Sistemas Modulares, Mezcla de Expertos y Sistemas Híbridos

Quiliano Isaac Moro Sancho

Departamento de Informática, Universidad de Valladolid, España
isaac@infor.uva.es

Informe Técnico DI-2000-001

Resumen Como es habitual en muchas disciplinas científicas, previo a la búsqueda de soluciones al problema que se plantea en cada momento, es razonable y preferible pararse un instante y realizar un *análisis* del problema. Fruto de este análisis será una visión de las distintas partes que forman el todo, habiéndose transformado la tarea inicial, probablemente compleja, en un conjunto de subtareas más elementales, susceptibles de ser abordadas de manera más sencilla y eficiente. Una vez hecho esto el problema se transforma en el opuesto: integrar los resultados parciales obtenidos de cada una de esas subtareas y generar la solución al problema completo.
En este trabajo inicialmente se planteará el problema de poner en práctica un método de "divide y vencerás" de forma genérica, buscando alguna estrategia que pueda sistematizar el procedimiento, para seguidamente presentar algunos métodos habitualmente utilizados en sistemas basados en la mezcla de expertos. En la última parte del trabajo se repasarán algunos ejemplos concretos de sistemas que utilizan esta aproximación para buscar soluciones.

1 Introducción. Módulos, expertos e hibridación

Las redes neuronales se utilizan ampliamente en tareas genéricas (especialmente el Perceptrón Multicapa usando como algoritmo de aprendizaje el de Retropropagación del Error), o bien en tareas más específicas, típicamente de clasificación o *clustering* (cuyo exponente más habitual entre las redes neuronales artificiales lo forman los mapas autoorganizados de Kohonen).

Cada una de estas aproximaciones presenta sus ventajas e inconvenientes, y se abre la posibilidad de combinar ambos aspectos de la computación conexionista, consiguiendo de alguna manera la unión de sendas características: potencial de aprendizaje de los sistemas supervisados junto con el potencial de los clasificadores no supervisados.

El primer paso para poder aplicar técnicas de mezclas de expertos es dividir la tarea problema en subtareas, y posteriormente crear y organizar adecuadamente los subsistemas construidos para permitir la comunicación entre ellas y así integrarse como un todo que proporcione la solución buscada.

La idea de modularidad, al igual que ocurrió en los orígenes de la computación conexionista, puede haber surgido bajo la inspiración de los modelos biológicos. Un examen de las estructuras fisiológicas del sistema nervioso en los animales vertebrados revela la existencia de una representación y procesamiento modular jerárquica de la información, siendo este hecho particularmente evidente en el córtex visual [Hay94]:

1. En el córtex visual existen módulos diferenciados, cada uno de ellos especializado en una tarea específica. De esta manera se permite que la arquitectura neuronal de cada módulo sea óptima para el tipo de tarea que va a desempeñar.

2. La estructura creada se replica un número muy elevado de veces en la zona del córtex visual.
3. Como punto adicional, es necesario añadir un sistema de coordinación y reparto de la información (señales) entre los distintos módulos existentes, que posibilite un aprendizaje e integración correctos.

Tomando como base los indicios biológicos, una primera aproximación modular de sistemas complejos podría ser la descrita por **Jacobs y Jordan** [JJ94], que implicaría dos tipos bien diferenciados de aprendizaje:

- **Aprendizaje supervisado**, durante el cual un *maestro externo* suministra para cada estímulo de entrada la salida correcta. Sin embargo, este maestro no especifica qué módulo es el que debe aprender el correspondiente par [estimulo de entrada, salida deseada].
- **Aprendizaje no supervisado**, que básicamente consiste en un aprendizaje competitivo, en el que los distintos módulos compiten por aprender el ejemplo presentado.

Dado el modo de funcionamiento expuesto, el sistema puede ser representado por un esquema como el mostrado en la figura 1. En dicha figura se ven los k módulos propiamente dichos; cada uno constituye un experto en una tarea específica. Un módulo especial formado por una red de puertas (*"Gating Network"*) es el encargado de coordinar el funcionamiento de los módulos durante el aprendizaje, responsabilizándose de determinar cuál de los k módulos es el que debe aprender a generar la respuesta buscada, así como de generarla durante el funcionamiento del sistema completo. Cada una de las k salidas de dicho módulo es un escalar que hará las labores de compuerta para ajustar la influencia o peso que tendrá cada uno de los k módulos en la formación de la salida final del sistema. A parte, aparece un *módulo sumador* que produce la salida final.

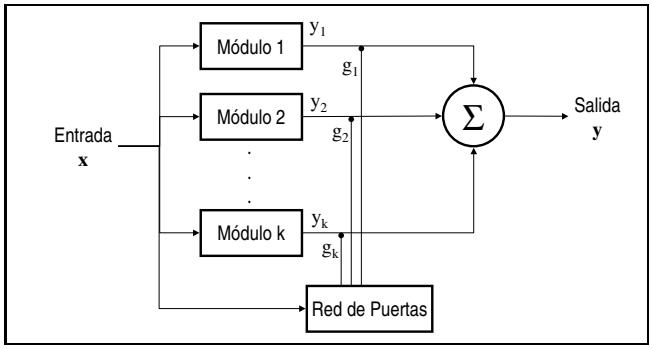


Figura1. Sistema Modular [Hay94].

El modelo aquí descrito representa muy bien al caso particular de los sistemas compuestos por **mezcla de gaussianas**, en el que cada módulo i constituye

una regla o experto que produce una salida y_i, fruto de un proceso aleatorio cuya función de distribución es gaussiana. El experto es escogido de entre los demás de acuerdo a la ley de probabilidad, siendo esta probabilidad $P(i/x)$ condicionada al valor del vector de entradas x que se presente al sistema en cada momento. La salida generada por el sistema será una suma ponderada por los valores generados por el módulo *gating network*. Con este esquema es inmediato identificar los valores g_i con las probabilidades $P(i/x)$.

Es posible reinterpretar el modelo de Jacobs-Jordan desde el punto de vista del control de sistemas, identificando cada experto como el mecanismo encargado del comportamiento de sistema en un determinado punto de funcionamiento. Esta es la que se denomina *aproximación del punto de funcionamiento* [MSJ97].

El esquema modular que se acaba de exponer, así como el mecanismo de aprendizaje propuesto, se fundamentan grandemente en hechos fisiológicos. Sin embargo, ¿por qué habría de confiarse la asignación de tareas para cada módulo experto a un mecanismo de aprendizaje no supervisado, cuando en muchos casos reales se dispone de información para asociar subtareas a módulos al menos con un cierto grado de certeza o intuición? Esto significaría que *el entrenamiento del sistema modular se podría realizar completamente en modo* ***supervisado***.

En cualquiera de los casos presentados, no se ha especificado la naturaleza final de los denominados **módulos o expertos**, pudiéndose realizar mediante múltiples posibilidades, como Sistemas Expertos que utilicen reglas (Inteligencia Artificial), lógica difusa, métodos estadísticos (como pueden ser los modelos bayesianos o los modelos ocultos de Markov), redes neuronales, etc.

En el caso más general, la estructura topológica que presentase el sistema completo no tendría por qué limitarse a una jerarquía de dos niveles, siendo posible estructuras más complejas y genéricas, e incluso cabe la posibilidad de integrar en el mismo sistema módulos de distinta naturaleza, dándose origen a los **sistemas híbridos**, donde tampoco es necesario que todos los módulos reciban el mismo conjunto de entradas.

Lamentablemente, cuando se habla de sistemas modulares híbridos en general no es posible establecer un *procedimiento sistemático* para ajustar los parámetros que los definen, como es el caso bien conocido de la mezcla de gaussianas. Esta desagradable circunstancia es debida a que la asignación de tareas a módulos y el ajuste de cada módulo dependerán de la naturaleza de la tarea en sí y de la subtarea asignada al módulo en cuestión, así como del modo en que se hallen interrelacionados los distintos módulos.

2 Sistemas Globales y Sistemas Locales

A la hora de plantearse la resolución de un problema generalmente se puede realizar de distintas maneras:

- Abordar el problema de forma **global**, es decir, como si fuera un todo.

– Analizar el problema procurando encontrar cada una de sus partes constituyentes más simples y buscar una solución para cada una de ellas. Este sería un enfoque **local**.

A pesar de que las redes neuronales son por su propia naturaleza un sistema modular (el elemento que se replica es la neurona en un sistema biológico o el elemento de proceso en una red neuronal artificial), los primeros modelos que aparecieron, y los que más se han difundido en su aplicación, son los paradigmas globales, en los que no se realiza ningún intento por encontrar subtareas, o partes más elementales del problema en cuestión.

Hay varios problemas que subyacen en la filosofía de las redes neuronales artificiales como sistemas globales, relacionados principalmente con una característica que les es inherente, como es el hecho de ser sistemas que aprenden.

El primer punto, que más que un problema se podría considerar como una situación "incómoda", es la incapacidad de dichos sistemas en justificar razonadamente (mediante reglas, leyes o algoritmos) la solución obtenida, tal como haría un experto *humano*. Es decir, no hay nada que justifique de una manera razonada la forma que ha tomado la información adquirida durante el proceso de entrenamiento (en el este caso, los valores de los pesos entre los distintos elementos de proceso). *¿Cómo fiarse de un sistema de proceso de datos o control del que no se sabe nada acerca de cómo se ha representado internamente la información que es utilizada para resolver el problema propuesto? ¿El sistema alcanzado es estable? ¿Cuál es su granularidad? ...* El sistema de **caja negra** no es muy bien aceptado por el usuario final.

En relación con un modelo muy extendido de redes neuronales artificiales como es el perceptrón multicapa entrenado con el algoritmo de retropropagación del error, se pueden señalar varias cuestiones:

– Dado el algoritmo de entrenamiento empleado (*retropropagación del error*), cabe la posibilidad de alcanzar un mínimo local en la superficie función del error, o bien que en la zona en la que se esté evaluando dicha función la pendiente sea muy escasa (superficie de error casi plana), produciéndose un aprendizaje demasiado lento.
– En general, cuando el sistema (la red neuronal) es demasiado grande, el número de parámetros a ajustar durante el entrenamiento es también muy elevado. Puede que esta red aún siendo capaz de aprender los ejemplos de entrenamiento, con ejemplos no vistos en la fase de aprendizaje los resultados sean malos, es decir, el sistema no es capaz de **generalizar**. Este problema recibe el nombre de **sobreentrenamiento** (*"overfitting"*)[Mac99,Koh95], y refleja el compromiso existente entre el sesgo y la varianza (*"biasi-variance dilemma"*). Cuando el número de parámetros que definen el sistema es grande, y el número de iteraciones de entrenamiento es también muy elevado, el sistema no sólamente modela la relación entre entrada y salida deseada (sesgo muy pequeño), sino que también modela el ruido que se encuentra mezclado con la señal bajo estudio. Para aliviarlo hay ciertas alternativas, como pueden ser:

- detener el proceso de aprendizaje en un número bajo de iteraciones (*"early stopping"*), con lo que no se llega a producir el sobreajuste de los parámetros,
- la reducción del tamaño de la red mediante poda, con lo que se reduce el número de parámetros involucrados en el sistema,
- utilizar un método incremental, aumentando el número de parámetros (elementos de proceso y conexiones) desde un nivel bajo, hasta alcanzar el grado de respuesta óptimo requerido,
- la utilización de técnicas como el *"weight decay"*, consistente en definir la actualización de los pesos de las conexiones entre los distintos elementos de proceso imponiendo la condición adicional de que sus correspondientes valores absolutos sean lo más pequeños posibles.

Si bien el primer método parece un poco drástico, ya que se enfrenta al problema del elevado número de parámetros por ajustar dejándolos sin ajustar del todo, el hecho de detener el proceso de aprendizaje cuando aún el número de épocas presentadas (iteraciones) es bajo, hace que el tiempo consumido en esta etapa sea pequeño. La segunda solución propuesta, *la poda*, parece más razonable, ya que desecha aquellos elementos de proceso que no forman parte relevante de la solución buscada, pero hay que tener en cuenta que, una vez que se haya decidido cuáles serán los elementos de proceso víctimas de la poda, hay que eliminarlos y volver a entrenar, aumentándose el tiempo de procesamiento. Algo parecido se puede indicar sobre el método de crecimiento de la red.

– Otro problema que aparece es que hay tareas complejas que un solo sistema no puede resolver. Este problema recibe el nombre de **interferencia** (*"cross-talk"*), y tiene dos variantes: la espacial y la temporal. En ambos casos, los elementos que aprenden (los elementos de proceso en las redes neuronales artificiales) reciben información contradictoria, ya sea de modo simultáneo (interferencia espacial) o en instantes distintos de tiempo (interferencia temporal).

Ejemplos En este apartado se mostrarán ejemplos en los que se pone de manifiesto la necesidad de alejarse de un enfoque global y utilizar sistemas modulares a la hora de resolver problemas complejos. Estos ejemplos se centran sobre todo en la aparición de *interferencia* tanto espacial como temporal durante el aprendizaje de la resolución de la tarea propuesta.

Interferencia Espacial. El problema "Qué y Dónde" ("What and Where"*).* El primer ejemplo consiste en la resolución de una tarea doble: determinar la posición y la forma de un objeto bidimensional. En concreto, los objetos que se usarán son dos: uno en forma de "T", y otro en forma de "C", que pueden tener cualquiera de las 4 orientaciones mostradas en la parte izquierda de la figura 2. En cuanto a la posición del objeto, se ha limitado a un retículo de 4×4, lo que ofrece un total de 4 posiciones distintas (las cuatro esquinas del retículo), pudiendo encontrarse en cualquiera de sus cuatro posibles orientaciones[1]. El problema consiste

[1] Algunos autores [Rij95,RG95] utilizan un área de 5×5 celdas.

en dada una situación como la mostrada en la parte derecha de la figura 2, el sistema habrá de identificar qué tipo de objeto se halla representado en ella e indicar su posición (en este caso una "T" en la esquina superior izquierda), con independencia de la orientación del objeto.

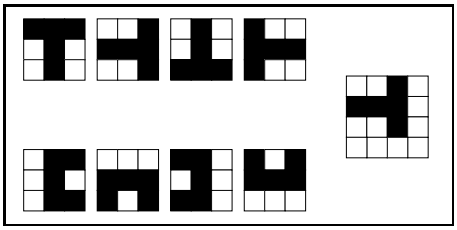


Figura2. El problema "Qué y Dónde" tal y como se plantea en [BKHSK93].

El problema así planteado presenta el efecto de la *interferencia espacial* cuando se quiere solucionar de forma global utilizando para ello una *única* red neuronal tipo perceptrón multicapa con una única capa oculta completamente interconectada con la capa de entrada y de salida. Más concretamente, las dificultades aparecen cuando hay que adaptar las conexiones los distintos elementos de proceso de la capa oculta de tal forma que se satisfagan simultáneamente las salidas deseadas para el problema de identificación de la forma y de su localización en el retículo. Esta circunstancia se refleja en la obtención de un sistema que no es capaz de solucionar de manera eficiente el problema planteado (considerando la ineficiencia tanto en términos de tamaño de la red, el tiempo de aprendizaje y en la obtención de las respuestas correctas).

En [RG95] se propone la división de la capa oculta en dos subcapas, y a partir de ahí, cada subcapa se conecta de forma exclusiva una con las salidas que proporcionan la identificación del objeto y la otra con las salidas que lo localizan, como se puede ver en la figura 3[2]. De esta manera, las conexiones entre los elementos de proceso de la capa oculta y las correspondientes salidas sólo habrán de hacerse responsables de una tarea.

Interferencia Temporal. Aparcar un remolque marcha atrás. El problema consiste en determinar el ángulo de giro del volante de un camión para conseguir llevarlo a una dársena, todo ello marcha atrás. El sistema toma como informaciones de entrada el estado del camión, descrito por la posición del camión respecto a la dársena (x_i, y_i) y los ángulos que forman el remolque con la dársena, la cabina con el remolque y las ruedas con la cabina (figura 4(a)). El objetivo final es colocar el remolque a lo largo del eje Y, de tal forma que el final del remolque

[2] Esta división en la práctica da lugar a dos sistemas o módulos totalmente independientes (salvo el detalle de que comparten las mismas entradas), uno dedicado a la detección de la forma del objeto, y otro de su posición.

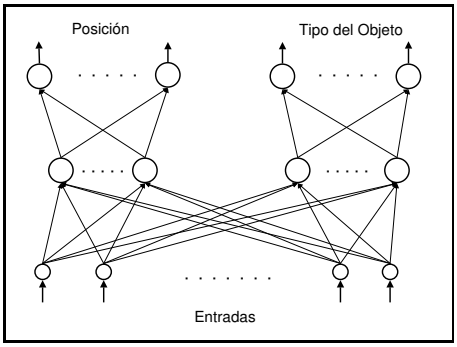


Figura3. Solución al problema "Que y Dónde" propuesta por [RG95]

esté en el origen de coordenadas (0,0). Los autores R. Jenkins y B. Yuhas propusieron en 1993 descomponer la resolución del problema en distintas subtareas, siendo todas ellas procesadas **simultáneamente**:

- *Subtarea 1: orientación del remolque*; debe girar hasta que el camión esté casi perpendicular a la dársena. Esta subtarea es independiente de la distancia del camión a la dársena, y por lo tanto, sólo necesita como entradas los ángulos de la cabina con el remolque y del remolque con la dársena.
- *Subtarea 2: dirigir el final del remolque hasta llegar a Y=0.* Esta tarea es independiente del valor de la coordenada x.

Cada una de estas subtareas se realiza mediante un único elemento Perceptrón, y se integran los resultados mediante la simple suma ponderada. Utilizando determinados conocimientos previos se ajustaron los pesos de las conexiones entre los dos módulos asociados a cada subtarea, y la estructura resultante es la mostrada en la figura 4(b). Si este problema se hubiera planteado como un todo, es decir, de forma global, la estructura de la red perceptrón multicapa resultante llegó a tener 25 elementos de proceso en la capa oculta. Con este ejemplo, Jenkins y Yuhas querían poner de manifiesto que el entrenamiento de una red neuronal artificial puede verse enormemente simplificado si se es capaz de identificar las subtareas en el problema e incorporar dicha información en la estructura de la red, reduciéndose así la dimensión del espacio de entrada a la red.

Navegación autónoma de un robot El objetivo es construir un sistema capaz de guiar un robot situado en un entorno desconocido desde una posición inicial hacia un objetivo, evitando los obstáculos que encuentra en el camino. En esta tarea se pueden ver dos elementos contrapuestos: alcanzar el objetivo y evitar los obstáculos. La solución inmediata es establecer la acción a llevar a cabo en todos y cada uno de los distintos estados (situación y orientación) en los que se puede encontrar el robot, lo que conduce a un uso exhaustivo de recursos como es la memoria del sistema, y además con toda probabilidad sería incapaz

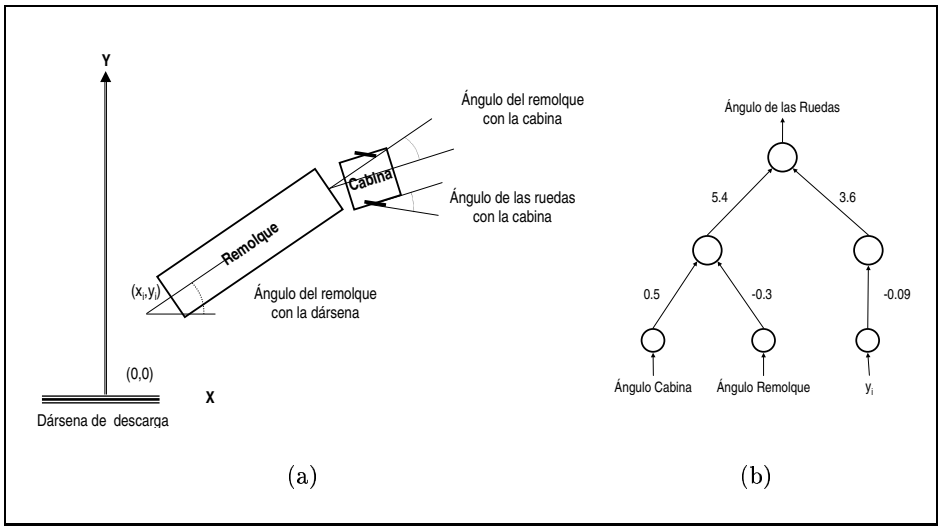


Figura4. El problema de aparcar un remolque.

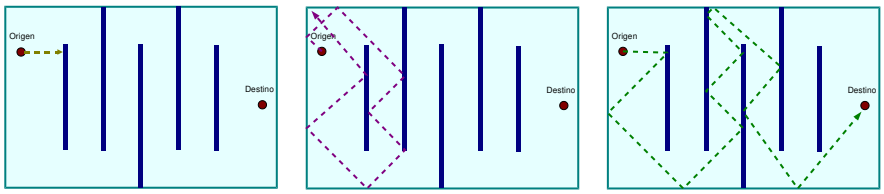
de realizar una generalización para nuevas situaciones de entorno, obstáculos y objetivos diferentes.

Un único módulo que intente aprender su resolución no ofrece buenos resultados, y en [FGP95] se propone el uso de dos módulos especializados en el cumplimiento de cada uno de las subtareas antes expuestas. Un tercer módulo coordina los resultados y genera la salida del sistema.

2.1 Ventajas e Inconvenientes de la Aproximación Modular

De un modo general, y en una primera aproximación, ya es posible enumerar algunas de las ventajas e inconvenientes que ofrecen los enfoques global y local [Has94], [Hay94], [RG95]:

- El uso de una aproximación local proporciona un aumento de la velocidad de aprendizaje, ya que cada módulo experto tiene menor tamaño (elementos de proceso), y se encarga de una subtarea, que por definición es de resolución más sencilla que la tarea global.
- Utilizado una aproximación local resulta mucho más fácil llegar a comprender la tarea de la que se ha hecho responsable un módulo (en el caso de haberse realizado una división en subtareas por medio de un conocimiento apriorístico, este punto resulta trivial).
- La aproximación modular es coherente con las limitaciones de espacio que se presentan en los modelos biológicos, ya que no podrían dar soporte a sistemas muy complejos (por ejemplo, mantener una red neuronal con un número muy elevado de subcapas y elementos de proceso, cada uno de ellos totalmente interconectados entre sí).



(a) Prioridad en el avance.

(b) Prioridad en esquivar los obstáculos.

(c) Combinación de ambas prioridades: avance y esquivar obstáculos.

Figura5. Navegación autónoma de un robot.

- Relativa sencillez de los módulos constituyentes. Los módulos expertos no necesitan ser complicados, ya que tal y como se ha indicado antes, cada subtarea se hace responsable de un subproceso elemental.
- Cada módulo puede ser construido de manera diferente, de forma que se ajuste a las exigencias de cada subtarea. Esta idea da paso a los *sistemas heterogéneos*, introduciéndose de manera intuitiva el concepto de *sistema híbridos*.
- Si se considera la aproximación global, el sistema (red) resultante puede alcanzar unas dimensiones demasiado grandes para poder aprender la tarea global propuesta. Esto dará lugar a un sistema con un número muy elevado de parámetros para ajustar (pesos entre las conexiones), y en consecuencia, también serían necesarios un gran número de ejemplos de entrenamiento, corriéndose el riesgo de producirse **sobreentrenamiento**.
  Sin embargo, cuando se utiliza una aproximación modular, los módulos que se encargan de resolver cada subtarea tienen una estructura más simple, un menor número de conexiones y pesos que ajustar, reduciéndose el problema del sobreentrenamiento.

3 Estado del Arte

La idea de combinar las respuestas de más de un mecanismo de pronóstico o estimación con el fin de mejorar los resultados obtenidos de manera individual no es nueva. Así, es posible remontarse hasta **Laplace** en el año 1818, quien ya propuso la combinación de varios estimadores. Ya más recientemente, a finales de los años 80, Jacobs y Jordan desarrollaron lo que denominaron **arquitectura de mezcla jerárquica de expertos** (HME), que utiliza la técnica del *"divide y vencerás"*, según la cual, se ha de dividir un sistema complejo en problemas más sencillos que son resueltos por separado, utilizando para cada uno de ellos un módulo (red neuronal). La solución final se construye mediante un módulo

denominado *"Gating Network"* o red de puertas. Otras variaciones sobre este mismo esquema han ido apareciendo a lo largo de los últimos años, como puede ser el *"Stacked Generalization"* propuesto por Wolpert,

En la tarea de aplicar mezcla de módulos o expertos se puede considerar varios puntos [RG95]:

1. Descomposición de la tarea principal en subtareas,
2. Determinación de la naturaleza del módulo que se asocia a cada subtarea.
3. Organización de los módulos conseguidos según una arquitectura apropiada, y
4. Establecimiento de las líneas de comunicación entre dichos módulos a dos niveles: reparto de la infornación durante la etapa de aprendizaje, e integración de la información a la hora de generar la salida del sistema completo.

Seguidamente se pasará a describir con más detalle estos puntos.

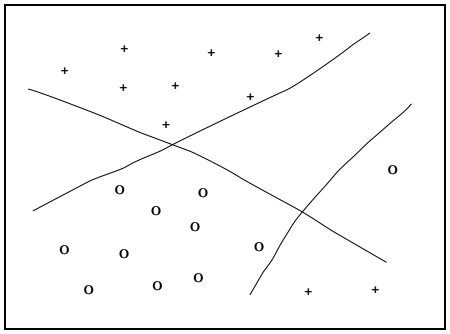
3.1 Descomposición específica ("ad hoc") y descomposición sistemática del entorno

Cuando la tarea principal es básicamente la resolución de un problema de clasificación, es decir, particionar el espacio de entrada en subespacios según las propiedades de los objetos (medidas o entradas al sistema), es bastante natural realizar una división en subtareas de acuerdo a las clases o categorías a detectar. En este caso, en general es indiferente el hecho de realizar una partición en subtareas de manera no supervisada (por ejemplo con sistemas autoorganizados) o bien de manera supervisada.

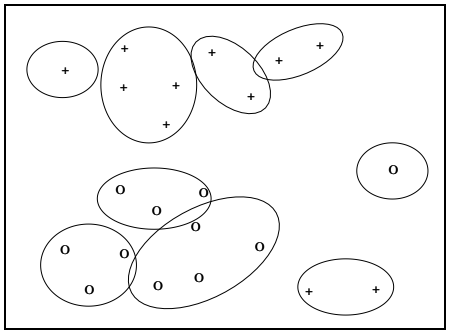
Buenos ejemplos de redes neuronales modulares para este tipo de problemas pueden ser:

- **Redes de Función de Base Radial** (RBF). Este tipo de redes realizan un agrupamiento ("clustering") hiperesférico del espacio de entrada. El problema de este tipo de sistemas es que al ir aumentando la dimensionalidad de la función objetivo a sintetizar, el número de elementos de proceso (funciones de base radial) a usar aumenta exponencialmente, produciéndose un *sistema intensivo en memoria*,

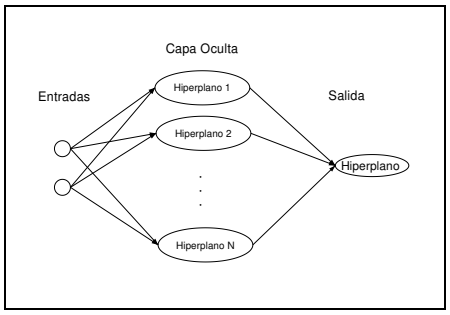
  El modo de entrenamiento del sistema para ajustar el centro del potencial de cada elemento de proceso RBF es autoorganizado. Si se dispone de conocimiento apriorístico, es posible mejorar el comportamiento del sistema, reflejándose esta mejora en la mayoría de las ocasiones en un reducción del número de elementos RBF necesarios, lo que implica que cada nodo RBF aumenta su "zona de responsabilidad".
- Otra variante es utilizar en vez de RBF un **mapa autoorganizado de Kohonen** para particionar el espacio de entrada, y a continuación un sistema local supervisado, es decir, un nodo experto que proporcione la salida deseada para cada clase que haya detectado el mapa de Kohonen.



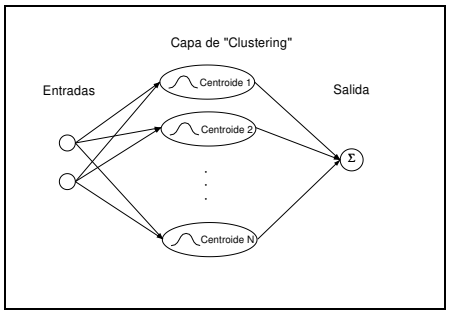
(a) Perceptrón multicapa.



(b) Funciones de base radial (RBF).



(c) Filosofía seguida por el perceptrón multicapa.



(d) Filosofía seguida por la red tipo RBF.

Figura6. Distintos modos de dividir el espacio de entradas según la red neuronal artificial usada.

– Un modelo que también puede ser incluido en este esquema es el constituido por las redes **TDNN** (*Time Delay Neural Network*), en el que los elementos de proceso de la o las capas ocultas no están totalmente conectados con los de las capas anteriores, sino que sólo reciben un subconjunto de las señales de las capas previas, lo que da lugar a un conjunto de sistemas locales.

El esquema de procesamiento que se acaba de indicar no siempre es susceptible de ser aplicado. Simplemente piénsese que se está realizando un agrupamiento o división del espacio de entradas en la mayoría de los casos de acuerdo con un criterio de *vecindad espacial*, es decir, aquellas entradas o estímulos que se encuentras próximos (por ejemplo según un criterio de distancia euclídea), serán asignados al mismo modo o clase. Sin embargo, existen problemas que no permiten usar de manera razonable ese mismo concepto de distancia; este es el caso de la conducción de un coche, para el que el espacio de entradas es tan

heterogéneo (velocidades, posiciones, ángulos, ...) que no es posible establecer un agrupamiento razonable de dichas entradas.

El modelo de Jacobs-Jordan [JJ94] ya expuesto en el apartado 1 no es capaz de realizar una división del espacio de entradas cuando dicho espacio no es "agrupable" en subtareas de manera autónoma.

En general se puede decir que en el caso de entradas heterogéneas será necesaria la utilización de una descomposición "had hoc" o específica en la que se utilice un conocimiento previo del funcionamiento del sistema. En esta descomposición del problema en subtareas, hay que considerar siempre las propiedades físicas o funcionales de dicha tarea, y realizar de acuerdo con estas propiedades las divisiones oportunas. Obviamente, este reparto en subtareas y módulos dependerá de la tarea problema, y por lo tanto no es generalizable el procedimiento que se alcance.

3.2 La comunicación entre los módulos

La pregunta a responder en este apartado es *¿Cómo hacer interaccionar los distintos módulos para que el sistema completo realice la tarea objetivo?* Normalmente esta tarea se lleva a cabo por medio de un elemento, módulo, o capa que permite establecer una decisión a la vista de los resultados ofrecidos por los distintos módulos.

La comunicación entre los módulos se puede describir desde dos puntos de vista:

1. Cómo repartir la información durante la etapa de entrenamiento entre los distintos módulos.
2. Cómo integrar los distintos resultados ofrecidos por los diferentes módulos constitutivos del sistema para generar la salida final.

Reparto de Información El problema de cómo repartir la información durante la etapa de entrenamiento entre los distintos módulos es equivalente a determinar qué módulo debe ajustarse para aprender un determinado ejemplo, y de qué magnitud debe ser ese ajuste. Pueden destacarse algunas soluciones inmediatas a este problema:

- Por un criterio de proximidad. Puede reducirse al caso "el ganador se lo lleva todo", o bien puede haber varios ganadores. Posibles ejemplos de este proceder son los mapas autoorganizados de Kohonen, los sistemas de mezclas de gaussianas o el modelo de mezcla jerárquica de expertos de Jacobs-Jordan.
- Por una función lógica que de acuerdo con el valor o valores de ciertas variables de entrada, determina de forma única el módulo que debe ajustarse. Como caso particular, en algunos sistemas es posible determinar la existencia de una variable índice que permite seleccionar para cada momento el módulo a ajustar.

- Por medio de un autómata de estados finito, que de acuerdo no sólo a los valores de las entradas actuales, sino también de acuerdo a una secuencia finita de entradas que la precedieron, determina el módulo a ajustar y la cuantía del ajuste.
- Por medio de lógica borrosa. Una función de pertenencia borrosa determina cuál o cuáles de los módulos han de ser ajustados.
- Por medio de técnicas estadísticas. Mediante una distribución de probabilidad (o una densidad de probabilidad) se determina qué módulo ha de ajustarse.

Integración Una vez efectuado el análisis del problema, construidos y ajustados los módulos que se hacen responsables de su resolución, es necesario especificar el mecanismo que reuna cada uno de las soluciones parciales alcanzadas para crear la solución al problema original.

De un modo más general y, por supuesto, acorde a la manera en que se ha hecho el reparto de la información, se pueden distinguir diferentes métodos de integración o combinación de los módulos que se hayan considerado [MSJ97]:

- El mecanismo de "el ganador se lo lleva todo" sólo se puede plantear en aquellos sistemas en los que los expertos realizan tareas parecidas y ofrecen resultados homogéneos, no siendo este el caso de tareas como el de aparcar un camión, en el que el resultado (el ángulo de giro de las ruedas) es función de la posición de la cabina y del remolque. Para estos casos, es habitual encontrarse con un esquema de módulos en serie.
- Modelos en serie: la salida de un modelo es utilizada como entrada para el siguiente.
- Mecanismos de **votación**: la función *softmax* Una arquitectura muy simple para construir sistemas modulares se basa en la creación de una batería de módulos expertos, cada uno de ellos especializado en una tarea específica que ha sido determinada a priori, y establecer un sencillo mecanismo de votación, en el que cada uno de los módulos indica con su salida la certeza (probabilidad a posteriori) que existe sobre la salida final a generar conocida la entrada actual en el sistema. La salida efectiva del sistema será la asociada al módulo experto que ofrezca mayor certeza.

  De una manera esquemática se puede observar esta procedimiento en la figura 7. En ella aparecen K expertos, cada uno de ellos con una única salida escalar con la que valoran de forma independiente unos de otros, si la presente entrada al sistema pertenece o no al tipo para el que ha sido entrenado cada uno de ellos. Así, cuanta mayor certeza se tenga sobre la pertenencia de dicha entrada al conjunto utilizado en su aprendizaje, más alta será la salida que genere el módulo.

  Pero para que el esquema expuesto tenga sentido, las salidas proporcionadas por cada uno de los módulos expertos han de ser equiparadas a probabilidades, es decir, deben cumplir dos condiciones: que sus valores estén comprendidos entre cero y uno, y que su suma siempre sea uno. Estas dos condiciones

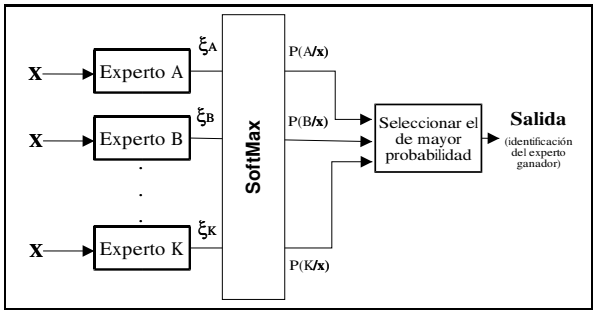


Figura7. Mecanismo de votación.

no se dan necesariamente para cualquier tipo de sistema con el que se construyan los módulos expertos, por lo que para que ocurra esto será necesaria la existencia de una etapa previa a la salida que adapte dichos valores. Una forma de lograrlo es a través de la función *softmax*. Si se denomina por ξ_i a las salidas o activaciones generadas por cada módulo experto, se asocia a cada uno un valor $P(i/\mathbf{x})$ que se obtiene al aplicar la fórmula:

$$P(i/\mathbf{x}) = \frac{\exp \xi_i}{\sum_{j=1}^{K} \exp \xi_j} \tag{1}$$

Los valores $P(i/\mathbf{x})$, que hacen las veces de probabilidades, permitirán seleccionar de todos los expertos aquel que ofrezca la mayor certeza. La salida del sistema será la etiqueta que se haya dado a los ejemplos utilizados durante el entrenamiento del experto ganador.
Cuando las etiquetas que se asignan a los expertos representan una variable cuantitativa, podría tener sentido el ofrecer como resultado final del sistema una combinación lineal de las etiquetas asociadas a cada módulo experto. El peso se cada módulo en la generación de la salida será proporcional a su probabilidad de éxito en el reconocimiento de la entrada como perteneciente a la clase para la que ha sido entrenado.

– La **combinación lineal de resultados** sólo tiene sentido cuando las salidas de los módulos son todas cuantitativas. Este es el esquema propuesto por [Has94] en su *combinación óptima lineal de redes neuronales mediante minimización del error cuadrático medio* (*MSE-OLC*).
En un principio se parte de un conjunto de distintas redes neuronales a las cuales se ha entrenado para la resolución del mismo problema. La idea subyacente es que combinación de los distintos resultados proporcionados por las diferentes redes neuronales permite integrar el conocimiento adquirido por cada una de ellas, consiguiendo así un aumento en la eficiencia del sistema. Esta combinación se hace mediante una suma ponderada de los resultados. El peso del resultado de cada una de las redes sobre el resultado final es ajustado de forma que el resultado final sea óptimo en el sentido de minimizar el error cuadrático medio.

Este sistema tiene dos variantes:

- *MSE-OLC con restricciones*, según el cual, los coeficientes que ponderan la influencia de cada red individual sobre la salida final del sistema están limitados por la condición de sumar 1.
- *MSE-OLC sin restricciones*, en el que los coeficientes de peso de cada red sobre la salida pueden tener cualquier valor.

– Mediante **Lógica Discreta**. Las circunstancias bajo las cuales se selecciona un módulo para generar la salida quedan definidas por una función lógica. Este método presenta problemas en las proximidades de las discontinuidades o transiciones entre las distintas zonas.

Si se dispone de un conocimiento previo que plasmado en una variable índice es capaz de identificar el módulo responsable de la generación de la salida, se podría utilizar un mecanismo multiplexor como el mostrado en la figura 8.

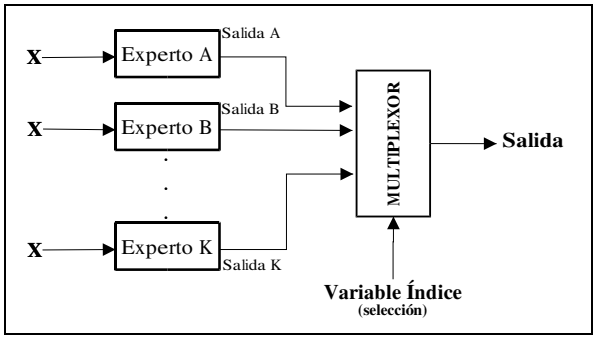
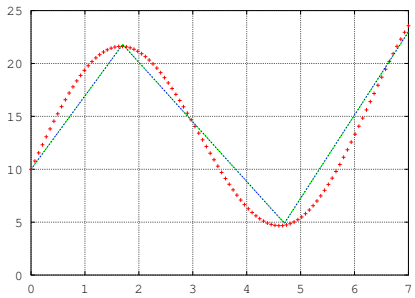


Figura8. Mecanismo multiplexor para seleccionar el módulo que genera la salida.

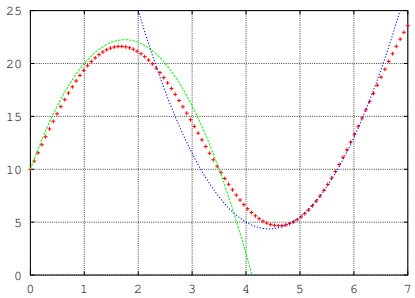
– Usando un **Autómata de Estados Finito** que determina el estado siguiente en función de una secuencia finita de los estados recorridos hasta el momento. Al igual que en el caso de la lógica discreta, pueden presentarse problemas en los instantes de transición entre un estado y el siguiente.

– Por medio de **Lógica Borrosa**. Se define una función de pertenencia difusa que indica el modelo a utilizar, lo que proporciona una transición suave entre los modelos al dar mayor o menor peso a cada modelo en función de un conjunto de variables borrosas.

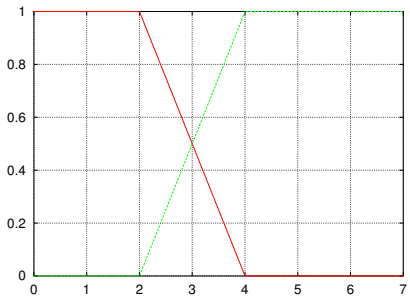
– Usando **técnicas probabilísticas**. En los tres casos anteriores la transición entre modelos era determinista, otra alternativa es utilizar métodos estadísticos para deducir qué modelo utilizar en cada instante. En este caso, cada modelo posee una densidad de probabilidad que indica lo apropiado que es para describir el funcionamiento del sistema en función de los datos disponibles.



(a) Funciones lineales y partición lógica discreta.



(b) Funciones cuadráticas y lógica difusa.



(c) Función de pertenencia para el caso de funciones cuadráticas.

Figura9. Aproximación de una función utilizando modelos locales. La gráfica punteada es la función a modelar, y las de trazo continuo las funciones locales utilizadas.

Otra manera de integrar a los expertos puede llevarse a cabo mediante una jerarquía, estrategia que principalmente se usa para reducir la complejidad de los problemas, siendo en estos casos bastante habitual la asignación de significados a cada experto en términos de representación interna de la tarea problema. La forma habitual de abordar el problema es buscar una variable que pueda describir la variación del comportamiento del sistema, con lo que se obtendrá una serie de puntos de funcionamiento que podrán dar origen a los correspondientes modelos y módulos.

3.3 Aprendizaje (ajuste de los módulos)

El ajuste o aprendizaje de cada uno de los módulos que conforman el sistema completo, de tal forma que cada módulo realice correctamente la subtarea que le ha sido asignada, en el caso de los sistemas en los que se ha efectuado una descomposición específica dependerá de la naturaleza del módulo y de la subtarea que se le haya asignado.

Cuando no existe un conocimiento previo del problema que pueda ayudar a realizar una división en subtareas, la principal cuestión será cómo realizar esta división. Inspirándose como siempre en los sistema biológicos, la idea que surge es la del **aprendizaje progresivo**: dividir la tarea principal en subtareas, seguidamente aprender a resolver todas las subtareas y por último integrar los resultados de cada subtarea para resolver el problema original. Este esquema es esencialmente un tipo de aprendizaje autoorganizado, en el que se desarrolla de forma autónoma una estructura para interactuar con el entorno.

En el caso de descomposición genérica, que principalmente se lleva a cabo de una manera no supervisada, hay ciertos procedimientos ampliamente utilizados que se basan en la homogeneidad de los módulos involucrados en la construcción del sistema; al ser todos del mismo tipo, el procedimiento a seguir es sistematizable. A continuación se presentan dos de estos procedimientos de ajuste, relacionados íntimamente con la mezcla jerárquica de expertos que representa el modelo de Jacobs-Jordan.

Algoritmo da aprendizaje estocástico Tal y como ya se vio al principio de este trabajo, uno de los tipos de sistemas modulares es el de Jacobs-Jordan. El modelo mostrado en la figura 1 se ajustaba fácilmente al caso particular de los sistemas compuestos por **mezcla o superposición de procesos estocásticos**, en el que cada módulo i constituye una regla o experto que produce una salida y_i, fruto de un proceso aleatorio cuya función de distribución para muchos casos prácticos suele considerarse gaussiana de media μ_i. Este valor μ_i es el valor medio de la respuesta deseada $\mathbf{y}$ condicionado a conocer el vector de entradas $\mathbf{x}$, con lo que su valores medios coincidirán: $y_i = \mu_i$.

Así, se puede escribir la función de distribución de la salida deseada $\mathbf{y}$ condicionada al conocimiento de la entrada $\mathbf{x}$ como sigue:

$$P(\mathbf{y}/\mathbf{x}) = \sum_{i=1}^{K} g_i P_i(\mathbf{y}/\mathbf{x}) \tag{2}$$

donde g_i son las correspondientes salidas de las redes de puertas.

Para el caso particular de mezcla de gaussianas con matriz de covarianzas identidad, la expresión anterior quedaría como sigue:

$$P(\mathbf{y}/\mathbf{x}) = \frac{1}{(2\pi)^{\frac{K}{2}}} \sum_{i=1}^{K} g_i \exp\left(-\frac{1}{2}||\mathbf{y_i} - \mu_i||^2\right) \tag{3}$$

También se habló al principio de este trabajo de la existencia de sistemas más genéricos, en los que se disponía de una jerarquía de expertos, tal y como se puede apreciar en la figura 10. En esta figura se representa un modelo jerárquico formado por un árbol de dos niveles de expertos. El primer nivel, que es el más profundo, está constituido por K bloques de L expertos cada uno, cuyos resultados se combinan por varios módulos de redes de puertas, dando origen a K conglomerados de expertos, y éstos a su vez se combinan por otra red de puertas para generar la salida.

Para este sistema en particular, se considera que cada una de las redes de expertos lleva asociada una distribución de probabilidad P_{ij}, que será función implícita de los parámetros de los que dependa el experto $\mathbf{w}_{ij}$ y de las entradas y salidas que se hayan utilizado para su ajuste $\{(\mathbf{x}^{(t)}, \mathbf{y}^{(t)}), t = 1 \cdots N\}$. Las denominadas redes de puertas del primer nivel generarán un conjunto de salidas $\{g_{j/i}\}, i = 1 \cdots K, j = 1 \cdots L$, y la red de puertas del segundo nivel generará un conjunto $\{g_i\}, i = 1 \cdots K$, que en ambos casos dependerán de una serie de parámetros $\{v_i\}$ y $\{v_{ji}\}$ respectivamente, y además de los pares de entrada y salida deseada utilizados durante su ajuste. Dado que las salidas de todas las redes de puertas se van a comportar como distribuciones de probabilidades que ponderan la participación de cada módulo experto en la salida final, los $\{g_i\}$ y $\{g_{j/i}\}$ habrán de ser todos positivos y sumar uno; una manera de conseguir esto es mediante la utilización de la función *softmax*. Así, si se denomina por ξ_i a la activación correspondiente a la salida i-ésima del la red de puertas del segundo nivel, los valores g_i se generarían por medio de la fórmula:

$$g_i = \frac{\exp \xi_i}{\sum_{j=1}^{K} \exp \xi_j} \tag{4}$$

Una fórmula análoga se tendría para los coeficientes $g_{j/i}$.

Ahora podría escribirse una fórmula semejante a la 2, pero para este nuevo sistema jerárquico a dos niveles, incluyendo de forma explícita la dependencia con los parámetros de todos los subsistemas:

$$P(\mathbf{y}/\mathbf{x}, \theta) = \sum_{i=1}^{K} g_i(\mathbf{x}, \mathbf{v_i}) \sum_{j=1}^{L} g_{j/i}(\mathbf{x}, \mathbf{v_{ji}}) P_{ji}(\mathbf{y}/\mathbf{x}, \mathbf{w_{ji}}) \tag{5}$$

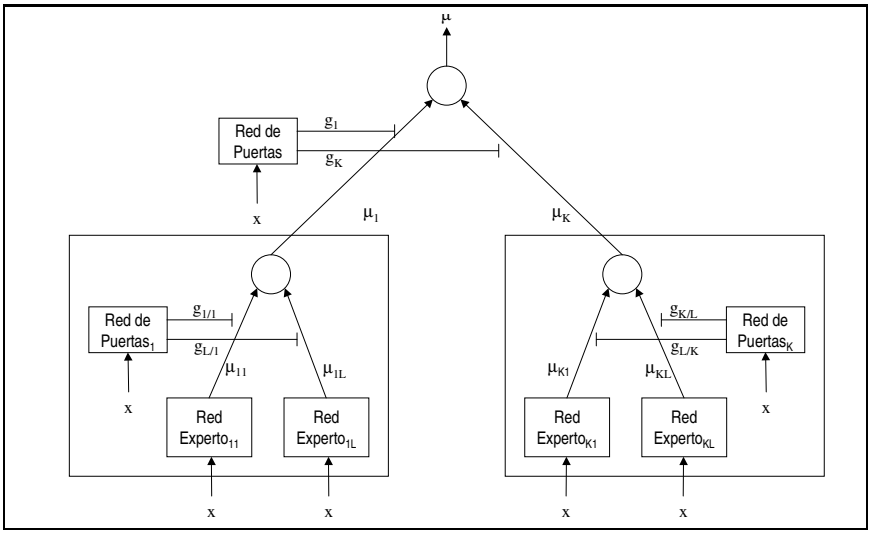


Figura10. El modelo de Jacobs-Jordan. El modelo aquí mostrado posee dos niveles de módulos expertos, y las redes de puertas (*gating networks*) generan tantos escalares de salida g_i como expertos tengan que controlar.

donde θ es el conjunto de parámetros que definen el sistema, y que incluye los de los expertos $\mathbf{w}_{ji}$, y los de las redes de puertas $\mathbf{v_i}$ y $\mathbf{v_{ji}}$.

Nótese que en el esquema que se acaba de exponer todos los niveles y módulos reciben como entrada el mismo vector $\mathbf{x}$.

Previo a la descripción de algún método de ajuste de los parámetros del sistema, se definen por cuestiones de notación las siguientes probabilidades condicionales a posteriori:

$$h_i = \frac{g_i \sum_{j=1}^{L} g_{j/i} P_{ji}(\mathbf{y})}{\sum_{i=1}^{K} g_i \sum_{j=1}^{L} g_{j/i} P_{ji}(\mathbf{y})} \tag{6}$$

El valor h_i representa la probabilidad de que el agrupamiento i-ésimo de expertos genere la respuesta deseada $\mathbf{y}$.

También se define otro conjunto de probabilidades a posteriori, que dan cuenta de la probabilidad de que el experto j-ésimo del agrupamiento i-ésimo genere una determinada salida deseada $\mathbf{y}$:

$$h_{j/i} = \frac{g_{j/i} P_{ji}(\mathbf{y})}{\sum_{j=1}^{L} g_{j/i} P_{ji}(\mathbf{y})} \tag{7}$$

Una manera de medir la bondad de los resultados obtenidos con el sistema es a través de la probabilidad de que dado un vector de entrada se obtenga su correspondiente vector de salida asociado. Si este mismo objetivo se debe cumplir

simultáneamente para todos los pares de entrada y salida usados en el ajuste del sistema, un buen parámetro de evaluación sería el producto de las distribuciones de probabilidad (ecuación 5) que ofrece el sistema para todos los pares de datos utilizados en el entrenamiento:

$$Q = \prod_{t=1}^{N} P(\mathbf{y}^{(t)}/\mathbf{x}^{(t)}, \theta) \tag{8}$$

Este parámetro Q recibe el nombre de *verosimilitud* (**"likelihood"** en inglés). Cuanto mayor sea este parámetro, mayor será la probabilidad de que el sistema asocie todos los vectores de entrada con sus correspondientes salidas, y como es natural, durante el proceso de ajuste del sistema, el objetivo será hacer máximo su valor, o lo que es equivalente, su logaritmo $\mathcal{L}$:

$$\mathcal{L} = \ln\left(\prod_{t=1}^{N} P(\mathbf{y}^{(t)}/\mathbf{x}^{(t)}, \theta)\right) = \sum_{t=1}^{N} \ln\left(\sum_{i=1}^{K} g_i^{(t)} \sum_{j=1}^{L} g_{j/i}^{(t)} P_{ji}(\mathbf{y}^{(t)})\right) \tag{9}$$

Aplicando la regla del gradiente decreciente para el caso de mezcla de gaussianas, y derivando el coeficiente $\mathcal{L}$ con respecto a las correspondientes activaciones se tiene:

- La ecuación

$$\frac{\partial \mathcal{L}}{\partial \xi_i} = h_i - g_i \tag{10}$$

  que refleja el hecho de que durante el aprendizaje la red de puertas del nivel superior va ajustándose de tal forma que las probabilidades a priori g_i se van aproximando hacia las probabilidades a posteriori h_i.
- La ecuación

$$\frac{\partial \mathcal{L}}{\partial \xi_{j/i}} = h_i(h_{j/i} - g_{j/i}) \tag{11}$$

  que refleja el hecho de que durante el aprendizaje de los conglomerados de expertos las probabilidades a priori $g_{j/i}$ se van acercando a las probabilidades a posteriori $h_{j/i}$.
- La ecuación

$$\frac{\partial \mathcal{L}}{\partial y_{ji}} = h_i h_{j|i}(\mathbf{y} - y_{ji}) \tag{12}$$

  indica que los módulos expertos durante el entrenamiento ajustan sus pesos en proporción al error cometido en la salida y también proporcionalmente a la probabilidad de que el experto en cuestión genere la salida deseada.

Aplicando a cada una de las ecuaciones anteriores la regla de la cadena, es posible llegar a conseguir fórmulas con las que poder actualizar adecuadamente los pesos (parámetros) de las redes de puertas y cada uno de los expertos.

En [Hay94] se puede encontrar de forma detallada los pasos a seguir para el entrenamiento de este tipo de sistemas.

El algoritmo de Maximización del Valor Esperado (EM) Otra posible alternativa para el ajuste de los parámetros que definen la mezcla jerárquica de expertos es el uso del algoritmo de maximización del valor esperado (*"Expectation Maximissation"*).

La idea fundamental en el algoritmo EM es que la tarea de maximizar el parámetro $\mathcal{L}$ se hace más sencilla si pudiera conocerse los valores que toman un conjunto de parámetros que permanecen desconocidos. Así, por ejemplo, la tarea de calcular el valor de la función de coste $\mathcal{L}$ sería trivial si se dispusiera de un conjunto $\mathbf{Z}$ constituido por $\{z_i\}$ y $\{z_{j/i}\}$. Estas variables respectivamente hacen las veces de etiquetas que identifican para cada vector de entradas $\mathbf{x}$ cuál es el conglomerado de expertos que debe tenerse en cuenta en el proceso de generación de la salida $\mathbf{y}$, y cuál de todos los módulos que forman el conglomerado i es el que en concreto genera la salida. Así se podría definir otro conjunto de variables $\{z_{ij} = z_i z_{j/i}\}$, tal que

$$z_{ij} = \begin{cases} 1 & \text{si es el experto j del conglomerado i el que genera la salida} \\ 0 & \text{en cualquier otro caso} \end{cases} \tag{13}$$

Como es obvio, estas variables z_i y $z_{j/i}$ no son conocidas, ya que si lo fueran, el problema del aprendizaje estaría resuelto, porque sólo se ajustaría el módulo y la conexión oportuna. Gracias a la introducción de estas variables, la expresión de $\mathcal{L}$ que aparece en la ecuación 9 se puede volver a escribir como sigue:

$$\begin{aligned} \mathcal{L} &= \sum_{t=1}^{N} \ln \left(\sum_{t=1}^{N} \sum_{i=1}^{K} g_i^{(t)} \sum_{j=1}^{L} g_{j/i}^{(t)} P_{ji}(\mathbf{y}^{(t)}) \right) = \\ &= \sum_{t=1}^{N} \ln \left(\prod_{i=1}^{K} \prod_{j=1}^{L} (g_i^{(t)} g_{j/i}^{(t)} P_{ji}(\mathbf{y}^{(t)}))^{z_{ij}^{(t)}} \right) \end{aligned} \tag{14}$$

gracias a que la variable z_{ij} hace que cada término producto sea 1 en el caso de que no sea el experto responsable de esa salida deseada $\mathbf{y}$ (y por lo tanto no afecte al resto de términos del producto), y que valga justamente $P_{ji}(\mathbf{y})$ cuando se trate del módulo experto correcto. De esta forma la función logaritmo se puede reescribir como suma de logaritmos:

$$\mathcal{L} = \sum_{t=1}^{N} \sum_{i=1}^{K} \sum_{j=1}^{L} z_{ij}^{(t)} (\ln(g_i^{(t)}) + \ln(g_{j/i}^{(t)}) + \ln(P_{ji}(\mathbf{y}^{(t)}))) \tag{15}$$

El algoritmo EM se lleva a cabo de forma iterativa en dos pasos:

1. **Paso E**: cálculo de la esperanza matemática de $\mathcal{L}$ sobre el conjunto formado por todos los pares de entrenamiento $\mathbf{X} = \{(\mathbf{x^{(t)}}, \mathbf{y^{(t)}}), t = 1 \cdots N\}$:

$$\mathcal{E}(\theta, \theta^{(p)}) = E(\mathcal{L}(\theta, \mathbf{Z})/\mathbf{X}) = \sum_{i=1}^{K} \sum_{j=1}^{L} h_{ji}^{(t)} (\ln(g_i^{(t)}) + \ln(g_{j/i}^{(t)}) + \ln(P_{ji}(\mathbf{y}^{(t)}))) \tag{16}$$

donde $\theta^{(p)}$ es la estimación de los parámetros en la iteración p, $\mathbf{Z}$ es el conjunto de variables ocultas, y se ha tenido en cuenta además que $E(z_{ij}^{(t)}/\mathbf{X}) = h_{ij}^{(t)}$.

2. **Paso M**: obtener la siguiente estimación de los parámetros $\theta^{(p+1)}$ que maximice el valor esperado estimado calculado en la fase E:

$$\theta^{(p+1)} = \arg\max_{\theta} \mathcal{E}(\theta, \theta^{(p)}) \tag{17}$$

Este problema de maximización se puede reducir a la maximización de cada uno de los tres sumandos más interiores que aparecen en la ecuación 15. Así se tienen las siguientes operaciones:

$$\mathbf{w}_{ji}^{(p+1)} = \arg\max_{\mathbf{w}_{ji}} \sum_{t=1}^{N} h_{ji}^{(t)} \ln P_{ij}(y^{(t)}) \tag{18}$$

$$\mathbf{v}_{i}^{(p+1)} = \arg\max_{\mathbf{v}_{i}} \sum_{t=1}^{N} \sum_{k=1}^{K} h_{k}^{(t)} \ln g_{k}^{(t)} \tag{19}$$

$$\mathbf{v}_{ji}^{(p+1)} = \arg\max_{\mathbf{v}_{ji}} \sum_{t=1}^{N} \sum_{k=1}^{K} h_{k}^{(t)} \sum_{l=1}^{L} h_{l/k}^{(t)} \ln g_{l/k}^{(t)} \tag{20}$$

En la literatura se pueden encontrar métodos para resolver estos problemas de maximización, como pueden ser el WLS (*"Weighted Least-Squares"*)[Moe97], o IRLS (*"Iteratively Reweighted Least Squares"*) [JJ94],

Analizando 16, los dos primeros términos $\sum_i \sum_j h_{ji} \ln g_i$, y $\sum_i \sum_j h_{ji} \ln g_{j/i}$ se pueden asimilar a entropías conjuntas, que medirían la entropía de la distribución de los patrones $\mathbf{x}$ entre los conglomerados de expertos y los expertos respectivamente. De acuerdo con esta interpretación, el valor esperado $\mathcal{E}$ se maximiza cuando los conglomerados son mutuamente excluyentes, y disminuye cuando existen datos de entrada que hacen que se activen simultáneamente más de un conglomerado. De forma análoga se puede razonar con el segundo término para cada uno de los expertos que forman los conglomerados. En cuanto al tercer término, $\sum_i \sum_j h_{ij} P_{ji}(\mathbf{y})$ indica que los expertos que más pesan en el valor de $\mathcal{E}$ son aquellos cuya probabilidad h_{ij} es mayor [Moe97].

3.4 Ajuste del tamaño y estructura de los módulos

Como siempre, no es posible indicar un único método que determine el número de capas, elementos de proceso en cada una de ellas, o las conexiones entre ellos. Algunas técnicas aplicables se exponen a continuación.

Los **métodos de poda** reducen el número de elementos constituyentes de cada módulo. Se basan en una vez entrenada la red neuronal artificial, buscar

conexiones entre elementos de proceso que al ser eliminadas produzcan una variación mínima en el comportamiento de la respuesta del sistema. Una manera de llevar a cabo esto es mediante el uso de técnicas como el "weight decay", consistente en agregar al algoritmo de actualización de pesos durante el aprendizaje la condición de minimizar el módulo de los pesos de las conexiones entre los elementos de proceso (ver el apéndice A). Otra alternativa es observar en el valor de los pesos: si es próximo a cero, la señal que circule a través de esa conexión afectará pobremente a la salida del elemento de proceso receptor, y por lo tanto, es susceptible de ser eliminada. En cualquier caso, el uso de esta técnica implica volver a entrenar la red después de haber eliminado dicha conexión.

Una alternativa más para la aplicación de los métodos de poda es usar el método denominado *"Optimal Brain Damage"* propuesto por *Le Cun*, que básicamente recalcula la función de error con y sin una determinada conexión (la ausencia de conexión implica que el peso es nulo). Si se parte de un sistema bien entrenado, la función *error* se estará evaluando en un mínimo (al menos un mínimo local), y por lo tanto, su expansión en serie de Taylor de segundo orden será de la forma $\delta E = \frac{1}{2} h_{ii} (\delta w_i)^2$, donde E es la función de error, $h_{ii} = \frac{\partial^2 E}{\partial u_i^2}$ (elemento i-ésimo de la diagonal del Hessiano) y δu_i es la variación que se va a dar al peso bajo estudio. Puede indicarse [Sva94] que uno de los problemas que conlleva este método es que se está calculando el cambio en la función error antes de volver a entrenar, y que el cálculo del Hessiano es computacionalmente muy costoso.

Los métodos **incrementales** se basan en la idea de la "modularización evolutiva", consistente en una modificación del bien conocido algoritmo de correlación en cascada (*"cascade correlation"*). Se parte de un módulo que es entrenado para responder adecuadamente a una tarea. Cuando se ha alcanzado una determinada cota de error, se añade un segundo módulo en cascada con el primero, y se le entrena para mejorar la respuesta del sistema, utilizando para ello los resultados generados por el primer módulo, al que ya no se le modifica (ahora sólo "aprende" el segundo módulo). Este esquema se repite añadiendo tantos módulos nuevos en serie con los anteriores como sea necesario para alcanzar la cota de error deseada.

Otra alternativa relacionada con los métodos incrementales son las redes **RCE** (*Restricted Coulonb Energy*) [Tve95]. En ellas se parte de un conjunto de datos (puntos en el espacio de representación). Cada elemento de proceso de la capa de entrada de la red neuronal artificial lleva asociada una hiperesfera en el espacio de entrada cuyo radio representa el entorno en el que dicho elemento de proceso es activo; si con la primera hiperesfera no es suficiente para resolver el problema considerado, se le añade un segundo elemento de proceso con su correspondiente hiperesfera, y se reajustan todas las hiperesferas (centros y radios) que se hayan construido hasta ese momento, de tal forma que se satisfaga la tarea problema sobre los puntos considerados. Como se puede ver, se trata de un mecanismo parecido al seguido con las redes tipo RBF, pero ahora se opera de una manera incremental.

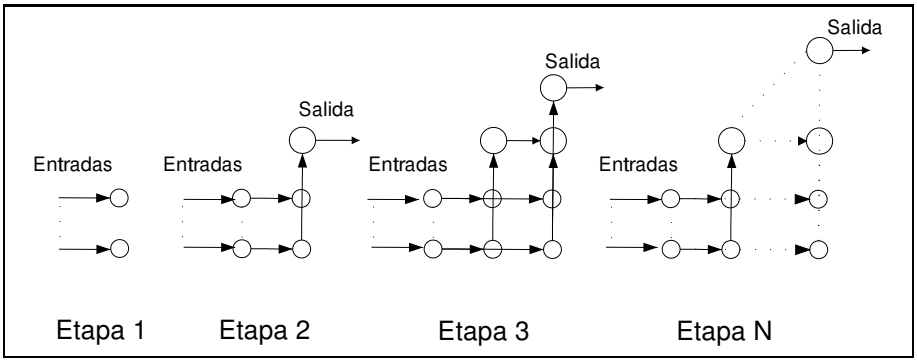


Figura11. Esquema del método Correlación en Cascada.

4 Ajuste de la Arquitectura Modular

En esta sección se plantea el problema de cómo ajustar los parámetros libres del sistema como pueden ser el número y tipo de módulos que lo componen, o la arquitectura que los relaciona.

Algunos autores estiman el número de expertos por el número de clases que se pueden encontrar en el conjunto de datos de entrada. Otros proponen incluir en los propios algoritmos de aprendizaje mecanismos para definir la estructura modular. Así, se tienen los métodos de poda y los incrementales que ya han sido descritos para el ajuste de los módulos individuales.

Muchas veces, cuando se dispone de **información a priori** de la tarea a realizar y del sistema que da lugar a dicha tarea, es posible llevar a cabo un análisis del problema planteado para así descomponerlo en subtareas que son asignadas de manera **supervisada** a los distintos módulos constitutivos, quedando así definido su número, función y relación entre ellos.

Cuando no se dispone de este tipo de información previa sobre el problema a resolver, se confiará en el uso de métodos no supervisados para la determinación del número de módulos a usar, su función asociada y la relación que se debe establecer entre ellos.

Si bien la interpretación *razonada* de los resultados obtenidos por medio de un enfoque basado en una asignación explícita utilizando conocimientos apriorísticos parece inmediata (o al menos debiera ser posible), la aproximación no supervisada abre la puerta al descubrimiento de una interpretación a posteriori de la funcionalidad de los módulos usados y de las relaciones existentes entre ellos, con lo que además de resolver el problema planteado (generalmente la síntesis de una función desconocida que relacione las entradas con las salidas deseadas), sería posible obtener información acerca del mecanismo interno intrínseco al sistema que genera esa función.

Una primera aproximación al problema del ajuste del sistema modular desde un punto de vista supervisado, es el ajuste de forma individual de cada uno de

los módulos de los que consta. Esto es factible puesto que es conocido el comportamiento que se espera de cada uno de ellos. Usando métodos tan elementales como el simple ensayo y error se ajustarían los correspondientes parámetros del módulo.

¿Qué hacer cuando no se tiene información sobre la estructura que debe tomar el sistema modular? En estos casos una posible solución es la utilización de algoritmos de búsqueda, como lo son los Algoritmos Genéticos, o el método de Monte Carlo [3].

La filosofía del método de Monte Carlo se puede describir en los siguientes pasos:

1. Generar K puntos aleatorios en el espacio de parámetros.
2. Evaluar la bondad del sistema en cada uno de esos puntos.
   Si en alguno de esos puntos el error cometido en el sistema es menor que una determinada cota, o se han superado el número prefijado de iteraciones del proceso de búsqueda, el conjunto de parámetros a utilizar será el definido por el punto que ofrezca mejores resultados. Ir al paso 3.
   Si el error en todos los puntos es superior a la cota elegida, se escoge aquel punto de resultado mejor y se generan de forma aleatoria otros K puntos próximos a él. Incrementar el número de iteraciones en uno, e ir al paso 1.
3. Fin.

La configuración de parámetros a utilizar sería la del ganador, pero dado que es una búsqueda aleatoria, todo el proceso debería repetirse un número lo suficientemente grande de veces. El valor final de los parámetros sería, por ejemplo, aquel que haya ofrecido el mejor resultado entre todas las pruebas realizadas.

En el método de Monte Carlo que se acaba de describir la búsqueda se realizaba generando puntos al azar dentro del espacio de parámetros, y se escogía aquel punto para el que el sistema ofrecía el mejor comportamiento. Los Algoritmos Genéticos [Gol89,Koz93]también se basan en la generación de un número muy elevado de puntos dentro del espacio de parámetros, para posteriormente calcular la bondad de los sistemas a los que dan origen. En este método, la bondad o aptitud de cada sistema para cumplir correctamente con su objetivo se mide a través de una función denominada "*fitness*" en inglés.

Las diferencias comienzan a partir de este punto. De entre todos los sistemas generados a partir de los distintos puntos en el espacio de parámetros, se hacen dos conjuntos disjuntos: los que dan buenos resultados y los que dan malos resultados. La población (sistemas) que ofrecen malos resultados son descartados, y los miembros de la población que ofrecen buenos resultados se dice que

[3] También se podría incluir aquí los métodos de ajuste basados en la minimización de una función coste mediante el uso de los algoritmos del tipo del gradiente decreciente, pero su aplicación sólo sería adecuada cuando se tratase del ajuste de parámetros de naturaleza continua, para las que existe el concepto de derivada. En el caso de ajuste de parámetros discretos, como el número de capas en una red neuronal, o el de elementos de proceso en una capa específica, carece de sentido este método.

sobreviven. A partir de estos últimos se genera una nueva población mediante los mecanismos de selección, cruce. inversión, y mutación.

Por medio de estos mecanismos aparecen varias posibilidades:

- conservar los rasgos de la población que ofrece buenos resultados y que esta población sobreviva, haciendo que estos rasgos se conserven en la siguiente generación,
- aparición de nuevos rasgos, ya sea por mezcla de los existentes, o por la introducción de características totalmente nuevas que no existían en la población original.

Para poder realizar las operaciones anteriormente descritas, es necesario definir la función *fitnness*, así como disponer de un método que permita la representación adecuada de los parámetros que definen a los distintos sistemas (*individuos* en la terminología de los algoritmos genéticos) bajo estudio, y que a la vez permita la realización de las operaciones de selección, cruce, inversión y mutación. De una manera intuitiva, se pueden definir los métodos de codificación de variables de tipo cuantitativo (por ejemplo el número de elementos de proceso en una red neuronal, o el número de capas), pero ya no es tan inmediato cuando se intenta describir aspectos tales como la arquitectura del sistema modular (los distintos módulos y su forma de interconexión, si forman capas o no, si reciben o no sus entradas de todas las salidas de una capa ya existente, ...).

En [Rij95] se puede encontrar detallado un método basado en lenguajes formales para la descripción de la arquitectura de un sistema modular de redes neuronales, así como la de los operadores genéticos que aplicados a una población darán origen a una nueva.

El problema "TC": ajuste del sistema modular por medio de los Algoritmos Genéticos. En [BKHSK93,Rij95] se intenta dar solución al problema "Qué y Dónde" descrito en el apartado 2, pero esta vez usando técnicas de Algoritmos Genéticos para encontrar la arquitectura óptima. El procedimiento se inicia generando una población de redes cuya estructura es generada aleatoriamente (con un número aleatorio de capas y subcapas, elementos de proceso y conexiones entre las distintas subcapas). Mediante un simulador de redes neuronales artificiales y el algoritmo de retropropagación del error se entrenan las redes de la población y el error residual que se produce al probarlas es usado como parámetro para seleccionar aquellas redes (individuos de la población) que serán utilizados para dar lugar a la siguiente generación mediante las operaciones de selección, cruce, inversión y mutación. Los experimentos se repitieron 50 veces, y la estructura final resultante para la red encargada de la identificación del tipo de objeto después de 7500 recombinaciones (iteraciones del algoritmo genético) se puede observar en la figura 12.

Los resultados que se obtuvieron, dada la naturaleza aleatoria de la inicialización de pesos y de la generación de los nuevos individuos, fueron que 45 de las 50 ocasiones en las que se entrenó la red, ésta ofreció siempre los resultados correctos, mientras que si se utilizaba un perceptrón multicapa con 6 elementos de proceso en la capa oculta, sólo se llegó a un acierto del 100% en 30 de

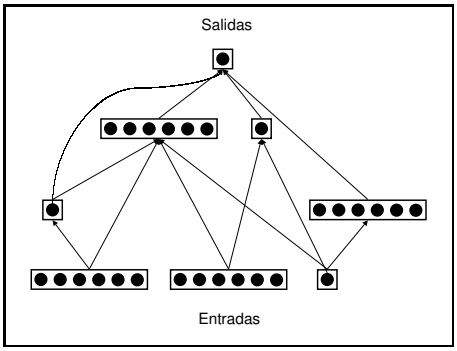


Figura12. Solución para la tarea de identificación en el problema "TC" propuesta por [BKHSK93] que usa algoritmos genéticos para el ajuste de la arquitectura del sistema

los 50 intentos (el resto de las arquitecturas que se probaron ofrecieron peores resultados). Una nota curiosa que podría destacarse es que el sistema ganador sólo posee 13 entradas de las 16 que cabría esperar. Esto es posible justificarlo teniendo en cuenta que cualquiera de los objetos ("T" ó "C" en cualquiera de las posibles posiciones) al ser colocado en la rejilla de 4×4, se puede identificar con sólo 13 de las 16 entradas (los valores de las otras 3 que faltan se pueden calcular perfectamente).

En el ejemplo que se acaba de presentar, cada individuo que se generaba en la nueva generación era creado compartiendo ciertas características arquitectónicas de sus progenitores, a las que se añadían las que pudieran aparecer por los procedimientos de cruce inversión y mutación, es decir, se heredaban ciertos rasgos arquitectónicos, pero en ningún caso se heredaban los conocimientos que habían adquirido los progenitores; lo que se hereda es la capacidad de aprendizaje y en consecuencia, de adaptación al entorno (esto se conoce como el *efecto Baldwin* [Rij95]).

5 Sistemas Híbridos

Algunos de los sistemas modulares que se han descrito utilizan un conocimiento apriorístico para establecer la división de la tarea objetivo en subtareas, y posteriormente se asigna un subsistema específico cuya naturaleza puede ser heterogenea para cada uno de ellos.

De entre la multitud de paradigmas de redes neuronales artificiales existentes en la actualidad, se encuentran algunos que en sí mismos constituyen sistemas híbridos y, que de alguna manera, combinan diferentes mecanismos de entrenamiento y procesamiento de la información. Así, es posible enumerar entre otros:

- **Redes de Contra Propagación** (*"Counter Propagation Net"*). Una primera parte de la red realiza un aprendizaje no supervisado competitivo,

dividiendo el espacio de entradas. Después del aprendizaje competitivo, se entrena de forma supervisada el segundo nivel de la red para que se obtenga la salida deseada.

- **Redes de Función de Base Radial (RBF)**. Una primera capa está constituida por elementos que implantan funciones de base radial, cuya activación es proporcional a la distancia entre la entrada y el centroide de la correspondiente función de base radial. El segundo nivel de la red RBF está constituido por elementos de proceso que aprenden de forma supervisada a producir la salida deseada a partir de la activación proporcionada por las funciones de base radial.
- **Redes ARTMAP**. Estas redes realizan un aprendizaje no supervisado tanto en la entrada como a la salida, con lo que se pretende obtener un agrupamiento adecuado de los datos. El entrenamiento supervisado se lleva a cabo en la parte intermedia de la red, siendo su objetivo el asociar patrones de entrada con patrones de salida.

6 Extracción de reglas del sistema ya ajustado

Uno de los puntos débiles de las redes neuronales artificiales es su incapacidad para poder justificar razonadamente sus resultados, y en consecuencia, esta situación da origen a una desconfianza implícita en el usuario final, cuando estos sistemas son utilizados por el público en general y se se aplican a problemas del mundo real. Si fuera posible extraer reglas simbólicas de las propias redes neuronales que justifiquen su comportamiento, esto tranquilizaría al usuario potencial.

Hablando de un modo general, es muy difícil la tarea de extraer reglas de una red neuronal ya entrenada, y aún así, el conocimiento embebido en la red a veces no es completo, por lo que el proceso de extracción puede que no produzca un conjunto de reglas del todo correctas. Hay ciertos tipos de redes neuronales artificiales que hacen fácil la tarea de extracción de reglas, ya sea por sus características arquitectónicas o por las funcionales; y de otro lado, existen paradigmas de redes en los que esta tarea de extracción no es tan inmediata. Un tipo especial de redes que se adapta muy bien a este proceso de extracción son las KBANN (*"Knowledge Based Artificial Neural Network"*) [OS93,MT94,OS94,OS97], de las que conviene aclarar que sus entradas y salidas son todas binarias.

El punto de partida de estas redes es la denominada *teoría de dominio*. Se trata de sistemas inductivos que utilizan un conjunto de reglas aproximadamente correcta inferidas de un dominio o problema específico, para intentar describir lo que se conoce sobre dicho dominio.

El algoritmo KBANN traduce cada regla de partida que describe el comportamiento de un sistema en su equivalente en forma de red neuronal artificial, ofreciendo una primera aproximación a la arquitectura y pesos de las conexiones entre los distintos elementos de proceso, y posteriormente aplica el algoritmo de retropropagación del error para reajustarla y refinar su comportamiento.

Al aplicar el algoritmo KBANN, las reglas conjuntivas se transforman en redes neuronales en las que los pesos de los enlaces correspondientes a antecedentes

no negados se les asigna valor ω, las conexiones con antecedentes negados se les asigna pesos con valor $-\omega$, y el término polarización del elemento de proceso que genera el consecuente toma un valor $2P - 1/(2\omega)$, donde P es el número de premisas no negadas. En la mayoría de los casos se ha encontrado que $\omega = 4$ es un valor que ofrece buenos resultados. Para codificar una regla disyuntiva, KBANN asigna el valor ω a los pesos a cada enlace con los antecedentes, mientras que al término polarización del elemento de proceso que genera el consecuente le asigna el valor $\omega/2$. En la figura 13 (a) y (b) se puede ver las subestructuras generadas por las reglas que se indican.

El algoritmo KBANN traduce una colección de reglas en una red neuronal tomando cada regla de forma individual y transformándola en una subred que reproduzca de manera precisa el comportamiento de dicha regla. Todas las subredes que se hayan ido construyendo se ensamblan en una red neuronal cuyo comportamiento ha de ser el de la regla completa. Una vez obtenida la red completa (figura 13(c)), se completa con aquellas conexiones que no se incluyen de forma explicita a través de las reglas, con un valor de peso aleatorio. A partir de aquí, y por medio del algoritmo de retropropagación del error, se ajusta el sistema, confiando en que el resultado final sea una refinación del original.

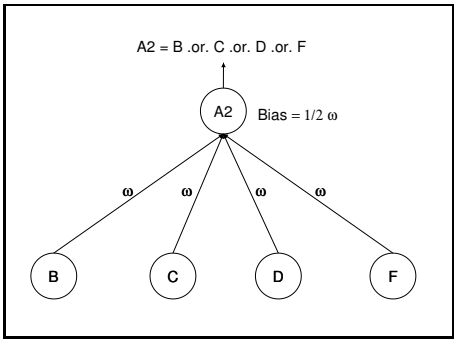
Una vez entrenado el sistema, se puede emprender el camino inverso: examinando los pesos de las interconexiones entre los distintos elementos de proceso, construir reglas que justifiquen el comportamiento de la red.

Esta operación presupone que durante el funcionamiento de la red neuronal las salidas de todos los elementos de proceso (incluidos los de las capas ocultas) van a ser binarias (cercana a 0 ó a 1). Cuando un elemento de proceso tiene una activación igual a uno indica que la premisa que representa es verdadera, y cuando está a 0 se supone que la correspondiente premisa es falsa. En consecuencia, para la extracción de reglas sólo se necesita examinar los valores de los pesos asociados con el elemento de proceso.

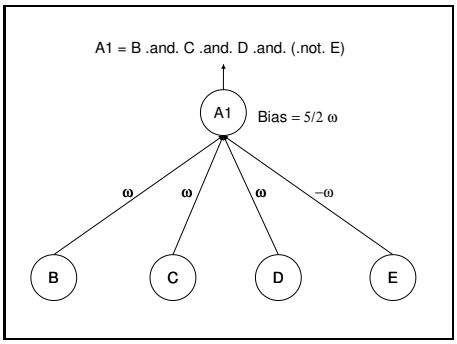
Un ejemplo: extracción de reglas en predicción de series económicas Uno de los tipos de redes neuronales artificiales más utilizados es el perceptrón multicapa, al que se le han asignado tareas de reconocimiento de patrones en series temporales obteniéndose resultados aceptables en ellas. Sin embargo, la forma de trabajar de estas redes obliga a presentar como entrada una ventana temporal sobre los datos, es decir, la red no guarda internamente ninguna noción semejante a la de "estado", pasando a ser responsabilidad del diseñador el presentar la historia adecuadamente ordenada al sistema de pronóstico.

Las redes neuronales recurrentes incluyen de forma natural la noción de estado gracias a los diferentes lazos de realimentación que se establecen en su diseño. Una consecuencia de esto es que parecen más adecuadas para tratar con problemas relacionados con la detección de patrones en las series temporales.

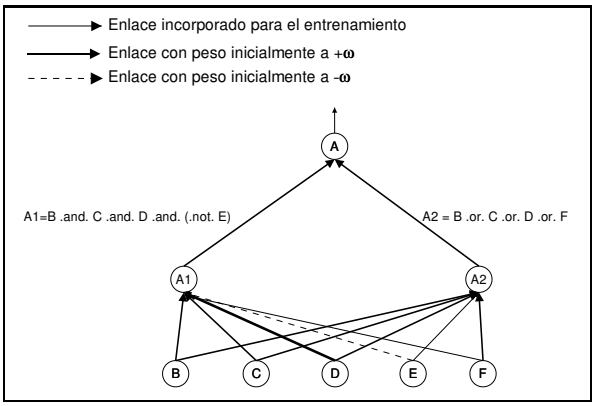
El método propuesto en [GLT97,LYC97] consiste en primeramente convertir la serie de valores de entrada en una secuencia de símbolos de un alfabeto discreto, y alimentar una red recurrente con estos símbolos adecuadamente codificados, siendo ésta la encargada de producir el pronóstico requerido. Este



(a)



(b)



(c)

Figura13. Ejemplos de codificación para dos reglas en las redes KBANN.

esquema aprovecha la habilidad de las redes neuronales recurrentes en la extracción del conocimiento simbólico. El uso de la red recurrente es significativo en dos aspectos: primero el modelo ataca el problema de la relación temporal entre los datos que forman la serie manteniendo un estado interno, y segundo, es posible la extracción de reglas generales de la red una vez entrenada, posibilitando al usuario humano la interpretación razonada de los resultados obtenidos. Las reglas así obtenidas pueden dar origen a un autómata finito determinista que refleje el funcionamiento del sistema.

El problema a resolver en este ejemplo es efectuar un pronóstico sobre la razón de cambio entre monedas de distintos países. El sistema de pronóstico utilizado se muestra en la figura 14, donde aparece un diagrama que puede considerarse dividido en dos grandes partes: los dos primeros módulos encargados del preprocesamiento y adaptación de los datos de partida, y los dos últimos que procesan la información y generan el pronóstico final.

Los datos de partida son la serie formada por la relación de cambio entre monedas al cierre de las sesiones a lo largo de varios días para una moneda en particular: $y(k), k = 1, 2, \ldots N$. La primera operación que se lleva a cabo es la diferenciación de la serie $\delta(k) = y(k) - y(k-1)$: no se van a predecir los cambios absolutos, sino su variación de un día para otro. Para facilitar su manipulación reduciendo su rango de variación, se utiliza una escala de tipo logarítmica: $x(k) = signo(\delta(k))(\log(|\delta(k)| + 1))$.

Sobre la serie transformada $x(k), k = 1, ..., N-1$ se considera una ventana de d datos que se va desplazando a lo largo del tiempo. En definitiva se tiene un conjunto de vectores $\mathbf{X}(k, d)$, donde el ancho de la ventana (historia a considerar en cada entrada al sistema de pronóstico) sólo fue considerado con dos valores: 1 ó 2.

El módulo encargado de realizar la cuantificación, es decir, transformar la serie $\mathbf{X}(k, d)$ en una secuencia de símbolos discretos, es un mapa autoorganizado de Kohonen, que ofrecerá como información de salida las coordenadas del nodo con la máxima activación para cada vector de entrada que se le presente: $S(k) = g(\mathbf{X}(k, d))$.

Una red recurrente tipo Elman toma la salida del módulo autoorganizado, se entrena para realizar la inferencia gramatical oportuna y obtener la predicción buscada.

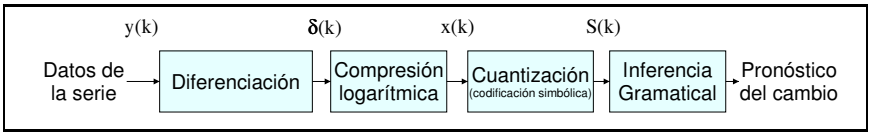


Figura14. Diagrama de bloques del procedimiento seguido en [GLT97].

Entender la forma de funcionamiento de una red neuronal artificial se puede intentar mediante la extracción de reglas. La terna que define un proceso dis-

creto de Markov $\{(estado;\ entrada \rightarrow siguiente\_estado)\}$ puede dar origen a un autómata de estados finito, cuyo funcionamiento ya es determinista, y por lo tanto proporcionaría las reglas buscadas. En el problema que se ha planteado, esto puede ser realizado mediante el agrupamiento de los valores de activación de los elementos de proceso de la capa de estado que almacenan el estado del sistema, y a partir de los estados (clases) detectados, un simple algoritmo puede asignar probabilidades de transición entre los distintos estados descubiertos. De forma más detallada, la activación de cada uno de los N elementos de proceso de la capa de estado es dividido en q intervalos de igual tamaño, produciéndose una partición del espacio de estados en q^N zonas o estados. Comenzando por el estado inicial, se considera que los símbolos de entrada siguen el orden. Si un símbolo de entrada causa una transición de estado, entonces se crea un estado del correspondiente autómata finito determinista, que lleva del estado de partida al nuevo a través de la transición descrita y con el símbolo de entrada correspondiente. Abordar el problema directamente puede ser intratable ya que el número de estados posibles es exponencial q^N, y por eso se realiza un agrupamiento, reduciendo la dimensionalidad del problema.

7 Algunos Ejemplos

7.1 El sistema híbrido BP-SOM

El sistema BP-SOM [WBH96,WHBP97] es un caso especial entre los sistemas híbridos, en el que coexisten varios subsistemas (un perceptrón multicapa y uno o más mapas autoorganizados de Kohonen) sólo durante la etapa de aprendizaje, mientras que una vez ajustado el sistema completo, se prescinde de los mapas autoorganizados, siendo el perceptrón multicapa el que procesa la información de entrada y ofrece la salida del sistema.

Las redes neuronales tipo perceptrón multicapa entrenadas con el algoritmo de retropropagación del error son uno de los paradigmas conexionistas más utilizados por su potencia y flexibilidad en la resolución de problemas genéricos. Sin embargo, este tipo de redes neuronales artificiales posee un problema inherente: la escasa generalización cuando se enfrentan a tareas para las que se usa un conjunto de datos para entrenamiento con ciertas características que lo hacen poco adecuado para este propósito. Así, el fenómeno del **sobreentrenamiento** consiste en la incapacidad de proporcionar respuestas correctas a entradas desconocidas aún habiendo aprendido perfectamente a producir las respuestas correctas con los ejemplos de entrenamiento. Muchas veces este efecto es debido a causas tales como la *dispersión* de los datos utilizados en el entrenamiento, o la existencia de una *no linealidad muy elevada* en la tarea propuesta.

El método propuesto en el modelo BP-SOM consiste en la definición de un *nuevo algoritmo de aprendizaje* que tiene como objetivo *guiar* el aprendizaje de una red perceptrón multicapa, de tal manera que las activaciones de los elementos de proceso de la capa oculta producidas por entradas de la misma clase sean lo más parecidas entre sí. Para conseguir esto se combina la red perceptrón

multicapa con un mapa autoorganizado de Kohonen asignado a cada una de las capas ocultas. Durante el entrenamiento de los pesos de cada capa oculta, también es entrenado el correspondiente mapa autoorganizado, tomando como entrada de éste último las activaciones de los elementos de proceso de la capa oculta asociada. El error calculado con el algoritmo de retropropagación es completado añadiéndole información sobre el mapa adjunto, y a partir de él se actualizan los pesos. El efecto buscado es que los patrones de activación de las capas ocultas generados por entradas de la misma clase sean similares entre sí.

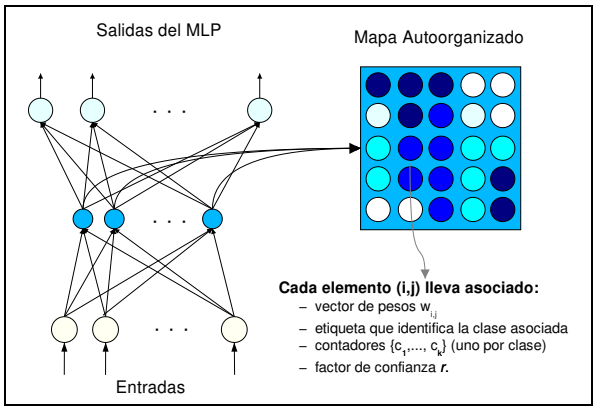


Figura15. Sistema híbrido BP-SOM

En la figura 15 se puede observar de forma esquemática un sistema BP-SOM con una única capa oculta. Allí se detalla la información asociada a cada uno de los elementos o nodos del mapa de Kohonen: el vector de pesos que lleva asociado, la etiqueta asignada, una serie de contadores (uno por cada clase con la que se esté trabajando), y un coeficiente de confianza o certeza.

Cada vez que se presenta un vector de activaciones de la correspondiente capa oculta al mapa, según el algoritmo de aprendizaje de Kohonen, se busca el nodo del mapa que tiene el vector de pesos $\omega_{i,j}$ más parecido en el sentido de distancia euclídea, y se ajusta a él y a los nodos que lo rodean para que se parezcan más (ver el apéndice B). Cuando ya se ha presentado todos los datos de entrenamiento, se podría etiquetar el mapa simplemente contando el número de veces que se activa cada nodo con las entradas de las distintas clases. Cada uno de los contadores asociados a cada nodo almacena el número de veces que se ha activado dicho nodo con cada entrada de la correspondiente clase. La etiqueta asociada será la de aquella clase cuyo contador tiene el valor más alto. El último parámetro, el factor de confianza, mide la certeza que se tiene sobre el valor de la etiqueta que se le ha asociado. Puede darse la existencia de nodos que no se han activado nunca, y por lo tanto no tienen etiqueta asociada.

Siguiendo con el ejemplo de la figura en el que sólo hay una capa oculta, el proceso de cálculo de la señal error asociada a cada capa oculta se puede describir como sigue:

1. Se presenta al perceptrón multicapa el vector de entradas y se generan las activaciones de la capa oculta y la salida.
2. El vector de activaciones de la capa oculta es utilizado como entradas al mapa autoorganizado asociado.
   De entre todos los nodos del mapa que han sido etiquetados con la clase a la que pertenece la entrada actual al sistema, se escoge aquel que posee la menor distancia euclídea entre su vector de pesos $\omega_{i,j}$ y el vector de activaciones de la capa oculta. Este elemento de proceso escogido puede que no sea el que tenga el vector de pesos más parecido al de activaciones si se considera todo el mapa (en términos de distancia euclídea).
3. Se calcula el error en la capa de salida del perceptrón multicapa y por medio del algoritmo de retropropagación del error se determina la señal de error en la capa oculta. Sea $V_{BP-error}$ este valor.
4. El mapa autoorganizado proporciona una señal de error adicional $V_{SOM-error}$, consistente en la diferencia entre el valor del vector de activaciones de la capa oculta y el vector de pesos del nodo ganador del mapa asociado con la misma clase que el dato de entrada. Es posible que no existan nodos con la etiqueta de la clase a la que pertenece la entrada actual del sistema; en este caso la nueva señal de error $V_{SOM-error}$ es cero.

De acuerdo con esto, la señal de error asociada a la capa oculta vendrá dada por:

$$V_{BP-SOM-error} = \begin{cases} ((1-\alpha) \times V_{BP-error}) + (r \times \alpha \times V_{SOM-error}) & \text{Si } r > t \\ V_{BP-error} & \text{Si } r \leq t \end{cases} \tag{21}$$

donde se han incluido tres parámetros:

- α que controla la influencia del elemento SOM sobre el aprendizaje. Cuando $\alpha = 0$ el algoritmo se reduce al de retropropagación del error, desapareciendo el término asociado al mapa autoorganizado.
- r es el factor de confianza que se tiene sobre el elemento ganador en el mapa. Una certeza elevada sobre el elemento ganador hace que la señal de error generada por el mapa tenga más peso sobre la señal de error total.
- t es un umbral de confianza, por debajo del cual no se tiene en cuenta el efecto de la señal de error generada por el mapa en el error total.

Una vez obtenido el valor de la señal error, es posible adaptar los pesos de las conexiones entre las distintas capas del perceptrón multicapa. Cuando se dé por finalizado el proceso de aprendizaje, simplemente se eliminan los correspondientes mapas autoorganizados, puesto que la información que proporciona a su salida es redundante con la que se obtiene en el perceptrón multicapa.

Dado que con este sistema se ha buscado una representación de los vectores de entrada en las correspondientes capas ocultas en la que se mantenga un criterio de proximidad (vectores de la misma clase han de tener representaciones internas en las capas ocultas similares), podría intentarse extraer reglas explícitas a partir de la simple inspección de los vectores de entrada que activan cada uno de los nodos del mapa de Kohonen. En [WBH96] se presenta el caso particular de una tarea de clasificación en la que los vectores de entrada se componen de una serie de atributos discretos.

Un último rasgo a destacar del sistema BP-SOM es que puede ayudar a ajustar el tamaño del perceptrón multicapa [WHBP97]. Supóngase que el tamaño de partida (número de elementos de proceso en la capa oculta) es demasiado grande, cabría la posibilidad de reducir su tamaño sin que se vea afectada la calidad de los resultados.

Considérese un elemento de proceso particular en la capa oculta en una red tipo perceptrón multicapa; si se observa que sobre el conjunto de ejemplos de entrenamiento su activación se mantiene a un nivel aproximadamente constante para cualquier ejemplo de entrenamiento que se presente a la red (o lo que es lo mismo, la desviación estándar de su activación es baja) indicaría que en promedio ese elemento de proceso aportaría una entrada a los elementos de proceso a los que envía su señal que prácticamente se mantiene constante para todos los ejemplos de entrenamiento, y por lo tanto, podría asimilarse a una entrada constante con efecto similar a la entrada de polarización existente en todos los elementos de proceso estándar en las redes tipo perceptrón. En consecuencia, dicho elemento de proceso bajo estudio podría ser eliminado de la red, eso sí, haciéndose imprescindible una nueva etapa de entrenamiento para la nueva estructura de la red.

Este efecto de baja desviación estándar en la activación de los elementos de proceso de las capas ocultas se puede observar sólo si durante el entrenamiento se fuerza a que entradas de la misma clase tengan representaciones en las capas ocultas también semejantes.

7.2 Procesamiento automático del habla

En [SL94] se propone un sistema híbrido perceptrón multicapa y mapa autoorganizado para el reconocimiento automático del habla. Las características del problema son: reconocimiento de palabras aisladas, con un vocabulario restringido e independencia del hablante. Como entrada al sistema se utilizaron 8 espectrogramas de 16 bandas de la señal emitidas por los locutores.

La salida del perceptrón multicapa consta de diez elementos de proceso, uno por cada dígito que ha de reconocer, y se entrena para que genere un valor máximo en el elemento de proceso asociado con la etiqueta correcta. Si se quiere dar una interpretación probabilística a estas salidas, se pueden pasar por una función *softmax*; el dígito pronunciado por el hablante sería el asociado a la salida de mayor valor de activación, o lo que es lo mismo, la que ofrece más probabilidad de reconocer la entrada como la del dígito asociado.

Sin embargo, cuando el sistema es probado con datos nuevos, es posible que haya varias salidas con activaciones semejantes, o lo que es lo mismo, con probabilidades parecidas, siendo imposible ofrecer una identificación del dígito sin ambigüedades. En estos casos entra en funcionamiento el mapa autoorganizado.

La idea subyacente en este modelo es que en la capa oculta del perceptrón multicapa se encuentra una representación simplificada (suponiendo que el número de elementos de proceso en la capa oculta es menor que en la capa de entrada) de la información de entrada al sistema, y que dicha representación es más eficiente para realizar con ella las tareas de reconocimiento.

Así, el módulo autoorganizado recibe como entrada las activaciones de los elementos de proceso de la capa oculta, y es entrenado para etiquetar los casos dudosos a partir de las activaciones de la capa oculta. Como es natural, sólo se emplea cuando el perceptrón multicapa es incapaz de ofrecer una respuesta clara.

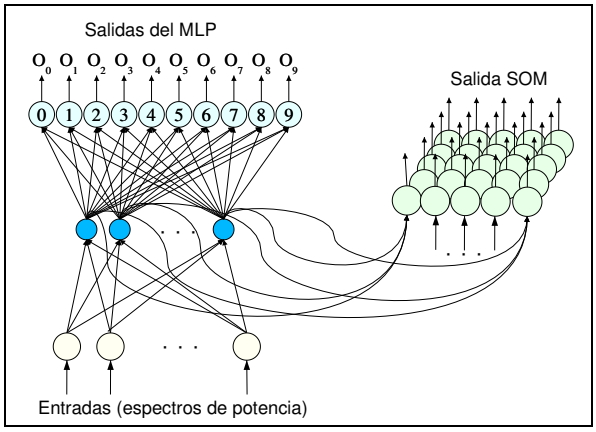


Figura16. Sistema híbrido perceptrón multicapa - mapa autoorganizado para el reconocimiento automático del habla

Dos problemas afines dentro de este mismo área de investigación son:

- **Verificación del Hablante**: consiste en verificar cuando una voz desconocida encaja con el patrón de voz almacenado de la persona que dice ser.
- **Identificación del Hablante**: consiste en identificar o clasificar una voz desconocida de entre un conjunto de voces conocidas.

Muchas de las aproximaciones que se han hecho a estos problemas se basan en la utilización de redes sin realimentación (*"feed-forward"*), y por lo tanto toda la dinámica de la información contenida en la señal hablada se pierde. Para evitar este efecto indeseable, muchas veces esta información que es función del

tiempo es extraída por algoritmos como la diferenciación a lo largo del tiempo, o por un análisis regresivo, todo ello fuera de línea y añadida como entrada al sistema, y así el clasificador utiliza la combinación obtenida como si fuera un patrón estático. Sin embargo, la dependencia a una escala de tiempo más grande no es recogida por estos métodos.

El uso de redes recurrentes ofrece un medio para incorporar esta información temporal que se quedaba perdida o enmascarada con otros métodos, a la vez que conservan la capacidad como herramientas de clasificación propia de las redes neuronales. De este modo, la extracción de estas características temporales ya no es necesario realizarla en la etapa de preprocesamiento, ya que ahora es llevada a cabo de forma natural por la propia red recurrente.

En concreto, en [Mak95] se usó una aproximación muy sencilla al problema de la identificación del hablante consistente en la creación de un módulo experto en la identificación de señales producidas por un hablante específico. Así, por ejemplo, cuando el sistema consiste en la identificación de uno entre N posibles hablantes, se tendrán N módulos, cada uno de ellos encargado de identificar la señal de uno solo de los N hablantes. Un mecanismo de toma de decisiones determina cuál es la identidad del hablante cuya señal se encuentra en la entrada del sistema (figura 17(a)).

Estos módulos de identificación se construyeron con redes neuronales RTRL (*"Real Time Recurrent Learning"*), cuya arquitectura genérica se muestra en la figura 17(b). Como allí se puede ver, consiste en una red con una capa de entrada y una capa de elementos de proceso totalmente interconectados entre sí. Cada salida $y_Q(t+1)$ vendrá determinada (a través de la función de activación f) por la suma ponderada de las entradas ($x_i(t)$ y de las salidas generadas por los elementos de proceso de la capa $y_j(t)$. El mecanismo de aprendizaje propuesto consiste en aplicar el algoritmo del gradiente decreciente.

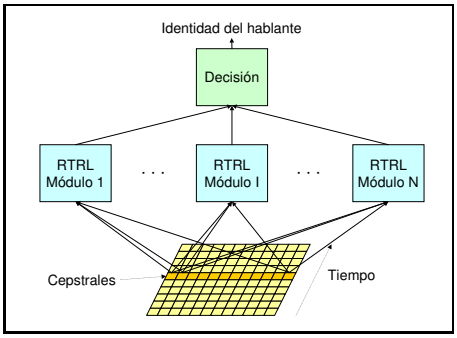
Gracias al empleo de este mecanismo de modularización se evita la *interferencia* en el proceso de aprendizaje a la hora de distinguir entre la señal de los distintos hablantes que utilizan el sistema.

7.3 Diagnóstico de enfermedades

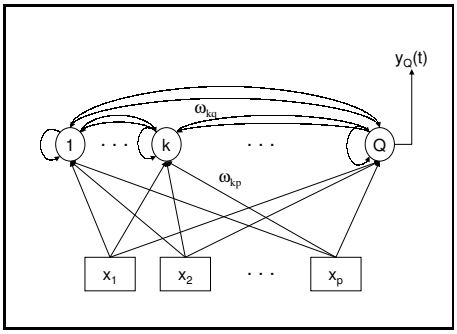
Un ejemplo de diagnóstico utilizando un sistema modular constituido por varias de redes neuronales en cascada se puede encontrar en [SHARAL$^+$99]. El autor propone con este sistema clasificar los defectos incipientes en el campo visual causados por diversas enfermedades.

Los datos utilizados están formados por los defectos de sensibilidad (medidos en decibelios) medidos en 59 puntos del campo visual del ojo humano, así como su desviación media (media de los defectos de sensibilidad presentes en cada punto) y la varianza de la pérdida (indica la variación de la sensibilidad en toda la extensión del campo visual).

En un primer intento de efectuar el diagnóstico partiendo de las medidas del campo visual se utilizó un único mapa autoorganizado, que una vez entrenado fue etiquetado de tres maneras diferentes:



(a) Sistema modular de identificación del hablante usado en [Mak95]



(b) Arquitectura de una red RTRL.

Figura17. Identificación del hablante.

- Clasificación en dos categorías: glaucoma y no glaucoma.
- Clasificación en tres categorías: normal, glaucoma y otras patologías.
- Clasificación en cinco categorías: normal, glaucoma, diabetes, cataratas y HTA (Retinopatía Hipertensiva).

Para evaluar la bondad de las clasificaciones se utilizaron dos parámetros: la *sensibilidad* y la *especificidad*. La sensibilidad indica la proporción de individuos de una determinada categoría detectados como tales, y la especificidad indica la proporción de individuos que no pertenecen a esa categoría y que no han sido clasificados dentro de ella.

Los resultados obtenidos en este primer intento usando un único mapa mostraron que, cuando se utilizan cinco categorías los valores de sensibilidad y especificidad para las categorías de diabetes, cataratas y HTA eran muy bajos como para considerarse fiables.

Para resolver este problema se desarrolló un *algoritmo jerárquico* de tres etapas con mapas autoorganizados entrenados de forma independiente. En la figura 18 se puede ver un esquema del procedimiento. El vector de entradas es presentado al primer mapa responsable de la *etapa 1*, que lo clasifica en normal o patológico. Si el vector es considerado como patológico por este primer módulo, es pasado al segundo mapa, que discrimina entre los que pertenecen a la categoría de glaucomatoso o no. Si según el módulo de la *etapa 2* el vector de datos es no glaucomatoso, este vector es enviado a la *etapa 3* que lo clasifica entre cataratas, HTA y diabetes.

La evaluación de los resultados de este nuevo algoritmo jerárquico indican una gran mejoría respecto al uso de un único mapa, encontrándose un aumento de la sensibilidad desde aproximadamente 20% en las categorías de diabetes, cataratas y HTA, a cerca del 80% en las categorías de cataratas y HTA.

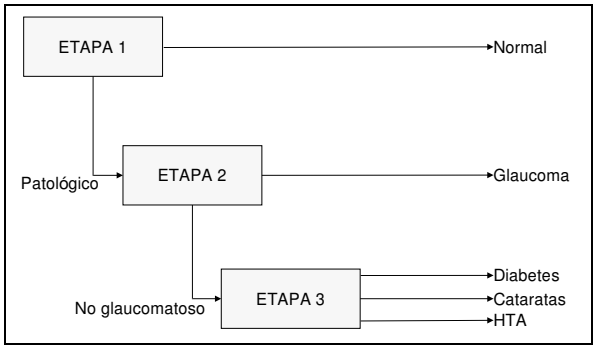


Figura18. Tres mapas autoorganizados en cascada para el diagnóstico del glaucoma.

Otra referencia de uso de sistemas jerárquicos de redes neuronales artificiales aplicados al diagnóstico puede encontrarse en [BSCH94], donde una jerarquía de

redes neuronales tipo perceptrón multicapa trabajan en cascada para clasificar células en distintas categorías que abarcan desde las no enfermas a las malignas. El objetivo final es evitar en lo posible la aparición de *falsos negativos*, es decir, que células malignas sean clasificadas como células no enfermas. Además de este método de clasificación mediante una cascada de clasificadores que filtran los resultados que se van obteniendo, se propone la utilización de una red tipo ART2 para discriminar de entre las 57 características morfológicas utilizadas como informaciones de entrada al sistema cuáles son aquellas que más influyen en el diagnóstico.

8 Resumen

Los sistemas modulares aplican la conocida táctica de "divide y vencerás", que implica un cierto nivel de cooperación. Este último aspecto, el de cooperación, ya fue utilizado por Laplace en el siglo XIX al combinar los resultados de varios predictores.

El uso de técnicas de aprendizaje y funcionamiento híbridas puede ayudar a minimizar ciertos efectos no deseables inherentes a algunas técnicas de resolución de problemas. Este es el caso del sobreentrenamiento y las redes neuronales tipo perceptrón multicapa entrenadas con retropropagación del error.

Gracias a la a combinación de distintos paradigmas de aprendizaje se abren nuevos caminos a la investigación y a la resolución de problemas que hasta el momento se habían considerado difíciles.

Cuando la estructura de los módulos con la que se está trabajando es lo suficientemente regular, como en el caso de los árboles constituidos por expertos todos del mismo tipo, se pueden diseñar estrategias de aprendizaje aplicables de manera sistemática, como los algoritmos estocásticos o el de maximización del valor esperado. En casos más heterogéneos, el diseño del procedimiento de ajuste de los parámetros debe hacerse de manera específica para cada situación.

Un punto a considerar es que siempre que se disponga de información apriorística acerca de la descomposición de la tarea en subtareas más elementales, debería utilizarse, gracias a lo cual se podría alcanzar configuraciones de sistemas más eficientes.

El ajuste de sistemas modulares (número, relación y tipo de los módulos) es una tarea compleja, donde pueden aplicarse técnicas tan elementales como el simple ensayo y error, o algo más sistemáticas como el Método de Monte Carlo o los Algoritmos Genéticos.

Por último, señalar que la extracción de reglas es un objetivo muy interesante pero poco factible. En la actualidad existen métodos para la extracción de reglas de un sistema ya ajustado, pero resultan ser unas técnicas muy poco generalizables, ya que dependen fuertemente de la naturaleza del problema bajo estudio y del o de los sistemas aplicados.

A Algoritmo de Retropropagación del Error

El algoritmo de aprendizaje de **retropropagación del error** (también conocido como la *regla delta generalizada*) es una aplicación del bien conocido método del **gradiente decreciente**. Esta regla de aprendizaje se debe a Werbos (1974), Rumelhart (1985) y Parker (1989) entre otros autores.

Debido a ciertas consideraciones probabilísticas y, ante todo, a su facilidad de cálculo, se establece como función objetivo a minimizar el error cuadrático:

$$E(t) = \frac{1}{2} \sum_{i \in Capa\ de\ Salida} (y_i'(t) - y_i(t))^2 \tag{22}$$

donde:

- $y_i(t)$ es la salida del elemento de proceso i en el instante t.
- $y_i'(t)$ es la salida deseada para el elemento de proceso i en el instante t.

De acuerdo con esta regla de aprendizaje, el peso w_{ij} que conecta el elemento de proceso j con el i se ajusta por

$$\Delta w_{ij}(t) = \alpha \delta_i(t) y_j(t) \tag{23}$$

donde:

- α es el denominado *coeficiente de aprendizaje* .
- $\delta_i(t)$ es la señal de error del elemento de proceso i en el instante t.

Si el elemento de proceso i es de la capa de salida, se tiene que

$$\delta_i(t) = (y_i'(t) - y_i(t))\phi_i'(t) \tag{24}$$

donde $\phi_i'(t)$ es la derivada de la función de activación o transferencia en el instante t para el elemento de proceso i.

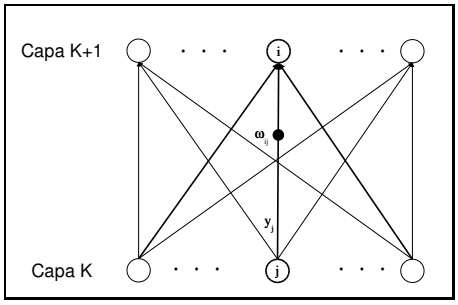
Si el elemento de proceso i no se encuentra en la capa de salida, sino que está en las interiores, la señal de error tiene el siguiente valor:

$$\delta_i(t) = \phi_i'(t) \sum_{h=1}^{N} \delta_h(t) w_{hi} \tag{25}$$

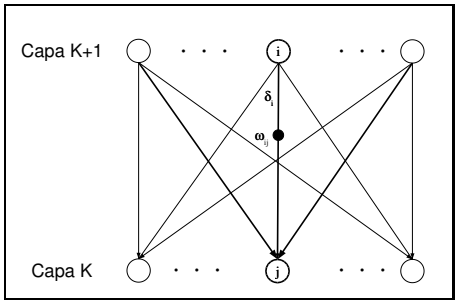
donde N es el número de elementos de proceso en la capa destino.

En la práctica para acelerar la convergencia se suele introducir un término **momento**:

$$\Delta w_{ij}(t) = \alpha \delta_i(t) y_j(t) + m \Delta w_{ij}(t-1) \tag{26}$$



(a) Generación de la salida.



(b) Retropropagación del error.

Figura19. Representación del flujo de información en las dos fases del algoritmo de retropropagación del error. En la primera fase durante la generación de la salida de la red la información fluye hacia adelante, desde las capas de entrada y ocultas hacia la de salida. En la segunda fase, la de retropropagación del error, la información (la señal de error) fluye en sentido inverso, desde la capa de salida hacia las capas interiores.

donde m es el *término momento* (momentum). Si se expande el término $\Delta w_{ij}(t)$ en función de todos los valores anteriores, se tiene:

$$\begin{aligned}\Delta w_{ij}(t) =& \alpha\delta_i(t)y_j(t) + m\Delta w_{ij}(t-1) \\ =& \alpha\delta_i(t)y_j(t) + m\alpha\delta_i(t-1)y_j(t-1) + m^2\Delta w_{ij}(t-2) \\ & \cdots \\ =& \alpha\sum_{l=0}^{t} m^l\delta_i(t-l)\end{aligned} \tag{27}$$

Esta última ecuación representa una serie temporal que converge sólo si el parámetro m cumple que $0 \leq |m| < 1$.

Una variación muy interesante del algoritmo de retropropagación del error es el que incluye un **término de regularización**, cuya misión es introducir la condición adicional de minimizar los valores absolutos de los pesos. El añadido de este término conduce a la variante conocida como *"Weight Decay"*, con el que se logra minimizar el efecto del *sobreentrenamiento*. .

En este caso, la expresión para la función de error queda:

$$E(t) = \frac{1}{2}\left(\sum_{i \in Capa\ de\ Salida}(y_i(t) - y_i'(t))^2 + \gamma \sum_{Todos\ los\ pesos} w_{ij}^2\right) \tag{28}$$

Donde γ es la constante de regularización. De este modo, la expresión para la actualización de los pesos queda así:

$$\Delta w_{ij}(t) = \alpha\delta_i(t)y_j(t) + m\Delta w_{ij}(t-1) + \gamma w_{ij}(t) \tag{29}$$

B Algoritmo de Aprendizaje para SOM

En los mapas autoorganizados de Kohonen [Koh89,KKL92] el mapa "aprende" una correspondencia topológica continua $f : B \subset \mathbb{R}^n \rightarrow C \subset \mathbb{R}^m$ por medio de ejemplos $x \in B$ ó $y \in C$, donde B es un subconjunto rectangular y C un subconjunto acotado sobre el cual se define una densidad de probabilidad $P(y)$.

Cada elemento de proceso i o nodo que forma el mapa está definido en el instante t por un conjunto de pesos $\boldsymbol{w}_i(t)$. El algoritmo de aprendizaje o ajuste de los pesos es como sigue:

Para cada entrada $\boldsymbol{x}_i(t)$ en el instante t

1. determinar el elemento de proceso *ganador*, en el sentido de mínima distancia:

$$g = \min_i(d(\boldsymbol{w}_i(t), \boldsymbol{x}_i(t))) \tag{30}$$

2. adaptar los pesos del elemento de proceso ganador y sus vecinos de acuerdo con la fórmula:

$$\boldsymbol{w}_j(t+1) = \boldsymbol{w}_j(t) + h(i,g,t)(\boldsymbol{x}(t) - \boldsymbol{w}(t)) \qquad \forall i \in S \qquad (31)$$

donde $h(i,g,t)$ depende de la posición del elemento de proceso j cuyos pesos se van a actualizar, del elemento de proceso ganador g y del tiempo t, tal que normalmente el valor de $h()$ decrece con el tiempo.

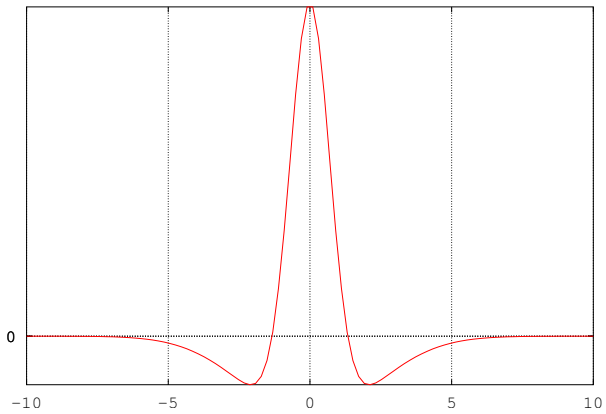


Figura20. La función "Sombrero Mejicano" para la interacción lateral en un mapa autoorganizado.

Debido a este algoritmo de aprendizaje, las señales de entrada que estén cercanas se harán corresponder a elementos de proceso que también están cercanos, permitiéndose así conservar la topología.

La forma de la función h es como la mostrada en la figura 20, que proporciona un ajuste positivo (refuerzo) en las proximidades del nodo ganador, que va decreciendo hasta volverse negativo (inhibidor) según se va alejando, para volver a crecer hasta hacerse cero en los nodos más lejanos.

C Red TDNN

Una nueva posibilidad es incluir elementos de retardo explícito en la red, tal y como ocurre en las redes **Time-Delay Neural Networks (TDNN)** [LWH89]. Estos elementos de retardo hacen las veces de memoria, y el efecto es parecido al de generar una ventana sobre los datos, pero ahora esa ventana se puede extender no sólo a las entradas, sino también a las salidas de las capas ocultas, lo cual proporciona una noción más cercana al *estado de la red neuronal* como un reflejo de lo ocurrido en instantes anteriores.

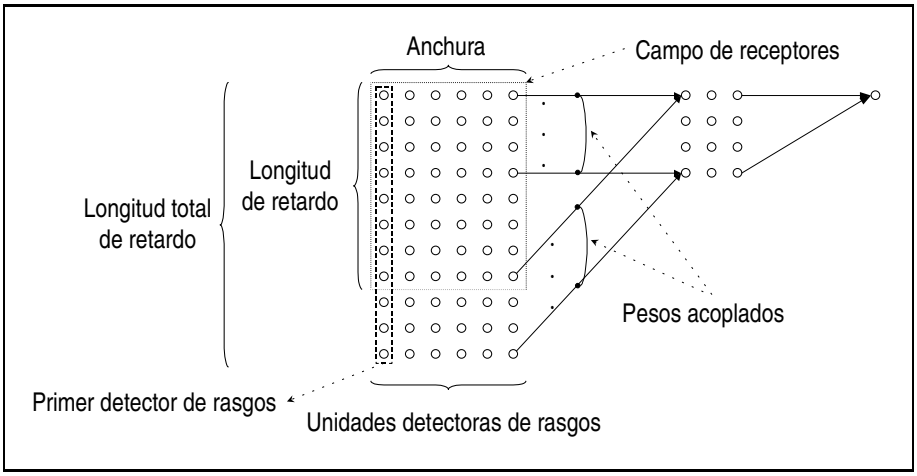


Figura21. Esquema de una red TDNN.

En la figura 21 se ve la estructura de una red TDNN en la que hay seis entradas. Cada entrada a la red se supone asociada a un *rasgo* que caracteriza a la serie temporal que quiere procesarse. El retardo permite almacenar los valores antiguos de activaciones y conexiones de los correspondientes elementos de proceso. Este efecto se logra copiando los valores de activación y pesos de las conexiones con la siguiente capa justo antes de actualizar sus valores en el proceso de aprendizaje.

Los *campos receptivos* están formados por elementos de proceso que se encuentran totalmente conectados con un subconjunto de las entradas (rasgos) y sus retardos. Los campos de receptores quedan definidos por una *anchura* y una *longitud de retardo*.

Durante el entrenamiento se calculan los valores de los pesos de las conexiones individuales para cada campo receptivo. Los valores que adoptarán dichas conexiones serán la media de las actualizaciones individuales, tratándose en consecuencia todos los campos de recepción como uno único.

Referencias

[BKHSK93] Egbert J.W. Boérs, Herman Kuiper, Bart L.M. Happel, and Ida G. Sprinkhuizen-Kuyper. Designing modular artificial neural networks. Technical report, Departement of Computer Science. Leiden University, 1993.

[BSCH94] Mehdi Bazoon, Deborah A. Stacey, Chen Cui, and George Harauz. A hierarchical artificial neural networks system for the classification of cervical cells. In *Proceedings of the International Congress on Computational Inteligence ICNN'94*, Julio 1994.

[FGP95] M. Figueiredo, F. Gomide, and W. Pedrycz. Fuzzy neurons and networks: Models and learning. *ECLA005*, 1995.

[GLT97] C. Lee Giles, Steve Lawrence, and Ah Chung Tsoi. Rule inference for financial prediction using recurrent neural networks. In IEEE, editor, *Proceedings of IEEE/IAFE Conference on Computational Inteligence for Financial Engineering*. IEEE, 1997.

[Gol89] David E. Goldberg. *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley Publishing Company, 1989.

[Has94] Sherif Hashem. Optimal linear combinations of neural networks. *Neural Networks*, 1994.

[Hay94] Simon Haykin. *Neural Networks. A Comprehensive Foundation*. Prentice Hall, 1994.

[JJ94] Michael I. Jordan and Robert A. Jacobs. Hierarchical mixtures of experts and the EM algorithm. *Neural Computations*, 5:181–214, 1994.

[KKL92] Teuvo Kohonen, Jari Kangas, and Jorma Laaksomen. *SOM_PAK. The Self-Organizing Map Program Package V1.2*. SOM Programming Team of the Helsinki University, Rakentajamaukio 2 C, SF- 02150 Espoo, Finland, 1992.

[Koh89] Teuvo Kohonen. *Self-Organization and Associative Memory*. Springer-Verlag, 1989.

[Koh95] Ron Kohavi. Wrappers for performance enhancement and oblivious decision graphs, 1995.

[Koz93] John R. Koza. *Genetic Programming*. MIT Press, 1993.

[LWH89] Kevin J. Lang, Alex H. Waibel, and Geoffrey E. Hinton. A time-delay neural network architecture for isolated word recognition. *Neural Networks*, 3:23–43, 1989.

[LYC97] Steve Lawrence, Peter Yianilos, and Ingemar Cox. Face recognition using mixture-distance and raw images. *International Conference on Systems, Man, and Cybernetics*, pages 2016–2021, 1997.

[Mac99] David J.C. Mackay. *Information Theory, Inference, and Learning Algorithms*. `http://wol.ra.phy.cam.ac.uk/mackay`, 1999.

[Mak95] Man-Way Mak. Speaker identification using modular recurrent neural networks. In *Proceedings 4th. IEEE International Conference on Artificial Neural Networks*, pages 1–6, Junio 1995.

[Moe97] Perry Moerland. Some methods for training mixtures of experts. Technical report, Dalle Molle Institute for Perceptive Artificial Intelligence, 1997.

[MSJ97] R. Murray-Smith and T. A. Johansen, editors. *Multiple Model Approaches to Modelling and Control*. Taylor and Francis, 1997.

[MT94] Ryszard Michalski and Gheorghe Tecuci, editors. *Machine Learning. A Multistrategy Approach*. Morgan Kaufmann Publishers, 1994.

[OS93] David W. Opitz and Jude W. Shavlik. Heuristically expanding knowledge-based neural networks. In *Proceedings of the Thirteenth International Joint Conference on Artificial Inteligence (IJCAI'93)*, 1993.

[OS94] David W. Opitz and Jude W. Shavlik. Genetically refining topologies of knowledge-based neural networks. In *Proceedings of the International Symposium on Integrating Khowledge and Neural Heuristics (ISIKNH'94)*, 1994.

[OS97] David W. Opitz and Jude W. Savlik. Connectionist theory refinement: Genetically searching the space of network topologies. *Journal of Artificial Inteligence Research*, 6, 1997.

[RG95] Eric Ronco and Peter Gawthrop. Modular neural networks: a state of the art. Technical report, Centre for System and Control. University of Glasgow, Glasgow, UK, 1995.

[Rij95] Alphen ann den Rijn. Structure optimisation of modular neural network. Master's thesis, Departement of Computer Science, Leiden University, 1995.

[SHARAL+99] María Aránzadu Simón Hurtado, Luis Alonso Romero, Alfonso Antón López, Daniel Bahillo Pérez, and Eduardo S. de la Fuente Gallego. Clasifiación de campos visuales mediante redes neuronales. Technical report, Proyecto AIRENE. CYTED subprograma VII Electrónica e Informática Aplicada. Universidad de Valladolid - España, Mayo 1999.

[SL94] Angel Luis Sánchez Lázaro. *Redes Neuronales Artificiales Aplicadas al Reconocimiento de Palabras Independiente de Locutor*. PhD thesis, Departamento de Informática. Universidad de Valladolid (España), 1994.

[Sva94] Claus Svarer. *Neural Networks for Signal Processing*. PhD thesis, CONNECT, Electronics Institute, Technical University of Denmark. DK-2800 Lyngby, Denmark, 1994.

[Tve95] Donald R. Tveter. The pattern recognition basis of AI - chapter 11, 1995.

[WBH96] Ton Weijters, Antal van den Bosch, and H. Jaap van den Herik. Beahvioural aspects of BP-SOM. Technical report, MATRIKS / Department of Computer Science. University of Maastricht, 1996.

[WHBP97] Ton Weijters, H. Jaap van den Herik, Antal van den Bosch, and Eric Postma. Avoiding overfitting with BP-SOM. In *15^{th} International Joint Conference on Artificial Inteligence*, pages 1140 – 1145. Norman Kaufmann, 1997.