este sentido, los parámetros fundamentales de la red son: el número de capas, el número de neuronas por capa, el grado de conectividad y el tipo de conexión entre neuronas.

A partir de su ubicación dentro de la red, se pueden distinguir tres tipos de capas:

1. **De entrada:** Es la capa que recibe directamente la información proveniente de las fuentes externas de la red. Esta capa no procesa información, simplemente la distribuye a las demás capas.

2. **Ocultas:** Son internas a la red y no tienen contacto directo con el entorno exterior. El número de niveles ocultos puede estar entre cero y un número elevado. Las neuronas de las capas ocultas pueden estar interconectadas de distintas maneras, lo que determina, junto con su número, las distintas topologías de RNA.

3. **De salida:** Transfieren información de la red hacia el exterior.

La conectividad entre los nodos de una RNA está relacionada con la forma en que las salidas de las neuronas están canalizadas para convertirse en entradas de otras neuronas.

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

Las conexiones entre los nodos de una red pueden ser:

1) **Intercapa.** Es la conexión entre nodos de distintas capas.

2) **Intracapa.** Cuando se conectan nodos dentro de la misma capa.

3) **Autoconectadas.** Se refiere a la conexión de un nodo a él mismo.

4) **Supracapa.** Cuando se conectan nodos de capas no adyacentes.

Una conexión de alto grado es una conexión que combina entradas de más de un nodo. El número de entradas determina el grado de la conexión mientras que el grado de una RNA es el grado de su conexión más grande.

Cuando se realiza una clasificación de las redes en términos topológicos, se suele distinguir entre redes con una sola capa o nivel de neuronas y las redes con múltiples capas (2, 3, etc.) Se asume que las RNA son de primer grado a menos que se especifique lo contrario.

1.8.1.1 Redes monocapa (1 capa)

En las redes monocapa, se establecen conexiones laterales entre las neuronas que pertenecen a la única capa que constituye la red. También pueden existir conexiones autorrecurrentes (salida de una neurona conectada a su propia entrada).

Las redes monocapa se utilizan típicamente en tareas relacionadas con lo que se conoce como autoasociación, por ejemplo, para regenerar informaciones de entrada que se presentan a la red incompletas o distorsionadas.

1.8.1.2 Redes multicapa

Las redes multicapa son aquellas que disponen de conjuntos de neuronas agrupados en varios niveles de capas. Normalmente, todas las neuronas de una capa reciben señales de entrada de otra capa anterior, más cercana a la entrada de la red, y envían las señales de salida a una capa posterior, más cercana a la salida de la red. A estas conexiones se les denomina conexiones hacia delante o *feedforward*.

En las redes *feedforward* no existen conexiones hacia atrás (ninguna salida de neuronas de una capa i se aplica a la entrada de neuronas de capas $i-1, i-2, \ldots$), y normalmente tampoco son autorrecurrentes (salida de una neurona aplicada a su propia entrada), ni laterales.

Sin embargo, en un gran número de estas redes también existe la posibilidad de conectar las salidas de las neuronas de capas posteriores a las entradas de las capas anteriores, a estas conexiones se les denomina conexiones hacia atrás o *feedback*.

Y finalmente, están las redes que disponen de conexiones tanto hacia delante como hacia atrás (*feedforward/feedback).*

En general las redes *feedforward/feedback* suelen ser bicapa, existiendo por tanto dos conjuntos de pesos: los correspondientes a las conexiones *feedforward* de la primera capa (capa de entrada) hacia la segunda capa (capa de salida) y los de las conexiones *feedback* de la segunda a la primera.

1.8.2 Taxonomía de acuerdo con el tipo de aprendizaje

El término de aprendizaje en las máquinas resulta poco claro. Existen muchas más definiciones del concepto general de aprendizaje, sin embargo, para nuestros fines, podemos enfocarnos en la siguiente: "La modificación del comportamiento inducido por la interacción con el entorno y como resultado de experiencias conducente al establecimiento de nuevos modelos de respuesta a estímulos externos"[11]. Esta definición fue enunciada muchos años antes de que surgieran las RNA, sin embargo puede ser aplicada también a los procesos de aprendizaje de estos sistemas.

Biológicamente, se suele aceptar que la información memorizada en el cerebro está más relacionada con los valores sinápticos de las conexiones entre las neuronas que con ellas mismas; es decir, el conocimiento se encuentra en las sinapsis. En el caso de las RNA, se puede considerar que el conocimiento se encuentra representado en los pesos de las conexiones entre las neuronas (pesos sinápticos).

Al igual que el funcionamiento de una red depende del número de neuronas de las que disponga y de cómo estén conectadas entre sí, cada modelo dispone de sus propias técnicas de aprendizaje.

En el contexto de las RNA puede definirse el aprendizaje como "El proceso por el cual una red neuronal modifica sus pesos en respuesta a una información de entrada"[12]. Los cambios que se producen durante el proceso de aprendizaje se reducen a la destrucción, modificación y creación de conexiones entre las neuronas y

[11] Hilera, J.R. y Martínez, V.J. (1995). *Redes neuronales artificiales. Fundamentos, modelos y aplicaciones*, Ra-Ma, Madrid. Pp. 63-64.
[12] *Ibid.* Pp. 75-76.

en algunos modelos incluso mediante la creación o muerte neuronal (en este caso se modifica la propia arquitectura de la red)[13].

En cualquier caso, en un proceso de aprendizaje la información contenida en los datos de entrada queda incorporada en la propia estructura de la red.

Este tipo de acciones, en especial la modificación de las intensidades sinápticas (plasticidad sináptica) serán las que utilicen los sistemas neuronales artificiales para llevar a cabo el aprendizaje.

De forma general, se suelen considerar dos tipos de reglas de aprendizaje: las que responden a lo que habitualmente se conoce como aprendizaje supervisado, y las correspondientes a un aprendizaje no supervisado. Ambas modalidades pretenden estimar funciones de entrada/salida multivariante o densidades de probabilidad. Las reglas de aprendizaje supervisadas suelen ser computacionalmente más complejas, pero también más exactas en sus resultados.

La diferencia fundamental entre ambos tipos estriba en la existencia o no de un agente externo (supervisor) que controle el proceso de aprendizaje de la red.

La regla de aprendizaje es uno de los atributos más importantes a especificar para una RNA. Con ella se determina cómo se adaptarán las conexiones de los pesos a fin de optimizar el funcionamiento de la red y cómo calcular los ajustes en los pesos durante cada ciclo. Esta regla se suspende después de que el entrenamiento se ha completado.

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

[13] La neurona es una célula muy especial que, en general, únicamente posee capacidad para reproducirse en los primeros estados de su vida, de modo que si una neurona muere, no nacerá otra que la reemplace (aunque recientemente se han encontrado evidencias de que en ciertas situaciones sí podría reproducirse).

Si denominamos $w_{ij}(t)$ al peso que conecta la neurona presináptica j con la postsináptica i en la iteración t, el algoritmo de aprendizaje, en función de las señales que llegan procedentes del entorno en el instante t, proporcionará el valor $\Delta w_{ij}(t)$ que da la modificación que se debe incorporar en dicho peso, el cual quedará actualizado de la siguiente forma:

$$\Delta w_{ij}(t+1) = w_{ij}(t) + \Delta w_{ij}(t)$$

El proceso de aprendizaje es usualmente iterativo, actualizándose los pesos de la manera anterior, una y otra vez, hasta que la red alcanza el rendimiento deseado.

Se puede afirmar que este proceso ha terminado (la red ha aprendido) cuando los valores de los pesos permanecen estables ($dw_{ij}/dt = 0$).

Un aspecto importante respecto al aprendizaje en las redes neuronales es el conocer cómo se modifican los valores de los pesos; es decir, cuáles son los criterios que se siguen para cambiar el valor asignado a las conexiones cuando se pretende que la red aprenda una nueva información. Estos criterios determinan lo que se conoce como la regla de aprendizaje de la red.

Las reglas de aprendizaje más conocidas son la regla de retropropagación (*back propagation*) la cual es una generalización de la regla Delta (empleadas para aprendizaje supervisado); la regla de aprendizaje Hebbiano y la regla de aprendizaje competitivo.

1.8.2.1 Redes con aprendizaje supervisado

El aprendizaje supervisado se caracteriza porque el proceso de aprendizaje se realiza mediante un entrenamiento controlado por un agente externo (supervisor o maestro) que determina la respuesta que debería generar la red a partir de una entrada determinada.

Este tipo de aprendizaje consiste en construir un modelo neuronal que permita estimar relaciones entre las entradas y las salidas sin la necesidad de proponer una cierta forma funcional *a priori*. La salida no coincidirá generalmente con lo deseado, de forma que se generará un error de salida e_i o residuo del modelo **(Figura 1.8.)**.

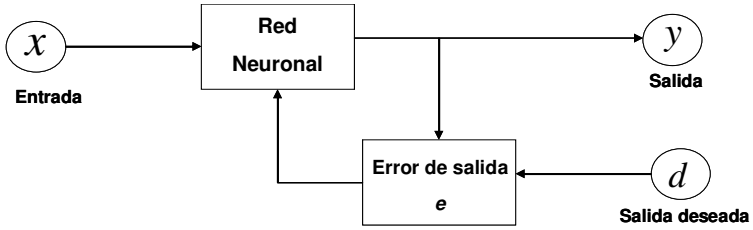

Figura 1.8. Ciclo del aprendizaje supervisado

En este tipo de aprendizaje se suelen considerar, a su vez, tres formas de llevarlo a cabo que dan lugar a los siguientes aprendizajes supervisados:

1) ***Aprendizaje por corrección del error.*** Consiste en ajustar los pesos de las conexiones de la red en función de la diferencia entre los valores deseados y los obtenidos en la salida de la red, es decir, en función del error cometido en la salida. Ejemplos de este tipo de aprendizaje son: la regla de aprendizaje del perceptrón, utilizada en el aprendizaje de la red perceptrón diseñada por Rosenblatt en 1957; la regla delta o regla del error cuadrático medio mínimo ("*Least-Mean-squared Error*"(LMS)) propuesta por Widrow en 1960, utilizada en los modelos neuronales *Adaline* y *Madaline* (estos modelos mejoran el modelo de perceptrón ya que incorporan la definición de error global cometido y mecanismos para reducirlo con mayor rapidez); y la regla delta generalizada o retropropagación del error.

2) ***Aprendizaje reforzado.*** Este método emplea la información del error cometido (calculado en este caso de forma global y no para cada una de las salidas), pero sin poseer la salida deseada. Dicho aprendizaje descansa en la idea dual premio-castigo, donde se refuerza toda aquella acción que permita una mejora del modelo mediante la definición de una señal crítica **(Figura 1.9.)**.

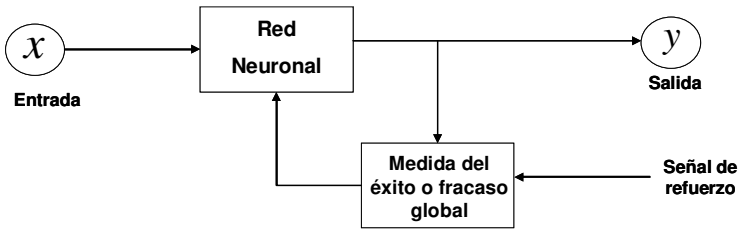

Figura 1.9. Ciclo del aprendizaje reforzado

3) ***Aprendizaje estocástico.*** Consiste en realizar cambios aleatorios en los valores de los pesos de las conexiones de la red y evaluar el efecto a partir del objetivo deseado mediante distribuciones de probabilidad.

1.8.2.2 Redes con aprendizaje no supervisado

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

Sus principales utilidades son entre otras, descubrir las regularidades presentes en los datos, extraer rasgos o agrupar patrones según su similitud, a través de la estimación de la función de densidad de probabilidad "$p(x)$" que permite describir la distribución de patrones "x" pertenecientes al espacio de entrada $\Re^n$ **(Figura 1.10.)**.

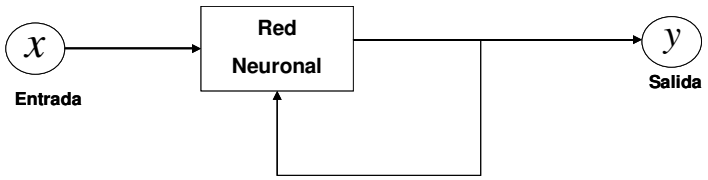

Figura 1.10. Ciclo del aprendizaje no supervisado

Estas redes deben encontrar las características, regularidades, correlaciones o categorías que se puedan establecer entre los datos que se presentan en su entrada. Existen varias posibilidades en cuanto a la interpretación de la salida de estas redes, las cuales dependen de su estructura y del algoritmo de aprendizaje empleado.

En algunos casos, la salida representa el grado de similitud o proximidad entre la información que se le está presentando a la entrada y las informaciones que se le han mostrado hasta entonces (en el pasado). En otro caso se podría realizar un agrupamiento indicando a la red a qué categoría pertenece la información presentada

a la entrada, siendo la propia red quien deba encontrar las categorías apropiadas a partir de las correlaciones entre las informaciones presentadas.

El aprendizaje sin supervisión también permite realizar una codificación de la entrada, generando en la salida una versión codificada de la entrada, con menos *bits* pero manteniendo la información relevante de los datos.

Finalmente, lo que realizan algunas redes es un mapeo de características (*feature mapping*), obteniéndose en las neuronas de salida una disposición geométrica que representa un mapa de las características de los datos de entrada, de tal forma que si se presentan a la red informaciones similares, siempre serán afectadas neuronas de salida próximas entre sí, en la misma zona del mapa.

En cuanto a los algoritmos de aprendizaje no supervisado, en general se suelen considerar dos tipos: aprendizaje Hebbiano y aprendizaje competitivo y cooperativo.

1.8.2.2.1 Aprendizaje Hebbiano

En el ámbito de la estrategia no supervisada, encontramos el aprendizaje Hebbiano, postulado por Hebb (1949), que consiste en el ajuste de los pesos de las conexiones de acuerdo con la correlación de los valores de las dos neuronas conectadas.

Este tipo de aprendizaje se basa en el siguiente postulado formulado por Donald O. Hebb: "Cuando un axón de una celda A está suficientemente cerca como para conseguir excitar una celda B y repetida o no persistentemente toma parte en su activación, algún proceso de crecimiento o cambio metabólico tiene lugar en una o ambas celdas, de tal forma que la eficiencia de A, cuando la celda a activar es B, aumenta".

Por celda, Hebb entiende un conjunto de neuronas fuertemente conectadas a través de una estructura compleja. La eficiencia podría identificarse con la intensidad o magnitud de la conexión; es decir, con el peso.

Se puede decir, por tanto, que el aprendizaje Hebbiano consiste básicamente en el ajuste de los pesos de las conexiones de acuerdo con la correlación (multiplicación en el caso de los valores binarios +1 y −1) de los valores de activación (salidas) de las dos neuronas conectadas:

$$\Delta w_{ij} = x_i \cdot x_j$$

Así, si las dos unidades son activas (positivas), se produce un reforzamiento de la conexión. Por el contrario, cuando una es activa y la otra pasiva (negativa), se produce un debilitamiento de la conexión. Se trata de una regla de aprendizaje no supervisado, pues la modificación de los pesos se realiza en función de los estados (salidas) de las neuronas obtenidas tras la presentación de cierto estímulo (información de entrada a la red), sin tener en cuenta si se deseaba obtener o no esos estados de activación.

Como característica general de las redes no supervisadas Hebbianas puede señalarse que en ellas un número elevado de neuronas de salida pueden activarse simultáneamente. Algunos modelos utilizan reglas de aprendizaje directamente basadas en la regla de Hebb, como las redes PCA (que realizan análisis de componentes principales).

1.8.2.2.2 Aprendizaje competitivo y cooperativo

En las redes con aprendizaje competitivo y cooperativo, las neuronas compiten o cooperan 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 cierta información de entrada, sólo una de las neuronas dé salida a la red, o una por cierto grupo de neuronas, se active (alcance su valor de respuesta máximo).

Por tanto, las neuronas compiten por activarse, quedando finalmente una, o una por grupo, como neurona vencedora (*winner- take- all*), quedando anuladas el resto, que son forzadas a sus valores de respuesta mínimos.

La competición entre neuronas se realiza en todas las capas de la red, existiendo en estas neuronas conexiones recurrentes de auto excitación y conexiones de inhibición (signo negativo) por parte de neuronas vecinas. Si el aprendizaje es cooperativo, estas conexiones con las neuronas vecinas serán de excitación (signo positivo).
El objetivo de este aprendizaje es categorizar 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.

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

Por tanto, la variación del peso de una conexión entre una unidad i y otra j será nula si la neurona j no recibe excitación por parte de la neurona i (no vence en presencia de un estímulo por parte de i), y se modificará (se reforzará) si es excitada por dicha neurona i.

Una variación del aprendizaje supervisado aplicado a redes multicapa consiste en imponer una inhibición mutua entre neuronas únicamente cuando están a cierta distancia unas de otras (suponiendo que las neuronas se han dispuesto geométricamente, por ejemplo formando capas bidimensionales).

Existe entonces un área o región de vecindad (*vicinity area*) alrededor de las neuronas que constituyen su grupo local.

El aspecto geométrico de la disposición de las neuronas de una red también es la base de un caso particular de aprendizaje competitivo inducido por Kohonen en 1982, conocido como *feature mapping*, aplicado en redes con una disposición bidimensional de las neuronas de salida, que permiten obtener mapas topográficos o autoorganizados, en los que, de algún modo, estarían representadas las características principales de la información presentada a la red.

De esta forma, si la red recibe información con características similares, se generarán mapas parecidos puesto que se afectarían neuronas de salida próximas entre sí (se profundizará sobre este tema en el siguiente capítulo).

Para concluir este apartado, hay que comentar la existencia de otro caso particular del aprendizaje competitivo, denominado Teoría de la Resonancia Adaptativa, desarrollado por Carpenter y Grossberg en 1986 y utilizado en la red *feedforward/feedback* de dos capas conocida como ART (en sus dos variantes: ART1, que trabaja con información binaria, y ART2, que maneja información analógica).

Esta red realiza un prototipado de la información que recibe de la entrada, generando como salida un ejemplar o prototipo que representa a toda la información que podría considerarse perteneciente a la misma clase o categoría.

La teoría de la resonancia adaptativa se basa en la idea de hacer resonar la información de entrada con los prototipos de las categorías que reconoce la red; si entra en resonancia con alguno (es suficientemente similar), la red considera que pertenece a dicha categoría y únicamente realiza una pequeña adaptación del prototipado (para que se parezca un poco más al dato presentado).

Cuando no resuena con ningún prototipo (no se parece a ninguno de los existentes recordados por la red) hasta ese momento, la red se encarga de crear una nueva categoría con el dato de entrada como prototipo de la misma.

1.8.2.3 Aprendizaje híbrido

Existe un tipo de aprendizaje denominado híbrido, en el cual coexisten en la red los dos tipos básicos de aprendizaje, el supervisado y el no supervisado, los cuales tienen lugar normalmente en distintas capas de neuronas. El modelo de Contra-propagación es un ejemplo de red que hace uso de este tipo de aprendizaje.

1.8.3 Taxonomía de acuerdo al tipo de aplicaciones

Respecto a las diferentes aplicaciones tenemos, en primer lugar, la memoria asociativa, consistente en reconstruir una determinada información de entrada que se presenta incompleta o distorsionada, asociando la información de entrada con el ejemplar más parecido de los almacenados conocidos por la red.

En segundo lugar, la optimización, es decir, la resolución de problemas de optimización combinatoria.

En tercer lugar, el reconocimiento de patrones, consistente, desde una óptica general, en la detección de formas simples.
En cuarto lugar, el mapeo de características, que parte de las ideas de Kohonen simulando la capacidad del cerebro humano de crear mapas topológicos de las informaciones recibidas del exterior.

En quinto lugar, está la predicción y en último lugar, la clasificación. Es importante señalar que una misma red puede utilizarse en aplicaciones diferentes.

1.9 RNA y Estadística

La estadística comprende un conjunto de métodos que sirven para recoger, organizar, resumir y analizar datos, así como para extraer conclusiones y tomar decisiones razonables basadas en tal análisis. Las RNA han sido descritas por algunos como técnicas de ajuste estadístico inspiradas en la biología. Modelos procedentes de ambas disciplinas se emplean en ajuste funcional (perceptrón y regresión), en reducción de la dimensionalidad (mapas de Kohonen y análisis de componentes principales) y otras tareas.

Se han realizado estudios comparando métodos estadísticos y neuronales, llegándose a la conclusión de que no se puede realizar la afirmación genérica de que los modelos neuronales sobrepasen siempre en eficiencia a las técnicas estadísticas. En Sarle (1994) se señala el claro paralelismo entre ciertos modelos estadísticos y neuronales sin embargo, hay algunos modelos de RNA para los que no existe una técnica estadística equiparable. La **Tabla 1.2.** muestra los más usuales.

La aproximación más común a los problemas de regresión son los perceptrones multicapa y generalizaciones de perceptrones de una sola capa. Además del análisis discriminante y la regresión, otra actividad común en la investigación es la que los estadísticos reconocen como análisis *cluster*, en la literatura de las RNA esto representa aprendizaje no supervisado.

Modelo Estadístico	Modelo de red neuronal
Regresión lineal múltiple	Perceptrón simple con función lineal
Regresión Logística	Perceptrón simple con función logística
Regresión no lineal múltiple	Perceptrón multicapa con función lineal en la salida
Función discriminante lineal	Perceptrón simple con función umbral
Función discriminante no lineal	Perceptrón multicapa con función logística en la salida
Análisis de Componentes Principales (PCA)	Perceptrón multicapa autoasociativo
Análisis *Cluster*	Mapas autoorganizados de Kohonen

Tabla 1.2. Equivalencia entre modelos estadísticos y modelos de red neuronal

Capítulo 2

2.- Los mapas autoorganizados

Una vez expuestos en el capítulo anterior los conceptos básicos relacionados con las redes neuronales artificiales, este capítulo se centrará en los mapas autoorganizados, uno de los sistemas neuronales no supervisados más conocidos y utilizados. Se presentará el modelo general de mapas autoorganizados así como algunos de sus algoritmos de aprendizaje.

2.1 Introducción

Se observa que en muchas regiones del córtex de los animales superiores aparecen zonas donde las neuronas detectoras de rasgos (o características) se distribuyen topológicamente ordenadas, circunstancia que el modelo neuronal de mapas autoorganizados, SOFM (*Self-Organizing Feature Maps*), SOM (*Self-Organizing Maps*), o mapas de Kohonen trata de reproducir.

Los SOM fueron desarrollados a lo largo de la década de los ochenta por el físico finlandés Teuvo Kohonen, como una continuación natural de la línea de desarrollo de las redes competitivas iniciada por Von der Malsburg. Aparte de su interés como una sencilla modelización de redes neuronales naturales, los SOM poseen un gran potencial de aplicabilidad práctica.

De entre las clases de problemas del mundo real en los que han demostrado su eficacia cabe citar: clasificación de patrones, cuantificación vectorial, reducción de dimensiones, extracción de rasgos y visualización. Por ejemplo, los SOM han sido empleados en reconocimiento del habla, control de robots, monitorización de procesos industriales, ayuda al diseño de circuitos integrados, reconocimiento de patrones financieros y minería de grandes bases de datos en Internet.

En este modelo, las neuronas se organizan en una arquitectura unidireccional de dos capas **(Figura 2.1.)**. La primera es la capa de entrada o sensorial, que consiste en m

neuronas, una por cada variable de entrada, que se comportan como *buffers*[14], distribuyendo la información procedente del espacio de entrada a las neuronas de la segunda capa. Las entradas son muestras estadísticas $\mathbf{x}(t) \in \Re^m$ del espacio sensorial.

El procesamiento se realiza en la segunda capa, que forma el mapa de rasgos, y consiste habitualmente en una estructura rectangular de $nx \times ny$ neuronas que operan en paralelo.

Aunque la arquitectura rectangular es la más común, a veces también se utilizan capas de una sola dimensión (cadena lineal de neuronas) o de tres dimensiones (paralelepípedo).

Etiquetaremos las m neuronas de entrada con el índice k $(1 \le k \le m)$, y las $nx \times ny$ neuronas del mapa con un par de índices $\mathbf{i} \equiv (i, j)$ $(1 \le i \le nx, 1 \le j \le ny)$ que determinarán su localización espacial. Cada neurona de entrada k está conectada a todas las neuronas (i, j) del mapa mediante un peso sináptico w_{ijk}.

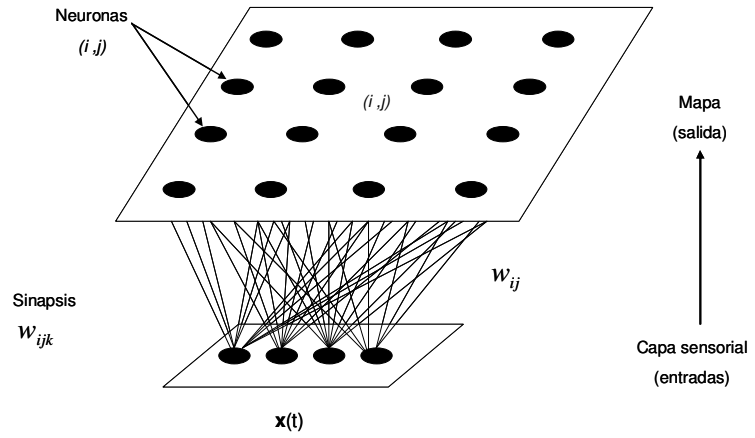

Figura 2.1 Arquitectura del SOM

[14] Un *buffer* (a veces traducido como "memoria temporal") es un área de datos compartida por dispositivos de *hardware* o procesos de programas que operan a distintas velocidades o con diferentes conjuntos de prioridades. El *buffer* permite que cada dispositivo o proceso opere sin verse interferido por otro. Como una memoria *caché*, un *buffer* es "un punto intermedio de almacenamiento". Este término se usa tanto en programación como en *hardware*. En la programación, el uso del *buffer* en ocasiones implica la necesidad filtrar datos de su destino final para poderlos editar o procesar de alguna otra forma antes de transferirlos a un archivo o base de datos regular.

En resumen, el mapa puede describirse como una matriz de procesadores elementales (i, j) ordenados en dos dimensiones que almacenan un vector de pesos sinápticos o vector de referencia (*codebook*) $\mathbf{w}_{ij}(t)$, con

$$\left\{\mathbf{w}_{ij}(t) : \mathbf{w}_{ij} \in \Re^m, 1 \le i \le nx, 1 \le j \le ny\right\}.$$

En la fase de ejecución (operación normal de la red), los pesos permanecen fijos. En primer lugar, cada neurona (i, j) calcula la similitud entre el vector de entradas $\mathbf{x}, \{x_k \mid 1 \le k \le m\}$ y su propio vector de pesos sinápticos $\mathbf{w}_{ij}$, según una cierta medida de distancia o criterio de similitud establecido.

A continuación, se declara vencedora la neurona $\mathbf{g} = (g_1, g_2)$, cuyo vector de pesos $\mathbf{w}_\mathbf{g}$ es más similar al de entradas. De esta manera, cada neurona actúa como un detector de rasgos específicos, y la neurona ganadora nos indica el rasgo o patrón detectado en el vector de entradas.

$$d(\mathbf{w}_\mathbf{g}, \mathbf{x}) = \min_{ij}\left\{d(\mathbf{w}_{ij}, \mathbf{x})\right\} \tag{2.1}$$

En la fase de aprendizaje cada neurona del mapa sintoniza con diferentes rasgos del espacio de entrada. El proceso es el siguiente: tras la presentación y procesamiento de un vector de entradas $\mathbf{x}(t)$, la neurona vencedora modifica sus pesos de manera que se parezcan un poco más a $\mathbf{x}(t)$. De este modo, ante el mismo patrón de entrada, dicha neurona responderá en el futuro todavía con más intensidad.

El proceso se repite para numerosos patrones de entrada, de forma que al final los diferentes vectores de referencia sintonizan con dominios específicos de las variables de entrada, y tienden a representar la función de densidad de probabilidad $p(\mathbf{x})$ (o función de distribución) del espacio sensorial.

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 entonces como un análisis *cluster*.

Lo descrito hasta el momento responde a un esquema competitivo clásico de relativa sencillez, en el que cada neurona actúa en solitario. Sin embargo, el modelo de SOM aporta una importante novedad, pues incorpora a este esquema relaciones entre las

neuronas próximas del mapa. Para ello introduce una *función de vecindad*[15], su efecto es que durante el aprendizaje se actualizan tanto los pesos de la vencedora como los de las neuronas pertenecientes a su entorno.

De esta manera, en el modelo de 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.

En esencia, por medio del proceso descrito los SOM realizan la proyección no lineal de un espacio multidimensional de entrada $\Re^m$ sobre un espacio discreto de salida, representada por la capa de neuronas. El mapa representa una imagen del espacio sensorial, pero de menor número de dimensiones, reflejando con mayor fidelidad aquellas dimensiones del espacio de entrada de mayor varianza (que suelen coincidir con los rasgos más importantes de las entradas).

La distribución de las neuronas sobre el mapa resulta ser un reflejo de la función de densidad de probabilidad $p(\mathbf{x})$: regiones en el espacio sensorial cuyos representantes $\mathbf{x}$ aparecen con más frecuencia ($p(\mathbf{x})$ mayor) serán proyectadas sobre un número mayor de neuronas en el mapa.

La función vecindad representa matemáticamente de una forma sencilla el efecto global de las interacciones laterales existente entre las neuronas en el cerebro, pues en vez de considerar en detalle que una neurona trata de activar a sus vecinas y de inhibir a las alejadas (como sucede en el córtex), esta situación se modela mediante una sencilla función que define el tamaño de la vecindad en torno a la vencedora, dentro de la cual todas las neuronas son premiadas actualizando sus pesos, y fuera de ella son castigadas al no actualizar sus pesos o al hacerlo en sentido contrario.

La utilización de la función vecindad en el modelo de mapas autoorganizados aporta respecto del modelo competitivo sencillo dos ventajas adicionales: el ritmo efectivo de convergencia se mejora y el sistema es más robusto frente a variaciones en los valores iniciales de los pesos.

---

[15] Una función vecindad *V=V[n]* (n= tiempo discreto) se puede definir de diferentes formas (cuadrada, circular, rombo, gaussiana o algún híbrido entre ellas). En realidad esta función no tiene como único argumento el tiempo discreto "*n*", si no que también tiene como argumentos a la posición del nodo ganador y la posición del nodo a actualizar. La función vecindad está centrada en el nodo ganador y su radio de influencia disminuye monótonamente a medida que avanza el entrenamiento.

2.2 Algoritmo de aprendizaje

Como hemos visto, la principal novedad de los SOM consiste en que la modificación de los pesos no se aplica solamente a una neurona específica (la ganadora), sino también a su vecindad. Al comienzo del entrenamiento la vecindad comprende una amplia región del mapa, lo que permite una ordenación global de los pesos sinápticos. Con el transcurso de las iteraciones, el tamaño de la vecindad se reduce, y al final solamente se modifican los pesos de la neurona ganadora. Así, el proceso de aprendizaje comprende dos fases fundamentales: una ordenación global, en la que se produce el despliegue del mapa; y un ajuste fino, en el que las neuronas se especializan.

2.2.1 Ejemplo de un algoritmo de aprendizaje autoorganizado

Se debe tener en cuenta que no existe un algoritmo de aprendizaje único ni totalmente estándar para los SOM. No obstante, el resultado final es bastante independiente de los detalles de su realización concreta, como pueden ser los pesos sinápticos de partida, el esquema de actualización del ritmo de aprendizaje, o la forma establecida para la vecindad. A continuación se expone un algoritmo de aprendizaje habitual:

1. **Inicialización de los pesos sinápticos** w_{ijk}**.** Se puede partir en $t=0$ de diferentes configuraciones: pesos nulos, aleatorios de pequeño valor absoluto (lo más habitual), o con un valor de partida predeterminado.

2. En cada iteración, **presentación de un patrón** $\mathbf{x}(t)$ tomado de acuerdo con la función de distribución $p(\mathbf{x})$ del espacio sensorial de entrada. En la muy habitual situación de disponer solamente de un conjunto pequeño de patrones de entrenamiento basta con tomar al azar uno de ellos y presentarlo a la red.

3. Cada neurona $\mathbf{i} \equiv (i, j)$ en paralelo del mapa calcula la **similitud** entre su vector de pesos sinápticos $\mathbf{w}_{ij}$ y el actual vector de entradas $\mathbf{x}$. Un criterio de medida de similitud muy utilizado es la distancia euclídea:

$$d^2\left(\mathbf{w}_{ij},\mathbf{x}\right)=\sum_{k=1}^{n}\left(w_{ij}-x_k\right)^2 \qquad (2.2)$$

4. Determinación de la **neurona ganadora** $\mathbf{g}=\left(g_1,g_2\right)$, cuya distancia sea la menor de todas.

5. **Actualización de los pesos sinápticos** de la neurona ganadora y de sus neuronas vecinas. La regla mas empleada es:

$$w_{ijk}\left(t+1\right)=w_{ijk}\left(t\right)+\alpha\left(t\right)h\left(\left|\mathbf{i}-\mathbf{g}\right|,t\right)\left(x_k\left(t\right)-w_{ijk}\left(t\right)\right) \qquad (2.3)$$

donde $\alpha(t)$ es un parámetro denominado ritmo de aprendizaje. La función $h(\cdot)$ se denomina función de vecindad, puesto que establece qué neuronas son las actualmente vecinas a la ganadora. Esta función depende de la distancia entre la neurona **i** y la ganadora **g**, valiendo cero cuando **i** no pertenece a la vecindad de **g** (con lo que sus pesos no son actualizados), y un número positivo cuando sí pertenece (sus pesos sí son modificados). Como veremos, la vecindad es un conjunto de neuronas centrado en la ganadora.

Tanto α como el radio de la vecindad usualmente disminuyen monótonamente con t (durante el proceso de ordenamiento)[16].

6. Si se ha alcanzado el número máximo de iteraciones establecido, entonces el **proceso de aprendizaje finaliza**. En caso contrario se vuelve al paso 2.

Se puede realizar a continuación una segunda fase en el aprendizaje, en la que se produce el ajuste fino del mapa, de modo que la distribución de los pesos sinápticos se ajuste más a la de las entradas. El proceso es similar al anterior, tomando $\alpha(t)$ constante e igual a un pequeño valor (por ejemplo, 0.01), y un radio de vecindad constante e igual a uno.

[16] Kohonen, T. (1995). *Self-Organizing Maps*. Springer Series in Information Sciences. Pp. 79.

En el aprendizaje, el número de iteraciones debe ser suficientemente grande por requerimientos estadísticos, así como proporcional al número de neuronas del mapa (a más neuronas, son necesarias más iteraciones), e independiente del número de componentes de $\mathbf{x}$. Aunque 500 iteraciones por neurona es una cifra adecuada, de 50 a 100 suelen ser suficientes para la mayor parte de los problemas. Entre $20,000$ y $100,000$ iteraciones representan cifras habituales en la simulación por computadora del entrenamiento de un SOM[17].

Una cuestión a tener presente es que el criterio de similitud y la regla de aprendizaje que se utilicen deben ser métricamente compatibles; así ocurre con la distancia euclídea (2.2) y la regla de aprendizaje (2.3). El empleo de diferentes métricas para la fase de recuerdo y para la actualización de los pesos puede causar problemas en el desarrollo del mapa. Más adelante se mostrarán las diferentes posibilidades para la elección del criterio de distancia o métrica, y los algoritmos de aprendizaje que de cada una de ellas se derivan. No obstante, podemos adelantar que una medida de similitud alternativa, más simple que la euclídea, es la correlación o producto escalar

$$C_{ij} = \sum_{k=1}^{n} w_{ijk} x_k \tag{2.4}$$

que suele incorporarse al algoritmo, junto con la regla de adaptación (2.3). Sin embargo, dicha métrica procede de la métrica euclídea, y la correlación solamente es compatible con esta métrica si se utilizan vectores normalizados de norma 1 (en cuyo caso la distancia euclídea y la correlación coinciden).

Por esta razón, en ocasiones se hace la afirmación errónea de que el modelo de Kohonen precisa vectores normalizados. Si utilizamos la distancia euclídea (2.2) y la regla (2.3), no es necesario tratar con vectores normalizados (otra cuestión diferente es que en determinados problemas dicha normalización pueda ser aconsejable para mantener las entradas dentro de un determinado rango dinámico).

[17] Pese a parecer cifras muy altas, las simulaciones de un SOM usualmente son rápidas pues su algoritmo es computacionalmente sencillo.

2.3 Interpretación del algoritmo de aprendizaje

La siguiente interpretación del proceso de aprendizaje puede resultar interesante para comprender la operación de los SOM. El efecto de la regla de aprendizaje (2.3) no es otro que en cada iteración acercar en una pequeña cantidad el vector de pesos $\mathbf{w}$ de la neurona de mayor activación (ganadora) al vector de entrada $\mathbf{x}$, donde la expresión

$$\Delta\mathbf{w}(t) = \alpha \cdot (\mathbf{x} - \mathbf{w}) \tag{2.5}$$

representa el incremento del vector de pesos de la neurona ganadora $(0 < \alpha < 1)$. Así, en cada iteración el vector de pesos de la neurona vencedora, rota hacia el presentado, y se aproxima a él en una cantidad que depende del ritmo de aprendizaje α. De modo que podemos observar que en cada iteración se elimina una cierta fracción del antiguo vector de pesos $\mathbf{w}(t)$ (es decir, la cantidad $-\alpha \cdot \mathbf{w}$ representa un término de olvido), el cual es sustituido por una fracción del vector actual $\alpha \cdot \mathbf{x}$, de modo que en cada paso el vector de pesos de la neurona ganadora $\mathbf{w}$ se parece un poco más al vector de entradas $\mathbf{x}$ que la hace ganar.

2.4 Consideraciones prácticas: ritmo de aprendizaje y función vecindad

El ritmo de aprendizaje $\alpha(t)$ es una función monótonamente decreciente con el tiempo, siendo habitual su actualización mediante una función lineal

$$\alpha(t) = \alpha_0 + (\alpha_f - \alpha_0)\frac{t}{t_\alpha} \tag{2.6}$$

con α_0 el ritmo de aprendizaje inicial (<1), α_f el final $(\cong 0.01)$ y t_α el máximo número de iteraciones hasta llegar a α_f. Una alternativa es usar una función que decrezca exponencialmente

$$\alpha(t) = \alpha_0 \left(\frac{\alpha_f}{\alpha_0}\right)^{t/t_\alpha} \tag{2.7}$$

También suele considerarse $\alpha(t) = 0.9\left(1 - t/1000\right)$ como una elección razonable[18].

El empleo de una u otra función no suele influir demasiado en el resultado final. No siendo así en el caso de mapas muy largos donde la selección de un $\alpha(t)$ óptimo es crucial para la convergencia.

La función vecindad $h(|\mathbf{i}-\mathbf{g}|,t)$ define en cada iteración t si una neurona $\mathbf{i}$ pertenece o no a la vecindad de la neurona vencedora $\mathbf{g}$. La vecindad es simétrica y centrada en $\mathbf{g}$, de ahí que se represente como uno de sus argumentos la distancia entre la neurona genérica $\mathbf{i} = (i, j)$ y la vencedora $\mathbf{g} = (g_1, g_2)$, debido a que

$$\|\mathbf{i}-\mathbf{g}\| = \sqrt{(i-g_1)^2 + (j-g_2)^2} \tag{2.8}$$

en general, $h(\cdot)$ decrece con la distancia a la vencedora, y depende de un parámetro denominado radio de vecindad $R(t)$, que representa el tamaño de la vecindad actual. En realidad, bajo la forma funcional de $h(\cdot)$ se encapsula el complejo sistema de interacciones laterales existente entre las neuronas del mapa.

La función de vecindad más simple es del tipo escalón, que denominaremos rectangular

$$h(|\mathbf{i}-\mathbf{g}|,t) = \begin{cases} 0 & si\ |\mathbf{i}-\mathbf{g}| > R(t) \\ 1 & si\ |\mathbf{i}-\mathbf{g}| \le R(t) \end{cases} \tag{2.9}$$

Por tanto, en este caso una neurona pertenece a la vecindad de la ganadora solamente si su distancia es inferior a $R(t)$. Con este tipo de función las vecindades adquieren forma circular, de bordes nítidos, en torno a la vencedora y la ecuación (2.3) se reduce a

$$\Delta w_{ijk}(t) = \begin{cases} 0 & si\ |\mathbf{i}-\mathbf{g}| > R(t) \\ \alpha(t)\left(x_k(t) - w_{ijk}(t)\right) & si\ |\mathbf{i}-\mathbf{g}| \le R(t) \end{cases} \tag{2.10}$$

[18] *Ibid.* Pp. 80.

por lo que en cada iteración únicamente se actualizan las neuronas que distan de la vencedora en una distancia menor a $R(t)$.

La función de vecindad Gaussiana tiene la siguiente forma:

$$h(|\mathbf{i}-\mathbf{g}|,t)=\alpha(t)\cdot\exp\left(-\frac{|\mathbf{i}-\mathbf{g}|^2}{2\sigma^2(t)}\right)$$

donde $\alpha(t)$ es otro ritmo de aprendizaje, y el parámetro $\sigma(t)$ define la amplitud de la vecindad. Tanto $\alpha(t)$ como $\sigma(t)$ son funciones monótonamente decrecientes con el tiempo.

Si la red SOM no es muy grande (digamos, unos cuantos cientos de nodos a lo más), la selección de los parámetros del proceso no es crucial, y de hecho, se puede usar la función de vecindad definida en (2.9).

La función de vecindad posee una forma definida, pero su radio $R(t)$ varía con el tiempo. Se parte de un valor inicial R_0 grande, (incluso puede ser más de la mitad del diámetro de la red), que determina vecindades amplias, con el fin de lograr la ordenación global del mapa. $R(t)$ disminuye monótonamente con el tiempo, hasta alcanzar un valor final de $R_f=1$ con el que solamente se actualizan los pesos de la neurona vencedora y las adyacentes. Una posible función de actualización de $R(t)$ es la siguiente:

$$R(t)=R_0+(R_f-R_0)\frac{t}{t_R} \tag{2.11}$$

donde t es la iteración y t_R el número de iteraciones para alcanzar R_f. Existen otras expresiones, como funciones exponencialmente decrecientes, de aspecto similar a (2.7).

Si únicamente se dispone de un número relativamente pequeño de muestras, entonces éstas deberán ser recicladas para alcanzar el número deseado de iteraciones[19].

[19] *Ibidem.*

2.5 Modelos de neuronas de Kohonen - Medidas de similitud

El modelo de neurona de Kohonen se basa en el cálculo de la similitud entre el vector de entradas y el de pesos. Así, dependiendo del criterio que se seleccione, se tendrá un modelo u otro.

Uno de los modelos más comunes es la correlación o producto escalar:

$$\mathbf{C}_{ij} = \sum_{k=1}^{n} w_{ij} x_k \tag{2.16}$$

según el cual, dos vectores serán más similares cuanto mayor sea su correlación. Es interesante observar que una neurona SOM que utilice este criterio de distancia coincide básicamente con el modelo de neurona estándar de las RNA. Sin embargo, esta medida es sensible al tamaño de los vectores; grandes diferencias en sus longitudes pueden introducir una importante distorsión en la medida de similitud.

Para resolver este problema puede dividirse por las normas de los vectores, con lo que se tiene el denominado criterio del coseno

$$\cos\left(\mathbf{w}_{ij}, \mathbf{x}\right) = \frac{\sum_{k=1}^{n} w_{ijk} x_k}{\left\|\mathbf{w}_{ij}\right\| \cdot \left\|\mathbf{x}\right\|} \tag{2.17}$$

su importancia radica en que esta medida se basa en una característica relativa a ambos vectores, como es su ángulo, independientemente de sus tamaños.

Otro de los criterios de más amplio uso es la distancia euclídea

$$d^2\left(\mathbf{w}_{ij}, \mathbf{x}\right) = \sum_{k=1}^{n} \left(w_{ijk} - x_k\right)^2 \tag{2.18}$$

si se utiliza una red de Kohonen para análisis *cluster*, la distancia euclídea es más adecuada cuando los grupos a extraer están compuestos por nubes esféricas de puntos en torno a su centro. Si no es así, el algoritmo tratará de ajustar los datos en múltiples grupos esféricos.

La métrica de Minkowski

$$d\left(\mathbf{w}_{ij},\mathbf{x}\right)=\left(\sum_{k=1}^{n}\left|w_{ijk}-x_k\right|^{\lambda}\right)^{1/\lambda} \quad ,\lambda\in\Re \tag{2.19}$$

cuando $\lambda=1$, se trata de la distancia de Manhattan.

La correlación, el coseno y la distancia euclídea son los criterios más utilizados, siendo fácil demostrar que coinciden para el caso de vectores normalizados. Por una parte, si las normas son iguales a uno en la ecuación (2.16) se obtiene (2.17), y por otra, desarrollando la ecuación de la distancia euclídea (2.18) y haciendo las normas igual a uno, se obtiene

$$d^2\left(\mathbf{w}_{ij},\mathbf{x}\right)=\left\|\mathbf{w}_{ij}-\mathbf{x}\right\|^2=\left\|\mathbf{w}_{ij}\right\|^2+\left\|\mathbf{x}\right\|^2-2\mathbf{w}_{ij}^T\cdot\mathbf{x}=2\left(1-\mathbf{w}_{ij}^T\cdot\mathbf{x}\right) \tag{2.20}$$

de lo que se deduce que una correlación máxima corresponde a una distancia euclídea mínima, luego ambas medidas también coinciden.

Para vectores normalizados se puede realizar una neurona de Kohonen empleando la correlación y la regla de actualización habitual (2.3), que se deduce del criterio de distancia euclídea. La forma de este modelo coincide con el de neurona estándar definida en el capítulo 1, y resulta de gran sencillez.

Existen algunos otros criterios de distancia, como la medida de similitud de Tanimoto, la de Mahalanobis o la de Hamming, aplicadas para el caso de patrones cuyas componentes no sean números reales, sino variables lógicas o cadenas de caracteres.

2.6 Modelos de aprendizaje en mapas autoorganizados

En primer lugar, se presenta un procedimiento sistemático para la deducción de reglas de aprendizaje para los SOM.

Se propone una cierta función objetivo o error E, dependiendo de los pesos de la red, y se obtiene la regla de actualización a partir de su optimización mediante descenso por el gradiente[20].

Sea $d\left(\mathbf{x}, \mathbf{w}_{ij}\right)$ una función de error la cual es una distancia genérica definida en el espacio de las señales, supondremos que es diferenciable, y que mide el error de cuantificación para el vector de entrada $\mathbf{x}$. La neurona ganadora $\mathbf{g}$ será la que cumple

$$\mathbf{g} = \min_{ij}\left\{d\left(\mathbf{w}_{ij}, \mathbf{x}\right)\right\} \tag{2.21}$$

la definición de una función error en el caso supervisado resulta bastante obvia, pues lo que se pretendía era que las salidas actuales tendieran a las deseadas, con lo cual una función objetivo a minimizar consiste en la suma de los errores asociados a cada patrón. En el caso no supervisado la definición no resulta tan evidente, puesto que no se dispone de un objetivo explícito al que deban tender las salidas de la red.

Aquí, el objetivo será encontrar una función de error que permita deducir sistemáticamente reglas de aprendizaje. Como se pretende que los pesos ajusten la distribución de entradas, un objetivo puede ser que los pesos sinápticos tiendan a ellas, es decir, que los errores de cuantificación sean lo más pequeños posibles. Con esta premisa, puede definirse una función objetivo global de la red de la siguiente manera

$$E = \int \sum_i h\left(\left|\mathbf{i} - \mathbf{g}\right|\right) f\left(d\left(\mathbf{x}, \mathbf{w}_i\right)\right) p\left(\mathbf{x}\right) d\mathbf{x} \tag{2.22}$$

[20] Es decir, habrá que modificar los pesos en la dirección opuesta al gradiente, esto es $\Delta w_{ij} = -\alpha \frac{\partial E}{\partial w_{ij}}$. Uno de los algoritmos de optimización de descenso por el gradiente más conocidos es el algoritmo de *back propagation*, que modifica los valores de los parámetros proporcionalmente al gradiente de la función de error con objeto de alcanzar un mínimo local.

con $p(\mathbf{x})$ la función de distribución del espacio sensorial, $f(\cdot)$ una cierta función del error de cuantificación (introducida por generalidad), y $h(\cdot)$ la función de vecindad. Esta función objetivo global al mapa se basa en la suma a todas las neuronas de los errores de cuantificación, ponderada en la vecindad, y promediado por medio de la función de distribución para todas las posibles entradas.

Para aplicar la aproximación estocástica definiremos la siguiente función

$$E_1(t) = \sum_i h(|\mathbf{i}-\mathbf{g}|, t) f\left(d\left(\mathbf{x}(t), \mathbf{w}_i(t)\right)\right) \tag{2.23}$$

que es una muestra tomada en t de la función global objetivo E. Para esta muestra, $\mathbf{g}$ es constante, una solución aproximada se obtiene mediante descenso por el gradiente

$$\mathbf{w}_i(t+1) = \mathbf{w}_i(t) - \lambda(t)\nabla_{\mathbf{w}_i} E_1(t) \tag{2.24}$$

con $\lambda(t)$ el ritmo de aprendizaje, que debe cumplir las dos condiciones habituales

$$\sum_{t=0}^{\infty} \lambda(t) = \infty, \qquad \sum_{t=0}^{\infty} \lambda^2(t) < \infty \tag{2.25}$$

al estar realizando descensos por los gradientes locales proporcionados por $E_1(t)$, y no descensos por la máxima pendiente que proporciona E, se obtienen soluciones no globalmente óptimas. No obstante, se ha mostrado que el punto que se alcanza está muy próximo al óptimo, y que puede considerarse que las soluciones que proporciona son casi óptimas. Este procedimiento permite deducir sistemáticamente algoritmos de aprendizaje sólo con cambiar el criterio de distancia $d(\cdot)$ y la función $f(\cdot)$.

2.7 Regla de aprendizaje euclídea

Si consideramos como criterio la distancia euclídea

$$d\left(\mathbf{w}_{ij},\mathbf{x}\right)=\sqrt{\sum_{k=1}^{n}\left(w_{ijk}-x_k\right)^2} \tag{2.26}$$

y como función $f\left(d\right)=d^2$, la muestra $E_1\left(t\right)$ de la función objetivo queda

$$E_1\left(t\right)=\sum_{ij}h\left(\left|\mathbf{i}-\mathbf{g}\right|,t\right)\left[\sum_{k=1}^{n}\left(w_{ijk}-x_k\right)^2\right] \tag{2.27}$$

si calculamos su gradiente

$$\nabla_{w_{ijk}}E_1\left(t\right)=\frac{\partial}{\partial w_{ijk}}\left\{\sum_{ij}h\left(\left|\mathbf{i}-\mathbf{g}\right|,t\right)\left[\sum_{k=1}^{n}\left(w_{ijk}-x_k\right)^2\right]\right\}= \tag{2.28}$$

por ser $\mathbf{g}$ constante para la muestra $E_1\left(t\right)$, se tiene

$$=\sum_{ij}h\left(\left|\mathbf{i}-\mathbf{g}\right|,t\right)\frac{\partial}{\partial w_{ijk}}\left[\sum_{k=1}^{n}\left(w_{ijk}-x_k\right)^2\right]=2h\left(\left|\mathbf{i}-\mathbf{g}\right|,t\right)\left(w_{ijk}-x_k\right) \tag{2.29}$$

y, llamando $\alpha\left(t\right)=2\cdot\lambda\left(t\right)$, de (2.29) se obtiene

$$\mathbf{w}_i\left(t+1\right)=\mathbf{w}_i\left(t\right)+\alpha\left(t\right)h\left(\left|\mathbf{i}-\mathbf{g}\right|,t\right)\left(x_k-w_{ijk}\right) \tag{2.30}$$

que es la regla de aprendizaje de Kohonen (2.3) por lo tanto, la regla convencionalmente utilizada en el aprendizaje de una red de Kohonen procede de la métrica euclídea.

2.8 Regla de aprendizaje de Manhattan

Otro de los criterios de distancia comentados es la norma de Manhattan

$$d\left(\mathbf{w}_{ij},\mathbf{x}\right)=\sum_{k=1}^{n}\left|w_{ijk}-x_k\right| \tag{2.31}$$

Para obtener su regla de aprendizaje asociada se toma $f\left(d\right)=d$.

$$E_1\left(t\right)=\sum_{ij}h\left(\left|\mathbf{i}-\mathbf{g}\right|,t\right)\left[\sum_{k=1}^{n}\left|w_{ijk}-x_k\right|\right] \tag{2.32}$$

y calculando el gradiente

$$\begin{aligned}\nabla_{w_{ijk}}E_1\left(t\right)&=\left\{\sum_{ij}h\left(\left|\mathbf{i}-\mathbf{g}\right|,t\right)\frac{\partial}{\partial w_{ijk}}\left[\sum_{k=1}^{n}\left|w_{ijk}-x_k\right|\right]\right\}=\\&=h\left(\left|\mathbf{i}-\mathbf{g}\right|,t\right)\frac{\partial}{\partial w_{ijk}}\left|w_{ijk}-x_k\right|=\end{aligned} \tag{2.33}$$

la función valor absoluto no es derivable en el origen. Considerando el caso $w_{ijk}>0$, se tiene

$$\nabla_{w_{ijk}}E_1\left(t\right)=h\left(\left|\mathbf{i}-\mathbf{g}\right|,t\right)\frac{\partial}{\partial w_{ijk}}\left(w_{ijk}-x_k\right)=h\left(\left|\mathbf{i}-\mathbf{g}\right|,t\right) \tag{2.34}$$

y para el caso $w_{ijk}\le 0$

$$\nabla_{w_{ijk}}E_1\left(t\right)=h\left(\left|\mathbf{i}-\mathbf{g}\right|,t\right)\frac{\partial}{\partial w_{ijk}}\left(-\left(w_{ijk}-x_k\right)\right)=-h\left(\left|\mathbf{i}-\mathbf{g}\right|,t\right) \tag{2.35}$$

Agrupando ambas expresiones haciendo uso de la función signo

$$y=sign\left(x\right)=\begin{Bmatrix}-1 & si & x<0\\0 & si & x=0\\+1 & si & x>0\end{Bmatrix} \tag{2.36}$$

y llamando $\alpha(t) = \lambda(t)$ la regla de aprendizaje queda

$$\mathbf{w}_i(t+1) = \mathbf{w}_i(t) + \alpha(t) h\left(|\mathbf{i} - \mathbf{g}|, t\right) sign\left(x_k - w_{ijk}\right) \tag{2.37}$$

la fórmula obtenida, aunque con apariencia similar a la euclídea, es mucho más simple de realizar, como se puede apreciar sólo con rescribirla así

$$\Delta w_{ijk}(t) = \begin{Bmatrix} +\alpha h & si & x_k(t) > w_{ijk}(t) \\ 0 & si & x_k(t) = w_{ijk}(t) \\ -\alpha h & & x_k(t) < w_{ijk}(t) \end{Bmatrix} \tag{2.38}$$

Se ha comparado el modelo basado en la distancia de Manhattan con el convencional euclídeo, haciendo uso en ambos de la regla de actualización euclídea (2.3) realizando numerosas simulaciones se llega a la conclusión de que, aunque ambos alcanzan resultados parecidos, los del modelo euclídeo son alrededor de un 2% mejores (no obstante, esta pequeña diferencia puede deberse a que en la referencia citada en el modelo de Manhattan se hace uso de (2.3) en lugar de (2.37), con lo que la regla de aprendizaje no es compatible con la métrica empleada).

En otros estudios se concluye que este modelo proporciona resultados similares a los de la regla euclídea, aunque es más sensible a la variación de los parámetros de aprendizaje, que deben ser más cuidadosamente elegidos[21].

2.9 Regla de aprendizaje derivada de la correlación o producto escalar

Si se toma como base el criterio de la correlación, la neurona vencedora es aquella cuyo vector de pesos presenta la máxima correlación con el vector de entrada actual, dado por (2.16).

Definiremos la muestra de una función error para un tiempo t en la forma

$$E_2(t) = \sum_i h\left(|\mathbf{i} - \mathbf{g}|, t\right) f\left(c_{ij}\left(\mathbf{x}(t), \mathbf{w}_i(t)\right)\right) \tag{2.39}$$

[21] Martín, B. y Sanz A. (2002). *Redes Neuronales y Sistemas Difusos*, 2da edición Alfaomega Ra-ma. Madrid. Pp. 117.

con $c_{ij}(\cdot)$ un cierto criterio de similitud, que es mayor cuanto más parecidos sean $\mathbf{x}$ y $\mathbf{w}_i$, y $f(\cdot)$, una cierta función que se introduce por generalidad. En esta ocasión, se trata de maximizar $E_2(t)$, y se obtiene una solución aproximada iterando de la forma conocida

$$\mathbf{w}_i(t+1) = \mathbf{w}_i(t) + \lambda(t)\nabla_{\mathbf{w}_i} E_2(t) \tag{2.40}$$

con $\lambda(t)$ el ritmo de aprendizaje. Obsérvese que en este caso hemos cambiado el signo en el gradiente, puesto que para maximizar debe efectuarse un ascenso por él.

Consideremos que $f(c) = c_{ij}$, la ecuación (2.39) se convierte en

$$E_2(t) = \sum_i h(|\mathbf{i}-\mathbf{g}|,t)\left[\sum_{k=1}^{n} w_{ijk} x_k\right] \tag{2.41}$$

y calculando el gradiente

$$\nabla_{w_{ijk}} E_2(t) = h(|\mathbf{i}-\mathbf{g}|,t)\frac{\partial}{\partial w_{ijk}}\left[\sum_{k=1}^{n} w_{ijk} x_k\right] = h(|\mathbf{i}-\mathbf{g}|,t)\, x_k \tag{2.42}$$

se obtiene la regla de actualización

$$w_{ijk}(t+1) = w_{ijk}(t) + \alpha(t)\, h(|\mathbf{i}-\mathbf{g}|,t)\, x_k \tag{2.43}$$

Esta regla presenta el problema de que con las sucesivas presentaciones de los $\mathbf{x}$, los pesos pueden crecer indefinidamente, para evitarlo hay que normalizar los pesos en cada iteración

$$w_{ijk}(t+1) = \frac{w_{ijk}(t) + \alpha(t)\, h(|\mathbf{i}-\mathbf{g}|,t)\, x_k}{\left\|\mathbf{w}_i(t) + \alpha(t)\, h(|\mathbf{i}-\mathbf{g}|,t)\,\mathbf{x}\right\|} \tag{2.44}$$

Esta es, junto con la euclídea, una de las reglas de aprendizaje más conocidas y empleadas. Un grave problema de este algoritmo de aprendizaje es que la normalización de los pesos que se debe efectuar en cada paso supone un alto costo computacional. Por ello, se deducirá a partir de (2.44) un nuevo algoritmo de

aprendizaje que preserve la norma de los vectores de pesos; así, si están normalizados inicialmente, también lo estarán en cada iteración, de manera que no sea preciso normalizarlos en cada paso.

Para ello, suponiendo que $\|\mathbf{w}_{ij}(t)\|=1$, y pretendiendo obtener una regla para que también lo estén en $t+1$. La expresión del desarrollo en serie de Taylor de una cierta función $f(x)$ en torno a un punto a es

$$f(x)=f(a)+f'(a)\cdot(x-a)+\frac{1}{2!}f''(a)\cdot(x-a)^2+... \tag{2.45}$$

reescribiendo la regla de aprendizaje (2.44) en la forma

$$w_{ijk}(t+1)=\frac{w_{ijk}(t)+\alpha h x_k}{\|\mathbf{w}_i(t)+\alpha h\mathbf{x}\|}=\frac{w_{ijk}(t)+\alpha h x_k}{L(\mathbf{w}_i(t)+\alpha h\mathbf{x})} \tag{2.46}$$

considerando la norma $L(x)$ como una función dependiente del parámetro α y desarrollando en serie en torno a $\alpha=0$, resulta

$$L(\alpha)=L(0)+\left(\frac{dL}{d\alpha}\right)_{\alpha=0}\cdot\alpha+O(\alpha^2)=1+\left(\frac{dL}{d\alpha}\right)_{\alpha=0}\cdot\alpha+O(\alpha^2) \tag{2.47}$$

donde se han supuesto pesos iniciales normalizados $L(0)=1$.

De la expresión de la norma al cuadrado

$$\begin{aligned} L^2(\mathbf{w}_i+\alpha h\mathbf{x}) &= (\mathbf{w}_i+\alpha h\mathbf{x})^T(\mathbf{w}_i+\alpha h\mathbf{x}) \\ &= \|\mathbf{w}_i\|^2+\alpha^2h^2\|\mathbf{x}\|^2+2\alpha h\mathbf{w}_i^T\cdot\mathbf{x} \\ &= 1+\alpha^2h^2\|\mathbf{x}\|^2+2\alpha h\mathbf{w}_i^T\cdot\mathbf{x} \end{aligned} \tag{2.48}$$

se obtiene su derivada

$$\frac{dL}{d\alpha}=\frac{2\alpha h^2\|\mathbf{x}\|^2+2h\mathbf{w}_i^T\cdot\mathbf{x}}{2\sqrt{1+\alpha^2h^2\|\mathbf{x}\|^2+2h\mathbf{w}_i^T\cdot\mathbf{x}}}$$

y por tanto

$$\left(\frac{dL}{d\alpha}\right)_{\alpha=0} = h\mathbf{w}_i^T \cdot \mathbf{x}$$

Así, de (2.47) resulta

$$L(\alpha) = 1 + \alpha h\mathbf{w}_i^T \cdot \mathbf{x} + O\left(\alpha^2\right)$$

con lo que los pesos en $t+1$ quedan

$$w_{ijk}(t+1) = \frac{w_{ijk}(t) + \alpha h x_k}{L\left(\mathbf{w}_i(t) + \alpha h\mathbf{x}\right)} = \frac{w_{ijk}(t) + \alpha h x_k}{\left(1 + \alpha h\mathbf{w}_i^T \cdot \mathbf{x} + O\left(\alpha^2\right)\right)}$$

$$= \left(w_{ijk}(t) + \alpha h x_k\right)\left(1 - \alpha h\mathbf{w}_i^T \cdot \mathbf{x} + O\left(\alpha^2\right)\right)$$

y desarrollando

$$\begin{aligned} w_{ijk}(t+1) &= w_{ijk}(t) + \alpha h x_k - \alpha h w_{ijk}(t)\,\mathbf{w}_i^T \cdot \mathbf{x} + O\left(\alpha^2\right) \\ &\cong w_{ijk}(t) + \alpha h\left(x_k - \left(\mathbf{w}_i^T \cdot \mathbf{x}\right) w_{ijk}(t)\right) \end{aligned} \tag{2.49}$$

considerando despreciables los términos $O\left(\alpha^2\right)$ por ser α pequeño. La expresión (2.49) coincide con la regla de aprendizaje que se propone en Kohonen, y que se puede escribir en la forma

$$w_{ijk}(t+1) = w_{ijk}(t) + \alpha h\left(x_k - y_{ij}(t) \cdot w_{ijk}(t)\right)$$

donde se denomina $y_{ij}(t)$ al producto escalar del vector de entradas por el de pesos de la neurona (i, j), que se consideró como salida de la neurona (i, j).

Capítulo 3

3.- Ejemplo comparativo

En este capítulo se mostrarán las soluciones generadas por cada uno de los tres métodos a comparar, se identificarán las ventajas y desventajas de cada uno desde su implementación hasta la interpretación de resultados.

3.1 Introducción

Cuando se desea encontrar una propuesta de solución a un problema de clasificación se recomienda evaluar diferentes alternativas, es decir, obtener soluciones mediante diferentes técnicas y con diversos escenarios con la finalidad de probar la consistencia de las soluciones o bien, elegir la que mejor se adecue al problema planteado.

Tanto el análisis *cluster* (también conocido como análisis de conglomerados) como los árboles de decisión son dos herramientas muy utilizadas para la clasificación, sin embargo, se ha demostrado la eficiencia de los mapas autoorganizados en problemas reales, incluyendo la clasificación, el reconocimiento de patrones y la reducción de dimensiones entre otras cosas.

En este capítulo se mostrará mediante un ejemplo clásico que los mapas autoorganizados pueden ser utilizados como un método alternativo o complementario a los métodos de clasificación tradicionales (análisis *cluster* y árboles de decisión), además de identificar las ventajas y desventajas así como las principales similitudes y diferencias entre estos tres métodos desde su implementación hasta la interpretación de los resultados generados.

En la actualidad existe una gran variedad de paquetes para implementar los métodos estadísticos tradicionales, por otro lado, existen varios paquetes para la implementación de redes neuronales artificiales, de hecho algunos son de distribución gratuita. Sin embargo, son pocos los que tienen integrados ambos métodos (métodos estadísticos y redes neuronales).

Se evaluaron diferentes alternativas para elegir el paquete con el que se realizaría la implementación tomando en cuenta la limitada oferta de paquetes que cuentan tanto con módulos estadísticos como de redes neuronales (específicamente mapas autoorganizados). Se evaluaron las siguientes opciones obteniendo los siguientes resultados:

1. **"Clementine":** se compone de un módulo Base + módulo de Clasificación (árboles de decisión) + módulo de Segmentación (análisis *cluster* en sus diferentes variantes) + módulo de Asociación + módulo de minería de datos (incluyendo mapas autoorganizados). Es decir, mediante este paquete es posible desarrollar cada uno de los métodos propuestos en este trabajo, sin embargo, no cuenta con versiones de prueba o para estudiantes y el costo es bastante elevado[22].

2. **"MatLab":** ofrece la posibilidad (en su versión completa) de realizar análisis *cluster*, redes neuronales y árboles de decisión; sin embargo, no incluye una amplia variedad de métodos para análisis *cluster* y de árboles de decisión.

3. **"SPSS":** cuenta con diversos métodos para el análisis *cluster* y árboles de decisión; sin embargo, la sección de redes neuronales no incluye mapas autoorganizados (sólo incluye modelos de Función de Base Radial y perceptrón multicapa).

Debido a que ninguno de los paquetes evaluados cumplía con las características deseables para poder realizar la implementación de los tres métodos, se optó realizar el ejercicio en dos de ellos (MatLab y SPSS) de tal forma que la implementación para los mapas autoorganizados se realizará mediante MatLab, mientras que el análisis *cluster* y los árboles de decisión mediante SPSS (los árboles de decisión mediante un módulo específico denominado AnswerTree).

[22] De acuerdo a una cotización en febrero de 2008 por parte de SPSS México, los precios en dólares para licencias monousuario son: Módulo Base $13,200; Módulo de Clasificación $3,600; Módulo de Segmentación $3,600; y Módulo de Minería de Datos $14,400. Es decir $34,800 dólares en total.

Se describen a continuación las principales características de los paquetes a utilizar:

MatLab es la abreviatura de *MATrix LABoratory*. Se trata de un software matemático muy versátil, entre sus características básicas se encuentran la manipulación de matrices, la representación de datos y funciones, la implementación de algoritmos, la creación de interfaces de usuario y la comunicación con programas en otros lenguajes y con otros dispositivos *hardware*. MatLab ofrece un entorno de desarrollo integrado con un lenguaje de programación propio (lenguaje M) así como ciertas librerías ("*Toolboxes*") con las que puede extender aún más sus capacidades. Es muy usado en universidades y centros de investigación y desarrollo.

SPSS es un programa estadístico informático ampliamente utilizado tanto para fines educativos como aplicaciones reales. Originalmente SPSS fue creado como el acrónimo de *Statistical Package for the Social Sciences*. En la actualidad, la sigla se usa tanto para designar el programa estadístico como la empresa que lo produce.

Como programa estadístico es muy popular su uso gracias a la capacidad que tiene para trabajar con bases de datos de gran tamaño. El programa consiste en un módulo base y módulos adicionales (regresión, tablas, reducción de datos, tendencias, categorías, etc.) que se han ido actualizando constantemente con nuevos procedimientos estadísticos. **AswerTree** es un módulo específico de SPSS que cuenta con varios métodos para la construcción de árboles de decisión.

Cada uno de estos paquetes por sí mismos proporcionan varias alternativas (métodos) para realizar el análisis, para el ejemplo comparativo se mostrarán únicamente los resultados de aquel método que haya arrojado los mejores resultados.

En la mayoría de los problemas reales, se desconoce el número de grupos en los que los datos tienen que ser clasificados, normalmente se parte de un número que el investigador considera razonable de acuerdo a su experiencia, incrementándolo o disminuyéndolo hasta que finalmente (en la mayoría de los casos) se elige el método cuya clasificación genere un mejor entendimiento y tratamiento de la información.

En nuestro caso, al tratarse de un ejemplo conocido del cual evidentemente ya se conocen los resultados, es decir, sabemos que hay tres grupos (las tres especies a clasificar) así como la especie real a la que pertenece cada lirio, la comparación de los resultados no tiene como propósito encontrar qué método puede determinar el número ideal de grupos que genere el menor error de clasificación; el objetivo más

bien, está enfocado en determinar qué método genera el menor error de clasificación partiendo del hecho de que los lirios se tiene que clasificar en tres grupos.

De igual forma, los análisis previos recomendados al realizar cualquier tipo de análisis multivariado (como gráficos para ver el comportamiento natural de los datos, correlación entre variables, etcétera), por las características de este ejercicio pueden omitirse, sin embargo, se recomienda siempre hacer este tipo de pruebas para aplicaciones reales en las que no sabemos *a priori* cómo se están comportando los datos.

3.2 Descripción del conjunto de datos

El conjunto de datos "Iris" (Fisher, 1936) es probablemente el más conocido entre los textos sobre clasificaciones. Si bien, la clasificación de Fisher es relativamente simple, su estudio representa un clásico en la materia y con frecuencia se recurre a él como material de referencia.

El conjunto de datos a estudiar contiene información sobre 3 diferentes especies de lirios ("setosa", "versicolor" y "virginica") **(Figura 3.1.)**, las cuales difieren en la medida (ancho y longitud) de sus pétalos y sépalos[23].

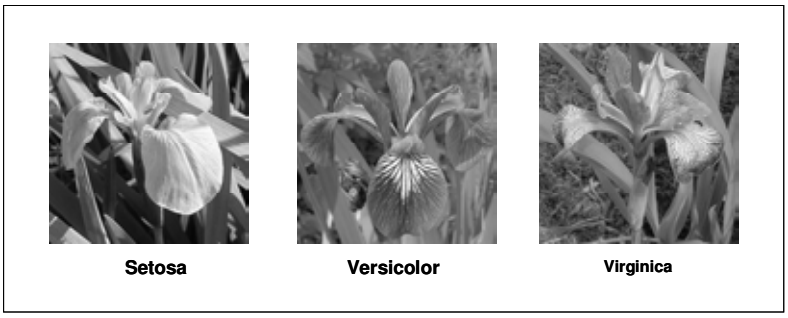

Figura 3.1. Tipos de lirios

El archivo de datos para este ejemplo contiene cuatro variables de medida continuas en cada observación (ancho del pétalo, longitud del pétalo, ancho del sépalo y longitud del sépalo) y una variable de clasificación denominada "especies" la cual muestra la especie real a la que pertenece cada lirio.

[23] Cada una de las hojas, generalmente de color verde, que forman el cáliz de una flor.

En el caso de SPSS el archivo de datos se encuentra en formato *sav* (formato mediante el cual SPSS almacena las bases de datos); en el caso de MatLab se trata de este mismo archivo pero en formato de archivo de datos (.data).

3.3 Clasificación de lirios mediante mapas autoorganizados

Para poder realizar la implementación de mapas autoorganizados en MatLab es necesario instalar "*SOM Toolbox*" la cual contiene las funciones necesarias para la creación, visualización y análisis de estos mapas[24].

Esta librería se encuentra disponible sin cargo en la siguiente dirección electrónica: ***http://www.cis.hut.fi/projects/somtoolbox/***

Al igual que en SPSS, en la librería SOM se pueden insertar etiquetas de texto asociadas a cada dato de la muestra; estas etiquetas resultan útiles para la interpretación de los resultados.

El archivo de datos de los lirios debe cargarse en MatLab mediante la función `som_read_data` y dado que los datos se encuentran en distintas escalas de medición deben ser estandarizados de tal forma que cada variable tenga varianza 1. Una vez que el conjunto de datos está listo, la red es entrenada.

Dado que el conjunto de datos está etiquetado, el mapa también debe ser etiquetado utilizando `som_autolabel`. Es importante señalar que aunque se está etiquetando el mapa, el aprendizaje es no supervisado, es decir, la variable "especies" que es la que muestra a qué especie pertenece cada lirio no se utiliza durante el análisis ya que son sólo las 4 variables de medidas las que lo conforman.

```
% Creación del conjunto de datos
sD = som_read_data('iris.data');
sD = som_normalize(sD,'var');
% Creación del mapa autoorganizado SOM
sM = som_make(sD);
sM = som_autolabel(sM,sD,'vote');
```

[24] En el Anexo A se detallan las funciones utilizadas para la implementación del ejemplo del conjunto de datos "Iris" en MatLab.

```
% Visualización básica
som_show(sM,'umat','all','comp',1:
4,'empty','Labels','norm','d');
som_show_add('label', sM, 'subplot', 6);
```

La función `som_make` inicializa y entrena el mapa. El tamaño del mapa, en este caso es de 6 x 14 de tal forma que el mapa está compuesto por 84 neuronas de salida. El entrenamiento se realizó en dos etapas. En la primera etapa, cuyo objetivo consiste en organizar el mapa, se utilizó una tasa de aprendizaje alta igual a 1 y un radio de vecindad también grande igual al diámetro del mapa.

A medida que avanzada el aprendizaje, tanto la tasa de aprendizaje como el radio de la vecindad iban reduciéndose de forma lineal hasta alcanzar los valores (mínimos) 0.05 y 1 respectivamente. En la segunda etapa, cuyo objetivo es el ajuste fino de 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 consto de 1,000 iteraciones mientras que la segunda fase de 2,000 iteraciones.

Una vez entrenado el mapa, se calculó el error de cuantificación promedio el cual es de 0.0156 lo que representa un error de casi el 2%.

Los mapas autoorganizados ofrecen grandes ventajas de visualización, como lo son la matriz de distancias[25] y las componentes. Cada componente muestra los valores de una variable en cada neurona del mapa usando la misma codificación de color descrita para la matriz de distancias.

Esto da la posibilidad de examinar visualmente cada celda (correspondiente a cada neurona del mapa). El mapa se puede visualizar utilizando `som_show`. La matriz de distancias se muestra junto con las cuatro componentes.

También las etiquetas de cada neurona del mapa se muestran en un bloque de celdas utilizando `som_show_add`. Los valores de los componentes se desentandarizan de modo que los valores mostrados en la barra de colores estén dentro del rango de valores original.

[25] La matriz de distancias nombrada en MatLab como U-matrix "*Unified distance matrix*" es la matriz en la que se visualizan las distancias entre las vecindades del mapa, ayuda a visualizar la estructura de los *clusters* del mapa de tal forma que las áreas uniformes o niveles bajos en la matriz muestran los *clusters* existentes mientras que valores altos valores en esta matriz indican la frontera entre ellos.

La siguiente imagen **(Figura 3.2.)** representa la matriz de distancias, misma que muestra en la parte superior izquierda, enseguida se encuentran las cuatro componentes, finalmente en la parte inferior derecha se ubican las etiquetas del mapa. En la matriz de distancias existen hexágonos adicionales entre todas las parejas de neuronas vecinas. Por ejemplo, la neurona en la esquina superior izquierda tiene valores pequeños para la longitud del sépalo, así como para la longitud y ancho del pétalo, y valores relativamente grandes para el ancho del sépalo.

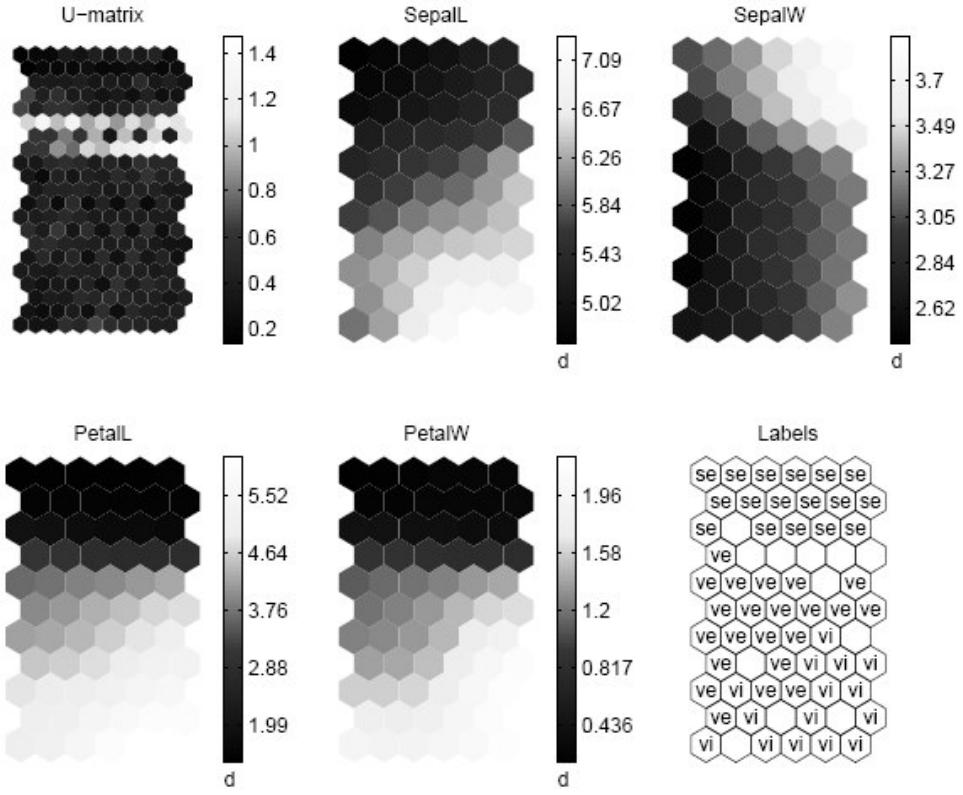

Figura 3.2. Visualización del mapa del conjunto de datos de lirios

La etiqueta asociada a estas neuronas es "se" ("setosa"), y podemos ver también en la matriz de distancias que las neuronas vecinas están muy unidas.

A partir de la matriz de distancias es fácil detectar que las primeras 3 filas del mapa forman un *cluster* claramente definido. Al ver las etiquetas se puede ver que corresponden a la especie "setosa".

Las otras dos especies, "versicolor" y "virginica", forman el otro *cluster*. La matriz U no muestra una clara separación entre estas últimas dos especies, pero por las etiquetas parece que corresponden a dos *clusters* diferentes.

A partir de las cuatro componentes se puede deducir que el largo y el ancho del pétalo están altamente relacionados el uno con el otro. También existe una correlación entre ellos y el largo del sépalo. La especie "setosa" exhibe pétalos pequeños y cortos pero sépalos anchos. El factor de separación entre "versicolor" y "virginica" es que esta última tiene hojas más grandes.

Las gráficas de las componentes son convenientes cuando se tiene que visualizar mucha información de una sola vez. No obstante, cuando solo unas pocas variables son de interés, las gráficas de dispersión son mucho más eficientes. La **Figura 3.3.** muestra la proyección de componentes principales tanto del conjunto de datos como del mapa.

```
% Proyección de componentes principales de los datos
[Pd, V, me] = pcaproj(sD, 3);
% Genera la gráfica de la proyección del mapa
som_grid(sM, 'Coord', pcaproj(sM, V, me), 'marker', 'none', ...
         'Label', sM.labels, 'labelcolor', 'k');
% Gráfica de datos originales con especies diferenciadas por
color
hold on, grid on
colD = [repmat ([1 0 0], 50, 1); ...
        repmat ([0 1 0], 50, 1); ...
        repmat ([0 0 1], 50, 1)];
som_grid('rect',[150 1], 'Line', 'none', 'Coord' ,Pd, ...
         'markercolor', colD);
```

Las tres especies fueron graficadas utilizando distintos colores. El mapa autoorganizado también es proyectado en el mismo sub espacio. Las neuronas vecinas en el mapa están conectadas mediante líneas. Se muestran también las etiquetas de las neuronas asociadas al mapa.

La **Figura 3.4.** visualiza las cuatro variables del SOM usando marcadores de tamaño y color. Se muestran tres coordenadas y marcadores de tamaño y color. Los marcadores de color indican los sub espacios. Los datos se muestran en la parte superior de la gráfica mediante cruces (x).

```
% Desestandarización de los pesos de los vectores
M = som_denormalize (sM. Codebook, sM);
colM = zeros (length (sM.codebook), 3);
un = unique(sD.labels);
for i=1:3, ind = find(strcm(sM.labels, un(i))); colM(ind,i) =1;
end
%Gráfica del mapa
som_grid(sM,'Coord',M(:,2:4),'Markersize',(M(:,1)-4)*5, ...
        'Markercolor', colM);
% Gráfica de los datos
hold on, grid on
D = som_denormalize (sD.data, sD);
Som_grid('rect', [150 1], 'Coord', D(:, 2:4), 'Marker', 'x',...
          'MarkerSize',(D(:,1)-4)*5,'Line','none','Markercolor',
         colD);
% Muestra el mapa y la información de la especie
som_cplane (sM.topol.lattice, sM.topol.msize, colM);
% Muestra las cuatros variables con gráficas de barras
hold on
som_barplane(sM.topol.lattice,sM.topol.msize,M,'w','unitwise')
```

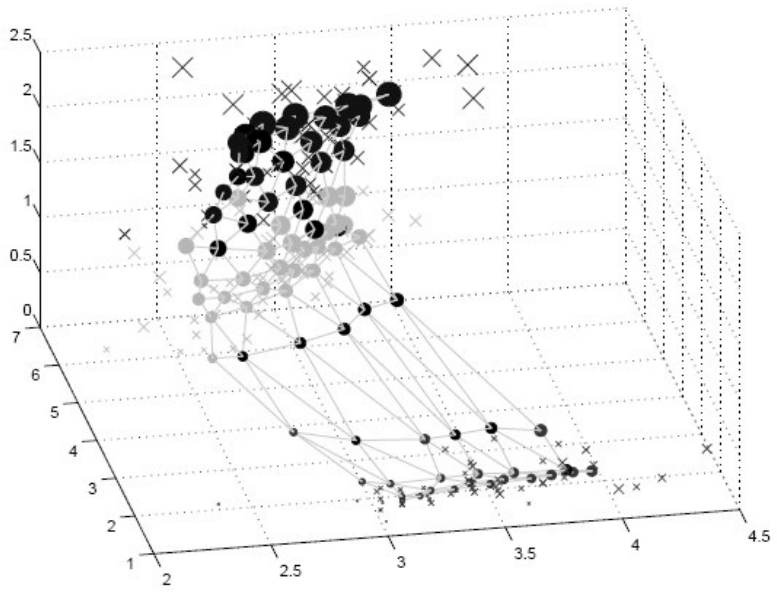

Figura 3.3. Proyección del conjunto de datos

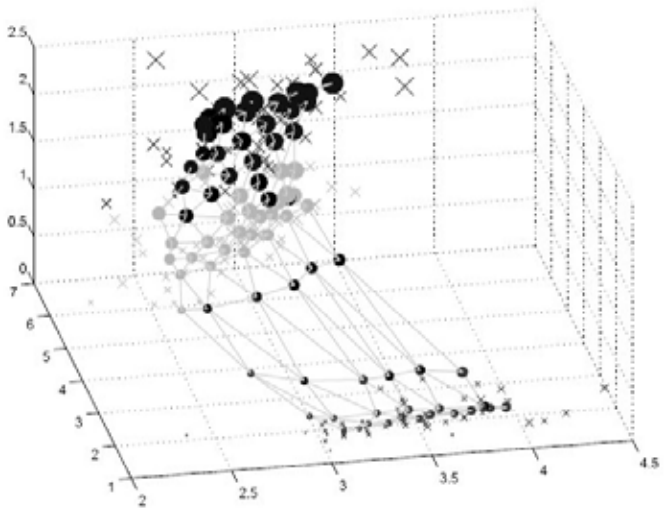

Figura 3.4. Las cuatro variables del SOM usando marcadores de tamaño y color

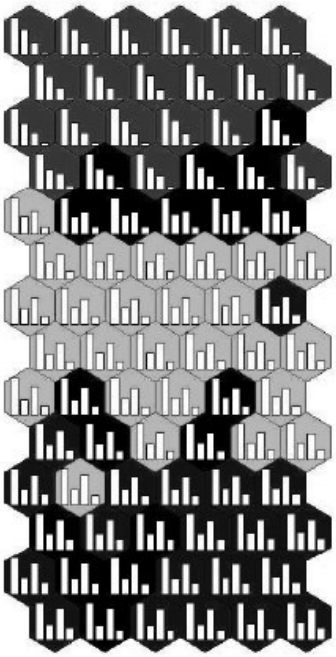

Figura 3.5. Las cuatro variables se muestran con gráficas de barras en cada neurona. El color indica la especie.

La **Figura 3.5.** muestra las cuatro variables del mapa junto con la información de la especie. En cada uno de los hexágonos del mapa se muestra una gráfica compuesta por cuatro barras, cada una de las cuales, asociada a las variables usadas para la clasificación (ancho y largo de los sépalos y pétalos).

3.4 Clasificación de lirios mediante análisis *cluster*

El algoritmo a utilizar es *k-medias*[26]. Este algoritmo (así como los métodos jerárquicos) depende de distancias y dado que las variables no se encuentran en las mismas escalas, es necesario estandarizar los datos antes de realizar el análisis.

Para ello, utilizamos la opción "Guardar variables tipificadas" que se encuentra en la opción de "Estadísticos descriptivos" del menú "Analizar" **(Figura 3.6.)**.

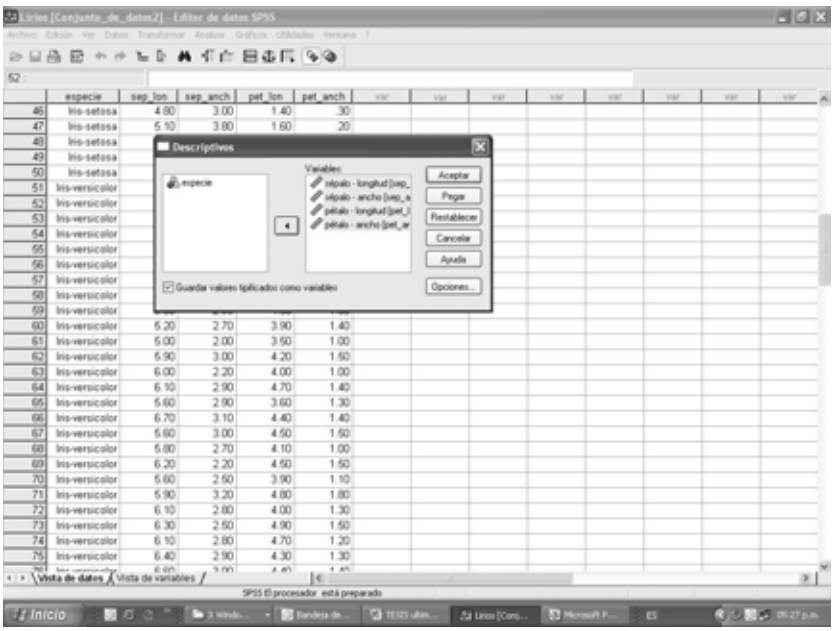

Figura 3.6. Estandarización de los datos

A continuación se selecciona el algoritmo *k-medias* ubicado en el menú "Analizar" en la opción "Clasificar". Como variables se introducen las cuatro medidas de los lirios y se establece como número de grupos tres. En este caso sabemos que en realidad se trata de tres especies diferentes de lirios, si no lo supiéramos de antemano tendríamos que realizar el análisis con diversos escenarios intercambiando el

[26] En el Anexo B se muestran los métodos de los análisis *cluster* incluidos en SPSS así como una visión simplificada de sus aspectos teóricos. Para mayor detalle sobre el análisis *cluster*, se recomienda consultar el libro de Everitt, B.S., Landau, S., and Leese, M. (2001). *Cluster Analysis, Applications and Programming Techniques*. Adison Wesley. USA.

número de grupos o bien utilizando algún otro método de agrupación, por ejemplo, algún método jerárquico y elegir aquel que arroje los mejores resultados tanto estadísticos como acordes con el problema planteado **(Figura 3.7.)**.

Dentro de la opción "Guardar" de esta misma pestaña se selecciona "Guardar conglomerados de pertenencia" y la "Distancia desde el centro del conglomerado" (esta opción debe seleccionarse sólo cuando se cuenta ya con el número ideal de grupos, que en este caso sabemos que es tres). Y en la pestaña "Opciones" seleccionar "Análisis ANOVA".

Figura 3.7. Selección del método a utilizar y especificación de las opciones

Una vez seleccionados los centros de los conglomerados, cada caso es asignado al *cluster* de cuyo centro se encuentra más próximo y comienza un proceso de ubicación iterativa de los centros. En la primera iteración se reasignan los casos por su distancia al nuevo centro y, tras la reasignación, se vuelve a actualizar el valor del centro y así sucesivamente. En la **Figura 3.8.** se resume el historial de iteraciones (8 en este caso) con indicación del cambio (desplazamiento) experimentado por cada centro en cada iteración.

Puede observarse que, conforme avanzan las iteraciones, el desplazamiento de los centros se va haciendo más pequeño, hasta llegar a la octava iteración, en la que ya no existe ningún desplazamiento.

Historial de iteraciones[a]

Iteración	Cambio en los centros de los conglomerados		
	1	2	3
1	1.985	1.786	1.811
2	.034	.281	.236
3	.018	.673	.297
4	.119	.258	.030
5	.122	.177	.000
6	.090	.100	.000
7	.039	.045	.000
8	.000	.000	.000

a. Se ha logrado la convergencia debido a que los centros de los conglomerados no presentan ningún cambio o éste es pequeño. El cambio máximo de coordenadas absolutas para cualquier centro es de .000. La iteración actual es 8. La distancia mínima entre los centros iniciales es de 5.058.

Figura 3.8. Historial de Iteraciones

La **Figura 3.9.** muestra los centros de los conglomerados finales, es decir, los centros de los conglomerados tras el proceso de actualización iterativa. Esta tabla es de utilidad para interpretar la constitución de los *clusters* pues resume los valores centrales en cada *cluster* en las variables de interés.

La **Figura 3.10.** indica la distancia entre cada caso y su centro de clasificación. Se observa que la menor distancia se da entre los tipos de lirios 1 y 2 lo cual puede generar que el mayor traslape se de entre estos dos grupos; de manera semejante la mayor distancia se da entre los tipos de lirios 1 y 3 por lo que el traslape entre estos dos grupos debe ser menor.

Centros de los conglomerados finales

	Conglomerado		
	1	2	3
Puntua: sépalo - longitud	1.03015	-0.16784	-0.99872
Puntua: sépalo - ancho	0.01384	-0.96684	0.89212
Puntua: pétalo - longitud	0.94054	0.25875	-1.29862
Puntua: pétalo - ancho	0.96902	0.17551	-1.25244

Figura 3.9. Centros de los conglomerados finales

Distancias entre los centros de los conglomerados

Conglomerado	1	2	3
1		1.869	3.852
2	1.869		2.934
3	3.852	2.934	

Figura 3.10. Distancias entre los centros de los conglomerados

La tabla ANOVA **(Figura 3.11.)** muestra la tabla resumen del análisis de varianza con un estadístico *F* univariante para cada una de las variables incluidas en el análisis. El análisis de varianza se obtiene tomando los grupos definidos por los grupos como factor y cada una de las variables incluidas en el análisis como variable dependiente.

Una nota al pie de página de la tabla informa que los estadísticos *F* sólo deben utilizarse con una finalidad descriptiva pues los casos no se han asignado aleatoriamente a los conglomerados sino que se han asignado intentando optimizar las diferencias entre los conglomerados.

Además, los niveles críticos asociados a los estadísticos *F* no deben ser interpretados de manera habitual pues el procedimiento de *k medias* no aplica ningún tipo de corrección sobre la tasa de error (es decir, sobre la probabilidad de cometer errores tipo I cuando se llevan a cabo contrastes). Lógicamente, la tabla de ANOVA no se muestra cuando todos los casos son asignados a un único *cluster*.

ANOVA

	Conglomerado		Error			
	cuadrática	gl	cuadrática	gl	F	Sig.
Puntua: sépalo - longitud	54.268	2	0.275	147	197.153	0.000
Puntua: sépalo - ancho	41.004	2	0.456	147	89.975	0.000
Puntua: pétalo - longitud	67.185	2	0.100	147	675.014	0.000
Puntua: pétalo - ancho	64.961	2	0.130	147	500.559	0.000

Las pruebas F sólo se deben utilizar con una finalidad descriptiva puesto que los conglomerados han sido elegidos para maximizar las diferencias entre los casos en diferentes conglomerados. Los niveles críticos no son corregidos, por lo que no pueden interpretarse como pruebas de la hipótesis de que los centros de los conglomerados son iguales.

Figura 3.11. Tabla ANOVA

En la **Figura 3.12.** se muestra el número de casos que se asignó a cada *cluster,* sabemos que en realidad el archivo de datos contiene 50 casos de cada especie, a simple vista parece que los casos faltantes en los *clusters* 1 y 2 fueron asignados al

cluster 1, sin embargo debemos realizar pruebas de validación para evaluar el error en cada asignación de casos.

Número de casos en cada conglomerado

Conglomerado	1	55.000
	2	46.000
	3	49.000
Válidos		150.000
Perdidos		0.000

Figura 3.12. Número de casos en cada conglomerado

3.4.1 Pruebas de validación

Como primera prueba de validación se realizará un análisis de medias en donde la variable dependiente es la variable de agrupación (SPSS genera esta variable y la incluye en la base de datos normalmente con el nombre "QCL_1") y las variables independientes son las variables usadas para el análisis *cluster* sin estandarizar (medidas de ancho y largo de los pétalos y sépalos).

Informe

Número inicial de casos		sépalo - longitud	sépalo - ancho	pétalo - longitud	pétalo - ancho
1	Media	6.6964	3.0600	5.4182	1.9382
	N	55	55	55	55
	Desv. típ.	0.50807	0.26006	0.63540	0.33802
2	Media	5.7043	2.6348	4.2152	1.3326
	N	46	46	46	46
	Desv. típ.	0.42109	0.26265	0.70395	0.31202
3	Media	5.0163	3.4408	1.4673	0.2429
	N	49	49	49	49
	Desv. típ.	0.34842	0.34876	0.17367	0.10801
Total	Media	5.8433	3.0540	3.7587	1.1987
	N	150	150	150	150
	Desv. típ.	0.82807	0.43359	1.76442	0.76316

Figura 3.13. Informe sobre el análisis de medias

En la **Figura 3.13.** observamos que los *clusters* 1 y 3 son los más diferenciados. El *cluster* 1 se encuentra conformado principalmente por aquellos lirios con la mayor longitud en sus pétalos y sépalos así como con los pétalos más anchos que el resto, mientras que en el *cluster* 3 se encuentran los lirios con la longitud de pétalos y sépalos así como el ancho de sus pétalos, más chicos que el resto, pero el ancho de

sépalos más grande. El *cluster* 2 de manera general se encuentra en la mitad de estos dos *clusters*.

Una segunda prueba de validación consiste en usar una variable externa y compararla contra la variable de agrupación generada en el análisis. En nuestro caso la variable de validación es la variable "especies" la cual nos dice el tipo de especie real al que pertenece cada lirio. En este caso la prueba consiste en generar una tabla de contingencia con la variable de validación "especies" en las columnas y la variable de agrupación "QCL_1" en las filas.

Tabla de contingencia Número inicial de casos * especie

			especie			
			Iris-virginica	Iris-versicolor	Iris-setosa	Total
Número inicial de casos	1	Recuento	42	13	0	55
	2	Recuento	8	37	1	46
	3	Recuento	0	0	49	49
Total		Recuento	50	50	50	150

Figura 3.14. Tabla de contingencia

Se observa en la **Figura 3.14.** que el *cluster* 3 contiene la mayor cantidad de la especie "setosa", mientras que los *clusters* 1 y 2 tuvieron el mayor traslape (tal como se había sugerido en la **Figura 3.10.** de acuerdo a las distancias entre los centros de los conglomerados), a pesar de los casos traslapados, la especie con mayor concentración en el *cluster* 2 es "versicolor" mientras que en el *cluster* 1 es "virginica".

La tabla de contingencia ayuda a identificar rápidamente cualquier caso clasificado erróneamente. En nuestro ejemplo, los valores en la diagonal principal (42, 37, 49) representan aquellos casos que fueron clasificados correctamente. Los valores en cualquiera de las demás casillas de la matriz representan casos de clasificación errónea, en total hubo 128 casos clasificados correctamente, representando un 15% de error.

3.5 Clasificación de lirios mediante árboles de decisión

Como primer paso, se selecciona en la pantalla el método a utilizar **(Figura 3.15.)**, en nuestro ejemplo el algoritmo a utilizar es C&RT[27]. A continuación, en la siguiente pantalla se selecciona "especies" como la variable criterio y longitud del pétalo, ancho del pétalo, longitud del pétalo y ancho del sépalo como las variables predictoras **(Figura 3.16.)**.

Dado que este conjunto de datos contiene una muestra muy reducida, estableceremos como reglas de parada que la profundidad máxima del árbol sea de 5 niveles bajo la raíz y que los números mínimos de casos para los nodos parentales y filiales sean de 25 y 1 respectivamente **(Figura 3.17.)**.

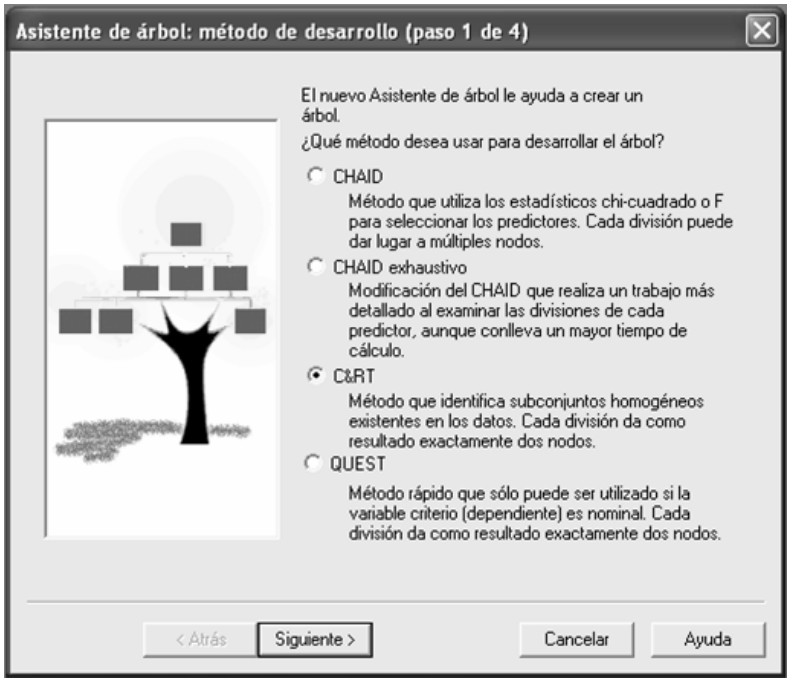

Figura 3.15. Elección del tipo de método a utilizar

[27] En el Anexo C se muestran los métodos de árboles de decisión incluidos en AnswerTree así como una visión simplificada de sus aspectos teóricos. Para mayor detalle, se recomienda consultar la Guía del usuario de AnswerTree 3.1. de SPSS Inc.

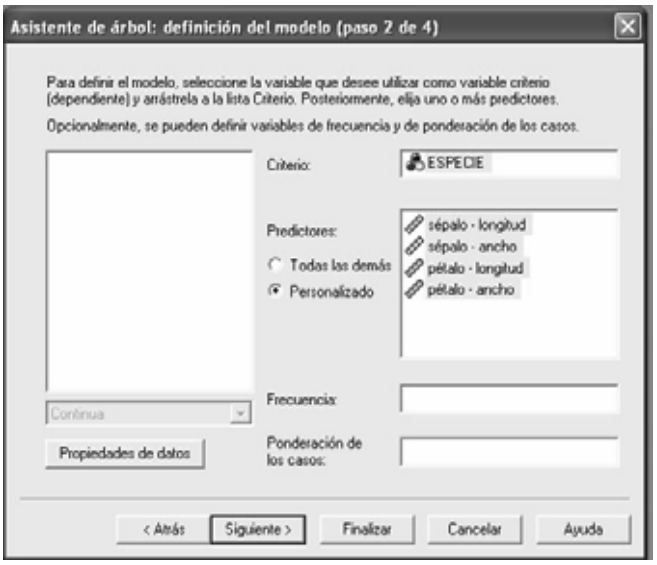

Figura 3.16. Especificación de la variable criterio y las variables predictoras

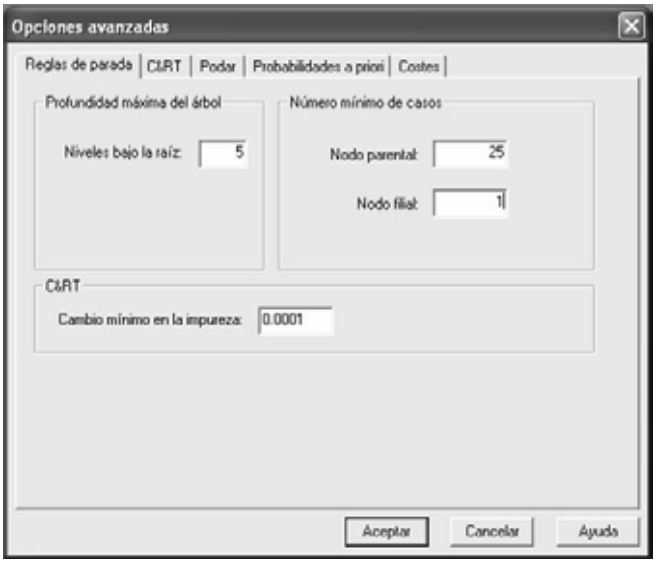

Figura 3.17. Especificación de las reglas de parada

El nodo raíz o nodo "0" representa las frecuencias de la variable criterio "especies" **(Figura 3.18.)**.

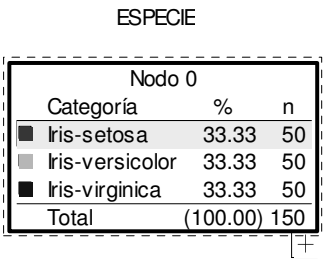

Figura 3.18. Árbol mínimo

Desarrollando el resto del árbol tenemos 4 niveles debajo del nodo raíz **(Figura 3.19.)**. Resulta interesante estudiar los árboles de decisión ya que en ellos se encuentra un historial detallado acerca del análisis que hemos realizado. Parte del desarrollo y de la comprensión que podamos obtener de un análisis basado en un árbol se deriva de las explicaciones de los resultados que seamos capaces de generar.

Haciendo un *zoom* en la parte superior del árbol **(Figura 3.20.)** vemos que para dividir el nodo raíz, se selecciona la longitud del pétalo; los valores utilizados para la división son mayores o menores que el valor de la medida 2.450. Todos los casos en los que la longitud del pétalo sea menor o igual que 2.450 se envían al nodo 1, en tanto que aquellos cuyo valor sea mayor que 2.450 se envían al nodo 2.

El algoritmo C&RT muestra la importancia relativa de la división de un nodo, utilizando la disminución en la impureza o mejora, como criterio de evaluación. En este ejemplo, utilizamos la medida de impureza predeterminada de Gini. En la primera división del árbol, la mejora corresponde a 0.3333.

Esto significa que la impureza de los dos nodos filiales resultante de la división era 0.3333 menor que la impureza del nodo raíz. El nodo 1 está compuesto por una sola especie ("setosa") y contiene todos los casos de dicha especie. En el nodo 2 se encuentran las 100 observaciones restantes, entre las que se observan todos los lirios "versicolor" y "virginica".

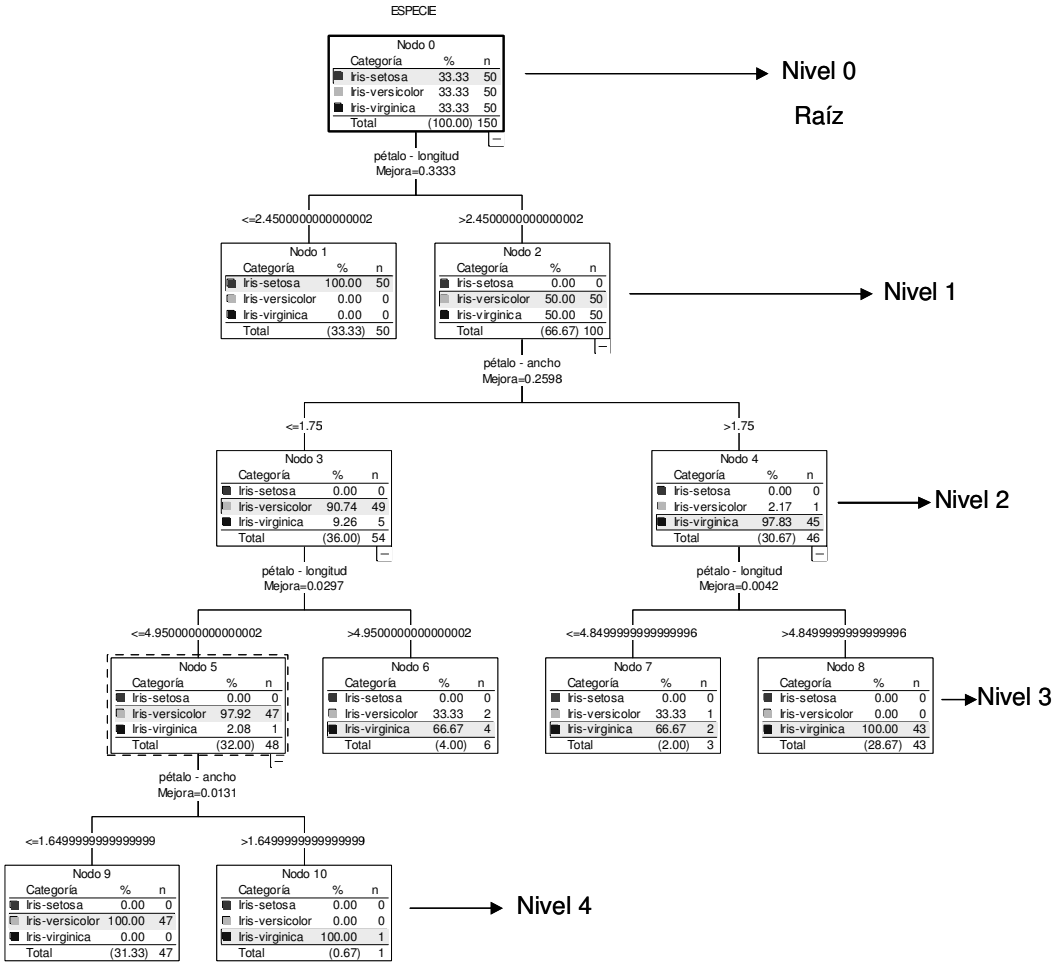

Figura 3.19. Árbol desarrollado

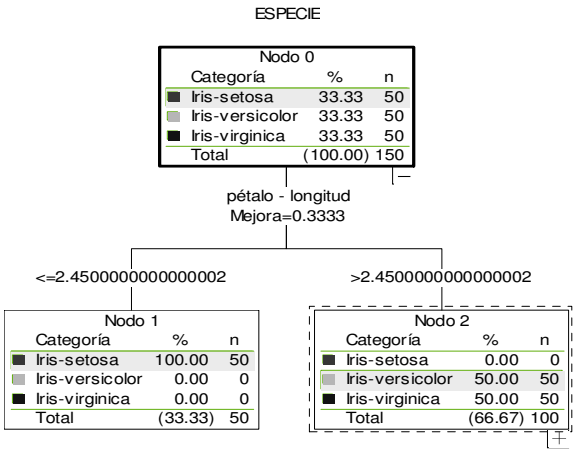

Figura 3.20. Desarrollo en la parte superior (primer nivel del árbol)

Desarrollando ahora el árbol en 2 niveles **(Figura 3.21.)**, se puede identificar que el nodo 1 se ha definido como un nodo terminal (no es posible dividir más este nodo). El nodo 2 se ha dividido utilizando la variable ancho del pétalo y la mejora corresponde a 0.2598. Los dos nodos filiales del nodo 2 describen a grandes rasgos los dos tipos restantes de lirios.

El nodo 3 incluye la mayoría de los lirios de la especie "versicolor", en tanto que el nodo 4 incluye la mayoría de los de la especie "virginica". El estudio de las últimas divisiones ayuda poco a comprender mejor el problema, ya que las divisiones sucesivas incluyen un número de casos muy reducido.

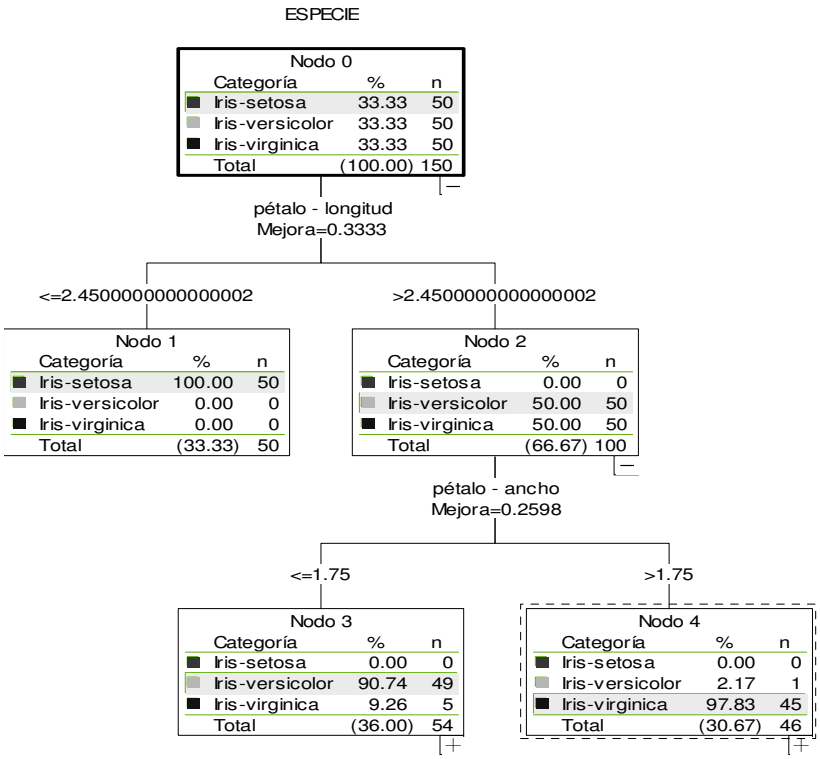

Figura 3.21. Desarrollo en el segundo nivel del árbol

En base a estos resultados podemos concluir que:

Si el lirio tiene pétalos pequeños (≤ 2.450), probablemente se trate del tipo "setosa". Si se tienen en cuenta los lirios con pétalos largos (> 2.450), observaremos que los pétalos angostos (≤ 1.750) corresponden al tipo "versicolor", en tanto que los pétalos anchos (> 1.750), corresponden al tipo "virginica".

Para valorar la capacidad del modelo a la hora de predecir el tipo de lirio, podemos examinar el resumen de riesgos **(Figura 3.22.)**. Dicho resumen compara el tipo de lirio asignado mediante el árbol con el tipo del lirio registrado realmente. La tabla de estadísticos de riesgo ayuda a identificar rápidamente cualquier caso clasificado erróneamente. En nuestro ejemplo, los valores en la diagonal principal (50, 49, 48) representan aquellos casos que fueron clasificados correctamente.

Los valores en cualquiera de las demás casillas de la matriz representan casos de clasificación errónea. La estimación de riesgo revela la proporción de casos clasificados de manera incorrecta. Se aprecia que la tasa de clasificación errónea es bastante baja: todos los casos, salvo tres fueron clasificados correctamente. Esto da como resultado una estimación de riesgos de 0.02 ya que el 2% de los casos se ha clasificado de manera errónea.

Matriz de clasificación errónea					
		Categoría real			
		Iris-setosa	Iris-versicolor	Iris-virginica	Total
Categoría estimada	Iris-setosa	50	0	0	50
	Iris-versicolor	0	49	2	51
	Iris-virginica	0	1	48	49
	Total	50	50	50	150
		Estadísticos de riesgo			
Estmación de riesgo		0,02			
ET de la estimación de riesgo		0,011431			

Figura 3.22. Resumen de riesgos

3.6 Resumen comparativo de los 3 métodos

En la **Tabla 3.1** se resumen las diferencias de cada método entre los aspectos evaluados.

Aspecto evaluado	SOM	Análisis *Cluster* *(k-medias)*	Árboles de decisión *(C&RT)*
Conocimientos previos recomendados	Análisis multivariado, Redes neuronales, Lenguaje M de MatLab.	Análisis multivariado. Análisis *cluster*. Conocimientos medios de SPSS.	Análisis multivariado. Árboles de decisión. Conocimientos básicos de SPSS.
Tipo de aprendizaje utilizado	No supervisado.	No supervisado.	Supervisado.
Dificultad de implementación	De media a alta. Se requiere programar la red y en aplicaciones reales la mayoría de las veces se tienen que modificar los parámetros manejados por default.	Baja. Cualquiera de los 3 métodos puede implementarse y modificarse desde el menú de herramientas de SPSS	Baja. La implementación se lleva a cabo paso por paso, el usuario sólo tiene que ir seleccionando las opciones que requiera en el análisis. Puede caerse en el mal hábito de dejar en automático las opciones que pueden ayudar a generar mejores árboles.
Tipos de datos con los que trabaja (continuos / discretos)	Cualquiera.	Cualquiera, en algunos casos se requieren previas adaptaciones.	Cualquiera.
Procesamiento previo de los datos	Estandarización de las variables a incluir en el modelo.	Estandarización de las variables a incluir en el modelo.	Ninguno.
Especificación *a priori* del número de grupos	No.	Sí.	No.

Tabla 3.1. Cuadro comparativo entre los tres métodos utilizados

Aspecto evaluado	SOM	Análisis *Cluster* (*k-medias*)	Árboles de decisión (*C&RT*)
Dificultad de ejecución	Media. En el ejemplo, el conjunto de datos era pequeño. Se requirieron 3,000 iteraciones para el entrenamiento del mapa.	Baja. Se requirieron 8 iteraciones para conseguir la convergencia del método.	Baja. El árbol se abrió sólo en 4 niveles.
Tasa de error	El error de cuantificación fue de casi el 2%.	15% de error.	2% de error.
Principales resultados / gráficos de visualización	Matriz de distancias y de componentes. Altamente efectivas y sencillas de interpretar.	Centros de los conglomerados finales. Se dificulta su interpretación cuando los datos fueron estandarizados.	Diagrama de árbol muy fácil de interpretar.
Interpretación de resultados	Sencilla a media.	Sencilla a media.	Sencilla.
Manipulación interactiva de los resultados	No.	No.	Sí.
Conclusiones generadas sobre cada tipo de lirio	Existe un *cluster* claramente definido: "setosa". Las otras dos especies forman otro *cluster*. La especie "setosa" exhibe pétalos pequeños y cortos pero sépalos anchos. El factor de separación entre "versicolor" y "virginica" es que esta última tiene hojas más grandes.	El *cluster* 3 contiene la mayor cantidad de la especie "setosa", mientras que los *clusters* 1 y 2 tuvieron el mayor traslape ("versicolor" y "virginica"). A pesar del traslape, la especie con mayor concentración en el *cluster* 2 es "versicolor" mientras que en el *cluster* 1 es "virginica".	Si el lirio tiene pétalos pequeños (≤ 2.45) probablemente se trate del tipo "setosa". Si se tiene en cuenta los lirios con pétalos largos (>2.45), observaremos que los pétalos angostos (≤ 1.75) corresponden al tipo "versicolor", en tanto que los pétalos anchos (>1.75) corresponden al tipo "virginica".
Posibilidad de generar reglas de decisión (reglas de clasificación para un análisis discriminante posterior)	Sí	Sí	Sí

Tabla 3.1. …*Continuación.* Cuadro comparativo entre los tres métodos utilizados

Conclusiones

En general las RNA han mostrado una capacidad clasificatoria igual o superior que las técnicas estadísticas. El modelo mediante análisis *cluster* obtuvo una tasa de error del 15%, mientras que, tanto el modelo de mapas autoorganizados como el modelo de árboles de decisión registraron una tasa de error del 2%; sin embargo, hay que recordar que este último parte de un método de clasificación supervisado lo cual representa una ventaja.

Si bien, la dificultad de implementación y ejecución de las RNA es ligeramente superior que para los métodos estadísticos, la interpretación resulta bastante sencilla.

Entre las propiedades de las RNA que han llamado la atención de los estadísticos destacan las relativas a su buen rendimiento ante problemas no lineales o datos con mucho ruido, y el poderse utilizar independientemente del cumplimiento de los supuestos teóricos relativos a las técnicas estadísticas.

Otro punto a considerar es el fácil acceso a los paquetes estadísticos actuales pues resultan hoy en día más económicos de aplicar que las RNA en cuanto a los recursos temporales y computacionales involucrados. Se recomienda valorar dependiendo de la complejidad de cada problema si vale la pena ganar en capacidad clasificatoria a costa de incrementar el costo computacional y de recursos involucrados (tanto en complejidad como en aspectos tecnológicos) en el entrenamiento de las RNA.

La consideración de todo lo dicho nos lleva a sugerir que técnicas estadísticas y RNA, pueden complementarse adecuadamente (modelos híbridos). De este modo, la estadística, centrada tradicionalmente en problemas lineales, y las RNA, más acostumbradas a tratar con problemas de categorías mal definidas, relaciones no lineales o datos con mucho ruido, se verán mutuamente enriquecidas. No es de extrañar por ello, que paquetes estadísticos ya comiencen a incorporar en sus últimas versiones módulos de redes neuronales artificiales.

De esta forma, se cubre satisfactoriamente el objetivo de este trabajo, dejando a consideración del lector la elección entre un método u otro (o bien la combinación de ellos). El presente material puede utilizarse como base para nuevos trabajos en los que se busque la solución a problemas reales.

Anexo A

SOM Toolbox en MatLab

A continuación se muestran las principales funciones que se emplean para la construcción y visualización de los SOM. Para mayor detalle se recomienda consultar la Guía de Usuario, "Neural Network Toolbox 5".

1. Construcción de conjuntos de datos

Se pueden utilizar las funciones normales de MatLab tales como `load` y `scan`; sin embargo, SOM Toolbox tiene la función `som_read_data` la cual puede ser usada para leer archivos ASCII.

2. Preprocesamiento de los datos

Dado que el algoritmo SOM usa la métrica euclidiana para medir la distancia entre vectores, los datos deben ser estandarizados en el caso de que las variables no se encuentren en la misma escala de medición (normalmente se requiere que las variables sean igualmente importantes, de otra forma las variables con valores más grandes dominarían la organización del mapa). La función `sD = som_normalize (sD, 'var')` o `D = som_normalize (D, 'var')` escala las variables de tal forma que su varianza sea igual a 1.

3. Inicialización y entrenamiento

Se manejan dos formas de inicialización (aleatoria o lineal) y dos formas de entrenamiento (secuencial y en bloques). Por *default* se maneja la inicialización lineal y el entrenamiento por bloques. La forma más simple de inicializar y entrenar un SOM es mediante la función `som_make`.

Esta función inicializa y entrena el mapa. El entrenamiento se realiza en dos etapas: una etapa de organización del mapa con valores grandes (iniciales) para el radio de la

vecindad y para la tasa de aprendizaje, la segunda etapa comprende el ajuste fino del mapa para lo cual se utilizan valores pequeños para el radio de la vecindad y la tasa de aprendizaje.

Esta función también selecciona el tamaño del mapa y los parámetros de entrenamiento automáticamente, sin embargo, se pueden modificar sus argumentos de tal forma que se puedan modificar las opciones, como por ejemplo, el tamaño del mapa.

Si se desea tener control sobre los parámetros de entrenamiento se puede optar por las funciones `som_lininit`, `som_randinit`, `som_seqtrain` y `som_batchtrain`.

4. Visualización y análisis

Las cuadrículas de visualización del SOM pueden ser usadas como una forma sencilla para mostrar las características de los mapas (y de los datos mismos).

4.1 Visualización de células

La función básica es `som_show`, por *default* esta función muestra la matriz de distancias (*U-matrix*) calculada en base a todas las variables y a las componentes.

Cada componente muestra los valores de una variable en cada neurona del mapa. Los valores son mostrados mediante colores. Esta función tiene varios argumentos de entrada que pueden ser modificados para tener más control sobre el tipo de planos a mostrar y en qué orden.

4.2 Visualización de gráficas

1. **Gráficas de pie** (`som_pieplane`): es ideal para mostrar valores proporcionales.

2. **Gráficas de barras** (`som_barplane`): es ideal para mostrar valores en diferentes categorías.

3. **Gráficas de señal** (`som_plotplane`): muestra los vectores de códigos como gráficas de líneas.

En todos los casos, los colores y tamaños de la gráfica pueden ser modificados mediante la manipulación de sus argumentos.

4.3 Visualización de mallas

La función `som_grid` puede ser utilizada para crear gráficas de mallas. La función se basa en la idea de que la visualización de un conjunto de datos, consiste simplemente en un conjunto de objetos con una única posición, color y tamaño.

Anexo B

Análisis *cluster*

SPSS incluye tres procedimientos de *clusterización*: en dos fases, jerárquico o de *K-medias*. Cada uno de estos procedimientos emplea un algoritmo distinto en la creación de grupos y contiene opciones que no están disponibles en los otros.

1. **Análisis de *K-medias*:** El uso del procedimiento requiere que el usuario especifique previamente el número de *clusters*. Ofrece una serie de funciones que se detallan a continuación:

    - Posibilidad de guardar las distancias desde los centros de los conglomerados hasta los distintos objetos.
    - Posibilidad de leer los centros de los conglomerados iniciales y guardar los centros de los conglomerados finales desde un archivo SPSS externo.
    - Puede analizar archivos con una gran cantidad de datos.

2. **Análisis jerárquico:** Su uso se limita a archivos de datos más pequeños y ofrece una serie de funciones que se detallan a continuación:

    - Posibilidad de agrupar casos o variables.
    - Posibilidad de calcular un rango de soluciones posibles y guardar los conglomerados de pertenencia para cada una de dichas soluciones.
    - Distintos métodos de formación de *clusters*, transformación de variables y medida de disimilaridad.
    - Siempre que todas las variables sean del mismo tipo, el procedimiento podrá analizar variables de intervalo (continuas), de recuento o binarias.

3. **Análisis en dos fases:** Ofrece una serie de funciones que se detallan a continuación:

    - Selección automática del número más apropiado de *clusters* y medidas para la selección de los distintos modelos.

- Posibilidad de crear modelos basados al mismo tiempo en variables categóricas y continuas.
- Posibilidad de guardar el modelo en un archivo *xml* externo y, a continuación, leer el archivo y actualizar el modelo con datos más recientes. Asimismo, este procedimiento puede analizar archivos con una gran cantidad de datos.

1. Análisis de *K-medias*

Este procedimiento intenta identificar grupos de casos relativamente homogéneos basándose en las características seleccionadas y utilizando un algoritmo que puede gestionar un gran número de casos. Sin embargo, el algoritmo requiere que el usuario especifique el número de *clusters*. Se puede elegir uno de los dos métodos disponibles para clasificar los casos: la actualización de los centros de los conglomerados de forma iterativa o sólo la clasificación.

Asimismo, puede guardar la pertenencia a los conglomerados, información de la distancia y los centros de los conglomerados finales. Si las variables son binarias o recuentos, se recomienda utilizar el análisis jerárquico.

Las distancias se calculan utilizando la distancia euclídea simple. Si desea utilizar otra medida de distancia o de similaridad, se recomienda utilizar el procedimiento de análisis jerárquico. El escalamiento de las variables es una consideración importante.

Así, si las variables utilizan diferentes escalas los resultados podrían ser erróneos. En estos casos, se debe considerar la estandarización de las variables antes de realizar el análisis (esta tarea se puede hacer en el menú "Descriptivos", ya que no está incluida dentro del mismo procedimiento como en el caso del análisis jerárquico).

1.1 Especificaciones del método

1.1.1 Iterar

Número máximo de iteraciones. Limita el número de iteraciones. La iteración se detiene después de este número de iteraciones, incluso si no se ha satisfecho el criterio de convergencia. Este número debe estar entre el 1 y el 999.

Criterio de convergencia. Determina cuándo finaliza la iteración. Representa una proporción de la distancia mínima entre los centros iniciales de los conglomerados, por lo que debe ser mayor que 0 pero no mayor que 1. Por ejemplo, si el criterio es igual a 0.02, la iteración cesará si una iteración completa no mueve ninguno de los centros de los conglomerados en una distancia superior al dos por ciento de la distancia menor entre cualquiera de los centros iniciales.

Usar medias actualizadas. Permite solicitar la actualización de los centros de los conglomerados tras la asignación de cada caso. Si no selecciona esta opción, los nuevos centros de los conglomerados se calcularán después de la asignación de todos los casos.

1.1.2 Guardar

Puede guardar información sobre la solución como nuevas variables para que puedan ser utilizadas en análisis posteriores:

Conglomerado de pertenencia. Crea una nueva variable que indica el conglomerado final al que pertenece cada caso.

Distancia desde centro del conglomerado. Crea una nueva variable que indica la distancia euclídea entre cada caso y su centro de clasificación.

1.1.3 Opciones

Estadísticos. Puede seleccionar los siguientes estadísticos: "Centros de conglomerados iniciales", "Tabla de ANOVA" e "Información del conglomerado para cada caso".

- **Centros de conglomerados iniciales.** Primera estimación de las medias de las variables para cada uno de los *clusters*. Los centros iniciales de los conglomerados se utilizan como criterio para una primera clasificación y, a partir de ahí, se van actualizando.

- **Tabla de ANOVA.** Muestra una tabla de análisis de varianza que incluye las pruebas F univariadas para cada variable de aglomeración.

- Las pruebas *F* son sólo descriptivas y las probabilidades resultantes no deben ser interpretadas. La tabla de ANOVA no se mostrará si se asignan todos los casos a un único *cluster*.

- **Información del conglomerado para cada caso.** Muestra, para cada caso, el *cluster* final asignado y la distancia euclídea entre el caso y el centro del *cluster* utilizado para clasificar el caso.

1.2 Orden de casos y centro de conglomerados iniciales

El algoritmo por defecto para elegir centros de conglomerados iniciales no es invariable con respecto a la ordenación de casos. La opción *"Usar medias actualizadas"* del cuadro de diálogo "Iterar" hace que la solución resultante dependa potencialmente del orden de casos con independencia de cómo se eligen los centros de conglomerados iniciales.

1.3 Eficacia del análisis de conglomerados de *K-medias*

El comando de análisis de *k-medias* es eficaz principalmente porque no calcula las distancias entre todos los pares de casos, como el utilizado por los procedimientos jerárquicos.

La principal ventaja del procedimiento radica en que es mucho más rápido que el análisis jerárquico. Sin embargo, el procedimiento jerárquico permite una mayor flexibilidad en los análisis: puede utilizar cualquiera de las diversas medidas de distancia o similaridad, incluidas las opciones para datos binarios o de datos de frecuencias y no es necesario especificar el número de *clusters a priori*.

Una vez que haya identificado los grupos, puede construir un modelo útil para la identificación de nuevos casos utilizando análisis discriminante.

2. Análisis jerárquico

Este procedimiento intenta identificar grupos relativamente homogéneos de casos (o de variables) basándose en las características seleccionadas, mediante un algoritmo que comienza con cada caso (o cada variable) en un *cluster* diferente y combina los *clusters* hasta que sólo queda uno. Las medidas de distancia o similaridad se generan mediante el procedimiento "Proximidades".

Las variables en este método pueden ser cuantitativas, binarias o de frecuencias. La estandarización de las variables es un aspecto importante, ya que las diferencias en las escalas pueden afectar a las soluciones de agrupación. Si las variables muestran grandes diferencias en sus escalas de medición se debe considerar la opción de estandarizarlas (esto puede llevarse a cabo automáticamente mediante el propio procedimiento de agrupación jerárquico).

2.1 Método de agrupación y medidas

Las opciones disponibles son: "Vinculación inter-grupos", "Vinculación intra-grupos", "Vecino más próximo", "Vecino más lejano", "Agrupación de centroides", "Agrupación de medianas" y "Método de Ward".

2.1.1 Medidas para datos de intervalo

Las siguientes medidas de disimilaridad están disponibles para datos de intervalo:

- **Distancia euclídea**. La raíz cuadrada de la suma de los cuadrados de las diferencias entre los valores de los elementos. Ésta es la medida por defecto para los datos de intervalo.

$$EUCLID(x, y) = \sqrt{\sum_i (x_i - y_i)^2}$$

- **Distancia euclídea al cuadrado**. La suma de los cuadrados de las diferencias entre los valores de los elementos.

$$SEUCLID(x, y) = \sum_i (x_i - y_i)^2$$

- **Correlación de Pearson.** La correlación producto-momento entre dos vectores de valores.

$$CORRELATION(x,y)=\frac{\sum_i (Z_{xi} z_{yi})}{N}$$

- **Coseno.** El coseno del ángulo entre dos vectores de valores.

$$COSINE(x,y)=\frac{\sum_i (x_i y_i)}{\sqrt{\left(\sum_i {x_i}^2\right)\left(\sum_i {y_i}^2\right)}}$$

- **Chebychev.** La diferencia absoluta máxima entre los valores de los elementos.

$$CHEVYCHEV(x,y)=\max_i |x_i - yi|$$

- **Bloque.** La suma de las diferencias absolutas entre los valores de los elementos. También se conoce como la distancia de Manhattan.

$$BLOCK(x,y)=\sum_i |x_i - yi|$$

- **Minkowski.** La raíz *p*-ésima de la suma de las diferencias absolutas elevada a la potencia *p*-ésima entre los valores de los elementos.

$$MINKOWSKY(x,y)=\left(\sum_i |x_i - yi|^p\right)^{1/p}$$

- **Personalizada.** La raíz *r*-ésima de la suma de las diferencias absolutas elevada a la potencia *p*-ésima entre los valores de los elementos.

$$POWER(x,y)=\left(\sum_i |x_i - yi|^P\right)^{1/r}$$

2.1.2 Medidas para datos de frecuencias

Las siguientes medidas de disimilaridad están disponibles para datos de frecuencias:

- **Medida de *chi*-cuadrado.** Esta medida se basa en la prueba de *chi*-cuadrado de igualdad para dos conjuntos de frecuencias. Ésta es la medida por defecto para los datos de recuento.

$$CHISQ(x,y)=\sqrt{\sum_i \frac{(x_i - E(x_i))^2}{E(x_i)} + \sum_i \frac{(y_i - E(y_i))^2}{E(y_i)}}$$

- **Medida de Phi-cuadrado.** Esta medida es igual a la medida de *chi*-cuadrado normalizada por la raíz cuadrada de la frecuencia combinada.

$$PH2(x,y)=\frac{CHISQ(x,y)}{\sqrt{N}}$$

2.1.3 Medidas para datos binarios

A continuación se muestran algunas de las siguientes medidas de similaridad / disimilaridad que están disponibles para datos binarios:

- **Jaccard.** Se trata de un índice en el que no se toman en cuenta las ausencias conjuntas. Se ofrece una ponderación igual a las concordancias y a las discordancias. Se conoce también como razón de similaridad.

$$JACCARD(x,y)=\frac{a}{a+b+c}$$

- **Russel y Rao.** Se trata de una versión binaria del producto interno (punto). Se ofrece una ponderación igual a las concordancias y a las discordancias. Ésta es la medida por defecto para los datos de similaridad binarios.

$$RR(x,y)=\frac{a}{a+b+c+d}$$

- **Ochiai.** Este índice es la forma binaria de la medida de similaridad del coseno. Varía entre 0 y 1.

$$OCHIAI(x,y)=\sqrt{\left(\frac{a}{a+b}\right)\left(\frac{a}{a+c}\right)}$$

- **Rogers y Tanimoto.** Se trata de un índice en el que se ofrece una ponderación doble a las discordancias.

$$RT(x,y)=\frac{a+d}{a+d+2(b+c)}$$

- **Sokal y Sneath.** Se trata de un índice en el que se ofrece una ponderación doble a las concordancias.

$$SS(x,y)=\frac{2(a+d)}{2(a+d)+b+c}$$

2.2 Gráficos

Dendograma: Representación visual de los pasos de una solución de análisis jerárquico que muestra, para cada paso, los *clusters* que se combinan y los valores de los coeficientes de distancia. El dendrograma re-escala las distancias reales a valores entre 0 y 25, preservando la razón de las distancias entre los pasos. Los dendrogramas pueden emplearse para evaluar la cohesión de los grupos que se han formado y proporcionar información sobre el número adecuado de grupos que deben conservarse.

Diagrama de témpanos: Muestra cómo se unieron los casos. En la base (la derecha en los gráficos horizontales), no hay casos unidos todavía; a medida que se recorre hacia arriba el diagrama (o de derecha a izquierda en los horizontales), los casos que se unen se marcan con una X o una barra en la columna situada entre ellos, mientras que los *clusters* separados se indican con un espacio en blanco entre ellos.

2.3 Estadísticos

Historial de conglomeración. Muestra los casos o grupos combinados en cada etapa, las distancias entre los casos o los grupos que se combinan, así como el último nivel del proceso de aglomeración en el que cada caso (o variable) se unió a su *cluster* correspondiente.

Matriz de distancias. Proporciona las distancias o similaridades entre los elementos.

Conglomerado de pertenencia. Muestra el *cluster* al cual se asigna cada caso, en una o varias etapas de la combinación de los *clusters*.

3. Análisis en dos fases

El procedimiento Análisis en dos fases es una herramienta de exploración diseñada para descubrir las agrupaciones naturales (o grupos) de un conjunto de datos que, de otra manera, no sería posible detectar. Ofrece las siguientes opciones:

- **Tratamiento de variables categóricas y continuas.** Al suponer que las variables son independientes, es posible aplicar una distribución normal multinomial conjunta en las variables continuas y categóricas.

- **Selección automática del número de *clusters*.** Mediante la comparación de los valores de un criterio de selección del modelo para diferentes soluciones de agrupación, el procedimiento puede determinar automáticamente el número óptimo de *clusters*.

3.1 Medida de distancia

Esta opción determina cómo se calcula la similaridad entre dos *clusters*.

- **Log-verosimilitud.** La medida de la verosimilitud realiza una distribución de probabilidad entre las variables. Las variables continuas se supone que tienen

una distribución normal, mientras que las variables categóricas se supone que son multinomiales. Se supone que todas las variables son independientes.

- **Euclídea.** La medida euclídea es la distancia según una "línea recta" entre dos *clusters*. Sólo se puede utilizar cuando todas las variables son continuas.

Las comprobaciones empíricas internas indican que este procedimiento es bastante robusto frente a las violaciones tanto del supuesto de independencia como de las distribuciones, pero aún así es preciso tener en cuenta hasta qué punto se cumplen estos supuestos.

3.2 Número de conglomerados

Esta opción permite especificar cómo se va a determinar el número de grupos.

- **Determinar automáticamente.** El procedimiento determinará automáticamente el número "óptimo" de *clusters*, utilizando el criterio especificado en el grupo.

- **Criterio de conglomeración.** Se puede especificar el número máximo de grupos que el procedimiento debe tener en cuenta. Se puede especificar tanto el criterio de información bayesiano (BIC) como el criterio de información de Akaike (AIC).

- **Especificar número fijo.** Permite fijar el número de *clusters* de la solución.

3.3 Estadísticos

Proporciona opciones para la presentación de tablas con los resultados de la agrupación. Se generan los estadísticos descriptivos y las frecuencias de los *clusters* para el modelo final, mientras que la tabla de criterio de información muestra los resultados correspondientes a varias soluciones.

- **Descriptivos por conglomerado.** Muestra dos tablas que describen las variables de cada *cluster*. En una tabla, se informa de las medias y las

desviaciones típicas para las variables continuas por cada grupo. La otra tabla informa de las frecuencias de las variables categóricas por grupo.

- **Frecuencias de los conglomerados.** Muestra una tabla que informa el número de observaciones existentes en cada *cluster*.

3.4 Gráficos

3.4.1 Gráfico del porcentaje intra-conglomerado

Muestra los gráficos que indican la variación dentro del *cluster* de cada variable. Para cada variable categórica, se genera un gráfico de barras agrupado, mostrando la frecuencia de la categoría por identificador de *cluster*.

3.4.2 Gráfico de sectores de los conglomerados

Muestra un gráfico de sectores que muestra el porcentaje y las frecuencias de observaciones correspondientes a cada *cluster*.

3.4.3 Gráfico de la importancia de las variables

Muestra varios gráficos diferentes que indican la importancia de cada variable dentro de cada grupo. Los resultados se ordenan según el nivel de importancia de cada variable.

- **Ordenar variables.** Esta opción determina si los gráficos que se crearán para cada *cluster* (por variable) o para cada variable (por *cluster*).

- **Medida de la importancia.** Esta opción permite seleccionar la medida de la importancia de las variables que se van a representar en el gráfico. *Chi*-cuadrado o prueba *t* de significación muestra un estadístico *chi*-cuadrado de Pearson como la importancia de una variable categórica y un estadístico *t* como importancia de una variable continua.

- **Nivel de confianza**. Esta opción permite establecer el nivel de confianza para la prueba de igualdad de la distribución de una variable dentro de un *cluster* frente a la distribución global de la variable. Se debe especificar un número inferior a 100 y superior o igual a 50.

- **Omitir variables no significativas.** Las variables que no son significativas para el nivel de confianza especificado no aparecen en los gráficos de la importancia de las variables.

Anexo C

Árboles de decisión

Los árboles de decisión (también llamados árboles de clasificación), es uno de los métodos de aprendizaje inductivo supervisado no paramétrico. Es muy utilizado principalmente por su sencillez tanto de generación de árboles como de interpretación de resultados.

Los árboles de decisión parten de un nodo raíz que contiene todas las observaciones de la muestra. A medida que se desplaza por el árbol, los datos se ramifican en subconjuntos de datos que se excluyen mutuamente.

1. Regla de parada

La regla de parada es una regla de decisión para detener el desarrollo de un árbol. Un nodo no se dividirá si se cumple alguna de las condiciones siguientes:

- Todos los casos de un nodo tienen valores idénticos para todos los predictores.
- El nodo se vuelve puro; es decir, todos sus casos tienen el mismo valor para la variable criterio.
- La profundidad del árbol ha alcanzado el valor máximo preestablecido.
- El número de casos que constituye el nodo es menor que el tamaño mínimo preestablecido para los nodos parentales.
- La división del nodo ha dado como resultado un nodo filial cuyo número de casos es menor que el tamaño preestablecido para los nodos filiales.

2. Métodos de desarrollo de árboles

AnswerTree incluye los siguientes métodos para el desarrollo de árboles: CHAID, CHAID exhaustivo, C&RT y QUEST. Cada uno de ellos funciona de modo ligeramente distinto y se utilizan en casos distintos. En esta sección se proporciona una visión general de todos esos algoritmos, además de una explicación sobre las ventajas e inconvenientes de cada uno y del modo que tratan los valores perdidos.

2.1 Método CHAID

Las siglas CHAID corresponden al término inglés *Chi-square Automatic Interaction Detector* (detector automático de interacciones mediante *chi*-cuadrado). Es una técnica estadística desarrollada por Kass (1980) muy eficaz para segmentar o generar árboles. El método CHAID, que utiliza la significación de una prueba estadística como criterio, evalúa todos los valores de una variable predictora[28] potencial. Funde los valores considerados estadísticamente homogéneos respecto a la variable criterio[29] y conserva inalterados todos los valores heterogéneos.

Como paso siguiente, selecciona la mejor variable predictora para formar la primera rama del árbol de decisión, de forma que cada nodo esté compuesto por un grupo de valores homogéneos de la variable seleccionada. Este proceso se repite hasta que el árbol se ha desarrollado por completo. La prueba estadística utilizada depende del nivel de medida de la variable criterio. Si la variable criterio es continua, se utiliza la prueba F; si es categórica se utiliza la prueba χ^2.

El método CHAID es probablemente el más popular. No se trata de un método binario, es decir, puede generar más de dos categorías en cualquier nivel del árbol. Por lo tanto, tiende a crear un árbol más ancho que los métodos de desarrollo binarios. Funciona con todos los tipos de variables y trata los valores perdidos como una categoría individual.

[28] Las variables predictoras son las que predicen el patrón de la variable criterio. También se denominan variables independientes.
[29] La variable criterio es aquélla cuyo resultado se desea predecir a partir de otras variables. También se denomina variable dependiente.

2.1.1 Algoritmo CHAID

CHAID funciona con todos los tipos de variables continuas y categóricas. Sin embargo, las variables predictoras continuas se categorizan automáticamente para el análisis. Algunas de las opciones mencionadas a continuación se pueden definir mediante las opciones avanzadas de CHAID. Estas opciones incluyen la posibilidad de seleccionar la prueba χ^2 de Pearson o la de la razón de verosimilitud, el nivel de α_{merge} (fusión) y el de α_{split} (división).

1. Para cada variable predictora X, buscar el par de categorías de X cuya diferencia presenta la menor significación (es decir, la que presente el mayor nivel crítico) respecto a la variable criterio Y. El método utilizado para calcular el nivel crítico depende del nivel de medida de Y.

    a. Si Y es continua utiliza la prueba F.
    b. Si Y es nominal, se forma una tabla de contingencia con las categorías de X como filas y las categorías de Y como columnas. Utilizar la prueba de *chi*-cuadrado de Pearson o la prueba de la razón de verosimilitud.
    c. Si Y es ordinal, se ajusta un modelo de asociación de Y. Utilizar la prueba de la razón de verosimilitud.

2. Para el par de categorías de X con el mayor nivel crítico, se compara el valor del nivel crítico con el nivel alfa preestablecido, α_{merge}.

    a. Si el nivel crítico es mayor que α_{merge}, fundir este par en una sola categoría compuesta. Como resultado, se forma un nuevo conjunto de categorías de X y el proceso se vuelve a iniciar desde el paso 1.
    b. Si el nivel crítico es menor que α_{merge}, ir al paso 3.

3. Calcular el nivel crítico corregido para el conjunto de las categorías de X y las categorías de Y, mediante la corrección de Bonferroni[30] correspondiente.

[30] Corrección aplicada a los *p-values* (probabilidades) en pruebas estadísticas cuando se realizan varias pruebas. Se utiliza para evitar que la tasa de error supere el criterio nominal (α) cuando se realizan varias pruebas. En forma sencilla, permite hacer pruebas sin que se lesione el nivel de significación.

4. Seleccionar la variable predictora X cuyo nivel crítico corregido sea el menor (la que sea más significativa). Comparar su nivel crítico con el nivel alfa preestablecido, α_{split} .

    a. Si el nivel crítico es menor o igual que α_{split} , dividir el nodo conforme al conjunto de categorías de X .
    b. Si el nivel crítico es mayor que α_{split} , no dividir el nodo. Este nodo es un nodo terminal.

5. Continuar con el proceso de desarrollo del árbol hasta que se cumpla una de las reglas de parada.

2.2 Método CHAID exhaustivo

El método CHAID exhaustivo es una modificación del método CHAID desarrollada por Biggs, de Ville y Suen (1991). Se desarrolló para resolver algunos de los puntos débiles del método CHAID. En concreto, hay ocasiones en las que CHAID no encuentra la división óptima de una variable, ya que detiene la fusión de categorías en cuanto constata que todas las categorías restantes son estadísticamente distintas. Para resolver este problema, CHAID exhaustivo continúa fundiendo las categorías de la variable predictora hasta que sólo quedan dos supercategorías. A continuación, examina la serie de fusiones del predictor, busca el conjunto de categorías que proporciona la mayor asociación con la variable criterio y calcula un valor *p* (nivel crítico) corregido para esa asociación. De esta manera, el método CHAID exhaustivo puede encontrar la mejor división para cada predictor y, a continuación, elegir el predictor que se va a dividir comparando los niveles críticos corregidos.

El método CHAID exhaustivo opera exactamente igual que el método CHAID en las pruebas estadísticas que utiliza y en la forma en que trata los valores perdidos. Dado que su método de combinación de categorías de variables es más minucioso que el CHAID, tarda más en realizar los cálculos. No obstante, cuando el tiempo no es un problema, el uso del método exhaustivo suele ser más seguro que el método simple. En ocasiones encuentra divisiones más útiles, sin embargo, se debe tener en cuenta que, dependiendo de los datos utilizados, es posible que no haya ninguna diferencia en los resultados obtenidos con ambos métodos.

2.3 Método C&RT

Las siglas C&RT corresponden al término inglés *Classification and Regresión Trees*. Es un algoritmo binario relativamente reciente, desarrollado por Breiman, Friedman, Losen y Stone (1984). Este método divide los datos en dos subconjuntos, de modo que los casos comprendidos dentro de cada uno de los subconjuntos, sean más homogéneos que en el subconjunto anterior. Se trata de un proceso recursivo, que se repite hasta alcanzar el criterio de homogeneidad o hasta llegar a otro criterio de parada (como ocurre con todos los otros métodos de desarrollo de árboles). Se debe tener en cuenta que la misma variable predictora puede ser utilizada varias veces en distintos niveles del árbol.

Este método es bastante flexible, permite considerar los costes de clasificación errónea en el proceso de desarrollo del árbol. También permite asignar la distribución de probabilidades *a priori* en los problemas de clasificación. No obstante, el método C&RT presenta algunos inconvenientes. Al ser un algoritmo binario, tiende a generar árboles de muchos niveles. Por ello, puede ocurrir que el árbol resultante no presente los resultados de manera eficaz, sobre todo si la misma variable ha sido utilizada para la división de varios niveles consecutivos.

El método C&RT es complejo; los cálculos pueden ser muy lentos cuando se utilizan grandes conjuntos de datos.

2.3.1 Algoritmo C&RT

C&RT funciona eligiendo en cada nodo una división, de modo que cada nodo filial sea más puro que su nodo parental. En este caso, la "pureza" se refiere a los valores de la variable criterio. En un nodo completamente puro, todos los casos tienen el mismo valor para la variable criterio. El algoritmo C&RT mide la impureza de la división de un nodo definiendo una medida de impureza.

2.3.2 Medidas de impureza

Para buscar las divisiones en los modelos C&RT se utilizan cuatro medidas de impureza distintas, que dependen del tipo de variable criterio. Para las variables

categóricas, se puede elegir Gini, binario (para criterios ordinales) o binario ordenado.

2.3.3 Índice de Gini

El índice de Gini en el nodo t, $g(t)$, se define como:

$$g(t)=\sum_{j\neq i} p(j/t)p(i/t)$$

Donde i y j son categorías de la variable criterio y

$$p(j/t)=\frac{p(j,t)}{p(t)}$$
$$p(j,t)=\frac{\pi(j)N_j(t)}{N_j}$$
$$p(t)=\sum_j p(j,t)$$

donde $\pi(j)$ es el valor de la probabilidad *a priori* para la categoría j, $N_j(t)$ es el número de casos en la categoría j del nodo t y N_j es el número de casos de la categoría j en el nodo raíz. Se debe tener en cuenta que cuando se utiliza el índice de Gini para buscar la mejora de una división durante el desarrollo de un árbol, sólo los casos del nodo t y del nodo raíz con valores válidos para el predictor de división se utilizan para calcular $N_j(t)$ y N_j respectivamente.

La ecuación del índice de Gini se puede expresar también como:

$$g(t)=1-\sum_j p^2(j/t)$$

Para ello, cuando los casos de un nodo están distribuidos uniformemente entre las categorías, el índice de Gini toma su valor máximo de $1-\frac{1}{k}$, donde k es el número de categorías de la variable criterio. Cuando todos los casos del nodo pertenecen a la misma categoría, el índice de Gini es igual a 0.

La función del criterio Gini $\Phi(s,t)$ para la división s en el nodo t se define como

$$\Phi(s,t) = g(t) - p_L g(t_L) - p_R g(t_R)$$

Donde p_L es la proporción de casos en t enviados al nodo filial de la izquierda y p_R es la proporción enviada al nodo filial de la derecha. Las proporciones de p_L y p_R se definen como

$$p_L = \frac{p(t_L)}{p(t)}$$

y

$$p_R = \frac{p(t_R)}{p(t)}$$

Se elije la división s para maximizar el valor de $\Phi(s,t)$, el cual se considera la "mejora" en el árbol.

Binario. El índice binario se basa en la división de las categorías criterio en dos superclases y, después, en la búsqueda de la mejor división de la variable predictora según estas dos superclases. La función del criterio binario para la división s en el nodo t se define como

$$\Phi(s,t) = p_L P_R \left[\sum | p(j/t_L) - p(j/t_R) | \right]^2$$

Donde t_L y t_R son nodos creados pro la división s. La división s se elije como la división que maximiza este criterio. Este valor, ponderado por la proporción de todos los casos del nodo t, es el valor que se considera la "mejora" en el árbol. Las superclases C_1 y C_2 se definen como

$$C_1 = \{ j : p(j/t_L) \geq p(j/t_R) \}$$

y

$$C_2 = C - C_1$$

donde C es el conjunto de categorías de la variable criterio.

Binario ordenado. Este índice es una modificación del índice binario para las variables criterio ordinales. La diferencia consiste en que, con el criterio binario ordenado, sólo las categorías contiguas se pueden combinar para formar superclases. Por ejemplo, consideremos la variable criterio "estado de cuenta", con las categorías

1 = *actualizada*
2 = 30 *días de vencimiento*
3 = 60 *días de vencimiento*
4 = 90 *o más días de vencimiento*

En algunas circunstancias, el criterio binario podría unir las categorías 1 y 4 para formar una superclase, y formar la otra superclase con las categorías 2 y 3. No obstante, si consideramos que estas categorías están ordenadas, no es bueno que se combinen las categorías 1 y 4 (sin incluir también las categorías intermedias) porque no son contiguas. El índice binario ordenado tiene en cuenta este orden y no combinará categorías que no sean contiguas como la 1 y la 4.

Desviación cuadrática mínima (LSD). Para variables criterio continuas se utiliza la medida de impureza LSD. La medida LSD, $R(t)$, es simplemente la varianza (ponderada) dentro del nodo t, y es igual a la estimación del riesgo mediante reestimación para dicho nodo. Se define como

$$R(t) = \frac{1}{N_w(t)} \sum_{i \in t} w_n f_n \left(y_i - \bar{y}(t)\right)^2$$

donde $N_w(t)$ es el número ponderado de casos en el nodo t, w_n es el valor de la variable de ponderación para el caso i (si existe), f_n es el valor de la variable de frecuencia (si existe), y y_i es el valor de la variable criterio y finalmente $y(t)$ es la media (ponderada) para el nodo t.

La función del criterio LSD para la división s en el nodo t se define como

$$\Phi(s,t) = R(t) - p_L R(t_L) - p_R R(t_R)$$

Se elige la división s para maximizar el valor de $\Phi(s,t)$. Este valor, ponderado por la proporción de todos los casos del nodo t, es el valor que se considera la "mejora" en el árbol.

2.3.4 Algoritmo C&RT

1. Para llevar a cabo un análisis C&RT, comenzando por el nodo raíz $t = 1$, buscar la división s^* entre todos los candidatos posibles S que dé lugar a la mayor reducción de la impureza:

$$\Phi(s^*,1) = \max_{s \in S} \Phi(s,1)$$

   A continuación, dividir el nodo 1 $(t = 1)$ en dos nodos $t = 2$ y $t = 3$ utilizando la división s^*.

2. Repetir el proceso de búsqueda de divisiones para cada uno de los nodos $t = 2$ y $t = 3$, y así sucesivamente.

3. Continuar con el proceso de desarrollo del árbol hasta alcanzar al menos las reglas de parada.

2.4 Método QUEST

Las siglas QUEST corresponden al término inglés *Quick, Unbiased, Efficient Statistical Tree*. Es un algoritmo binario relativamente reciente, desarrollado por Loh y Shih (1997). Este método trata por separado la selección de variables y la selección del punto de división.

La división univariante del método lleva a cabo una selección de variables aproximadamente insesgada; esto es, si todas las variables predictoras son igualmente informativas respecto a la variable criterio, QUEST selecciona cualquiera de las variables predictoras con la misma probabilidad. Este método está creado con vistas a la eficacia de los cálculos, presenta muchas de las ventajas del método C&RT, pero al igual que este último, los árboles pueden ser poco manejables.

2.4.1 Algoritmo QUEST

El nivel para α a utilizar se puede especificar en las opciones avanzadas de QUEST. El valor predeterminado es $\alpha = 0.5$.

1. Para cada variable predictora X, si X es una variable categórica nominal, calcular el nivel crítico de una prueba de independencia χ^2 de Pearson entre X y la variable dependiente categórica. Si X es continua u ordinal, utilizar la prueba F para calcular el nivel crítico.

2. Comparar el menor nivel crítico con el nivel α preestablecido, corregido mediante Bonferroni.

    a. Si el nivel crítico es menor que α, seleccionar la variable predictora correspondiente para dividir el nodo. Ir al paso 3.
    b. Si el nivel crítico es mayor que α, para cada X ordinal o continua, utilizar la prueba de Levene (F de Levene)[31] sobre varianzas desiguales, para así calcular su nivel crítico. En otras palabras, intentar averiguar si X presenta varianzas desiguales en los distintos niveles de la variable criterio.
    c. Comparar el menor nivel crítico de la prueba de Levene con el nuevo nivel α, corregido mediante Bonferroni.
    d. Si el nivel crítico es menor que α, seleccionar la variable predictora correspondiente con el menor nivel crítico en la prueba de Levene para dividir el nodo. Ir al paso 3.
    e. Si el nivel crítico es mayor que α, seleccionar la variable predictora del paso 1 cuyo nivel crítico se el menor (ya sea a partir de la prueba de χ^2 o de la prueba F) para dividir el nodo. Ir al paso 3.

3. Supongamos que X es la variable predictora del paso 2. Si X es continua u ordinal, ir al paso 4. Si X es nominal, transformar X en una variable ficticia Z y calcular la mayor coordenada discriminante de Z. De manera general, se transforma X para maximizar las diferencias entre las categorías de la variable criterio.

[31] Estadístico de prueba utilizado para comprobar las diferencias en la varianza de las variables predictoras en todas las categorías de la variable criterio.

4. Si Y tiene sólo 2 categorías, ir al paso 5. De lo contrario, calcular la media de X para cada categoría de Y y aplicar un algoritmo de agrupación de dos medias para obtener dos superclases de Y.

5. Aplicar el análisis discriminante cuadrático (QDA) para determinar el punto de división. Se debe tener en cuenta que el QDA normalmente genera dos puntos de corte, se debe elegir el más cercano a la media muestral de cada clase.

Bibliografía

Cheng, B. and Titterington, D.M. (1994). Neural Networks: A Review from a Statistical Perspective. *Statistical Science,* **9**, No.1, 2-54.

Demuth, H., Beale, M. and Hagan, M. (2007). *Neural Network Toolbox 5: User´s Guide*. The MathWorks, Inc.

Eccles, J.C. (1973). *The understanding of the Brain*. McGraw-Hill.

Everitt, B.S., Landau, S., and Leese, M. (2001). *Cluster Analysis,* 4th ed. Oxford University Press Inc., New York.

Freeman, J.A. and Skapura, D.M. (1991). *Neural Networks. Algorithms, Applications and Programming Techniques*. Adison Wesley. USA.

Hecht-Nielsen, R. (1988). Neurocomputing: picking the human brain. *IEEE. Spectrum,* **25**, No.3, 36-41.

Hilera, J. y Martínez, V. (1995). *Redes Neuronales Artificiales: Fundamentos, Modelos y aplicaciones*. Ra-ma. Madrid.

Johnson, D.E. (2000). *Métodos multivariados aplicados al análisis de datos*. International Thomson Editores.

Kohonen, T. (1995). *Self-Organizing Maps*. Springer Series in Information Sciences.

Martín, B. y Sanz, A. (2002). *Redes Neuronales y Sistemas Difusos,* 2ª ed. Alfaomega Ra-ma. Madrid.

SPSS Inc. (2205). *Guía del usuario de AnswerTree 3.1*. SPSS Inc.

Vesanto, J. and Alhoniemi, E. (2000). *Clustering of the Self-Organizing Map. IEEE Transactions on Neural Networks,* **11**, No.3, 586-600.

Warner, B. and Misra, M. (1996). Understanding Neural Networks as Statistical Tools. *The American Statistician,* **50**, No.4, 284-293.