CAPÍTULO 2 HERRAMIENTAS DE EXTRACCIÓN DE INFORMACIÓN: *REDES NEURONALES*

2.1. Introducción.

El procesamiento de información de carácter redundante, imprecisa y distorsionada posee un papel primordial para la resolución de problemas reales de clasificación o de predicción en muchas áreas científicas.

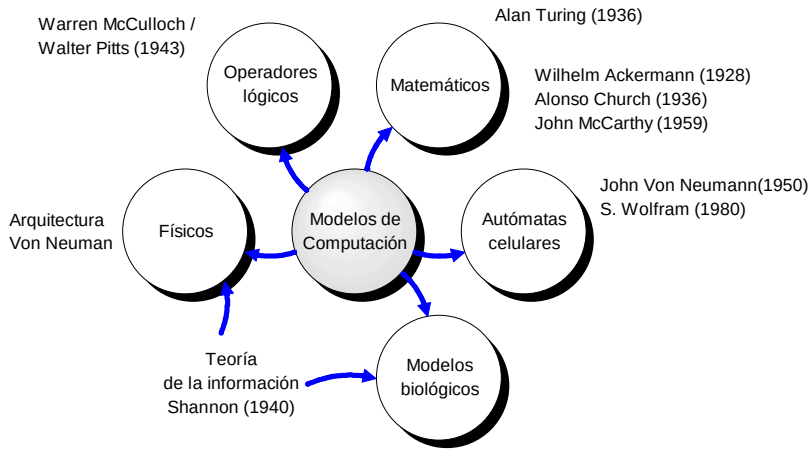


Ilustración 1.2.1. Modelos de *computación*

Fuente: Rojas, R. (1996). **Neural Networks. A Systematic Introduction** , pp. 7, Springer y elaboración propia

Una de las metodologías con un mayor auge en la última década son los modelos de redes neuronales (*Neural Networks*)[1], que en esencia son estructuras formales de carácter matemático y estadístico con la propiedad del *aprendizaje*[2], es decir, la adquisición de conocimientos que en la mayoría de los casos es a partir de ejemplos.

[1] Una posible definición de red neuronal es, por ejemplo, *una nueva forma de computación, inspirada en modelos biológicos* (véase Hilera, J.R. y Martínez, V.J. (1995). **Redes neuronales artificiales. Fundamentos, modelos y aplicaciones**, pp. 9, Ra-Ma, Madrid).

[2] Véase un mayor desarrollo de las propiedades del aprendizaje mediante sistemas computerizados, Weis, S.M.; Kulikowski, C.A. (1991). **Computer Systems That Learn**, Morgan Kaufmann Publishers.

Este aprendizaje se produce mediante un estilo de computación[3] denominado en *paralelo* que intenta simular algunas de las capacidades que posee nuestro cerebro, (véase ilustración 1.2.1.). Por esta razón se las definen como redes neuronales artificiales para distinguirlas de los modelos biológicos.

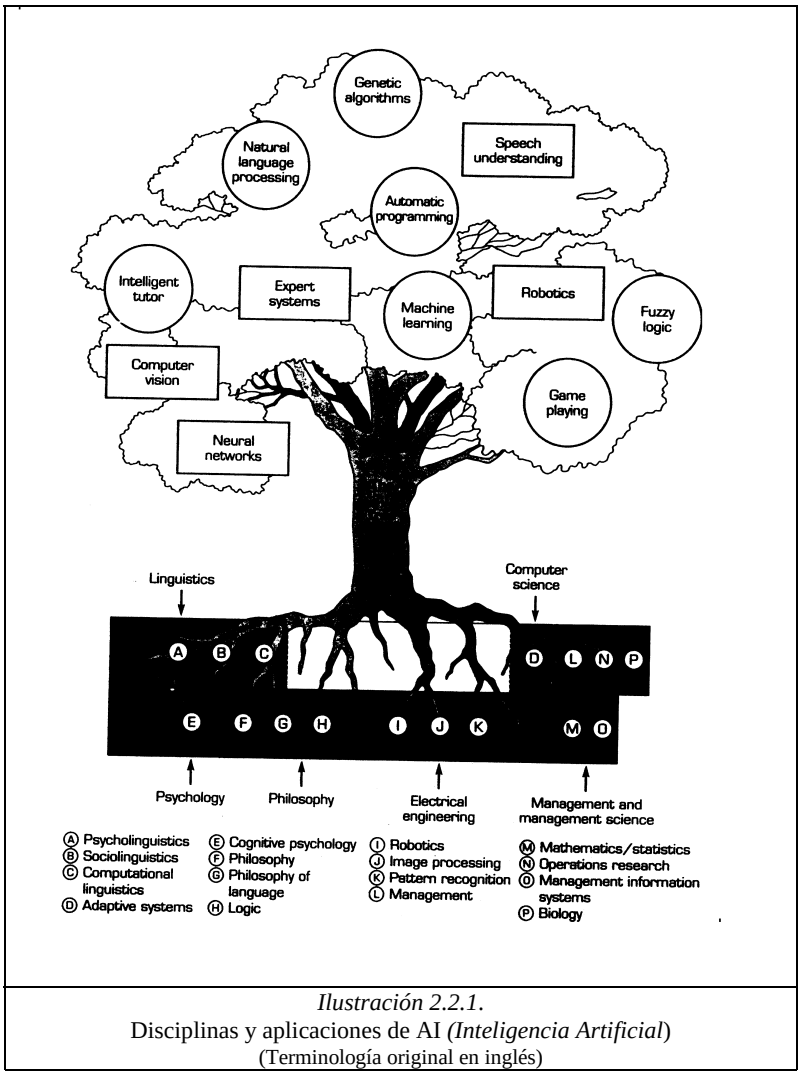


Ilustración 2.2.1.
Disciplinas y aplicaciones de AI *(Inteligencia Artificial*)
(Terminología original en inglés)

Los tres elementos clave de los sistemas biológicos que pretenden emular los artificiales[4] son, el *procesamiento en paralelo,* la *memoria distribuida* y la *adaptabilidad.*

[3] Los elementos de un modelo de computación son, *almacenamiento, transmisión* y *procesamiento.*
[4] Véase Martín del Brío, B. Sanz Molina, A. (1997). **Redes Neuronales y sistemas**, pp. 10-12, Ra-Ma, Madrid.

En referencia al primero de los elementos clave, es importante remarcar que aún siendo las neuronas biológicas más simples, lentas y menos fiables que las artificiales, el cerebro resuelve problemas complejos imposibles para sistemas simulados, a través de su trabajo en paralelo. En segundo lugar, la memoria distribuida permite a los modelos biológicos ser tolerantes a los fallos, debido a que muchas neuronas pueden realizar tareas similares produciéndose intercambios de funciones. Y por último, la adaptabilidad garantiza el proceso de aprendizaje.

Pero en rigor, debemos notar que existen otras metodologías que permiten atajar los mismos problemas[5], como por ejemplo, los sistemas borrosos (*Fuzzy systems*), algoritmos genéticos (*Genetic algorithms*), sistemas expertos (*Expert systems*), etc. Todas estas metodologías conforman un tronco común definido como Inteligencia Artificial[6] (*Artificial Intelligence*) de clara inspiración biológica, cuya finalidad es replicar parcialmente la naturaleza de la inteligencia humana[7] apostando en muchos casos por estrategias mixtas que potencien aún más su desarrollo[8], (véase ilustración 2.2.1.).

El concepto filosófico de inteligencia artificial se debe a un gran matemático, Alan Turing[9]. Este lógico y matemático sometió a debate, desde la óptica filosófica, si las máquinas podían o no pensar[10].

---

[5] Véase aplicaciones para los mercados financieros de: herramientas neuronales, algoritmos genéticos y sistemas borrosos en Deboeck, G.J. (1994). **Trading on the Edge, Neural, Genetic, and Fuzzy Systems for Chaotic Financial Markets**, Wiley.

[6] Interesante el esfuerzo por vincular la disciplina estadística con la Inteligencia Artificial, véase Hand, D.J. (1993) **Artificial Intelligence Frontiers in Statistics**, (AI and Statistics III), Chapman & Hall.

[7] Una ejemplo en el entorno de la toma de decisiones de la aplicación de sistemas inteligentes es, Hill, T. y Remus, W. (1994). **Neural network models for intelligent support of managerial decision making**. *Decision Support Systems*, 11, pp. 449-459 y de su crecimiento, Schocken, S.; Ariav, G. (1994). **Neural netwoks for decision support: Problems and opportunities**, *Decision Support Systems*, 11, pp. 393-414.

[8] Las aplicaciones comerciales sobre modelos neuronales acostumbran a presentarse conjuntamente con otras metodologías, como por ejemplo, CBR (*Case-Based Reasoning*) que consiste en descubrir analogías entre los datos a partir de grupos homogéneos, véase Kolodner, J. (1993). **Case-Based Reasoning**, Morgan Kaufmann Publishers y Yoon, Y.; Guimaraes, T. y Swales, G. (1994). **Integrating artificial neural networks with rule-based expert systems**, *Decision Support Systems*, 11, pp. 497-507.

[9] Durante la II guerra mundial trabajó como descifrador de códigos, ideando una máquina que podía decodificar rápidamente un código secreto alemán llamado *Enigma.* Turing fue un adelantado a su época poniendo las bases de los fundamentos lógicos del diseño de computadores.

[10] Para mayor detalle véase, Copeland, J. (versión Española de Armero San José, JC)(1996). **Inteligencia Artificial. Una introducción filosófica**, Alianza Editorial, Madrid.

Ideó un experimento de laboratorio para demostrar tal afirmación, que denominó "*prueba de Turing*". Tal experimento consistió en considerar que una máquina posee inteligencia si puede ser confundida por un ser humano. Ahora bien, el campo de la inteligencia artificial (AI) propiamente dicho recibió su nombre de John McCarthy[11]. Fue el primer investigador que logró reunir a principios de los años 60 a todas aquellas personas interesadas en la inteligencia de los computadores. McCarthy realizó aportaciones de gran importancia, como por ejemplo, la invención del LISP (lenguaje de programación) y la creación del primer sistema de tiempo compartido.

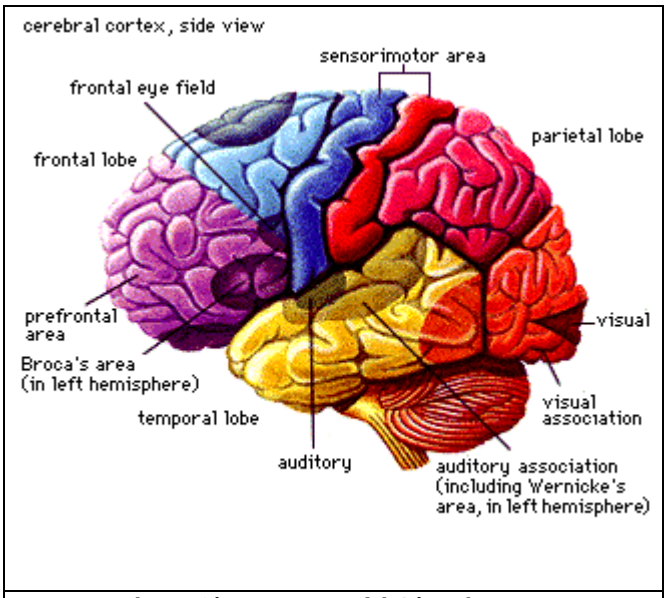


Ilustración 3.2.1 Zonas del Córtex humano
Fuente: http://freud.tau.ac.il/~shakhar/neuro/images.html
(Terminología original en inglés)

El cerebro humano es una red biológica muy compleja con cientos de miliardos (10^{12}) de células[12] de varios tipos. La neurona biológica, como caso particular, posee la propiedad conductiva de señales eléctricas o químicas, que permiten formar una red, en la que la interconexión es muy alta. El número aproximado de neuronas del cerebro humano es de cien mil millones, las cuales están interconectadas entre sí, generando un volumen de conexiones en promedio del orden de diez mil para cada una de ellas, lo que lo convierte en un sistema con una complejidad muy superior a cualquier producto humano. El proceso de comunicación entre neuronas se produce gracias a los neurotransmisores liberados por la neurona emisora y

[11] Véase http://www-formal.stanford.edu/jmc/ para una mayor descripción de la obra del científico.
[12] Las *células nerviosas* son sistemas complejos con propiedades de autoorganización, que realizan el almacenamiento de la información en el proceso de sinapsis.

captados por la neurona receptora. Este último aspecto permite afirmar que gracias a este número tan elevado de conexiones (del orden de 10^{14}), puede considerarse como una gran red neuronal con fuerte especialización funcional entre las diferentes partes del *cortex*, (véase ilustración 3.2.1.).

A continuación se detallan algunos de los principios computacionales[13] más importantes muy relacionados con detalles de la organización del cerebro, que posteriormente se utilizaron en el diseño de los modelos artificiales, (véase tabla 1.2.1.).

Tabla 1.2.1.

1	*Paralelismo masivo*, las operaciones de cálculo se realizan al mismo tiempo en diferentes regiones del *córtex* y en cada región funcional del mismo, el cálculo lo realizan simultáneamente todas las neuronas.
2	Alto grado de *complejidad en las conexiones* definidas entre las neuronas.
3	*Habilidad para el aprendizaje*, es decir, existe una adaptación constante de las neuronas a partir de la experiencia acumulada.
4	*Estados binarios y variables continuas*. Las neuronas poseen dos estados, activado o desactivado y la información continua está codificada por la frecuencia de pulsos. (Pulsos por unidad de tiempo)
5	*Numerosos* tipos de neuronas y de señales.
6	Interacción entre neuronas de *carácter no lineal*.
7	El cerebro se organiza *descomponiendo* las conexiones en subredes.
8	Asociado al anterior aspecto cada subred tiene una *función* específica.

El diseño de las redes neuronales artificiales (véase ilustración 4.2.1.), también llamadas sistemas de procesamiento en paralelo (*Parallel distributed processing systems* (PDPs)) o sistemas conexionistas (*Connectionist systems*), incorporan características biológicas generando un claro paralelismo entre estos y el modelo neuronal biológico (véase ilustración 5.2.1.). Así conceptos, como por ejemplo, *dentritas*, *axón*, *sinapsis* y cuerpo o *soma* que son de uso corriente en el entorno biológico, toman un papel relevante en los modelos matemáticos que replican la forma de transmisión de la señal eléctrica que por ellas circula. En 1901 *Santiago Ramón y Cajal* postuló que la dirección de la transmisión de la información es determinada por la naturaleza en forma de red de las células nerviosas, a partir de los conocimientos sobre la naturaleza eléctrica de los impulsos nerviosos y de la velocidad de su transmisión.

[13] Véase el paralelismo entre computación, aprendizaje y topologías neuronales en, Massón, E.; Wang, Y-J. (1990). **Introduction to computation and learning in artificial neural networks**, *European Journal of Operational Research*, 47, pp. 1-28.

Tal aspecto fue el caballo de batalla para los primeros intentos de formalizar su comportamiento artificial, nos referimos al primer modelo artificial, el modelo de *McCulloch-Pitts*[14].

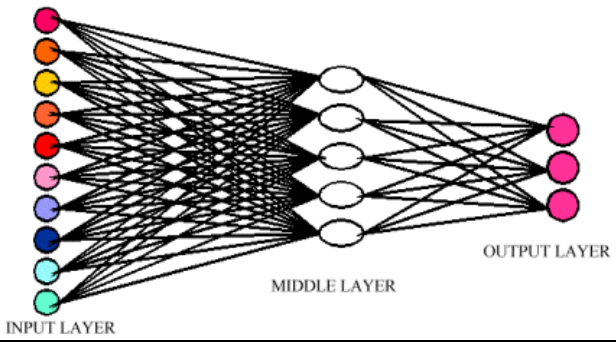


Ilustración 4.2.1. Modelo neuronal artificial
Fuente: http://www.culture.com.au/brain_proj/

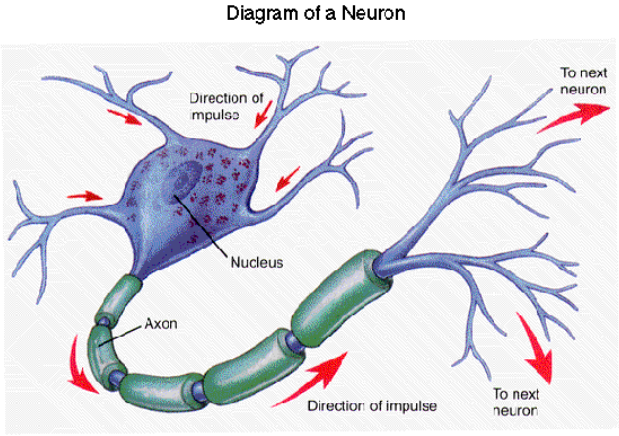


Ilustración 5.2.1. Neurona biológica
*Fuente:*http://psych.wisc.edu/faculty/pages/croberts/topic3.html

La ilustración 6.2.1. refleja el recorrido histórico que ha sufrido la metodología neuronal desde sus orígenes hasta la actualidad de forma sintética. Resaltamos el intenso incremento en las aplicaciones de carácter general a partir de los años 90, que más adelante serán detalladas[15], (véase apartado 2.4.).

[14] McCulloch, W; Pitts, W. (1943). **A logical calculus of the ideas immenent in nervous activity**. *Bull. Math.Biophys.* 5:115-133. pp. 96-104. Referenciado en Hastie, T; Tibshirani, R.; Friedman, J. (2001). **The Elements of Statistical Learning. Data Mining, Inference, and Prediction**, Springer.

[15] Véase referencias del desarrollo de las redes neuronales orientado a los negocios, véase, Smith, K.A.; Gupta, J.N.D. (2000). **Neural networks in business: techniques and applications for the operations researcher**, *Computers &Operations Research*, 27, pp. 1023-1044.

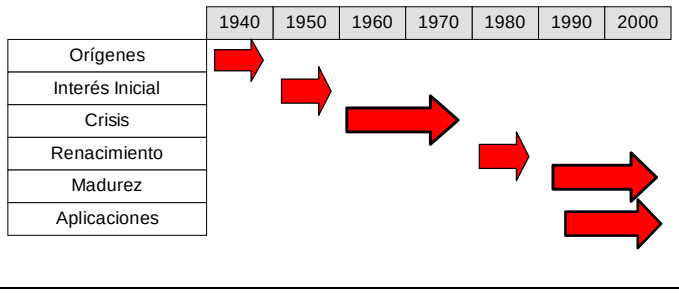


Ilustración 6.2.1.
Fases de la *historia* de las redes neuronales.

Como consecuencia del incremento espectacular del número de referencias (véase la ilustración 7.2.1.) y de la presencia de un gran número de instituciones internacionales que fomentan la investigación en los campos de la inteligencia artificial[16], consideramos que el desarrollo futuro está asegurado. En particular, en España, resaltamos los esfuerzos de asociaciones como por ejemplo, la Asociación Española para la Inteligencia Artificial[17] (AEPIA) y la Asociación catalana[18] entre otras instituciones.

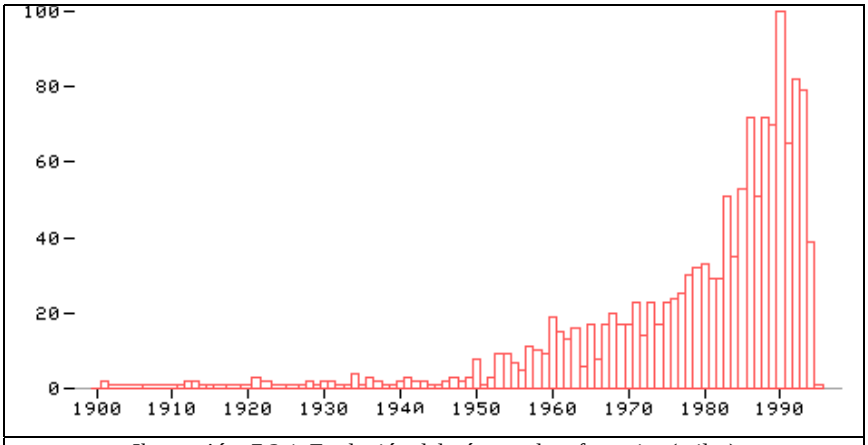


Ilustración 7.2.1. Evolución del número de referencias (miles)
Fuente: http://www.cs.berkeley.edu/~russell/aima-bib.html

[16] En la actualidad existe un gran número de asociaciones internacionales que fomentan el avance de metodologías neuronales en todos los campos de la ciencia, por ejemplo, *European Neural Network Society* (ENNS), http://www.ida.his.se/ida/enns/, *International Neural Network Society* (INNS), http://www.inns.org/, *Japanese Neural Network Society* (JNNS), *Asia Pacific Neural Network Assembly* (APNNA), *European Coordinating Committee for Artificial Intelligence* (ECCAI), http://www.eccai.org. Cabe destacar en Europa con el soporte de las acciones ESPRIT, el proyecto NEUFODI (Neural Networks in Diagnosis and Forecasting Applications) así como la creación de una base de datos de todas las aportaciones, DEANNA (véase Taylor, J.G. (1993). **The promise of neural networks**, pp. 99-121, Springer).

[17] Véase http://aepia.dsic.upv.es/

[18] Véase http://www.acia.org/

En la actualidad el crecimiento más importante se sitúa en el entorno de las aplicaciones[19], en especial, los campos relacionados con la industria y la empresa. Dicho desarrollo está supeditado al avance científico paralelo en el conocimiento del comportamiento de la neurona biológica, permitiendo su posterior implementación en los modelos artificiales.

El incremento constante en el número de aplicaciones está provocando un impacto trascendente en la industria en general[20], específicamente sectores[21] como por ejemplo, sector bancario, asegurador, etc. A través de estas aplicaciones las ventajas competitivas son claras en términos de productividad.

Smith y Gupta (2000) consideran que el desarrollo histórico de la investigación en este campo posee cinco fases vinculado a su impacto en la industria, la primera de ellas, desde 1834 a 1943, coincide con la creación de IBM , la segunda, de 1946 a 1957, donde merece resaltar por parte de General Electric Co, la creación del primer sistema computerizado en paralelo, en tercer lugar, de 1969 a 1982, aparece el primer microprocesador de Intel Corp, se crean las empresas SPSS, Inc., SAS Institute y Apple Computer Coporation, etc, en cuarto lugar, desde 1983 a 1990, resaltamos el informe DARPA, la creación de la empresa NeuralWare Inc y en la última fase, 1990 hasta hoy en día, la investigación continua con fuerza pero más orientada a la generación de aplicaciones que solucionen problemas reales[22].

[19] Véase los capítulos 9 a 11 sobre las posibilidades de los modelos neuronales, Swingler, K. (1996). **Applying Neural Networks. A Practical Guide**, Academic Press.

[20] Para un mayor detalle histórico, véase, Smith, M. (1993). **Neural Networks for statistical Modeling**, Van Nostrand Reinhold, New York, pp. 3-14 y aplicaciones empresariales, Bonsón, E. (1999). **Tecnologías Inteligentes para la gestión empresarial**, Ra-Ma, Madrid.

[21] Empresas como por ejemplo, *Neuraltech Inc*., *NeuralWare*, Inc, *Trajecta, Inc*, *Ward Systems Group*, etc. poseen en el mercado aplicaciones comerciales que permiten su utilización inmediata en los diferentes ámbitos de la economía y/o finanzas. En España, especial mención a la empresa AIS (Aplicaciones de inteligencia Artificial, SA; http://www.ais-int.com) por sus aplicaciones en los sectores financieros y de seguros. Véase para el impacto en Finanzas en Burell, P.R.; Folarin, B.O. (1997). **The impact of Neural Networks in Finance**, *Neural Comput & Applic*, 6, pp.193-200.

[22] Un caso claro son los problemas de control en los proceso industriales, véase Dracopoulos, D.C. (1997). **Evolutionary Learning Algorithms for Neural Adaptive Control**, Springer. Una visión más general véase en Lawrence, J. (1994). **Introduction to Neural Networks**. *California Scientific Software Press*, 6ª Edición.

Un aspecto importante es la aparición de un nuevo tipo de industria basada en el análisis de la información, denominada industria del *Data mining*[23], donde la metodología neuronal posee una parcela importante de aplicaciones.

Finalmente, la tabla 2.2.1. nos muestra una selección de los eventos más importantes en el avance de las redes neuronales, a través de sus investigadores más destacados.

Tabla 2.2.1.

1	Trabajo pionero de **McCulloch y Pitts** (1943).
2	Trabajos de **Hebb** (1949) sobre el proceso de aprendizaje, "aprendizaje Hebbiano".
3	**Rosenblatt** (1958, 1961), psicólogo que consideró que el cerebro aprendía por asociación, creando los modelos de tres capas.
4	**Marvin Minsky** y **Seymour Papert** (1969), criticaron las limitaciones de los modelos propuestos por Rosenblatt.
5	**Bernard Widrow**, desarrolló los modelos Adaline y Madaline y la regla de aprendizaje Widrow-Hoff (1960).
6	**James Anderson** (1968,1977) desarrolló una extensión del asociador lineal, Brain-State-in-Box (BSB). **Teuvo Kohonen** (1977) obtuvo un modelo similar de forma independiente.
7	**Stephen Grossberg** estudió los mecanismos de la percepción y la memoria.
8	**John Hopfield** (1982) presentó una variación del asociador lineal basado en la función de energía de Lyapunov para ecuaciones no lineales.
9	El algoritmo *backpropagation* fue descubierto simultáneamente por **Werbos** (1974), **Parker** (1985), **Rumelhart, Hinton** y **Williams** (grupo PDP).
10	**Kunihiko Fukushima** (1969, 1982) desarrolló el neocognitrón, un modelo de red neuronal para el reconocimiento de patrones visuales.
11	**DARPA, Neural Networks Study**, MIT Lincoln Laboratory, 1988. Es un compendio del conocimiento neuronal hasta la fecha.
12	**Halbert White** (1992), realiza un importante esfuerzo por vincular la tecnología neuronal con la teoría del aprendizaje y de la aproximación y su interpretación econométrica, **Kuan y White** (1994) y **Sarle** (1994).
13	Desde la óptica del vínculo existente con el aprendizaje estadístico las aportaciones de **Ripley B.D**. (1994) y de **Cheng y Titterington** (1994) son de una gran importancia.
	Fuente: Patterson, D.W. (1996). **Artificial neural networks. Theory and applications**, Prentice y elaboración propia.

[23] Para conocer con mayor detalle el estado del *business data mining*, véase Bose, I. And Mahapatra, R.K. (2001). **Business data mining–a machine learning perspective**, *Information & Management*, 39, pp. 211-225 y Berry M.J.A.; Linoff Gordon. (1997). **Data Mining Techniques. For Marketing, Sales, and Customer Support**, Wiley.

2.2. Características generales de los modelos *neuronales artificiales*.

Los modelos neuronales asumen muchas simplificaciones del modelo biológico para poder plantear su desarrollo matemático[24], así en esta línea, el primer modelo artificial fue diseñado por McCulloch-Pitts (1943), el cual utilizaba unidades de procesamiento denominadas *neuronas* que poseían dos estados discretos. Asociados a cada uno de ellos, se conseguía un *output* que se transmitía a lo largo de la estructura vinculada a la red neuronal, pero con la limitación que sólo permitían computar funciones *booleanas*[25]. La ilustración 1.2.2. compara una neurona biológica con el modelo matemático homólogo.

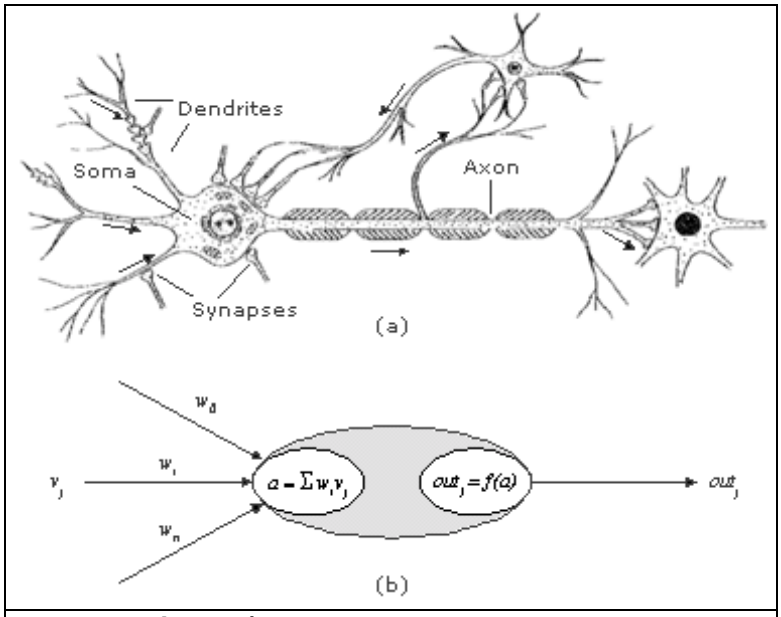


Ilustración 1.2.2. (Terminología original en inglés)
Paralelismo entre el Modelo Biológico (a) y el Modelo McCulloch-Pitts (b)

El modelo de McCulloch-Pitts posee las siguientes hipótesis. En primer lugar, el estado de una neurona en el tiempo, "$t+1$", depende solamente del estado que poseía en el período anterior, "t". En segundo lugar, una neurona estará activada o no si supera un umbral ,"θ", y en último lugar, se asume la sincronía entre *input* y *output* aún conociendo la asincronía en la neurona biológica.

[24] Para conocer los fundamentos matemáticos y algebraicos de la metodología neuronal, véase Ellacott, S.; Bose, D. (1996). **Neural Networks: Deterministic Methods of Analysis**, International Thomson Computer Press.
[25] El modelo de McCulloch-Pitts BDN (*Binary decisión*) es una simple aproximación a la operativa de una neurona biológica.

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

$$y_i^{t+1} = f\left(\sum_{j=1}^{m} w_{ij} x_j^t - \theta_i\right) = f(a)$$
$$f(a) = \begin{cases} 1 & \text{si } a \geq 0 \\ 0 & \text{en otro caso} \end{cases}$$

donde la función $f(\cdot)$, está descrita por $n+1$ parámetros, *n*-pesos $(w_1, w_2, \cdots, w_n)$ o eficacia sináptica y el umbral o tendencia (θ, "*bias*").

En términos generales la metodología que subyace en un modelo neuronal es la utilización de arquitecturas y reglas de aprendizaje que permitan la extracción de la estructura estadística presente en los datos. Para este proceso son necesarios tres elementos importantes: la *estructura* de nodos, la *topología* de la red y el *algoritmo de aprendizaje* utilizado para "*estimar*" los pesos o parámetros de la red.

Las ventajas inherentes que poseen dichos modelos son, entre otras, el *aprendizaje*, la *generalización* y la *robustez* frente al "ruido" en los datos. Es decir, la adquisición de la información es posible a partir del aprendizaje de la estructura interna de los datos, de forma que son las propias conexiones o pesos entre los diferentes elementos que constituyen una red neuronal donde se retiene el conocimiento. Es de gran importancia notar que no existe a priori una definición explícita de la forma del conocimiento, el propio algoritmo iterativo de estimación de los parámetros desconocidos o pesos, se encarga de extraer la presencia de regularidades en los datos[26].

Los aspectos de mayor relevancia de los modelos neuronales son, primeramente, sus *arquitecturas*, en segundo lugar, la *tipología* de las unidades de procesamiento, en tercer lugar, el tipo de *conexiones* de estas unidades o neuronas, en cuarto lugar, los paradigmas del *aprendizaje*, y para finalizar, la *teoría de la información* asociada a los algoritmos de aprendizaje.

[26] De ahí la expresión de *Aproximador Universal*, véase White, H. (1992). **Artificial Neural Networks. Aproximation and Learning Theory**, pp. 13-28, Blackwell Publishers.

El primero de los aspectos nombrados son las arquitecturas, es decir, la forma de las conexiones entre las unidades neuronales. Este aspecto se desarrollará en capítulos posteriores enfatizando las similitudes entre las mismas y los modelos econométricos tradicionales. Su forma genera toda una familia de posibles modelos, cuya gran variedad obliga a la vertebración de los mismos mediante clasificaciones o *taxonomías*.

En una primera aproximación, podemos encontrar una clasificación en función de la tipología del *output* que genera el modelo, divididos en: *modelos deterministas* y *modelos estocásticos*[27]. Para el caso determinista tenemos que cada neurona sigue una ley del tipo,

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

donde $f(.)$es la función de activación[28], en cambio para las redes con neuronas estocásticas, la activación de la red se interpreta como una probabilidad de un estado lógico tal y como se expresa en las expresiones siguientes,

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

y

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

donde el *output* es un valor continuo entre $[0,1]$ que se interpreta como una probabilidad.

El segundo aspecto es la tipología existente en las unidades de procesamiento o neuronas. Existen neuronas *visibles* y neuronas ocultas[29] (*hidden*). Por neuronas visibles se entienden tanto los *inputs* (variables exógenas) como los *outputs* (variables endógenas), en cambio las neuronas ocultas, poseen la función de capturar la representación interna de los datos. Éstas pueden no estar conectadas directamente con las neuronas visibles, (véase ilustración 2.2.2.).

[27] En el apartado 2.4. se realizará una *taxonomía* más detallada.
[28] Acostumbran a utilizarse funciones como por ejemplo, *identidad*, *sigmoide*, *tangente hiperbólica*, *gaussiana* etc., véase con más detalle apartado 2.3.
[29] El comportamiento en el interior de las neuronas ocultas es de interés de una parte de esta tesis doctoral, véase apartado 6.4.

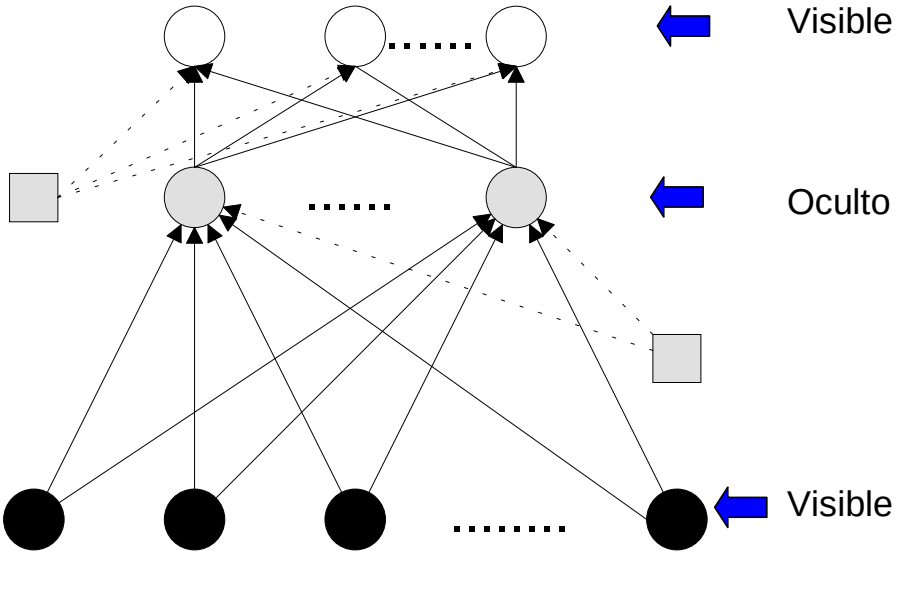


Ilustración 2.2.2.
Tipología de las *unidades de procesamiento* o neuronas.

El tercer aspecto descansa en el tipo de conexiones que se establecen entre las unidades de procesamiento o neuronas. Así tenemos, en primer lugar, los modelos que se propagan en una sola dirección, denominados *feed-forward* (véase ilustración 3.2.2.) y en segundo lugar, los modelos *recurrentes*, cuyas conexiones se establecen en todas las direcciones incluso con procesos de realimentación, es decir, las propias neuronas consigo mismas, (véase ilustración 4.2.2.). Estas últimas son especialmente útiles para captar el comportamiento dinámico en presencia de retardos[30].

El cuarto aspecto hace referencia a los paradigmas de aprendizaje. Existen principalmente dos tipos, *supervisado* y *no supervisado*. El primero de ellos, consiste en adaptar los pesos o conexiones de las neuronas para conseguir el mejor resultado en términos de aprendizaje, dirigido por un "*supervisor o profesor*[31]" (dicho aprendizaje puede presentarse en su versión determinista o estocástica).

[30] Véase un ejemplo de aplicación en, Torra P., S; V. Pérez-Rodríguez J.; Borrell Vidal, M. (1998). **Predicción multivariante de los tipos de interés en el mercado interbancario español con redes neuronales y varianza condicional**, *Comunicación presentada en I Encuentro de Economía Aplicada*, Universitat de Barcelona i Formació Continuada-Les Heures, Barcelona.

[31] Es decir, en cualquier momento del proceso de aprendizaje, éste está dirigido hacia unos objetivos predefinidos. En nuestro caso, representar lo mejor posible la relación entre los *inputs* y los *outputs* deseados.

Para el aprendizaje no supervisado, no existe la figura del "*supervisor*" y su objetivo es establecer una representación de los *inputs* en una nueva dimensión de base estadística (normalmente se le asocia, por similitud, la técnica estadística de *cluster*).

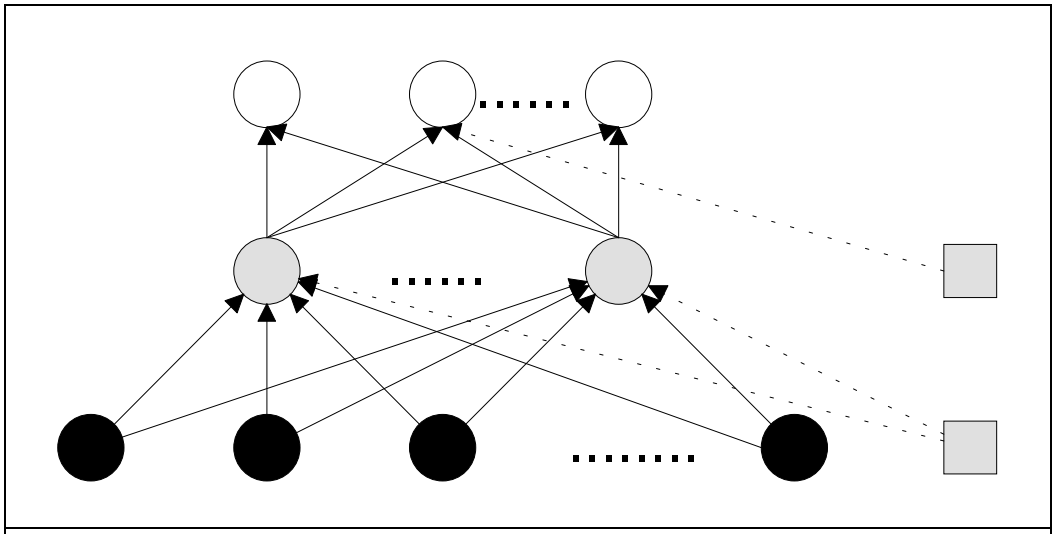


Ilustración 3.2.2.
Redes *feed-forward*

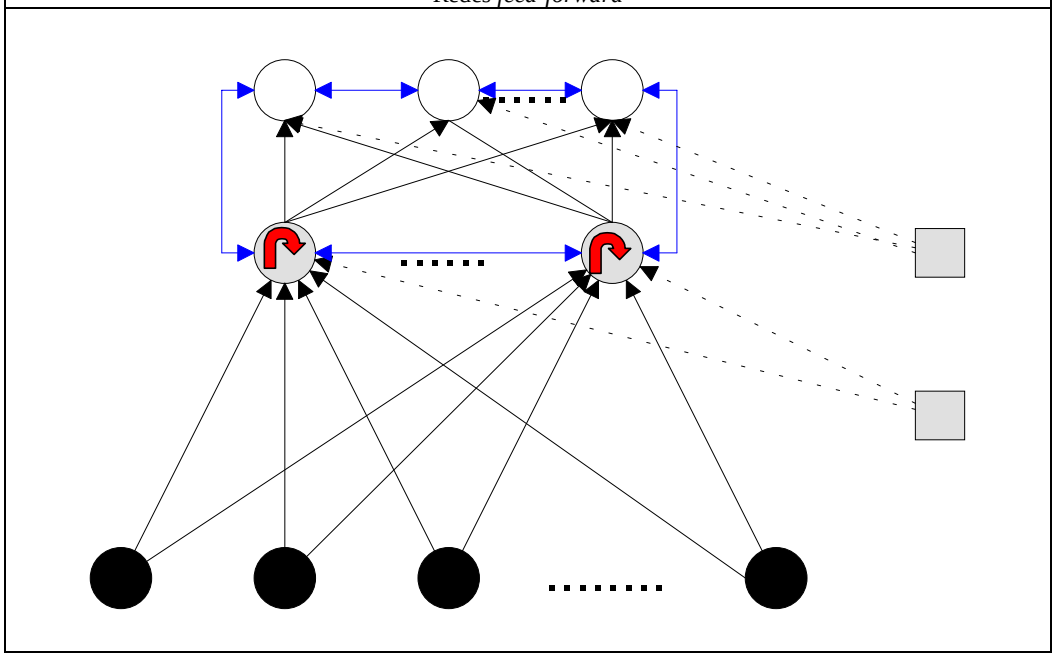


Ilustración 4.2.2.
Redes *recurrentes*

Por último, y sólo con la idea de mostrar algunas de las líneas de investigación actuales, presentamos, de forma sucinta, los conceptos de la teoría de la información que se

utilizan en los propios modelos neuronales[32]. Los conceptos de dicha teoría son muy útiles en el ámbito de la ingeniería, especialmente en el entorno de la teoría de comunicaciones, donde una de sus aplicaciones más comunes es la *compresión* de los datos. Los conceptos a que nos referimos son, entre otros, *entropía*, *información mutual*, *entropía de Kullback-Leibler*, etc, que se incorporan en el ámbito de la construcción de los modelos neuronales tanto en aprendizaje no supervisado como en el aprendizaje supervisado y que se desarrollarán de forma sintética a continuación.

El concepto básico por excelencia es la entropía, definida como una medida de incertidumbre acerca del valor de una variable aleatoria (x). Especificada por Shannon[33] en 1948 como $H(X)$, posee la siguiente expresión formal,

$$H(X) = -\sum_{x \in \aleph} p(x)\log(p(x)) = E\left[\log\left(\frac{1}{p(x)}\right)\right]$$

donde, E(.) es el operador esperanza y $p(x)$ la distribución de probabilidad de la variable aleatoria. Si consideramos más de una variable, la entropía tendrá un carácter *conjunto* y si condicionamos su comportamiento, la entropía será de naturaleza *condicional*, ambos conceptos se sitúan en el ámbito bidimensional. Así, sean "*x*" e "*y*", dos variables aleatorias de tipo discreto, donde $p(x, y)$ es la probabilidad conjunta y $p(y|x)$ la probabilidad condicionada, entonces definimos $H(x, y)$ como la entropía *conjunta*,

$$H(x, y) = -\sum_{x \in \aleph}\sum_{y \in \Im} p(x, y)\log[p(x, y)] = E\left[\log\left(\frac{1}{p(x, y)}\right)\right]$$

una medida de incertidumbre conjunta entre las dos variables aleatorias y la entropía *condicional* $H(y|x)$,

$$H(y|x) = -\sum_{x \in \aleph}\sum_{y \in \Im} p(x, y)\log[p(y|x)]$$

como la medida del grado de incertidumbre de "*y*", una vez se conocen los valores concretos de "*x*".

[32] Véase Deco, G.; Obradovic, D. (1996). **An information Theoretic approach to neural computing**, Springer.
[33] Véase Shannon, C.E. (1948). **A Mathematical Theory of Communication**, *Bell Sys. Tech. Journal*, 27, pp. 379-423 y pp. 623-656, referenciado en Deco, G.; Obradovic, D. (1996). **An information Theoretic approach to neural computing**, Springer.

En el ámbito de la probabilidad ha existido siempre la necesidad de definir medidas que capten la diferencia entre dos distribuciones, véase contrastes no paramétricos, de ahí que dentro de la teoría de la información podamos encontrar medidas con esta finalidad. En esta línea está la divergencia de *Kullback-Leibler*, $K(p,q)$, también denominada entropía *relativa*, que se define como,

$$K(p,q)=\sum_{x\in\aleph} p(x)\log\left(\frac{p(x)}{q(x)}\right)$$

siendo una *cuasidistancia* por no cumplir la característica de simetría.

Por último, un concepto de gran importancia es la *información mutua* $I(\cdot)$ que mide la independencia estadística entre dos variables aleatoria a través de sus distribuciones de probabilidad. La información mutua entre "x" e "y" se define como,

$$I(x;y)=K(p(x,y),p(x)p(y))=\sum_{x\in\aleph}\sum_{y\in\Im} p(x,y)\log\left(\frac{p(x,y)}{p(x)p(y)}\right)$$

siendo en este caso simétrica. Además se cumple que la información mutua[34] de una variable aleatoria consigo misma es igual a su entropía.

De forma más concreta y en el entorno del aprendizaje supervisado, la teoría de la información[35] juega un papel importante en la validación de los modelos obtenidos y en la obtención de estimadores máximos verosímiles de los parámetros. Desde la óptica de la validación, el criterio de información de *Akaike* (AIC) incorpora algunos de los aspectos anteriores siguiendo el principio de *Occam*, el cual sugiere escoger modelos con el menor números de parámetros. Si nos centramos en la estimación máximo verosímil, el método en sí puede interpretarse como la minimización de la distancia de información entre el modelo verdadero y el obtenido[36].

[34] Tal medida nos proporciona un valor que mide la cantidad de información que posee la reducción de la incertidumbre acerca de "y" cuando se conoce "x".

[35] Véase la relevancia de la misma en el entorno neuronal, Neelakanta, P.S. (1999). **Information-Theoretic Aspects of Neural Networks**, CRC Press.

[36] El problema conjunto de selección del modelo más idóneo y la estimación de sus parámetros está cercano a la teoría de la codificación, existiendo nuevos conceptos como puede ser el *Mínimal Description Length* (MDL) o *Stochastic Complexity* (SC), utilizados para modelizar sistemas dinámicos y de series temporales.

Finalmente existen fuertes vínculos entre la teoría de la mecánica estadística, el aprendizaje supervisado y la generalización que buscan los modelos neuronales, donde aspectos como, por ejemplo, el principio de máxima entropía, beneficio informativo contra complejidad, etc, son utilizados para el diseño de nuevos métodos de aprendizaje.

En segundo lugar, y referente al aprendizaje no supervisado, la utilización del método de análisis de componentes principales (PCA) para aproximarse al problema clásico de la extracción de características de los datos, está recibiendo otras definiciones en el entorno de la teoría de información[37]: tranformación de *Karhunen-Loève* o la transformación de *Hotelling* y además existen técnicas alternativas a su cálculo, como por ejemplo, el principio *Infomax* o el principio de minimización de la pérdida de información, conceptos muy vinculados a la información mutua[38] que poseen el mismo objetivo final.

[37] Véase una aproximación a la técnica de componentes principales neuronales, Diamantaras, K.I. y Kung, S.Y. (1996). **Principal Component Neural Networks. Theory and Applications**, Wiley.

[38] Creemos de mucho interés el Análisis de Componentes Independientes (ICA), siendo PCA un caso particular de un modelo *Gaussiano* ICA. (véase Aapo Hyvärinen; Ella Bingham. (2003). **Connection between multilayer perceptrons and regression using independent component analysis**, *Neurocomputing*, Vol. 50, pp. 211-222.

2.3. Teoría de la *aproximación* versus modelos neuronales.

Los modelos neuronales, definidos en el apartado 2.2., pueden ser considerados de momento como una *caja negra*, cuya estructura abstracta tiene *n*-inputs, que producen ciertos *m*-outputs deseados. Así la red es capaz de estimar una función, " f ", desconocida, que supone la siguiente aplicación, $F:\Re^n \to \Re^m$. La *teoría de aproximación* está muy cercana al planteamiento anterior, de ahí que, una vez expuestos los aspectos históricos, vamos a desarrollar un conjunto de características que permiten conocer el proceso computacional que se produce en el interior de un modelo neuronal y de esta forma desentrañar su comportamiento interno.

Un primer concepto inicial, son las *funciones primitivas*[39] (f), cuyo papel es permitir el cálculo computacional en el interior de la neurona abstracta, de forma que transforman el *input* en *output*, (véase ilustración 1.2.3.). Las diferentes especificaciones de modelos neuronales difieren precisamente en la forma de las propias funciones primitivas.

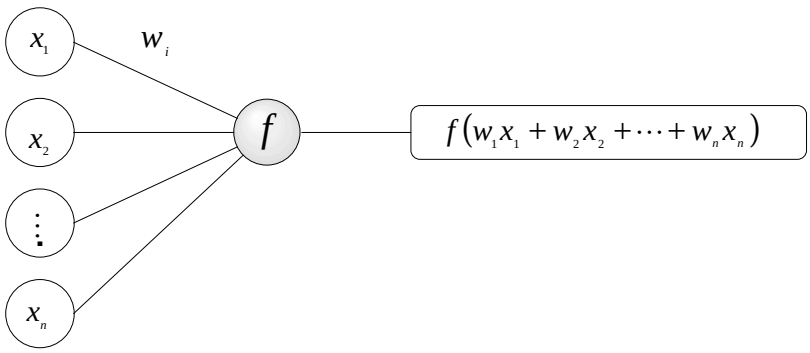


Ilustración 1.2.3.
Modelo de neurona *abstracta*

En segundo lugar, las *reglas de composición* para el modelo computacional, de forma que, existen dos casos, el proceso de *composición* y la presencia de procesos computacionales *recursivos*.

[39] Los diferentes modelos neuronales difieren principalmente en las hipótesis sobre las funciones *primitivas*, la interconexión establecida y la forma en la transmisión de la información. Por lo tanto los tres elementos básicos, ya comentados en apartados anteriores, son: la *estructura* o *arquitectura*, la *topología* y el *algoritmo* utilizado para estimar los *pesos* o *ponderaciones*.

Para el primero de los casos, la función primitiva que se computa en cada neurona posee dos componentes claramente diferenciados, el primero de ellos, consiste en un proceso de integración," g ", que reduce los *n*-argumentos en uno sólo, y en segundo lugar, la función de salida," f ", que produce un *output*, (véase ilustración 2.2.3.).

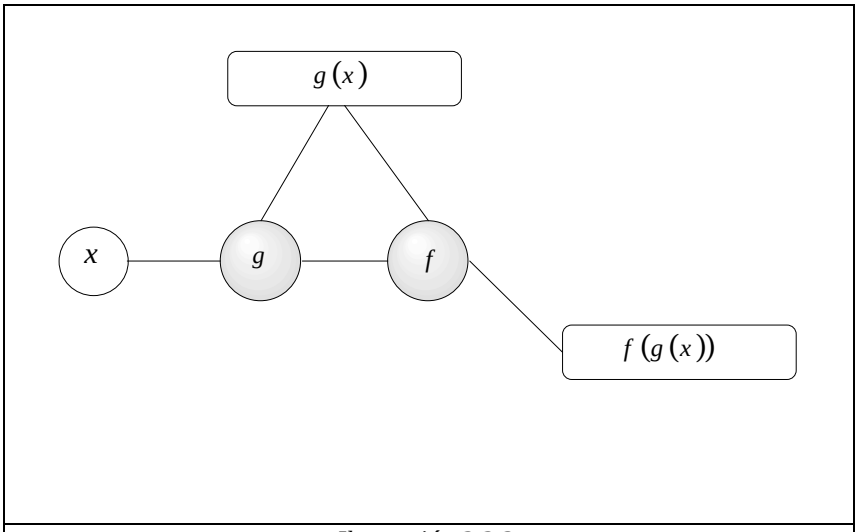


Ilustración 2.2.3.
Esquema de *composición* de funciones

En el segundo de los casos, los esquemas recursivos, son procesos de realimentación, (véase la ilustración 3.2.3.).

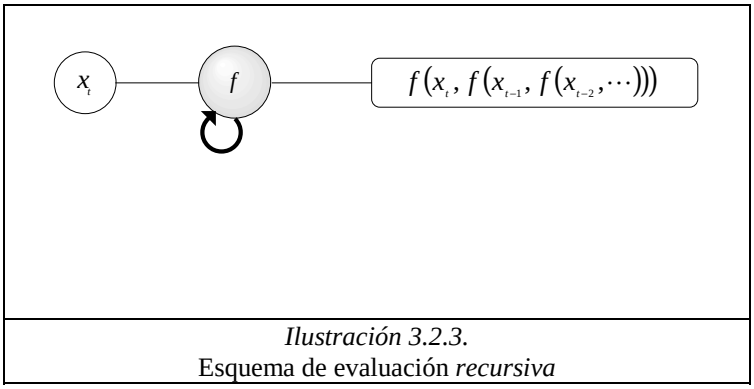


Ilustración 3.2.3.
Esquema de evaluación *recursiva*

Finalmente, el modelo de computación diseñado para el modelo neuronal posee en la mayoría de los casos, una función integración," g ", que es de naturaleza aditiva (más adelante se la definirá como función de transferencia o de activación), (véase la ilustración 4.2.3.).

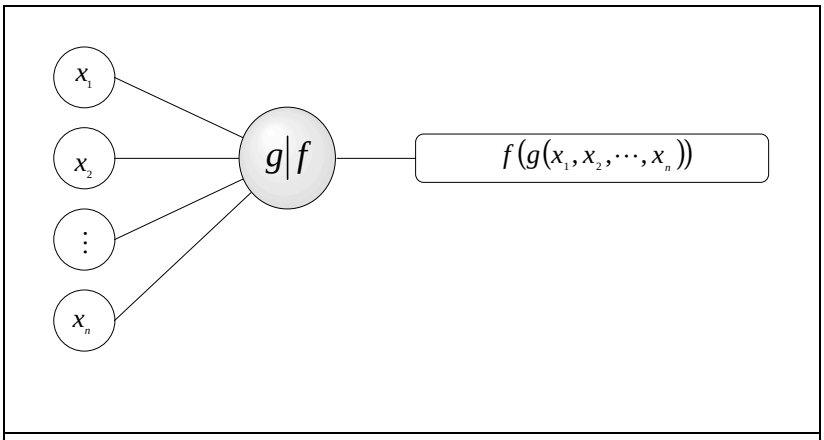


Ilustración 4.2.3.
Modelo de *computación*

Todo lo expuesto anteriormente nos permite acercarnos de forma progresiva a la teoría de aproximación. Uno de los ejemplos más referenciados es la aproximación con funciones de una sola dimensión mediante *polinomios* o series de *Fourier*, (véase ilustraciones 5.2.3. y 6.2.3.).

Para el primero de los casos, su expresión formal es la siguiente,

$$F(x) = a_0 + a_1(x - x_0) + a_2(x - x_0)^2 + \cdots + a_n(x - x_0)^n + \cdots,$$

donde, $a_0, \cdots a_n$ son constantes que dependen de la función, F, y sus derivadas en x_0.

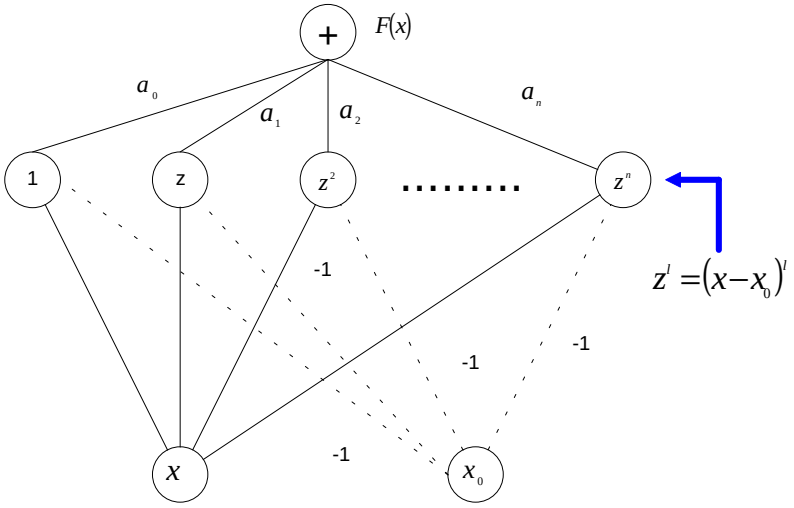


Ilustración 5.2.3.
Topología para series *polinomiales*

Para el segundo de los ejemplos, series de *Fourier*, su expresión es,

$$F(x)=\sum_{i=0}^{\infty}\left(a_i\cos(ix)+b_i\sin(ix)\right)$$

donde, $k_0,\cdots k_n$ son las constantes que determinan el número de ondas para los argumentos de las funciones "*seno*", las constantes $d_0,d_1,d_2,\cdots,d_n$ son los factores[40] "*fase*" y $w_0,\cdots w_n$ son las "*amplitudes*" de los términos de *Fourier*.

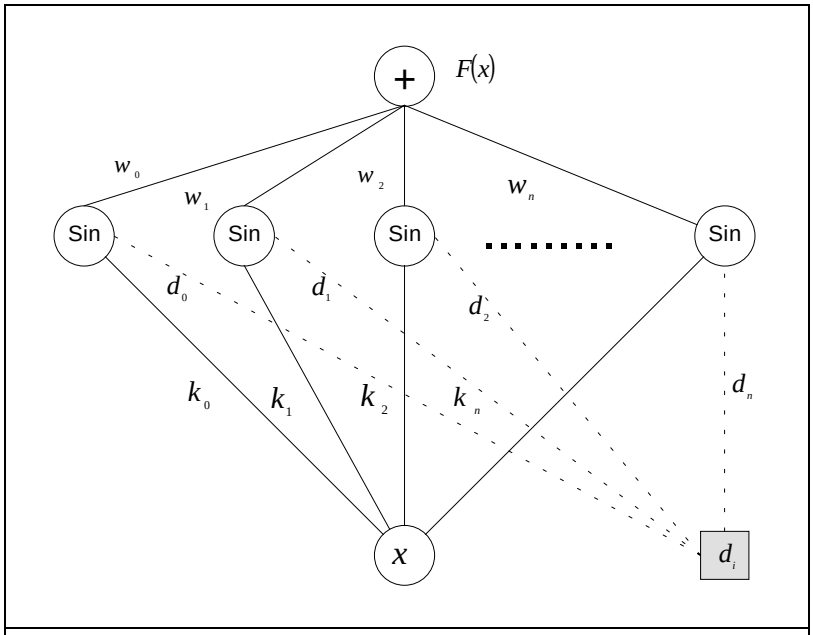


Ilustración 6.2.3.
Topología para series de *Fourier*

La principal diferencia entre las series *polinomiales* y de *Fourier* respecto a los modelos neuronales, es que estos últimos pueden aproximarse a la función $F(x)$desconocida y no explícita mediante la relación intrínseca entre los ejemplos de *outputs* e *inputs*. El algoritmo de aprendizaje del modelo neuronal modifica los parámetros de forma iterativa, hasta que capture la relación existente de la forma más fidedigna posible[41]. Los dos ejemplos anteriores enfatizan el potencial de los modelos neuronales para aproximar funciones, este aspecto será desarrollado en el apartado 6.2.

[40] Si $d_0=\pi/2$ significa que, $\sin(x+d_0)=\cos(x)$, en este caso no será necesario implementar de forma explícita la función coseno.

[41] La estimación de los parámetros de estos modelos se sitúa en el ámbito de la estimación no lineal, véase Ross, G.J.S. (1990). **Nonlinear Estimation**, Springer.

A modo de síntesis y siguiendo R. Rojas[42] (1996), el modelo completo computacional deberá poseer tres elementos, la función de *transferencia*, "*g*", la función de *salida*, "*f*", y las *unidades de contexto*, "*C*", utilizadas en los procesos recurrentes, (véase ilustración 7.2.3.). Con los tres elementos anteriores, el modelo genérico de neurona artificial incorpora todas las piezas básicas para poder iniciar su paralelismo con los modelos estadísticos y econométricos desarrollados en el capítulo 3 de la presente tesis.

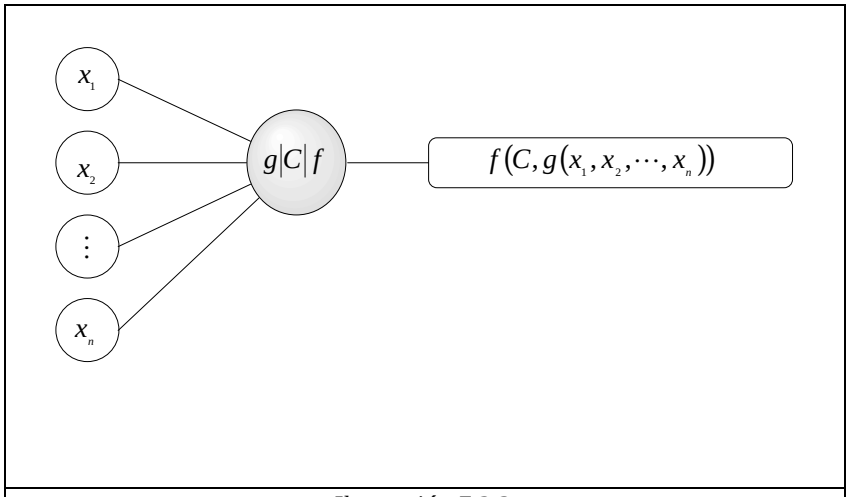


Ilustración 7.2.3.
Modelo de *computación* completo
Nota: Funciones de salida (*f*), transferencia (*g*) y neuronas de contexto (*C*)

Los elementos anteriores pueden desarrollarse con un mayor detalle. Así tenemos, en primer lugar, los *inputs* o *entradas*, $x_j(t)$. En segundo lugar, las *ponderaciones*, β_{ij}, que representan la intensidad de la interacción entre neuronas, es decir, indicadores del conocimiento retenido. En tercer lugar, la *regla de propagación*, $h_i(t) = \sigma(\beta_{ij}, x_j(t))$ y la función de *activación* o *transferencia*, $g_i(h_i(t))$(que suele ser determinista, monótona creciente y continua). En último lugar, la función de *salida*, $f_i(g_i(h_i(t))) = f_i(g[(\sigma(\beta_{ij}, x_j(t)))])$, (véase ilustración 8.2.3.).

Los elementos descritos permitirán una generalización formal, que permitirá aglutinar tanto los modelos supervisados como los no supervisados, (véase el apartado 2.2.).

[42] Véase Rojas, R. (1996). **Neural Networks. A Systematic Introduction**, Springer.

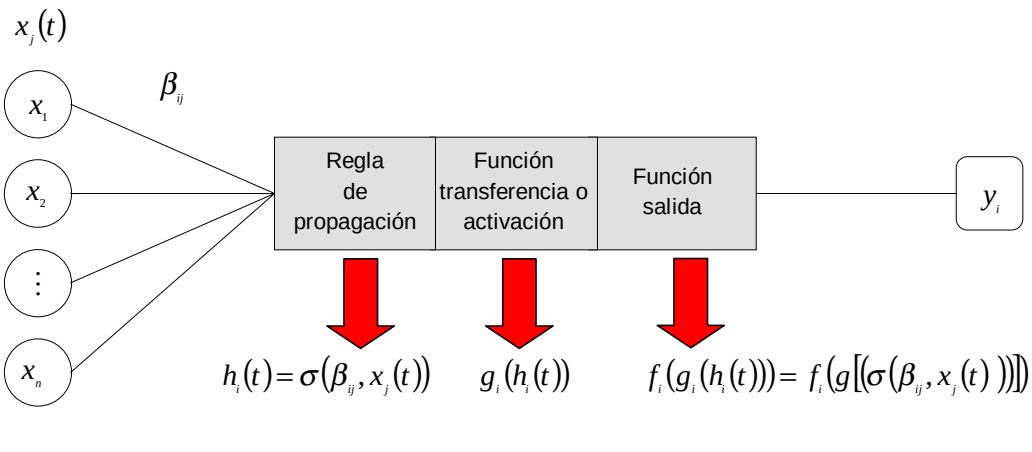


Ilustración 8.2.3.
Modelo genérico de *neurona artificial*

Fuente: Martín del Brío, B. y Sanz Molina, A. (1997)
Redes Neuronales y sistemas borrosos, RA-Ma, Madrid y elaboración propia

La regla de propagación es un elemento relevante que puede poseer diferentes formas, (véase ilustración 9.2.3.). En primer lugar, *cuadrática*, en segundo lugar, modelos basados en el cálculo de distancias entre vectores, como por ejemplo, la *distancia euclídea*[43], en tercer lugar, de carácter no lineal como la expresión polinómica y por último, la *lineal*, cuya expresión más utilizada es, $h_i(t)=\sum_{j=0}^{p}\beta_{ij}x_j$.

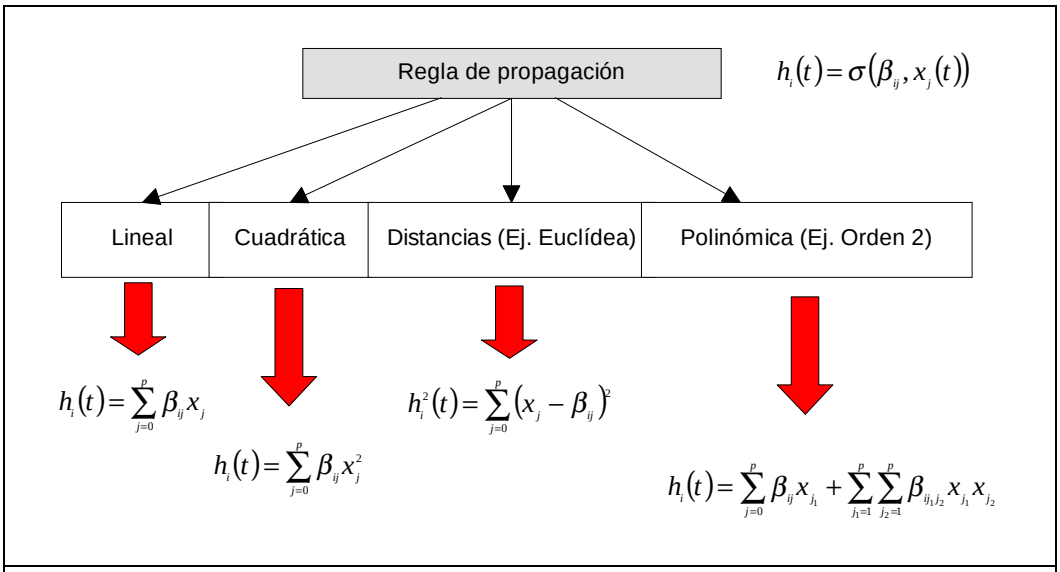


Ilustración 9.2.3.
Modelos de *reglas de propagación*

Fuente: Martín del Brío, B. y Sanz Molina, A. (1997)
Redes Neuronales y sistemas borrosos, RA-Ma, Madrid y elaboración propia

[43] Existen otras muchas posibilidades, *Manhattan*, *Mahalanobis*, etc.

El valor que surge de la regla de propagación elegida debe ser con posterioridad filtrado mediante la <u>función de transferencia o activación</u>, $g(\cdot)$. Ésta posee también diferentes formas (véase la ilustración 10.2.3.), cuyas expresiones se detallan en la siguiente tabla 1.2.3. En última instancia nos queda la <u>función salida</u>, $f(\cdot)$, que acostumbra a ser una función *identidad*, (véase los modelos *Adaline* o *MLP*). Pero existen otras posibilidades como, por ejemplo, la función *escalón*.

Tabla 1.2.3.

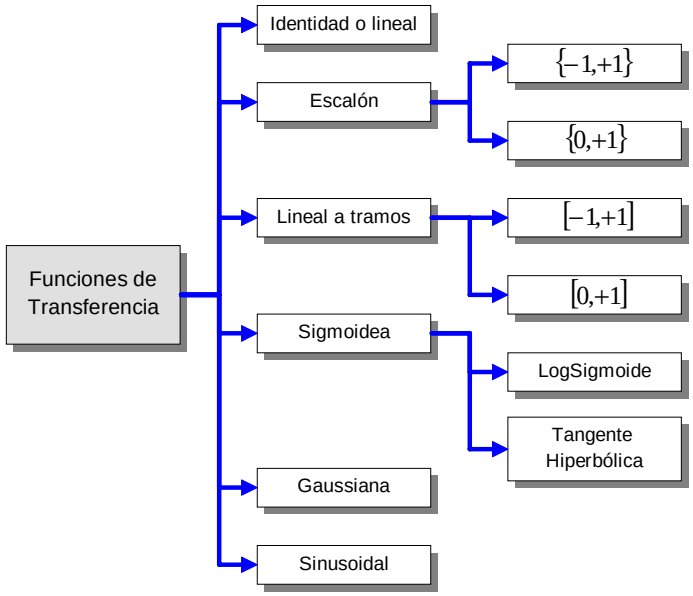
Expresión / Función	
$y_i = g\left(\sum_{j=0}^{p} \beta_{ij} x_j\right)$ $g(a) \equiv a$ Función *identidad* o *lineal*	
$y_i = g\left(\sum_{j=0}^{p} \beta_{ij} x_j\right)$ $g(a) = \begin{cases} 1 & a \geq 0 \\ 0 & a < 0 \end{cases}$ Función *escalón*	
$y_i = g\left(\sum_{j=0}^{p} \beta_{ij} x_j\right)$ $g(a) = \begin{cases} 1 & a \geq 0 \\ -1 & a < 0 \end{cases}$ Función *escalón* simétrica	
$y_i = g\left(\sum_{j=0}^{p} \beta_{ij} x_j\right)$ $g(a) = \begin{cases} 0 & a < 0 \\ a & 0 \leq a \leq 1 \\ 1 & a > 1 \end{cases}$ Función lineal a *tramos*	
$y_i = g\left(\sum_{j=0}^{p} \beta_{ij} x_j\right)$ $g(a) = \begin{cases} -1 & a < -1 \\ a & -1 \leq a \leq 1 \\ 1 & a > 1 \end{cases}$ Función lineal a *tramos* simétrica	
$y_i = g\left(\sum_{j=0}^{p} \beta_{ij} x_j\right)$ $g(a) \equiv \frac{1}{1+\exp(-a)}$ Función Logística o Log*sigmoidea*	$y_i = g\left(\sum_{j=0}^{p} \beta_{ij} x_j\right)$ $g(a) \equiv \tanh(a) = \frac{\exp(a) - \exp(-a)}{\exp(a) + \exp(-a)}$ Función Tangente hiperbólica *sigmoidea*
$y_i = g\left(\sum_{j=0}^{p} \beta_{ij} x_j\right)$ $g(a) \equiv \exp(-a^2)$ Función *gaussiana*	$y_i = g\left(\sum_{j=0}^{p} \beta_{ij} x_j\right)$ $g(a) \equiv sen(a)$ Función *sinusoidal*
<u>Nota</u>: se ha considerado en todos los casos, regla de *propagación lineal* y función de *salida identidad*.	

Ilustración 10.2.3.
Funciones de transferencia o *activación*

2.4. Modelos de redes Neuronales (*Taxonomía*).

La gran variedad de modelos de redes neuronales existentes en la actualidad obliga en cierta medida a la realización de clasificaciones o *taxonomías*. De esta forma, los modelos neuronales se pueden clasificar desde una triple óptica: en función de la forma del aprendizaje[44] ("*learning paradigm*"), en función de la arquitectura ("*network architecture*") y la tercera, que está situada en la área de las *aplicaciones*, (véase ilustración 1.2.4.).

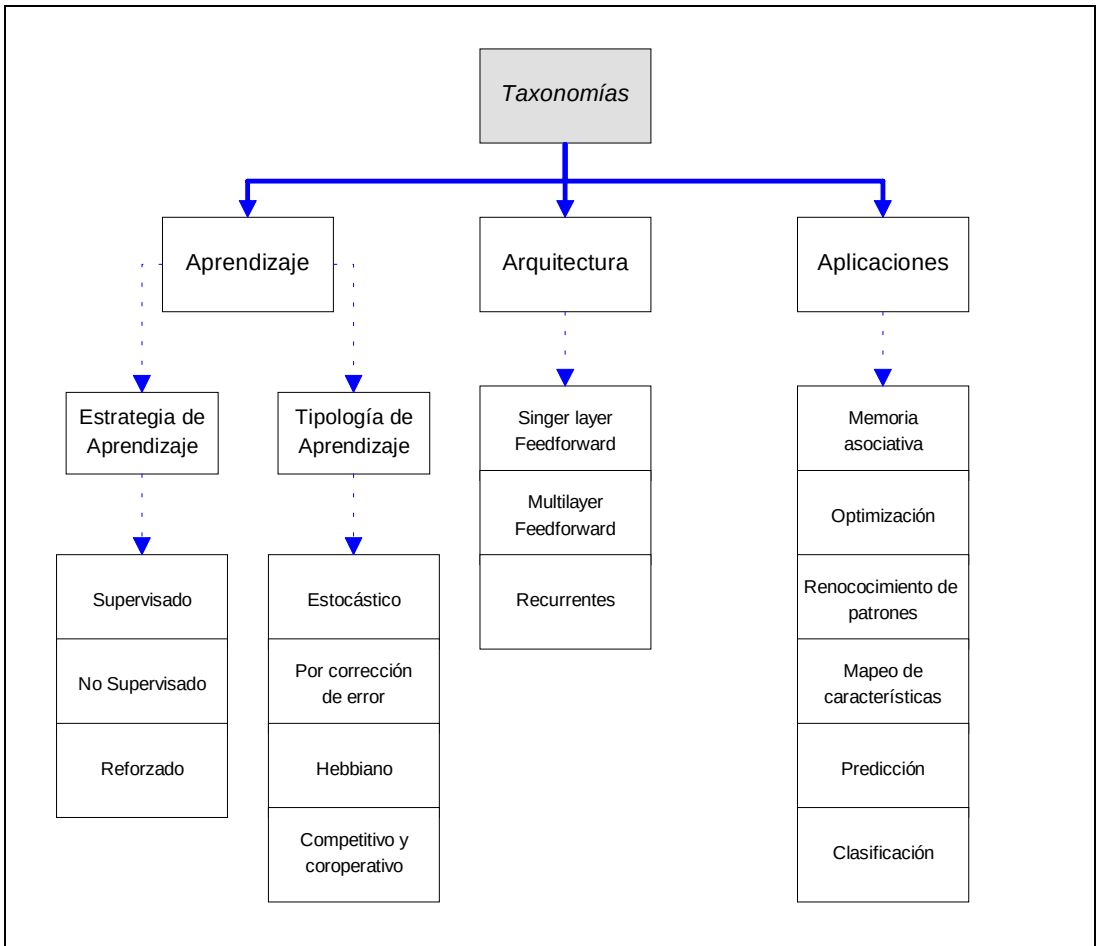


Ilustración 1.2.4.
Taxonomías

[44] El concepto de *aprendizaje* en el entorno neuronal se refiere a la forma en que se produce el ajuste de los parámetros, permitiendo realizar correctamente el tipo de procesamiento deseado.

De forma previa presentamos, sin ánimo de querer ser exhaustivos, una relación de aquellos modelos más utilizados con las siglas que permiten identificarlos[45]. Posteriormente serán agrupados[46] según las características definidas, (véase tabla 1.2.4.).

Tabla 1.2.4.

	Modelo Red Neuronal	*Siglas*
1	Adaline (*Adaptative Linear Neural Element)*	ADA
2	Adaptive Resonance Theory Networks	ARTN
3	Bidirectional Associative Memory	BAM
4	Boltzmann Machine	BOLTMA
5	Brain-State-in a Box Networks	BSBN
6	Cauchy Machine	CAUMA
7	Cerebellar Model Articulation Controller	CMAC
8	Counter-Propagation Networks	CPN
9	Delta Bar Delta Networks	DBDN
10	Finite Impulse Response Multilayer Perceptron	FIR-MP
11	Functional-link Networks (Polynomial Neural Networks)	FLN
12	Fuzzy ARTMAP Classification Networks	FARTMAP
13	General Regression Neural Networks	GRNN
14	Group Method of Data Handling (Polynomial Neural Networks)	GMDH
15	Hamming Networks	HAMN
16	Hierarchical Networks-Neocognitron	HNN
17	Hopfield	HOPF
18	Jordans´s sequential networks	JORDAN
19	Learning Vector Quantization	LVQ
20	Logicon Projection Network	LPN
21	Madaline (Multiple adalines)	MAD
22	Modular Neural Network	MNN
23	Multilayer Feedforward	MLFF
24	Nonlinear Autoregressive Moving Average Network	NARMA
25	Pipelined Recurrent Neural Networks	PPRN
26	Probabilistic Neural Networks	PNN
27	Radial Basis Function Networks	RBFN
28	Real-Time Recurrent Networks	RTRN
29	Recirculation Networks	RCN
30	Self-Organizing feature Map	SOFM
31	Sequential Cascaded Recurrent Networks	SCRN
32	Sigma-Pi Network (Polynomial Neural Networks)	SPN
33	Simple recurrent networks (Elman)	ELMAN
34	Spation-Temporal Pattern Recognition	STPR

[45] De todos modos existen otras posibles clasificaciones, por ejemplo véase R. Rojas (1996), que presenta una clasificación un poco más general, distinguiendo en primer lugar, entre aquellos modelos que pueden o no modificar su topología durante el proceso de aprendizaje, en segundo lugar, aquellos modelos que permiten que todas las neuronas pertenecientes a una misma capa se actualicen de forma simultánea, es decir, con dinámica *síncrona* o en su defecto cada neurona se actualiza de forma independiente de las demás (éste es realmente el funcionamiento biológico, forma *asíncrona*). En último lugar, la posibilidad de modelos que permitan la presencia de unidades de contexto, es decir, neuronas que almacenan información en iteraciones anteriores, permitiendo procesos recursivos o de realimentación.

[46] Véase Hagan, Martín T. (1996). **Neural Network Design**, PWS Publishing Company, ITP, Boston, MA; Chin-Teng Lin y George Lee, C.S. (1996). **Neural Fuzzy Systems. A Neuro-Fuzzy Synergism to Intelligent Systems**, Prentice Hall PTR, Upper Saddle River, NJ y Kosko, B. (1992). **Neural Networks and Fuzzy Systems. A Dynamical Systems Approach to Machine Intelligence**, Prentice-Hall.

35	Support Vector Machine	SVM
36	Time-Delay Neural Networks	TDNN
37	Tree Neural Networks	TNN
38	Wavelet Neural Networks	WNN

Desde la óptica de la forma del aprendizaje[47] podemos discernir entre un aprendizaje *supervisado*, *no supervisado*, *híbrido* y *aprendizaje reforzado* (estos dos últimos amplían el contenido del apartado 2.2.).

El primero de ellos, el *supervisado*, consiste en construir un modelo neuronal que permita estimar relaciones entre los *inputs* y los *outputs* sin la necesidad de proponer un cierta forma funcional a priori, siendo desde la óptica computacional más complejo pero con resultados más exactos. Como se observa en la ilustración 2.2.4., el *output* no coincidirá generalmente con el deseado, de forma que se generará un error de salida (e_i) o *residuo* del modelo.

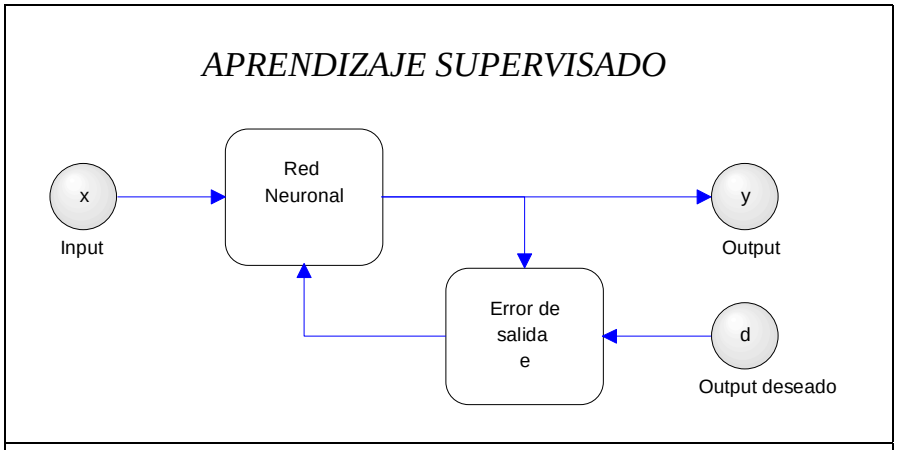


Ilustración 2.2.4.
Ciclo del aprendizaje *supervisado*

La segunda forma de aprendizaje es el *no supervisado* o *autoorganizado*, donde no se dispone de *output* deseado. Sus principales utilidades son entre otras, descubrir las regularidades presentes en los datos, extraer rasgos o agrupar patrones según su similitud (es decir, *cluster*), a través de la estimación de la función de densidad de probabilidad "$p(x)$" que permite describir la distribución de patrones "*x*" pertenecientes al espacio de entrada, R^n, (véase ilustración 3.2.4.).

[47] Véase Patterson, Dan W. (1996). **Artificial Neural Networks. Theory and Applications**, Prentice Hall.

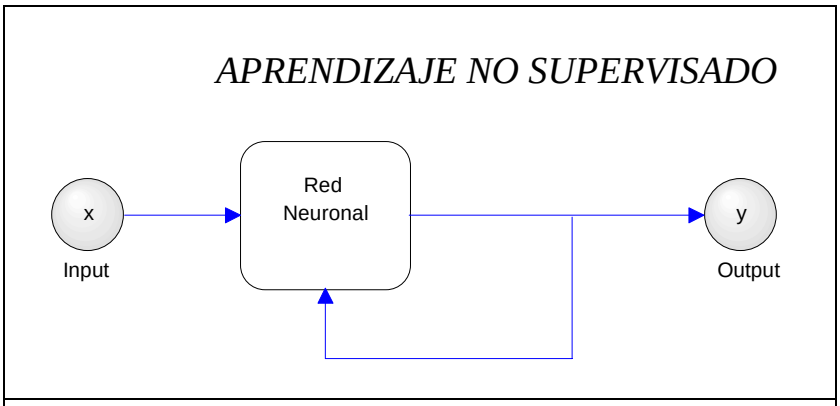


Ilustración 3.2.4.
Ciclo del aprendizaje *no supervisado*

El tercer método de aprendizaje es el *aprendizaje reforzado*, el cual se sitúa entre los dos anteriores, de forma que, por una parte se emplea la información del error cometido (calculado en este caso de forma global y no para cada uno de los outputs), pero se sigue sin poseer el *output* deseado. Dicho aprendizaje descansa en la idea dual *premio-castigo*, donde se refuerza toda aquella acción que permita una mejora del modelo mediante la definición de una señal crítica, (véase ilustración 4.2.4.). Esta estrategia de aprendizaje permite tratar con "targets" diferidos que aparecen, por ejemplo, en aplicaciones de robótica.

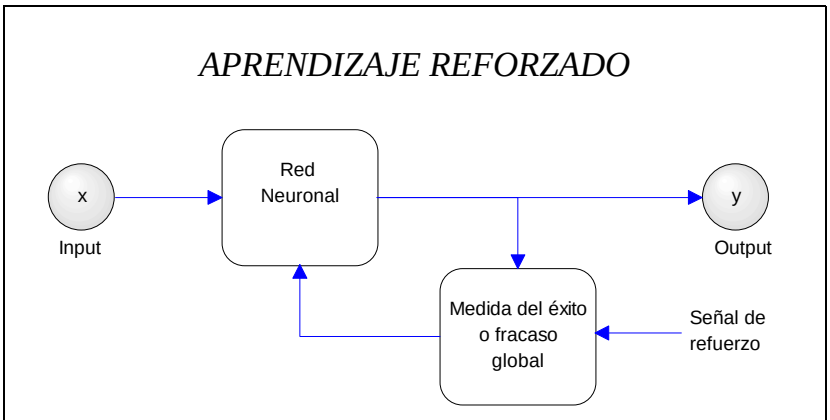


Ilustración 4.2.4.
Ciclo del aprendizaje *reforzado*

Por último tenemos el *aprendizaje híbrido*, donde coexisten en el modelo neuronal los dos tipos básicos de aprendizaje, el supervisado y el no supervisado, normalmente en distintas capas de neuronas. Modelos de este tipo son, por ejemplo, *Counter-Propagation Networks* y *Radial Basis Function Networks*, (ésta última topología será desarrollada en el apartado 2.5.5.1.).

Asociada a las anteriores tipologías de aprendizaje están las diferentes *estrategias* para poder conseguirlas, así la ilustración 5.2.4. las recoge de forma esquemática. En primer lugar, para el caso supervisado diferenciamos entre un aprendizaje por *corrección de error* y uno de carácter *estocástico*. En segundo lugar, para el caso no supervisado, tenemos el aprendizaje *hebbiano* y *competitivo*. En último lugar están las estrategias reforzadas, donde encontramos un aprendizaje por refuerzo.

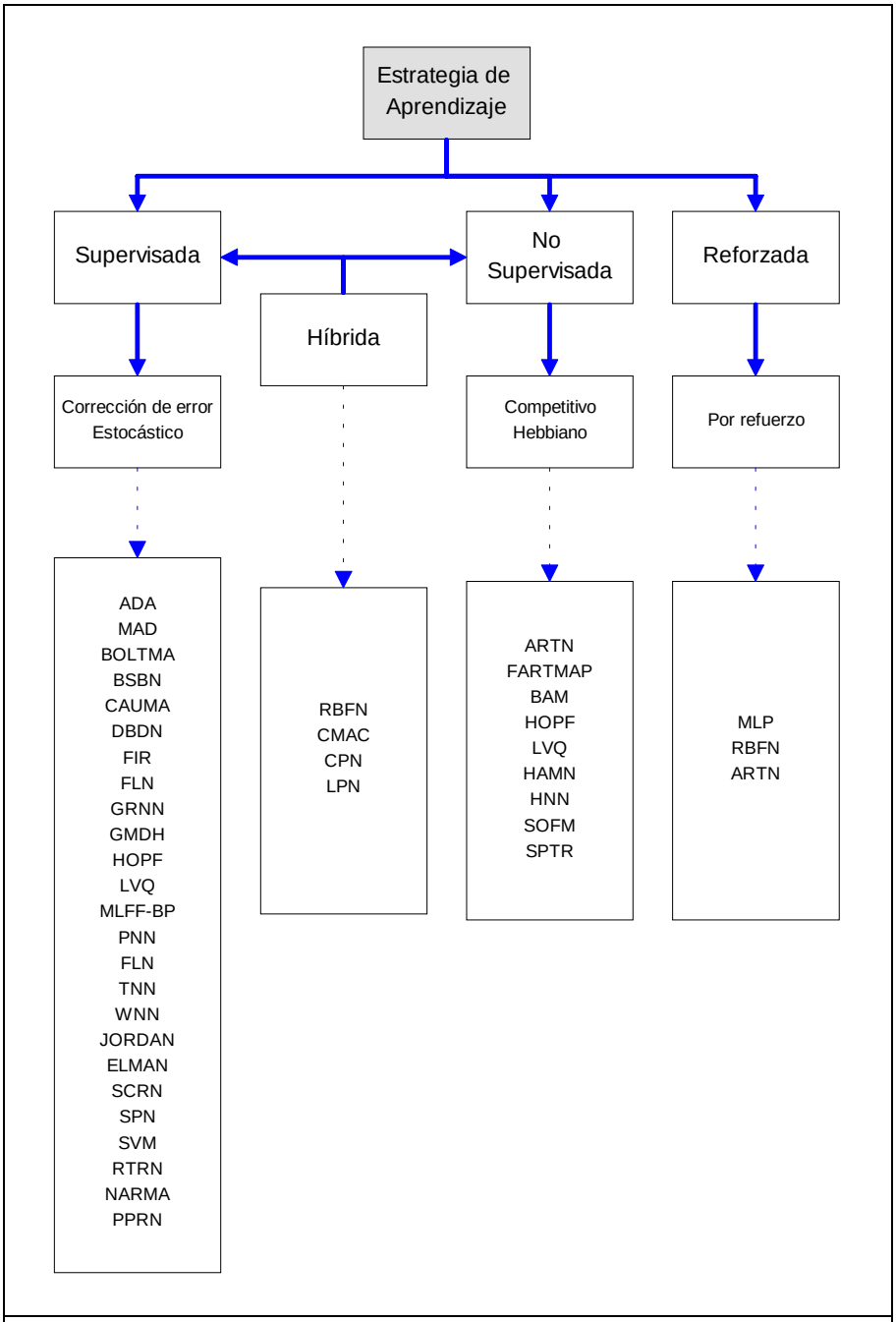


Ilustración 5.2.4.
Clasificación según la *estrategia de aprendizaje*

Con un mayor detalle para dos de las estrategias mayoritarias tenemos, primeramente y para el caso supervisado, el aprendizaje por *corrección de error*. Este consiste en ajustar los pesos de las conexiones de la red en función de la diferencia entre los valores deseados y los obtenidos en la salida de la red, es decir, en función del error cometido. Ejemplos de este tipo de aprendizaje son los tres algoritmos siguientes: la *regla de aprendizaje del Perceptron*, utilizada en el aprendizaje de la red Perceptron diseñada por Rosenblatt en 1958; la *regla delta o regla del error cuadrático medio mínimo* ("Least-Mean-squared Error"(LMS)) propuesta por Widrow en 1960, utilizada en los modelos neuronales Adaline y Madaline (estos modelos mejoran el modelo de Perceptron ya que incorporan la definición de error global cometido y mecanismos para reducirlo con mayor rapidez); y la *regla delta generalizada* o *retropropagación del error*.

El segundo tipo de aprendizaje es el *estocástico*, el cual consiste en realizar cambios aleatorios en los valores de los pesos de las conexiones de la red y evaluar el efecto a partir del objetivo deseado mediante distribuciones de probabilidad. A partir de conceptos de termodinámica, se asocia a la red un nivel de estado energético, de forma que, si su nivel es de mínimos entonces corresponde a una red bien entrenada. En algunos casos es necesario introducir "ruido" en el proceso de aprendizaje que facilita escapar de los mínimos locales o relativos. Nos referimos al procedimiento temple simulado (*simulated annealing*) que junto a la asignación probabilística forman parte del aprendizaje estocástico.

En el ámbito de la estrategia no supervisada, encontramos el aprendizaje *hebbiano*, postulado por Hebb (1949), que consiste en el ajuste de los pesos de las conexiones de acuerdo con la correlación de los valores de las dos neuronas conectadas. El aprendizaje *competitivo*, donde las neuronas compiten entre ellas con el fin de llevar a cabo un objetivo, es decir, las neuronas "luchan" para activarse quedando una por grupo ganador (*winner-take-all unit*). El objetivo de este aprendizaje es formar clusters de los datos que se introducen en la red, pero con el matiz de que el número de categorías es decidido por la propia red.

La ilustración 7.2.4. nos muestra otra posible clasificación en función del tipo de arquitectura. Las redes *feed-forward* que consisten en un tipo de arquitectura donde no existen conexiones hacia atrás. Las redes *feedback*, que permiten conexiones laterales y hacia atrás. Y las *redes recurrentes o realimentadas*, cuyas conexiones están permitidas tanto hacia atrás, como adelante, como la realimentación.

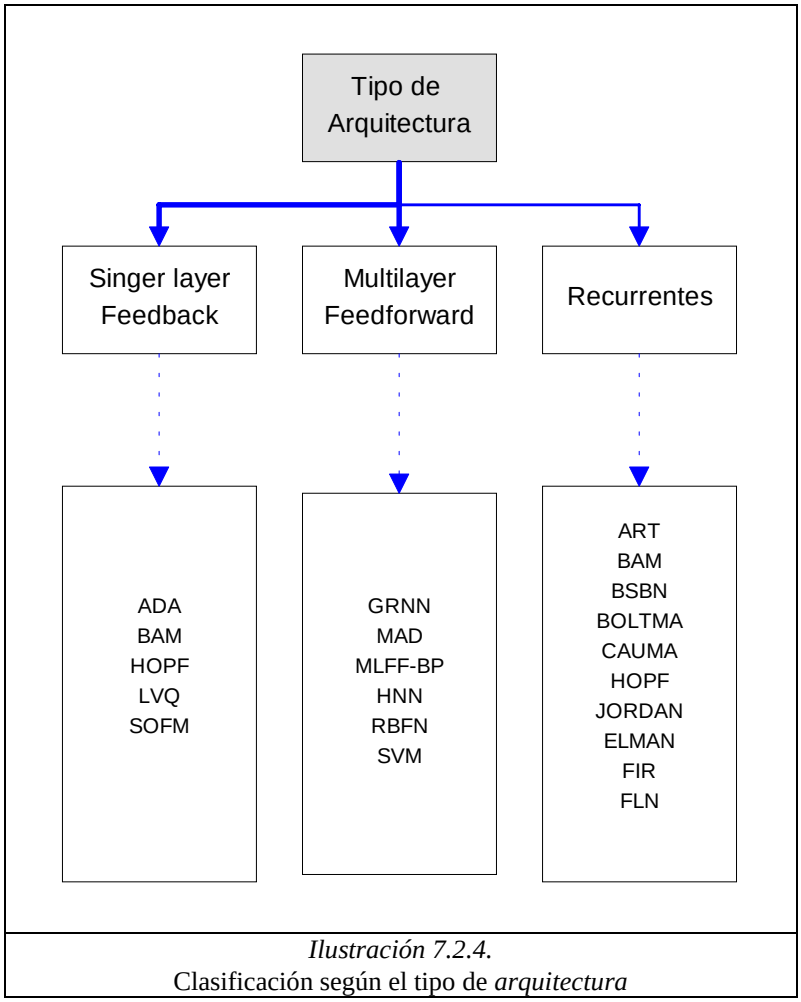


Ilustración 7.2.4.
Clasificación según el tipo de *arquitectura*

Respecto a las diferentes aplicaciones (véase ilustración 8.2.4.) tenemos, en primer lugar, *memoria asociativa*, consistente en reconstruir una determinada información de entrada que se presenta incompleta o distorsionada, asociando la información de entrada con el ejemplar más parecido de los almacenados conocidos por la red.

En segundo lugar, la *optimización*, es decir, la resolución de problemas de optimización combinatoria[48]que si se utilizan sistemas convencionales requieren mucho tiempo de computación. En tercer lugar, el *reconocimiento de patrones*, consistente, desde una óptica general, en la detección de formas simples (podríamos hablar de la *percepción artificial*). En cuarto lugar, el *mapeo de características*, que parte de las ideas de Kohonen (1982) simulando la capacidad del cerebro humano de crear mapas topológicos de las informaciones recibidas del exterior. En quinto lugar, está la *predicción* y en último lugar, la *clasificación*[49].

Podemos observar finalmente que una misma red puede utilizarse en aplicaciones diferentes.

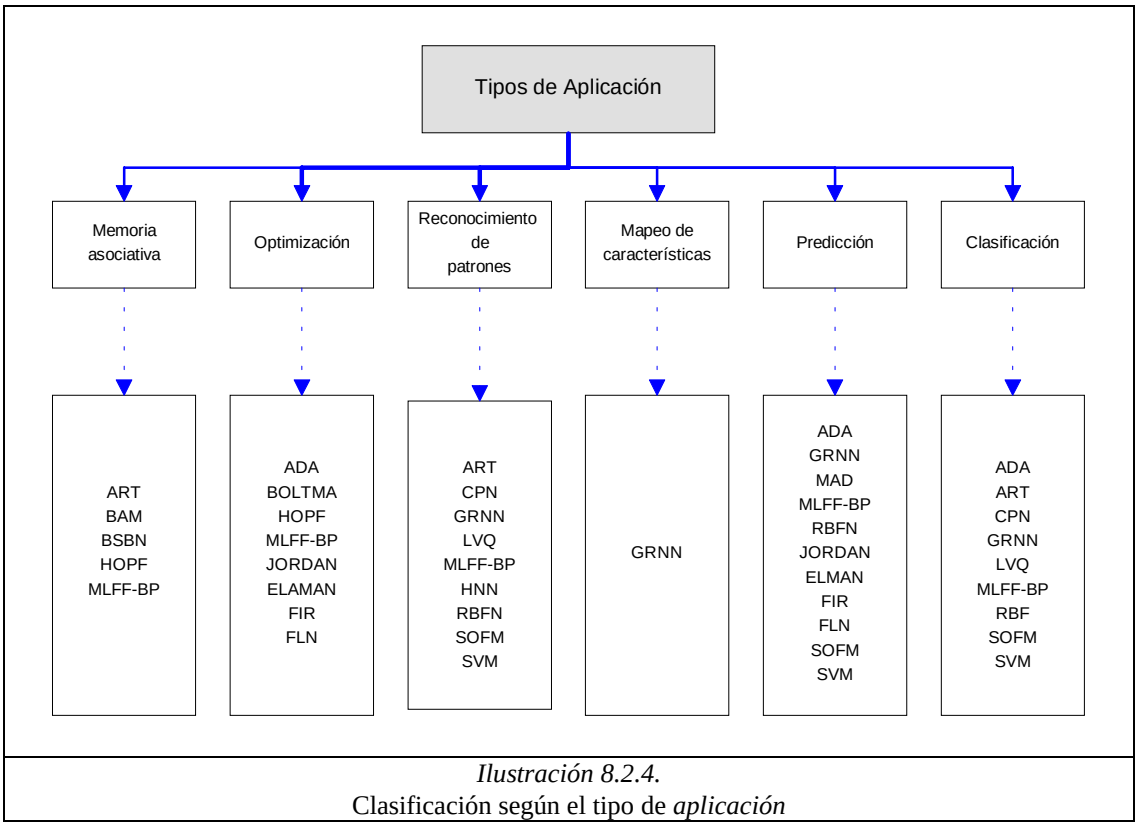


Ilustración 8.2.4.
Clasificación según el tipo de *aplicación*

[48] Por ejemplo el problema del viajante de comercio ("*Travelling Salesman Problem*")

[49] Véase Raudys, S. (2001). **Statistical and Neural Classifiers. An Integrated Approach to Design**, Springer; Borra. S.; Rocci, R. Vichi, M. Schader, M. (2000). **Advances in Classification and Data Analisys**, Springer y Ripley, B. D. (1994). **Neural Networks and Related Methods for Classification**, *J. Royal Statistical Society*, 56, No. 3, pp. 1-29.

Especial interés poseen los avances recientes en finanzas[50], ya sea desde la óptica de los modelos neuronales como de otras disciplinas relacionadas, como por ejemplo, los *algoritmos genéticos*[51]. Estos son utilizados en muchas aplicaciones de forma conjunta[52].

Desde una óptica más general Krycha, K.A. y Wagner, Udo[53] (1999) realizan una excelente recopilación de las aplicaciones existentes, tanto en el entorno financiero como en el ámbito del marketing, producción[54] y metodológico desde 1990 a 1996. Algunas de las posibilidades de aplicación en el entorno financiero[55] están recogidas en la ilustración 9.2.4., las cuales acaban derivando hacia *sistemas inteligentes*. Dichos sistemas inteligentes (*Intelligent Systems*) pueden definirse como aquella categoría de computación que puede encontrar patrones y descubrir relaciones en bases de datos. Estos sistemas están muy relacionados con otros, como por ejemplo, los sistemas de toma de decisiones[56] (*decisión support system*), donde encontramos muchas de las técnicas ya nombradas en el capítulo anterior: *redes neuronales*, *algoritmos genéticos*, *sistemas difusos*, *sistemas expertos* y los *sistemas inteligentes híbridos*.

---

[50] Véase Feldman, K. y Kingdon, J. (1995). **Neural networks and some applications to finance**, *Applied Mathematical Finance*, 2, pp. 17-42; Wong, Bo K. y Yakup Selvi. (1998). **Neural network applications in finance: A review and analysis of literature (1990-1996)**, *Information & Management*, 34, pp. 129-139; Baestaens, D.E.; Van Den Bergh, W.M.; Wood, D. (1994). **Neural Network Solutions for trading in Financial Markets**, Pitman Publishing; Van Eyden, R.J. (1996). **The Application of Neural Networks in the Forecasting of Share Prices**, *Finance & Technology Publishing* y en especial Olmeda, J.I. (1995). **Modelos no lineales en Finanzas**, Tesis Doctoral, Universidad de Alcalá, Madrid.

[51] Véase Kingdon, J. y Feldman, K. (1995). **Genetic algorithms and applications to finance,** *Applied Mathematical Finance*, 2, pp. 89-116.

[52] Véase como guía de referencia, Davis, L. (1991). **Handbook of genetic algorithms**, Van Nostrand Reinhold, New York. Los algoritmos genéticos se suelen utilizar para estimar los parámetros necesarios para el funcionamiento de ciertos modelos neuronales, como por ejemplo, los GRNN.

[53] Véase Krycha, Karl A. ; Wagner, Udo. (1999). **Applications of artificial neural networks in management science: a survey**, *Journal of Retailing and Consumer Services*, 6, pp. 185-203.

[54] Véase Fuente G. D. y Pino, R.D. (1996). **Aplicación de Sistemas Expertos y Redes Neuronales en Organización de la Producción y Previsión**, ESIC, Octubre-Diciembre, Madrid.

[55] Véase con carácter general, Trippi R.R. y Turban. E. (1993). **Neural Networks in Finance and Investing**, Probus Publishing Company y como un ejemplo de hibridación entre un sistema neuronal y estadístico aplicado a las finanzas, Oh, K.J.; Kim, K-J. (2002). **Analyzing stock market tick data using piecewiese nonlinear model**, *Expert Systems with applications*, 22, pp. 249-255.

[56] Véase ejemplos en, Caporaletti, L.E.;Dorsey, R.E.;Johnson, J.D. y Powell, W.A. (1994). **A decision support system for in-sample simultaneous equation systems forecasting using artificial neural systems,** *Decision Support Systems*, 11, pp. 481-495; Hill, T. ; Remus, W. (1994). **Neural network models for intelligent support of managerial decision making**, *Decision Support Systems*, 11, pp. 449-459; Schocken, S. (1994). **Neural networks for decision support: Problems and opportunities**, *Decision Support Systems*, 11, pp. 393-414 y Christopher W., J. (1992). **Self-organizing excecutive information networks**, *Decision Support Systems*, 8, pp. 41-53.

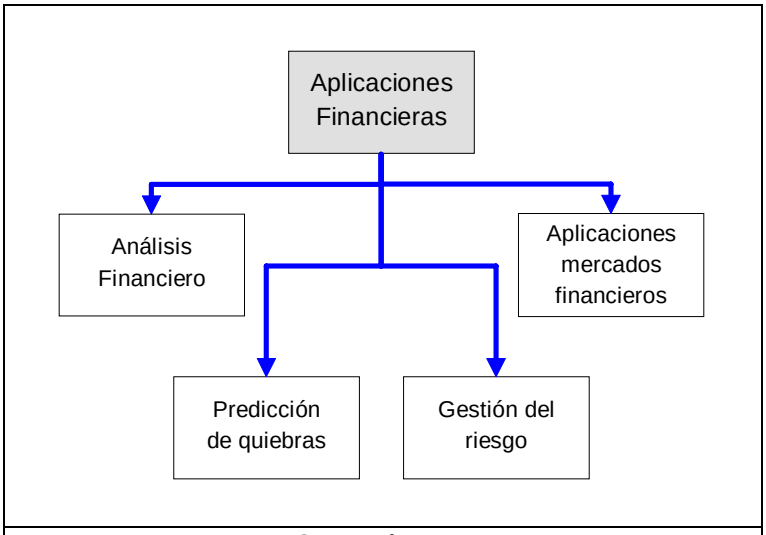


Ilustración 9.2.4.
Clasificación *aplicaciones* en el campo de las finanzas

Todas las técnicas anteriores poseen las siguientes características comunes, *adaptación*, *flexibilidad*, *explicación*, *descubrimiento*, las cuales pueden ser combinadas de forma diferente.

La tabla 2.2.4. muestra el grado de cumplimiento de las mismas en función de la tecnología empleada. La metodología neuronal y los algoritmos genéticos obtienen la máxima puntuación en tres de las características o propiedades deseables para un sistema inteligente[57].

Tabla 2.2.4.

Tecnologías	*Aprendizaje*	*Flexibilidad*	*Adaptación*	*Explicación*	*Descubrimiento*
Redes Neuronales	❋❋❋❋❋	❋❋❋❋❋	❋❋❋❋❋	❋	❋❋
Algoritmos Genéticos	❋❋❋❋❋	❋❋❋❋	❋❋❋❋	❋❋❋	❋❋❋❋❋
Sistemas Difusos	❋	❋❋❋❋❋	❋	❋❋❋	❋
Sistemas Expertos	❋	❋	❋	❋❋❋❋❋	❋

Nota: ❋ el número de símbolos nos indica un mayor grado de cumplimiento en la característica definida como deseada

[57] Véase para mayor detalle, Goonatilake, S. y Treleaven, P. (1995). **Intelligent Systems for Finance and Business**, Wiley, y Kingdon, J. (1997). **Intelligent Systems and Financial Forecasting**, Springer.

Finalmente la ilustración 10.2.4. nos muestra con mayor detalle las posibilidades actuales de los sistemas inteligentes definidos, en los campos de seguros, marketing[58], planificación, vigilancia, banca de inversiones[59] y banca al por menor.

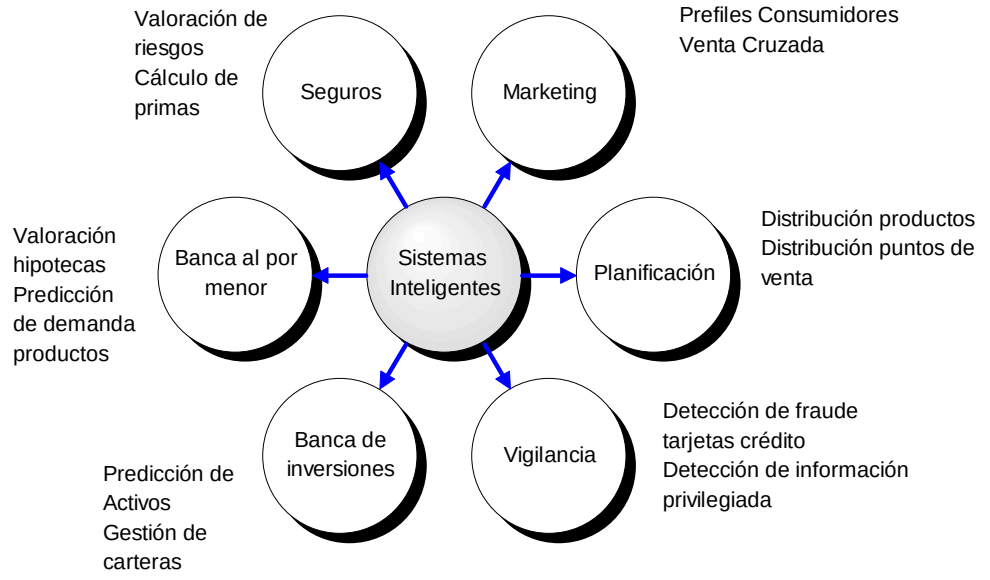


Ilustración 10.2.4
Clasificación de sistemas inteligentes en *finanzas* y *economía*.

El crecimiento de los sistemas inteligentes, en el entorno de las nuevas tecnologías[60], está asegurado en aquellos sectores donde la información es un elemento clave de las organizaciones, como por ejemplo, Banca, Seguros, etc. El incremento del volumen de información que se genera diariamente obliga a diseñar sistemas con cierto automatismo, para garantizar el mantenimiento de la productividad y fiabilidad en los sistemas de información.

[58] Véase Curry, B.; Morgan, P.; Silver, M. (2002). **Neural networks and non-linear statistical methods: an application to the modelling of price–quality relationships**, *Computers & Operations Research*, 29, pp. 951-969.
[59] Véase una aplicación en Indro, D.C.; Jiang, C.X.; Patuwo, B.E.; Zhang, G.P. (1999). **Predicting mutual fund performance using artificial neural networks**, *Omega, Int. J. Mgmt, Sci*., 27, pp. 373-380.
[60] Véase Chorafas, D.N. (1992). **The New Technology of Financial Management**, Wiley y Trippi, R.R.; Lee, J.K. (1992). **Artificial Intelligence in Finance & Investing. State-of-the-Art Technologies for Securities Selection and Porfolio Management**, IRWIN Profesional Publishing.

2.5. *Descripción* **de los Modelos Neuronales.**

2.5.1. Modelos *feed-forward* **y aprendizaje supervisado.**

La red conocida como *perceptron simple* es una red neuronal tipo *feed-forward*[61] supervisada, sin capa oculta, constituida por un vector de "*p*" *inputs*, $X = (x_1, x_2, \cdots, x_p)'$, un vector de "*n*" *outputs* deseados, $Y = (y_1, y_2, \cdots, y_n)'$, (véase ilustración 1.2.5.1.). La relación entre ambos vectores, (*inputs* ; *outputs*) se obtiene mediante la regla de aprendizaje, *perceptron learning rule*. Se demuestra que converge de forma correcta en un número finito de iteraciones[62] (*perceptron convergence theorem*). Si adicionalmente las clases son linealmente separables, permite su utilización en problemas de clasificación con más de una categoría.

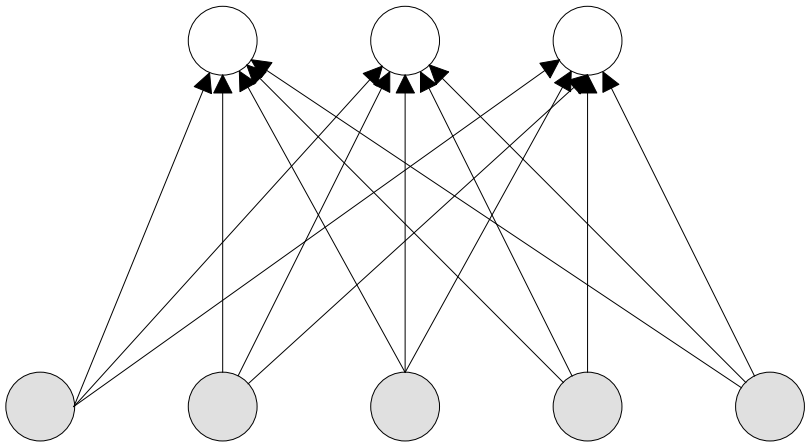


Ilustración 1.2.5.1.
Ejemplo de Modelo de *perceptron* simple
(5 inputs y 3 outputs)

[61] Las redes *feed-forward* son redes multicapa con conexiones hacia adelante, es decir, las neuronas de una capa reciben señales de entrada de otra capa y envían señales de salida a una capa posterior. Los modelos más conocidos son entre otros, *perceptron*, *adaline*, *madaline*, *linear adaptive memory*, etc. , véase Hilera, J.R. y Martínez, V.J. (1995). **Redes neuronales artificiales. Fundamentos, modelos y aplicaciones**, pp. 71-72, Ra-Ma, Madrid.

[62] Véase Hertz, J.; Krogh, A. Palmer R.G. (1991). **Introduction to the theory of neural computation**, Addison-Wesley Publishing Company.

Un modelo neuronal que representa la relación lineal entre *input* y *output* es la red *Adaline*[63] (*adaptive linear element*). Este modelo utiliza una neurona similar a la del *perceptrón* simple pero de respuesta lineal. Su utilización es posible siempre que los *inputs* sean linealmente independientes, lo cual implica, de forma unidireccional, la condición de separabilidad entre los mismos. El mecanismo que posee para su aprendizaje es la regla de *Windrow-Hoff* o *least mean square* (LMS)[64], que puede considerarse un caso particular de la regla de aprendizaje delta, *delta learning rule*. Ésta última considera como función de activación no lineal la función sigmoidea.

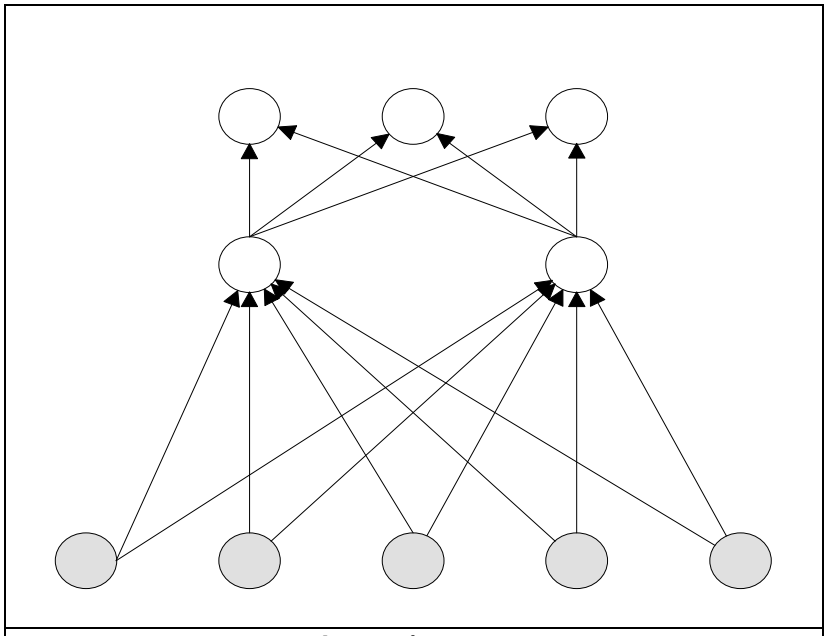


Ilustración 2.2.5.1.
Ejemplo de modelo de red *multilayer feed-forward*
(5 inputs, una capa oculta con dos neuronas y 3 outputs)

La versión multicapa de la red Adaline se denomina *Madaline*[65] y su homólogo en términos de perceptron simple son las redes *multilayer feed-forward,* (véase ilustración 2.2.5.1.).

[63] Véase Chin-Teng Lin y George Lee, C.S. (1996). **Neural Fuzzy Systems. A Neuro-Fuzzy Synergism to Intelligent Systems**, Prentice Hall PTR, Upper Saddle River, NJ.

[64] La diferencia entre la regla de aprendizaje del *perceptron* y la LMS, es que la primera penaliza la red cuando se produce un error, es decir, perturba los pesos de forma proporcional a la entrada o input, mientras que el segundo, los perturba siempre de forma proporcional al error y la entrada.

[65] Véase Freedman, F.A. Skapura, D.M. (1993). **Redes Neuronales. Algoritmos. Aplicaciones y Técnicas de Programación**, Addison-Wesley / Díaz de Santos, Madrid.

Las redes *Multilayer feed-forward* (MLP) pueden considerarse unos aproximadores funcionales universales, es decir, una red con una única capa oculta, puede aproximar hasta el nivel deseado dentro de un conjunto compacto cualquier función continua[66]. Las redes multilayer feed-forward son entrenadas habitualmente con el algoritmo de aprendizaje denominado *Back-propagation* o BP, uno de los algoritmos con más importancia histórica en el desarrollo de las redes neuronales. Las redes neuronales asociadas al algoritmo Back-propagation se definen como *redes back-propagation*[67], (véase ilustración 3.2.5.1.).

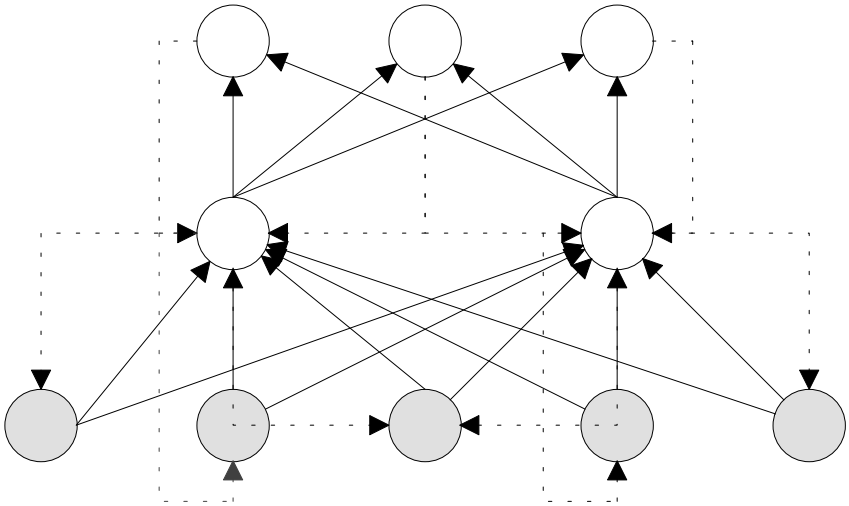


Ilustración 3.2.5.1.
Ejemplo de Modelo de red *back-propagation*

Nota: la línea discontinua representa el retroceso del error en el proceso *iterativo.*

Los factores que influyen en el proceso de aprendizaje del algoritmo back-propagation (BP) son, entre otros,

[66] Véase Hornik, K. Stinchcombe M. 1992. **Multilayer Feedforward Networks are Universal Approximators**. (referenciado en White, H. (1992). **Artificial Neural Networks. Approximation and Learning Theory**, Blackwell Publishers).

[67] Es habitual en el entorno de series temporales comprobar la eficiencia del modelo MLP-BP frente la metodología Box-Jenkins (ARIMA), véase Chakraborty, K.;Mehrotra, K.; Mohan, Ch.K. y Ranka, S. (1992). **Forecasting the Behavior of Multivariate Time Series Using Neural Networks**, *Neural Networks*, Vol.5, pp. 961-970.

- los *pesos iniciales* que son normalmente inicializados de forma aleatoria, pero existen otras posibilidades, como por ejemplo, $\left(-3/\sqrt{k_i}, 3/\sqrt{k_i}\right)$ donde "k_i" es el número de conexiones entre inputs y neuronas en la capa oculta.
- la *constante de aprendizaje*, factor de gran importancia en el proceso de convergencia, tanto en lo referente a cómo afecta su valor, como cual es el mejor valor a utilizar en las aplicaciones.
- las *funciones de coste*, usualmente se utiliza la función cuadrática.
- el *momentum*, filtrado de paso bajo (alisado) del gradiente del error.
- técnicas de optimización utilizadas, métodos de *gradiente descendente*, método de *Newton*, método de *quasi-Newton*, método de *dirección conjugada*.
- Aprendizaje y generalización
- *Número de neuronas* en las capas ocultas.

Muchas aplicaciones requieren que las redes neuronales diseñadas respondan a secuencias en el tiempo de patrones, es decir, series temporales[68]. Si utilizamos el algoritmo de aprendizaje BP, podemos convertir una red MLP en una red *time-delay neural networks* (TDNN)[69] simplemente utilizando *inputs* o entradas con retardos. Adicionalmente existen otros tipos de redes *feed-forward*, agrupadas con el nombre de redes *polinomiales*. Ejemplos de ellas son: las redes *functional-link*, que son redes con una sola capa oculta en la que las entradas se procesan con un conjunto de funciones elegidas en función del problema; las *redes neuronales arbóreas* o TNN, que utilizan pequeñas redes MLP en cada nodo de partición para una clasificación binaria que permita extraer las características no lineales; las redes neuronales *Wavelet* o WNN, que son una alternativa a las redes neuronales *feed-forward* para aproximarse de forma arbitraria a funciones no lineales. Y por último, las redes GMDH que serán tratadas de forma individual en el apartado 2.5.5.4.

[68] Véase una aplicación neuronal para series temporales que se comportan con estructuras ARMA(p,q), Hwarng, H.B.; Ang, H.T. (2001). **A simple neural netwok for ARMA(p,q) time series**, *Omega*, 29, pp. 319-333.

[69] TDNN (Waibel, A. (1989). **Modular construction of time-delay neural networks for speech recognition**. *Neural Comput*. 1, pp. 39-46); Functional-Link Networks (Pao, Y.H. (1989).**Adaptive Pattern Recognition and Neural Networks**. *Addison-W*esley); TNN (Sankar A., and Mammone, R.J. (1991). **Neural tree networks. Neural Networks: Theory and Applications**, pp. 281-302, *Academic Press*); WNN (Zhang, Q and Benveniste, A. (1992). **Wavelet networks**. *IEEE Trans. Neural Networks* 3(6), pp. 889-898. Todos los modelos anteriores están referenciados en Chin-Teng Lin y George Lee, C.S. (1996). **Neural Fuzzy Systems. A Neuro-Fuzzy Synergism to Intelligent Systems**, Prentice Hall PTR, Upper Saddle River, NJ. pp. 250-256.

2.5.2. **Modelos** *Single-Layer Feedback o Asociativos* **y Memorias** *Asociativas***.**

En el apartado anterior se han presentado modelos con dos capas (*heteroasociativos*), una de entrada y otra de salida, donde además no se permitían posibles *feedback* entre neuronas, es decir, conexiones hacia atrás. Ahora y aquí definimos un segundo grupo de modelos neuronales que poseen nuevas posibilidades. Algunas de ellas son conexiones laterales, topologías multidimensionales de capas y modelos realimentados, es decir, el flujo de la información se genera entre las neuronas de la única capa[70].

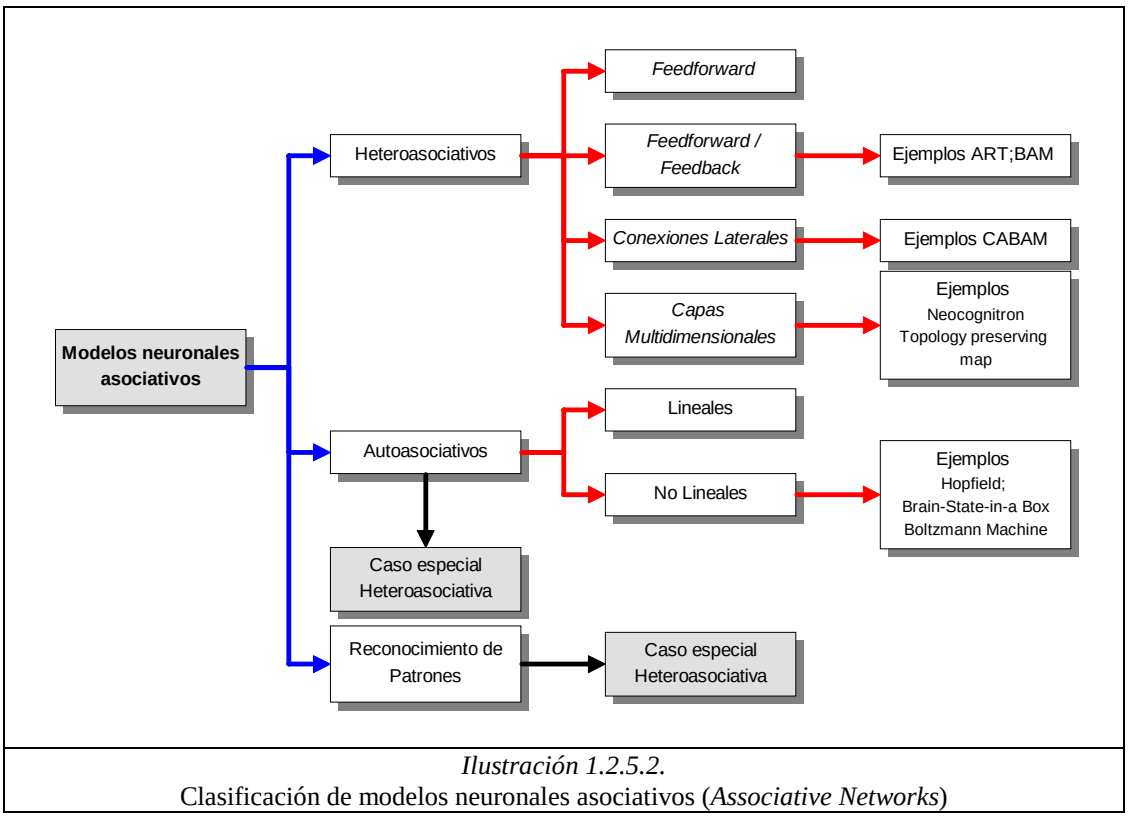


Ilustración 1.2.5.2.
Clasificación de modelos neuronales asociativos (*Associative Networks*)

La ilustración 1.2.5.2. muestra una posible clasificación de los modelos asociativos, que abarca la existencia de tres clases de modelos: los heteroasociativos (ya especificados), los autoasociativos y los modelos de reconocimiento de patrones.

[70] Tal comportamiento genera *sistemas dinámicos* cuya dificultad reside en la estabilidad de los mismos.

Siguiendo a Hilera, J. (1995), la distinción anterior de modelos heteroasociativos y autoasoaciativos, descansa en el tipo de relación existente entre las informaciones de entrada y de salida. Debemos recordar, antes de abordar su diferencia que, cuando un modelo neuronal realiza un proceso de aprendizaje, significa que dicha información retenida está distribuida entre los pesos asociados a las conexiones entre neuronas. Así, si hablamos de un modelo heteroasociativo, el conocimiento retenido será leído a la salida como respuesta a cierta información de entrada, en cambio, si dicha respuesta se produce como salida a la propia entrada, estamos frente el caso autoasociativo[71]. Además tenemos los modelos de reconocimiento de patrones, los cuales se distinguen de los autoasociativos porque cada vector de entrada se asocia un escalar, donde su principal objetivo es identificar el "nombre" de los patrones de entrada o *inputs*. Ambos son casos particulares del modelo heteroasociativo.

Los modelos *autoasociativos*[72] poseen naturaleza de carácter lineal o no lineal. Para el caso lineal, dichos modelos pueden considerarse como una generalización del *perceptrón*, donde es posible utilizar la regla de *Hebb* para su aprendizaje. Dicha regla determina que la intensidad entre la j-ésima neurona de salida y la i-ésima neurona de entrada sea proporcional a la intensidad entre el estímulo y la respuesta deseada. En esencia, el mecanismo descrito es una correlación entre *inputs* y *outputs*, pero posee la siguiente limitación, permite obtener un resultado esperado siempre que los *inputs* sean ortogonales. Para resolver este problema pueden utilizarse otras reglas de aprendizaje, como por ejemplo, la regla de *Windrow-Hoff* que, desde la óptica de la estadística multivariante, es formalmente equivalente a la técnica de componentes principales. Su proceso de implementación necesita de los siguientes conceptos: la descomposición de *valores singulares* de una matriz y el cálculo de *seudoinversa* de una matriz (aproximación de su inversa). La regla de *Windrow-Hoff* puede ser representada matricialmente en términos de los valores propios y vectores propios de la matriz de pesos de las conexiones entre las neuronas, siendo su solución matricial siempre única.

[71] Consisten en asociar una información de entrada con el ejemplar más parecido de los almacenados *conocidos* por el modelo neuronal.

[72] Un caso particular de las mismas es las memorias *autoasociativas*, que permiten asociar patrones de entrada consigo mismo.

De todos modos un modelo autoasociativo lineal siempre generará una salida lineal, lo cual en ciertos casos puede ocasionar problemas[73]. Para solucionarlos existen los modelos no lineales en este contexto, como por ejemplo, el modelo *Hopfield*[74] (véase ilustración 2.2.5.2), el cual puede realizarse tanto en el dominio discreto o continuo, o el modelo *Brain State in a Box* (BSB) Anderson (1977), donde se observa la presencia de realimentación, no presente en el modelo de Hopfield, (véase ilustración 3.2.5.2.).

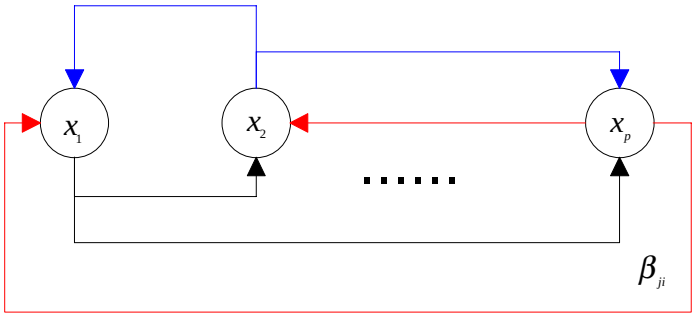


Ilustración 2.2.5.2.
Modelo neuronal *Hopfield*

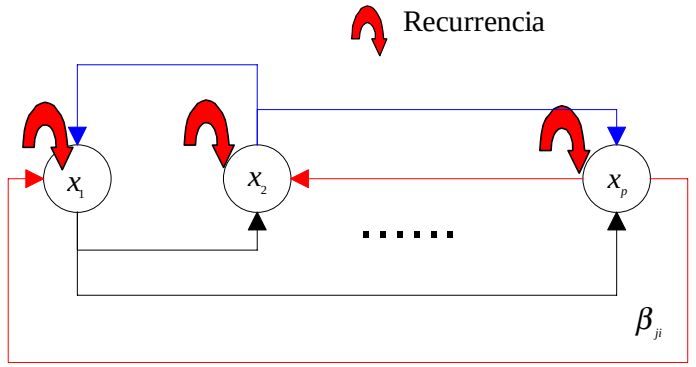


Ilustración 3.2.5.2.
Modelo neuronal *Brain State in a Box* (BSB)

[73] Véase Abdi, Hervé. (1994). **Les réseaux de neurones**, Presses universitaires de Grenoble, Grenoble.
[74] Desarrollado por el físico Hopfield, J.J. (1982) apostando por una modelización muy cercana a la física, específicamente, la *mecánica estadística*. Especialidasd de de la Física que estudia los sistemas que contienen un elevado número de partículas, tantas que, normalmente, no es posible determinar las propiedades globales del sistema evaluando la contribución de cada partícula sino a partir de un comportamiento medio.

Respecto al primero de los modelos autoasociativos no lineales, el modelo *Hopfield* tenemos que, para el caso discreto, las entradas y salidas no supervisadas son binarias. Sus aplicaciones se sitúan es la confección de *memorias asociativas*[75] y la eliminación de distorsiones en las imágenes. En cambio para el caso continuo, su utilidad radica en la resolución de problemas de optimización[76], donde la función de transferencia utilizada acostumbra a ser la tangente hiperbólica. Dicho modelo posee problemas de estabilidad, que son resueltos parcialmente por el teorema de *Cohen-Grossberg* (1983), que permite, bajo ciertas condiciones, garantizar la estabilidad en sistemas no lineales[77].

Otro modelo de interés es la *Boltzmann Machines*[78]. Es un modelo basado en aprendizaje supervisado, de carácter más general que el modelo de Hopfield, cuyo algoritmo de entrenamiento utiliza una analogía entre conceptos de mecánica estadística y conceptos de optimización[79]. El algoritmo de entrenamiento que utiliza el modelo *Boltzmann Machines* es el proceso de templado simulado (*simulated annealing*), el cual permite escapar de mínimos locales de la función de error[80]. El estudio de la dinámica de estos modelos neuronales estocásticos necesitan de la definición de una matriz de probabilidades de transición, denominada matriz de *Markov*, de hecho pueden considerarse como un proceso de *Markov* de primer orden, ya que dichas probabilidades de transición solo poseen memoria del estado anterior, (véase ilustración 4.2.5.2.).

---

[75] El concepto de *memoria asociativa* está muy relacionado con la forma en que el cerebro humano almacena los recuerdos, busca dicha información a partir de alguna pista, información parcial o por contexto. Así los modelos neuronales operan siguiendo este mismo esquema, por ejemplo un modelo MLP implementa una memoria asociativa de tipo *heteroasociativo*, véase Martín del Brío, B. ; Sanz Molina, A. (1997). **Redes Neuronales y sistemas borrosos**, pp. 127-132, Ra-Ma, Madrid.

[76] Por ejemplo, resolver el problema del viajante de comercio o TSP (*Traveling Salesman Problem*).

[77] Véase referenciado en Page M. (2000). **Connectionist modelling in psychology: A localist manifesto**, *Behavioral and Brain Sciences*, 23, pp. 443-512.

[78] Las aplicaciones de dicho modelo son muchas, problemas de codificación, reconocimiento de patrones estadísticos, reconocimiento de palabras y problemas de optimización, véase, Hertz, J.;Krogh, A. Palmer R.G. (1991). **Introduction to the theory of neural computation**, pp. 163-173, Addison-Wesley Publishing.

[79] Véase el diseño de una *Máquina de Boltzmann* para el modelado de funciones de densidad, Albizuri Irigoyen, F.J. (1995). **Máquina de Boltzmann de alto orden: una red neuronal con técnicas de Monte-Carlo para el modelado de Distribuciones de Probabilidad. Caracterización y estructura**, Tesis Doctoral, Universidad del País Vasco.

[80] Véase capítulo 5 de Freeman, J.A. y Skapura, D.M. (1993). **Redes neuronales. Algoritmos, aplicaciones y técnicas de programación,** Addison-Wesley / Díaz de Santos, Madrid.

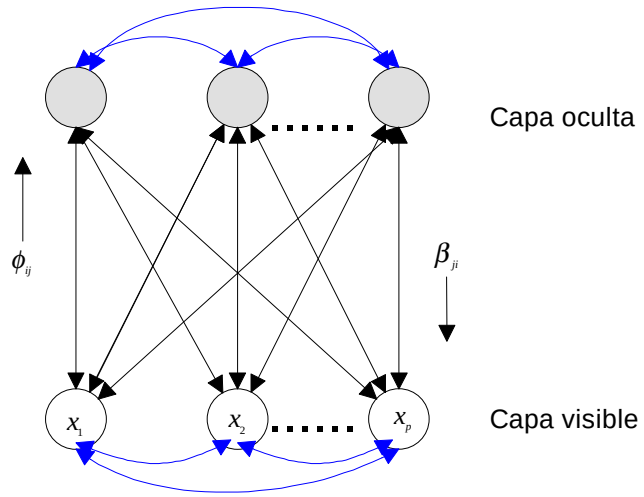


Ilustración 4.2.5.2.
Ejemplo de un modelo neuronal, *Boltzmann machine*

Finalmente, dentro del entorno heteroasociativo, tenemos otros modelos, como por ejemplo, el modelo neuronal *Bidirectional Associative Memory* (BAM), especificado por Bart Kosko (1992). Consiste en un modelo recurrente heteroasociativo con dos capas que puede considerarse una extensión del modelo Hopfield, véase ilustración 5.2.5.2. El análisis de estabilidad del mismo, descansa sobre la evolución temporal de la función de energía y existe una versión multivariante del mismo, el modelo *Multidirectional Associative Memory*. Las variaciones de este modelo dependen de la función de la regla de aprendizaje utilizada. Así, si utilizamos función de correlación, tenemos el modelo, *adaptative Bidirectional Associative Memory* (ABAM), en cambio si incorporamos un aprendizaje competitivo, tenemos el modelo *competitive* ABAM, es decir, CABAM.

Otros de los modelos a tener presente, son los modelos neuronales de resonancia adaptativa (*Adaptive Resonance Theory*) denominados ART, Carpenter y Grossberg (1987), tanto en su versión simple, ART1, como en su versión generalizada, ART2 e incluso en una nueva versión ART3 cuya dinámica es determinada por ecuaciones diferenciales similar a los procesos químicos[81]. Dicha familia de modelos puede considerarse una clase híbrida que

[81] Existen otros modelos derivados del ART modificando el modelo original, como por ejemplo, ARTMAP modelo híbrido con lógica difusa, véase Carperter, G.A.; Grossberg, S. ; Reynols, J.H. (1991). **A Self Organizing ARTMAP Neural Architecture for Supervised Learning and Pattern Recognition**, pp. 43-80, editado por Mammone R.J.; Zeevi, Y.Y. **Neural Networks. Theory and Applications**, Academic Press.

permite generar clusters de secuencias aleatorias[82]. Sus aplicaciones más importantes son entre otras, detección de fallos en circuitos digitales, análisis del mercado de valores, procesos de monotorización, procesamiento de imágenes y palabras, etc., (véase ilustración 6.2.5.2.).

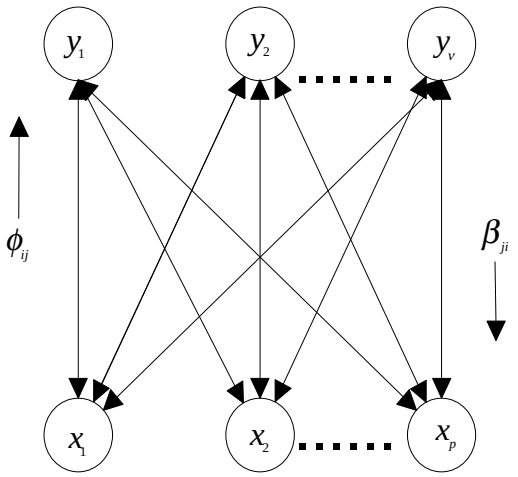


Ilustración 5.2.5.2.
Modelo neuronal *BAM*

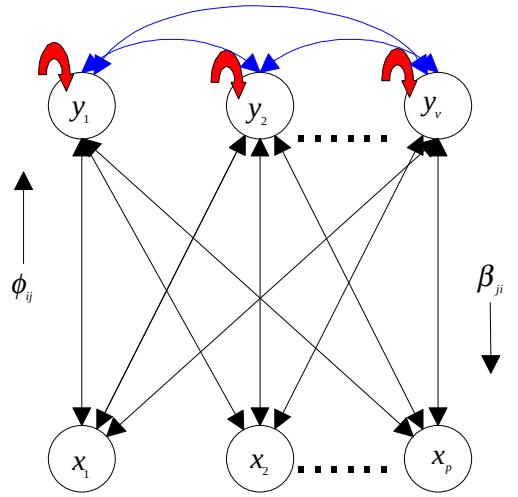


Ilustración 6.2.5.2.
Modelo neuronal *ART1*

[82] Respecto a los *modelos híbridos* sólo se detallarán en el apartado 2.5.5. aquellos más utilizados y que poseen un menor coste computacional.

2.5.3. Modelos neuronales de aprendizaje *no supervisado*.

En apartados anteriores se han descrito los modelos supervisados y los autoasociativos. Aquí presentamos otra tipología, los modelos *autoorganizados*, los cuales se caracterizan porque los patrones de entrada no están asociados a un valor de salida. El concepto de *autoorganización* se refiere a sistemas que estiman los parámetros a partir de las regularidades presentes en los datos. Los modelos no supervisados[83] suelen dividirse en dos grupos, el primero de ellos, los modelos neuronales *no supervisados hebbianos* (apartado 2.5.2.) y el segundo grupo, los modelos *no supervisados competitivos*[84], donde solo una neurona o grupo de ellas puede quedar finalmente activadas. Se les suele definir como modelos WTA (*winner-take-all*).

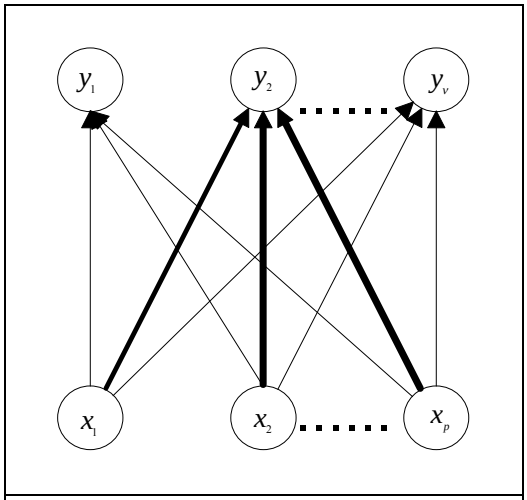


Ilustración 1.2.5.3.
Modelo neuronal WTA

La ilustración 1.2.5.3. muestra el comportamiento de la neurona, "y_2", que mediante un sistema de competencia entre neuronas, es activada frente a las otras al poseer la mayor *similitud* entre el vector de entradas o *inputs* y su propio vector de pesos o conexiones.

[83] Véase Hertz, J.; Krogh, A. Palmer R.G. (1991). **Introduction to the theory of neural computation**, Addison-Wesley Publishing .

[84] Podemos considerar la familia de modelos ART y el *neocognitron*, nombrados en el apartado 2.5.2. como modelos autoorganizados.

Uno de los modelos más conocidos es el de los mapas de rasgos autoorganizados o SOFM (*Self-Organizing Feature Maps*) (Kohonen, 1989, 1990) cuya topología intenta reproducir la siguiente propiedad: las zonas próximas del córtex están asociadas con zonas próximas del cuerpo (*homúnculo*), véase ilustración 2.2.5.3. Es decir, la representación de la información en la corteza cerebral aparece con frecuencia organizada espacialmente, éste aspecto es el que intentan reproducir dichos modelos.

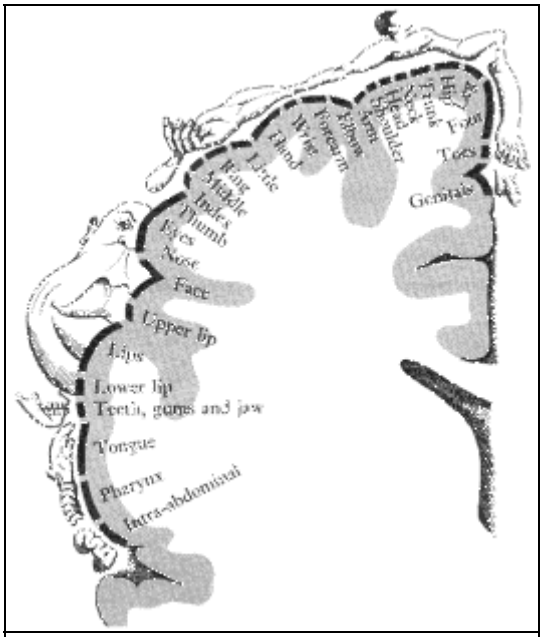


Ilustración 2.2.5.3.
Estructura córtex "*homúnculo*"
Fuente: http://artsci.wustl.edu/~jprinz/cog32.htm

Dicho modelo consiste en una arquitectura de dos capas, la primera de ellas, es la capa de los *inputs* cuya información se distribuye a la segunda capa, la cual realiza un mapa de la información. El *output* final acostumbra a presentarse en un mapa rectangular de neuronas, de dimensión $(n_x; n_y)$. Dicha representación final necesita de un aprendizaje competitivo introduciendo una función de *vecindad*[85] que define un entorno alrededor de la neurona ganadora[86]. La ilustración 3.2.5.3. muestra la estructura de este tipo de modelo neuronal.

[85] La función de vecindad define en cada iteración si una neurona pertenece o no a la vecindad de la neurona vencedora, posee las características de *simetría* y *centrada*. Algunas de las funciones que podemos utilizar son, "sombrero mejicano", gaussiana, rectangular, etc.
[86] Véase el capítulo 3 de Martín del Brío, B. ; Sanz Molina, A. (1997). **Redes Neuronales y sistemas borrosos**, Ra-Ma, Madrid.

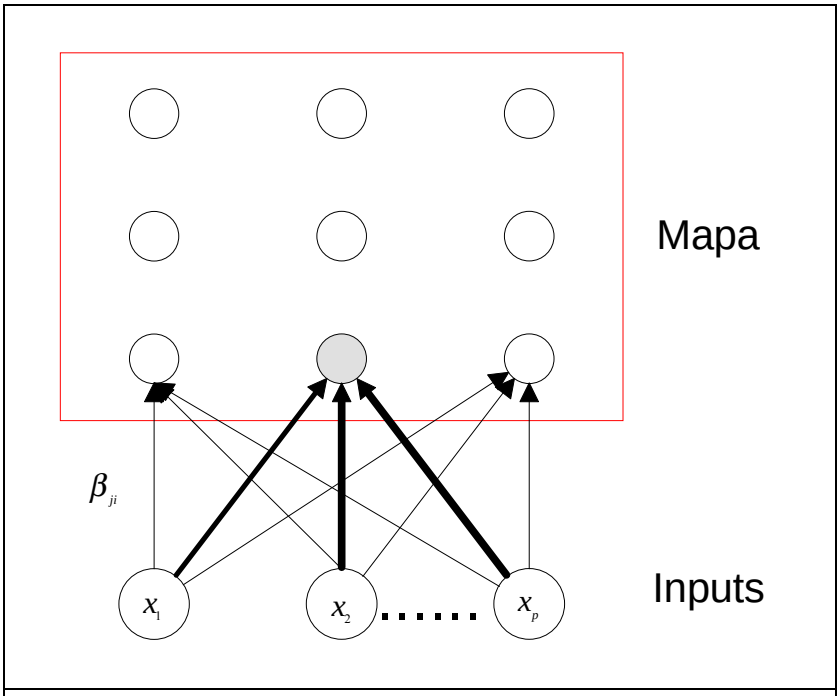


Ilustración 3.2.5.3.
Modelo neuronal *Kohonen Self-Organizing Feature Maps* (SOFM)

En esencia, el modelo permite una proyección no lineal de un espacio multidimensional de entrada, los *inputs*, sobre un espacio discreto de salida, representado por la capa de neuronas ganadoras[87]. A modo de definición, podemos decir que en un mapa autoorganizado los elementos de proceso situados físicamente cerca unos de otros serán los que responderán a los vectores de entrada que estén realmente agrupados en clases que sean similares. El modelo de neurona de Kohonen necesita el cálculo de la "similitud" entre el vector de entradas, *inputs*, y los pesos. Así en función del criterio de distancia escogido obtendremos un modelo u otro. A continuación se expone algunos de los criterios más habituales respecto a estos modelos, véase Martín del Brío (1997), el primero de ellos, la *correlación* o producto escalar,

$$C_{ij} = \sum_{k=1}^{n} \beta_{ijk} x_k ,$$

en segundo lugar, el *coseno* (una vez normalizada la correlación anterior), en tercer lugar, la distancia *euclídea*,

[87] Véase González Pareja, A. (1993). **Redes neuronales artificiales: análisis de un nuevo método de modelización aplicable en economía**, Tesis Doctoral, Universidad de Málaga, Málaga.

$$d^2(\beta_{ij}, x) = \sum_{k=1}^{n} (\beta_{ijk} - x_k)^2$$

y en último lugar, la distancia de *Manhattan*, que utiliza la norma igual a l,

$$d(\beta_{ij}, x) = \sum_{k=1}^{n} |\beta_{ijk} - x_k|$$

El aprendizaje competitivo a que se somete las entradas genera un proceso de reducción de la dimensión de los datos, permitiendo una exploración de los mismos situando al mismo en el ámbito del *Dataminig*. El vector de datos de las entradas representa puntos en el espacio n-dimensional, el modelo neuronal SOFM permite su visualización en dos o tres dimensiones[88] generando un proceso de cluster. Dicho modelo es una red *feedforward* que utiliza un algoritmo de aprendizaje no supervisado basado en la similitud o vecindad. Las aplicaciones generales que permite el modelo SOFM son muy variadas[89], reducción de dimensiones, preprocesado de datos, monitorización de procesos, generación de clusters[90], cuantificación vectorial, modelado de funciones de densidad, etc. En especial en el ámbito financiero[91] han surgido en los últimos años numerosas aplicaciones, como por ejemplo, análisis de estados financieros y modelización de la solvencia empresarial, proyección de la evolución de la estructura de tipos de interés a largo plazo, representación de los fondos de inversión, análisis de quiebras empresariales, representación sectorial, generación de mapas de inversión para los mercados emergentes, generación de sistemas híbridos para el *trading* de mercados, análisis del mercado inmobiliario, detección de comportamientos diferenciales en los consumidores, etc.

---

[88] Los métodos tradicionales que permiten la visualización de datos en espacios de menor dimensión, pueden ser divididos en dos grandes grupos, el primero de ellos, son los métodos de cluster (definiendo o no a priori el número de clusters) y el segundo grupo, los métodos de proyección lineales, como por ejemplo la técnica de *componentes principales* (PCA) y no lineales, como por ejemplo, *multidimensional scaling* (MDS) o los *mapas de Sammon*.

[89] Existen muchas aplicaciones comerciales y de utilización libre que poseen implementado el modelo SOM, véase SOM_PAK, http://www.cis.hut.fi/research/som_lvq_pak.shtml y para reconocimiento de datos textuales, véase http://websom.hut.fi/websom/.

[90] Véase Back, B.; Toivonen, J. Vanharanta, H. Visa, A. (2001). **Comparing numerical data and text information form annual reports using self-organizing maps**, *International Journal of Accounting Information Systems*, 2, pp. 249-269 y Kuo, R.J.; Ho, L.M.; Hu, C.M. (2002). **Cluster analysis in industrial market segmentation through artificial neural networks**, *Computers & Industrial Engineering*, Vol. 42, 2-4, pp. 391-399.

[91] Véase de forma general Deboeck, G,; Kohonen, T. (1998). **Visual Explorations in Finance with Self-Organizing Maps**, Springer. En particular, véase, Serrano-Cinca, C. (1996). **Self organizing neural networks for financial diagnosis,** *Decision Support Systems*, Vol. 17, No. 3, pp. 227-238; Martin-del-Brio, B. y Serrano-Cinca, C. (1993). **Self-organizing Neural Networks for the Analysis and Representation of Data: Some Financial Cases,** *Neural Comput & Applic*, 1, pp. 193-206 y el capítulo 11 de Zirilli, J.S, (1997). **Financial Prediction using Neural Networks**, International Thomson Computer Press.

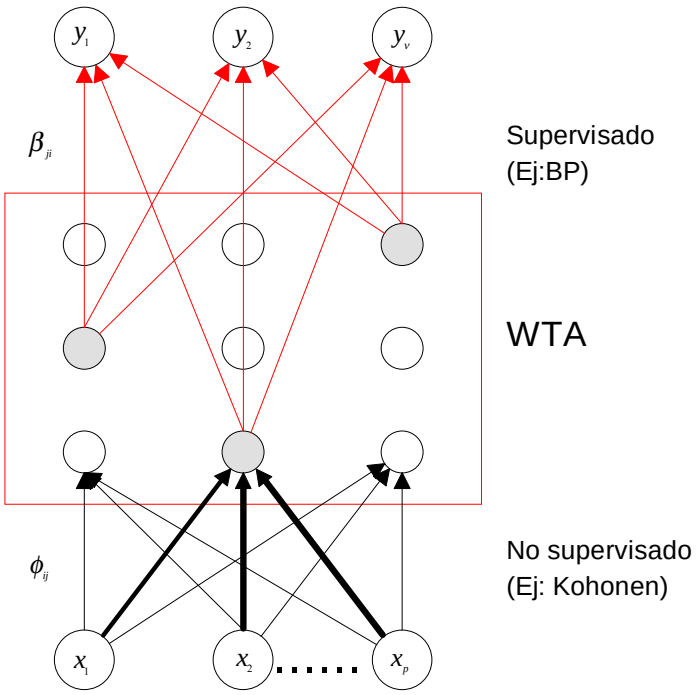


Ilustración 4.2.5.3.
Modelo neuronal *counter-propagation* (CPN)

Existen variantes de los SOFM clásico que incorporan mejoras en su funcionamiento, por ejemplo, el modelo autoorganizado VQP (*Vector Quantization and Projection*), que parte de un mapa de reducidas dimensiones y de forma iterativa se incrementa el número de neuronas, buscando la mejor representación. También, el modelo contrapropagación[92], (*counter-propagation* (CPN)) que consiste en una red híbrida (Hecht-Nielsen (1990)), donde en una primera fase, se aplica el carácter no supervisado (de la capa *inputs* a la capa oculta) y posteriormente, de la capa oculta a la de salida, el supervisado (véase ilustración 4.2.5.3.). En último lugar, tenemos el modelo LVQ (*Learning Vector Quantization*), quien mediante la modificación de sus pesos en sentido contrario, premia aquellas neuronas que clasifican mejor un determinado patrón y penaliza a las que realizan el proceso de clasificación de forma errónea.

[92] Véase referenciado en Chin-Teng Lin y George Lee, C.S. (1996). **Neural Fuzzy Systems. A Neuro-Fuzzy Synergism to Intelligent Systems**, pp. 326-328, Prentice Hall PTR. Respecto a las posibles aplicaciones véase, Juren Zupan; Marjana Novič; Itziar Ruisanchéz. (1997). **Kohonen and counterpropagation artificial neural networks in analytical chemistry**, *Chemometrics and Intelligent Laboratory Systems*, 38, pp. 1-23.

2.5.4. Modelos Neuronales Recurrentes (*Recurrent neural networks*).

Las redes neuronales *recurrentes* también definidas como, *feedback network*, permiten conexiones de las neuronas consigo mismas y con neuronas en capas inferiores, dotando a las mismas de una dinámica de comportamiento que no es posible ser replicada con las redes *feed-forward*.

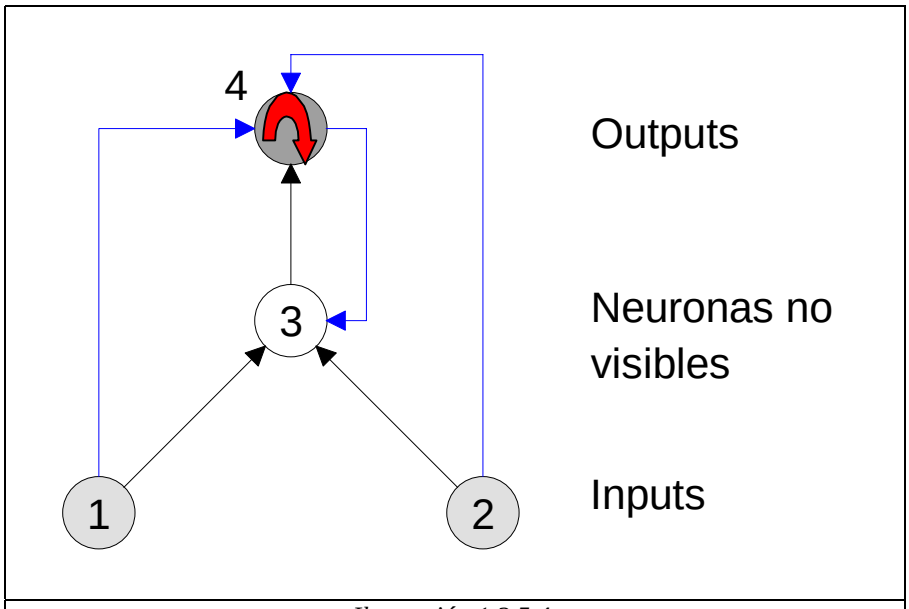


Ilustración 1.2.5.4.
Ejemplo de *Recurrent Back Propagation* (RBP) con dos inputs.

Los modelos neuronales recurrentes son muy adecuados para entender el comportamiento de las series espacio-temporales[93], véase ilustración 1.2.5.4. Poseen un número importante de aplicaciones, desde memorias asociativas, clasificación de patrones espacio-temporales, optimización, control, predicción y generación de secuencias[94].

[93] Véase capítulos 8 y 9 de Bose, N.K.; Liang, P. (1996). **Neural Networks Fundamentals with Graphs, Algorithms, and Applications**, McGraw-Hill Series in Electrical and Computer Engineering.

[94] De hecho pueden ser equivalentes a autómata finito, que debe aprender una secuencia temporal de patrones, de tal forma que durante su funcionamiento, dado un patrón de entrada, es capaz de generar como salida los siguientes de las series que tiene almacenada, véase Hilera, J.R. y Martínez, V.J. (1995). **Redes neuronales artificiales. Fundamentos, modelos y aplicaciones**, pp. 350, Ra-Ma, Madrid.

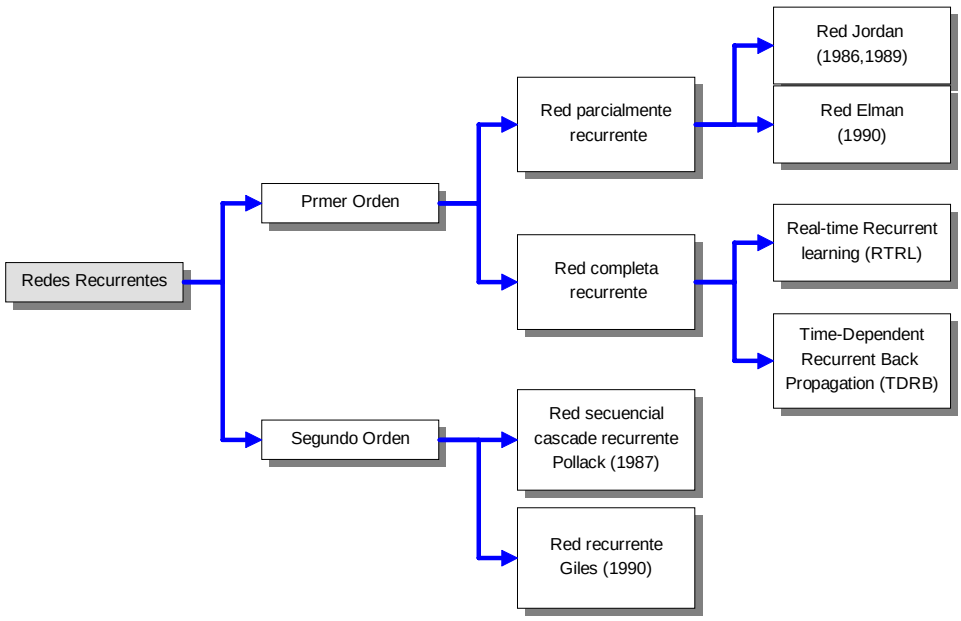


Ilustración 2.2.5.4.

Fuente: Chin-Teng Lin y George Lee, C.S. (1996).
Neural Fuzzy Systems. A Neuro-Fuzzy Synergism to Intelligent Systems,
Prentice Hall PTR y elaboración propia.

La ilustración 2.2.5.4. representa una posible clasificación de los diferentes modelos neuronales recurrentes. Distinguimos entre redes de primer orden, es decir, cuando el *output* que deseamos conseguir en el proceso de aprendizaje está definido y es conocido y las redes de segundo orden, donde no se posee dicha información. A su vez las redes de primer orden pueden subdividirse en *redes parcialmente recurrentes* y *redes recurrentes completas*.

Las redes parcialmente recurrentes pueden memorizar secuencias a partir del pasado, pero no comprenden la posible estructura interna, de forma que, en la mayoría de los casos los *feedback* están predefinidos en la propia topología. Por esta razón podemos afirmar que son redes *multilayer feed-forward*[95] donde se permite la presencia de *feedback*[96].

[95] Cuando ninguna salida de las neuronas es entrada de neuronas del mismo nivel o de niveles precedentes, el modelo es *feed-forward* o propagación hacia delante, en cambio, cuando las salidas pueden ser conectadas como entradas de neuronas de niveles previos o del mismo nivel, incluyéndose ellas mismas, el modelo se define como *feedback*, es decir, propagación hacia atrás.

[96] Es decir, pueden ir de las capas ocultas o capa de salida a la capa de contexto, capa oculta o la propia capa de las entradas. Recordamos que la capa de *contexto* o neuronas de contexto, son las unidades que reciben las señales de propagación hacia atrás.

Las redes parcialmente recurrentes, a su vez poseen dos tipos de modelos[97], el modelo de red secuencial de Jordan (1986, 1989) y la red recurrente simple de Elman[98] (1990). Este tipo de redes posee algunas propiedades que las hacen especialmente adecuadas para la modelización de series temporales. Así, desde la perspectiva econométrica, la estructura neuronal adoptada aquí es una clase particular de modelo dinámico con variables latentes. Dicha representación es ventajosa porque vincula las redes recurrentes a una clase de modelos conocidos en nuestro ámbito como son los modelos de espacio de estados.

Respecto a las otras topologías, es decir, las redes recurrentes completas y algunos de sus posibles modelos, tenemos los modelos RTRL ("*Real-Time Recurrent Learning*") y TDRB ("*Time-Dependent Recurrent Back Propagation*") que son utilizados en aplicaciones *on-line*, muy habituales en el ámbito industrial. En último lugar, están las redes recurrentes de segundo orden, que no poseen de forma explícita un *output* objetivo en el proceso de aprendizaje y son utilizadas para el aprendizaje regular de lenguajes, (véase ilustración 3.2.5.4.).

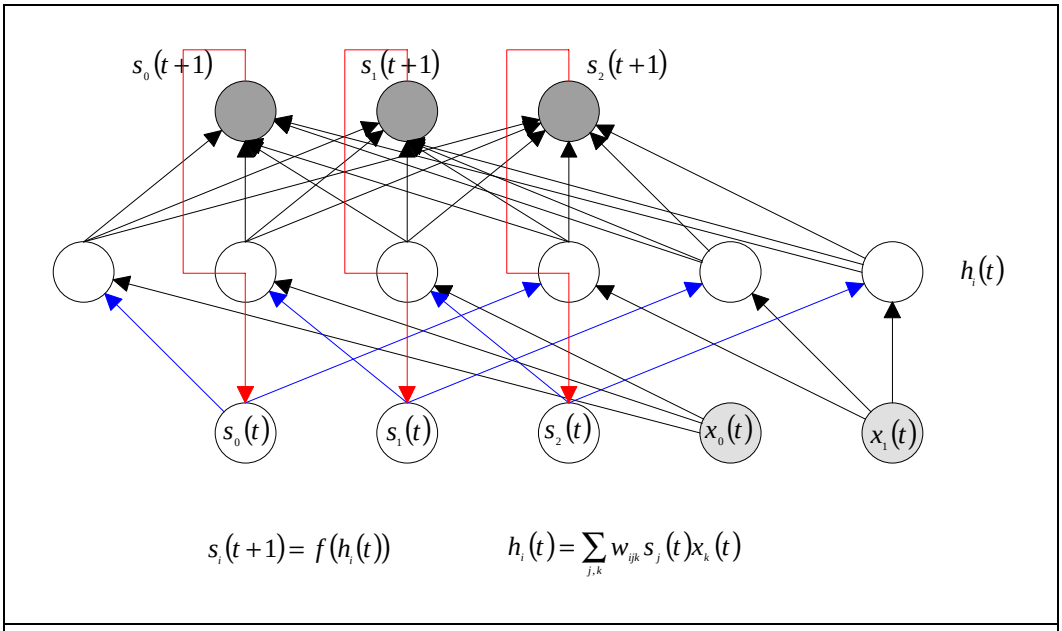


Ilustración 3.2.5.4.
Ejemplo de *red recurrente* de segundo orden

[97] Véase referenciado en Chin-Teng Lin y George Lee, C.S. (1996). **Neural Fuzzy Systems. A Neuro-Fuzzy Synergism to Intelligent Systems**, pp. 346-348, Prentice Hall PTR.
[98] Ambas tipologías se especificarán en el capítulo 3.4.3.

Una de las áreas con mayor número de aplicaciones es la *predicción*, en éste ámbito las redes neuronales poseen ventajas de consideración, en especial las redes recurrentes[99]. Principalmente, no necesita realizar a priori hipótesis sobre el comportamiento del modelo, además, generaliza aún existiendo mucho ruido en los datos utilizados. Y por último, son modelos no lineales, así en este contexto podría afirmarse que la aproximación tradicional a la predicción de series temporales mediante modelos ARIMA es un caso particular por descansar en la presunción lineal del proceso[100]. Aunque no exenta de críticas, el uso de las redes neuronales artificiales se ha popularizado en los últimos años[101], dada la mayor comprensión de estos modelos, que permiten representar automáticamente complejas relaciones no lineales[102].

Sus aplicaciones en el ámbito de la predicción son muy variadas, de especial relevancia son las contribuciones de las aplicaciones financieras de los modelos neuronales artificiales[103], Tripi y Turban (1993), Azoff (1994), Refenes (1995), Gately (1996).

---

[99] Véase un ejemplo aplicado a la ETTI española en Pérez-Rodríguez, J.V.; Torra Porras, S.; Borrell Vidal, M. (1998). **Tipos de interés a corto plazo, volatilidad condicional y redes neuronales**, *Documento de trabajo*, 39/98 Universidad de las Palmas de Gran Canaria. Departamento de Economía Aplicada y Pérez-Rodríguez, J.V. y Torra Porras, S.; Borrell Vidal, M. (2000). **Modelos para la predicción de los tipos de interés en el mercado interbancario: estructuras lineales, GARCH y redes neuronales artificiales**, *Revista Asturiana de Economía*, 18, pp. 123-139. Pero existen otros muchos campos de aplicación como por ejemplo la geociencia, véase Tiira, T. (1999). **Detecting teleseismic events using artificial neural networks**, *Computers & Geosciences*, 25, pp. 929-938, donde realiza una comparativa entre redes MLP, Jordan y Elman.

[100] Existe un fuerte esfuerzo en la modelización no lineal de series temporales, véase Zhang, G.P. (2001). **An investigation of neural networks for linear time-series forecasting**, *Computers & Operations Research*, 28, pp. 1183-1202; Zhang, G.P.; Patuwo, B.E.; Hu, M.Y. (2001). **A simulation study of artificial neural networks for nonlinear time-series forecasting**, *Computers & Operations Research*, 28, pp. 381-396; Donaldson, R.G. y Kamstra, M. (1996). **Forecast Combining with Neural Networks**, *Journal of Forecasting*, Vol. 15, pp. 49-61; Gençay, R.; Lui, T. (1997). **Nonlinear modelling and prediction with feedforward and recurrent networks**, *Physica* D, 108, pp. 119-134 y Swanson, N.R.; White, H. (1997). **Forecasting economic time series using flexible versus fixed specification and linear versus nonlinear econometric models**, *International Journal of Forecasting*, 13, pp. 439-461.

[101] Véase Olmeda, I.; Romero, S.B. (1993). **Redes Neuronales Artificiales. Fundamentos y Aplicaciones**, Universidad de Alcalá, servicio de publicaciones, Madrid.

[102] Véase para el diseño de modelos neuronales globales y locales orientados a la predicción de series temporales, Huerta Rico, R. (1994). **Algoritmos para la predicción de series temporales basados en modelos deterministas de baja dimensión**, Tesis Doctoral, Universidad Autónoma de Madrid, Madrid.

[103] En general, casi todos los trabajos analizan la capacidad *predictiva* de las redes comparando diversos modelos, tanto lineales como no lineales, véase Ho, S.L:; Xie, M.; Goh, T.N. (2002). **A comparative study of neural network and Box-Jenkins in time series prediction**, *Computers & Industrial Engineering*, Vol. 42, 2-4, pp. 371-375.

2.5.5. Modelos *Híbridos*.

2.5.5.1. Redes neuronales *Radial Basis Function* (RBF).

Las redes de *función de base radial* (RBF) son en este momento uno de los modelos más utilizados debido a su rápida formación, generalidad y simplicidad. Se trata de un modelo híbrido, ya que incorpora tanto el aprendizaje supervisado como el no supervisado. La parte no supervisada se sitúa desde el vector de *inputs* a la capa oculta, donde debe decidirse el número de nodos radiales que cubran buena parte del espacio definido por los *inputs* o entradas y la parte supervisa desde la capa oculta y la capa del *output*. Los métodos *Radial Basis* (RBF) normalmente se comparan con los modelos *feed-forward*[104] (MLP), pero aún siendo los más utilizados, poseen serios problemas que en la mayoría de los casos los modelos RBF solucionan, como por ejemplo la rapidez en el aprendizaje o proceso de estimación[105]. La arquitectura de un modelo neuronal con funciones de base radial o RBF posee tres capas, de entrada, oculta y capa de salida, muy parecida al modelo MLP[106] ya estudiado en el apartado 2.5.1. y cuya expresión es,

$$y = \beta_0 + \sum_{j=1}^{q} \beta_j g_j(x_i) + \varepsilon$$

$$g_j(x_i) = \exp\left(- \frac{\sum_{j=1}^{q} (x_i - \mu_j)^2}{2\sigma_j^2} \right)$$

$$X = (x_1, x_2, \cdots, x_p)'$$

$$\{ \beta_j, j = 0,1,\cdots,q \}$$

[104] Véase un estudio comparativo aplicado al ámbito industrial, West, D.A.; Mangiameli, P.M.; Chen, S.K. (1999). **Control of complex manufacturing proceses: a comparison of SPC methods with a radial basis fuction neural network**, *Omega, Int. J. Mgmt. Sci.* 27, pp. 349-362.

[105] Dichos modelos son *aproximadores universales*, (Girosi y Poggio (1990), Hartman y Keeler (1990)), es decir, dada una red con suficientes neuronas en la capa escondida, pueden aproximar cualquier función continua con una exactitud arbitraria. Esta es una propiedad que comparten con otras redes *feed-forward* que posean una capa escondida de neuronas no lineales. Stinchcombe y White (1989) han probado que la no linealidad no necesita ser sigmoidal y puede ser cualquiera de una amplia gama de funciones. Referenciado en Wasserman, Philip D. (1993). **Advanced Methods in Neural Computing**, pp. 147-155, Van Nostrand ReinHold.

[106] La diferencia esencial está en la operación que se produce en la capa oculta. Así en el caso MLP se calcula la suma ponderada de los *inputs* y posteriormente se le aplica una función de transferencia, en cambio en los modelos RBF, opera sobre la base de la distancia que separa el vector *inputs* respecto del vector *centroide* almacenado en cada neurona oculta (de la misma forma que el modelo no supervisado de Kohonen). Posteriormente al resultado anterior se le aplica una *función radial*, véase Martín del Brío, B ; Sanz Molina, A. (1997). **Redes Neuronales y sistemas borrosos**, pp. 152-158, Ra-Ma, Madrid.

donde, la función $g_j(x_i)$ es una de las posibles funciones radiales, la función gaussiana[107]. Pero existen otras muchas posibilidades, como por ejemplo,

$$g(x_i)=\left(\sum_{j=1}^{q}(x_i-\mu)^2+\sigma^2\right)^{-\alpha},\quad \alpha>0$$

que posee las mismas cualidades de localización. Existen otras funciones que no cumplan dicha propiedad[108], como por ejemplo,

$$g(x_i)=\sum_{j=1}^{q}(x_i-\mu)^2\cdot\ln\left(\sum_{j=1}^{q}(x_i-\mu)^2\right)$$

$$g(x_i)=\left(\sum_{j=1}^{q}(x_i-\mu)^2+\sigma^2\right)^{\beta},\quad 0<\beta<1$$

caso cúbico,

$$g(x_i)=\sum_{j=1}^{q}(x_i-\mu)^3$$

y finalmente la función lineal,

$$g(x_i)=\sum_{j=1}^{q}(x_i-\mu)$$

Podemos comprobar que en todos los casos, previamente al diseño de la función radial, existe la definición de una distancia estadística, habitualmente la euclídea, que separa el vector de *inputs* y el centroide. Así, si el vector de *inputs* coincide con el centroide de la j-neurona oculta, ésta responde con el máximo valor, es decir, la unidad[109]. El parámetro, "σ" o factor de escala mide la anchura de la función *gaussiana* y se puede definir como el área de influencia[110] de la neurona en el espacio de los *inputs*. A mayor valor, la región que la neurona oculta domina, cerca del centroide es más amplia[111], (véase ilustración 1.2.5.5.1.).

---

[107] Las funciones localizadas, como la *Gaussiana*, cumplen, $g(x)\to 0$ cuando $|x|\to\infty$.

[108] Véase para mayor detalle, Bishop, C.M. (1995). **Neural Networks for Pattern Recognition**, pp. 165-192, Clarendon Press – Oxford.

[109] Es decir, si los *inputs* están cerca del centroide, la neurona se activa y reconoce el patrón de entrada, si el patrón de entrada es muy diferente, la salida de la capa oculta tenderá a cero.

[110] Hay que notar la diferencia existente entre el modelo neuronal RBF y el propio funcionamiento de una neurona biológico en este aspecto. La neurona biológica determina el área de influencia mediante la conectividad de la misma, es decir, la distancia existente entre neuronas adyacentes conectadas por sus *dentritas*. En cambio la neurona artificial determina su campo de acción a través de la determinación del tipo de *función radial*.

[111] En algunos casos pueden escogerse factores de escala diferentes, adquiriendo formas *elipsoidales.*

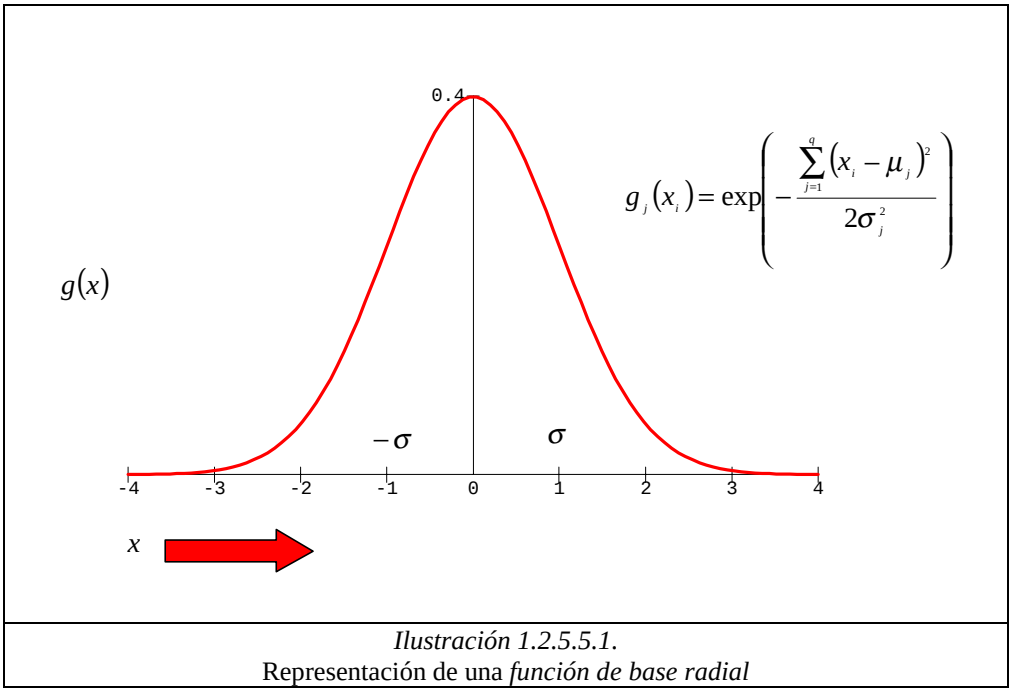


Ilustración 1.2.5.5.1.
Representación de una *función de base radial*

La elección del número de nodos radiales, es decir, la parte no supervisada, debe tener en cuenta el compromiso entre el error que se comete y su capacidad de generalización. Existen procedimientos para automatizar dicho aspecto, como por ejemplo, el *algoritmo autoorganizado jerárquico* que evalúa la necesidad de incrementar el número de nodos en función de su nivel de activación[112]. El proceso habitual es un aprendizaje por etapas, en la primera de ellas, se determinan el número de nodos y el cálculo de los *centroides*[113]. En la segunda etapa, se calculará el parámetro de escala[114],"σ_j", donde su estimación se suele realizar por métodos heurísticos. Finalmente, en la última etapa y mediante el aprendizaje supervisado se entrenará el modelo en la capa de salida. Este carácter híbrido consigue acelerar notablemente el entrenamiento respecto al modelo neuronal MLP, (véase ilustración 2.2.5.5.1.).

[112] Véase Martín del Brío, B. ; Sanz Molina, A. (1997). **Redes Neuronales y sistemas borrosos**, pp. 155, Ra-Ma, Madrid.
[113] Pueden utilizarse cualquier algoritmo no supervisado de *clustering*, como por ejemplo, el *algoritmo k-means* o algoritmos neuronales tipo *Kohonen*, existiendo un gran paralelismo entre ellos.
[114] El objetivo del factor de escala es cubrir el espacio multidimensional de los *inputs* con áreas de influencia lo más uniforme posibles. Si el espacio entre los centros no es uniforme, podría ser necesario que cada neurona oculta tuviera su propio valor de "σ_j".

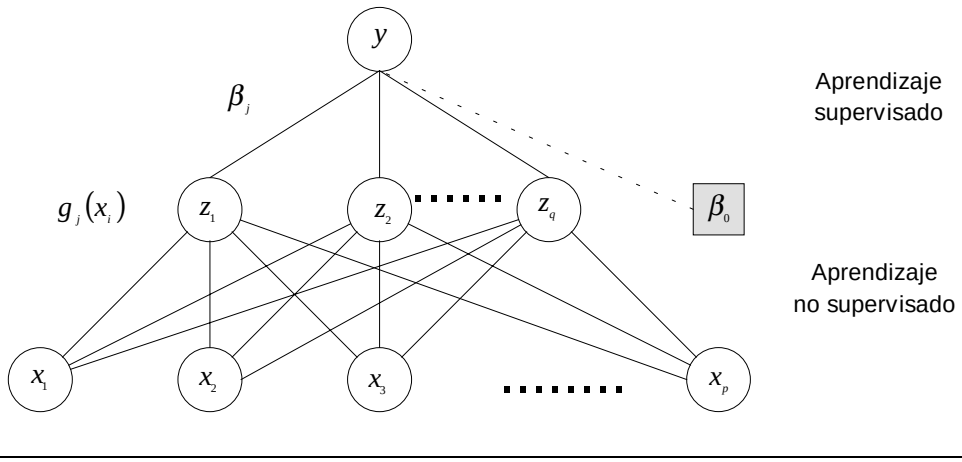


Ilustración 2.2.5.5.1.
Modelo neuronal *RBF*

Existen algunos aspectos relevantes de dichos modelos que son de interés. Primeramente, las relaciones existentes entre los modelos RBF y otras formas de aproximar funciones, como por ejemplo, la *regresión de kernel*[115]. En segundo lugar, la aplicación de los mismos a tareas de clasificación, muy próximos a los modelos probabilísticos neuronales. En tercer lugar, la aplicación de técnicas de mínimos cuadrados ortogonales para la selección de los *centroides* de las áreas de influencia y finalmente, las funciones radiales pueden ser especificadas como las componentes de modelos mixtos de densidad, cuyos parámetros son optimizados por máxima verosimilitud[116]. La ilustración 3.2.5.5.1. muestra la topología de un modelo neuronal RBF con más de un *output*, cuya expresión es la siguiente,

$$y_t = \beta_{0t} + \sum_{j=1}^{q} \beta_{jt} g_j(x_i) + \varepsilon$$

$$g_j(x_i) = \exp\left(-\frac{\sum_{j=1}^{q} (x_i - \mu_j)^2}{2\sigma_j^2} \right)$$

$$X = (x_1, x_2, \cdots, x_p)' \{ i = 1, \cdots, p \}$$

$$Y = (y_1, y_2, \cdots, y_r)'$$

$$\{ \beta_{jt}, j = 0,1, \cdots, q; t = 1, \cdots, r \}$$

[115] Véase para un amplio desarrollo el capítulo 6 de Hastie, T; Tibshirani, R.; Friedman, J. (2001). **The Elements of Statistical Learning. Data Mining, Inference, and Prediction**, Springer.

[116] Véase Bishop, C.M. (1995). **Neural Networks for Pattern Recognition**, pp. 177-179, Clarendon Press – Oxford y para aplicaciones en el entorno financiero, Wolberg, J.R. (2000). **Expert Trading Systems. Modeling Financial Markets with Kernel Regression**, Wiley Trading.

Si estos valores son posteriormente normalizados de la siguiente forma,

$$y_t = \frac{\sum_{j=1}^{q} g_j(x_i)\beta_{jt}}{\sum_{j=1}^{q} g_j(x_i)}$$

dicha topología representa un modelo general de regresión neuronal (GRNN) que será detallado en el apartado 2.5.5.3.

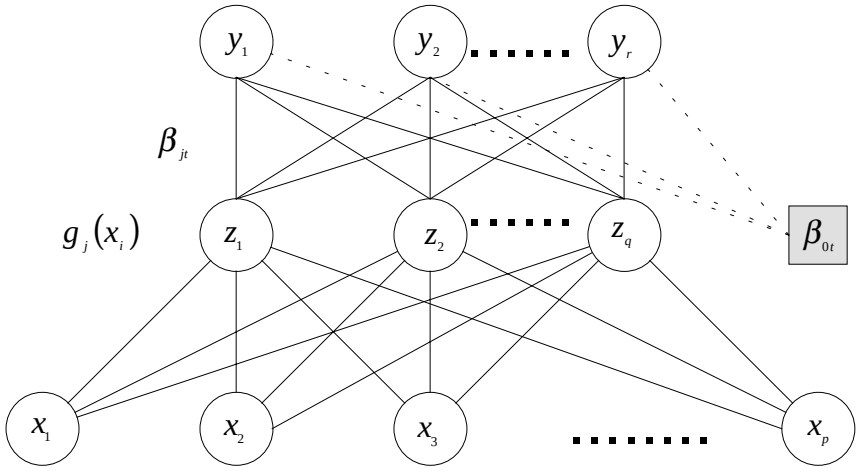


Ilustración 3.2.5.5.1.
Modelo neuronal *RBF* con más de un output

2.5.5.2. Redes neuronales *Probabilísticas* (PNN).

Los modelos neuronales probabilísticos (*probabilistic neural networks*, PNN) surgen en los años 70 y fueron descritos por Meisel[117] (1972). Pero no empezaron a tener importancia hasta los años 90 con Specht. Dichos modelos aparecen de forma natural en el marco de la teoría de la probabilidad y son en esencia un clasificador *bayesiano*. Desde esta óptica y con un perfil clásico, los algoritmos de clasificación asumen el conocimiento de la distribución de las variables aleatorias que se utilizan para clasificar. Pero en la mente de todos está la idea de la problemática que ocasiona en los análisis, la existencia de mayores desviaciones o la aparición de distribuciones multimodales. Con el objetivo de clasificar también pueden ser utilizados los modelos *feed-forward* (comentadas en apartados anteriores, MLP), aunque adolecen de ciertos problemas. Según Master (1995), la primera de las dificultades radica en su forma de operar y la segunda, en el desconocimiento de cual es el comportamiento teórico esperado.

Las PNN poseen por su parte tres limitaciones. Su lentitud en clasificar, un alto coste computacional y una elevada necesidad de memoria de sistema. Pero a su favor juega su alta resistencia y estabilidad frente la presencia de *outliers*. Los modelos probabilísticos, tal y como se ha comentado, utilizan la metodología de *Bayes*[118] para su proceso de clasificación[119], pero de forma habitual se desconocen las funciones de densidad necesarias para su aplicación. En este entorno los métodos de estimación existentes son los siguientes. El primero de ellos es el *paramétrico*, donde se parte de la definición de una forma funcional específica que posee un número limitado de parámetros ajustables.

---

[117] Véase Meisel, W. (1972). **Computer-Oriented Approaches to Pattern recognition**, Academic Press, referenciado en Masters, T. (1995). **Advanced Algorithms for Neural Networks. A C++ Sourcebook**, Wiley.

[118] Los métodos más comunes de estimación de estos parámetros son, el método de *máxima verosimilitud* y el método *Bayesiano*. Ambos nos conducen aproximadamente a los mismos resultados pero su concepción es muy diferente. El primero de ellos obtiene los valores óptimos de los parámetros maximizando una función de verosimilitud (obtenida a partir de los datos observados), en cambio en el segundo método, los parámetros son definidos en primera instancia sin observar los datos, probabilidad a priori; una vez observados el aprendizaje *bayesiano* utiliza probabilidades a posteriori. Debemos recordar que el método utilizado por Multilayer Perceptron (MLP) impone *equiprobabilidad* de las clases.

[119] Una de las aplicaciones más frecuente es modelizar el riesgo de quiebra, véase Yang, Z.R.; Platt, M.B.; Platt, H.D. (1999). **Probabilistic Neural Networks in Bankruptcy Prediction**, *Journal of Business Research*, 44, pp. 67-74.

En segundo lugar, existe otra forma de aproximación al problema, los denominados métodos secuenciales de estimación basados en técnicas iterativas, como por ejemplo, los procedimientos de aproximación estocástica o de Robbins-Monro[120] (1951). Y en último lugar, los métodos *no paramétricos*, que no especifican a priori la forma funcional de la función de densidad. Ésta se obtiene de los propios datos observados y existen varias aproximaciones para poder estimar la función de densidad de forma no paramétrica. Por ejemplo, el método del *histograma*, que en esencia es un método semiparamétrico consistente en representar la función de densidad desconocida mediante un "histograma", aumentando progresivamente el número de intervalos definidos, los métodos basados en *Kernel*[121] y en tercer lugar, el método de *k-nearest-neighbours*[122].

Todos los métodos anteriores no paraméricos necesitan la definición de un parámetro de gran importancia que dirige el proceso de adaptación a los datos, denominado *smoothing parameter*. Ahora bien, existen métodos que combinan las ventajas de los métodos paramétricos y no paramétricos, los llamados modelos *mixtura*[123], cuyo modelo más conocido es el modelo mixtura *Gaussiano*[124], muy cercano a los modelos RBF del apartado 2.5.5.1 y posee aplicaciones en el campo actuarial.

[120] Véase Bishop, C.M. (1995). **Neural Networks for Pattern Recognition**, pp. 46-50, Clarendon Press – Oxford.

[121] El método *Parzen* de estimación de funciones de densidad univariantes a partir de una muestra aleatoria utiliza una función que actúa como ponderación denominada función potencial o simplemente *Kernel*. La función kernel es conocida también como ventana de Parzen, véase para mayor detalle, Masters, T. (1995). **Advanced Algorithms for Neural Networks. A C++ Sourcebook**, pp. 157-171, Wiley.

[122] Existen versiones del modelo de regresión que también se utilizan para estimar funciones de densidad, el *modelo de regresión localmente ponderada* o *Moving Polynomial Regresión* (MPR), véase Lejeune, M.; Sarda, P. (1992). **Smooth estimators of distribution and density funcions**, *Computational Statistics & Data Análisis*, 14, pp. 457-471.

[123] Véase capítulo 6.8. de Hastie, T; Tibshirani, R.; Friedman, J. (2001). **The Elements of Statistical Learning. Data Mining, Inference, and Prediction**, Springer.

[124] Podemos estar frente a mixturas Gaussianas con la misma dispersión o con diferente, es decir, heterocedástica, véase Rong Yang, Z.; Chen, S. (1998). **Robust maximum likelihood training of heterocedastic probabilistic neural networks**, *Neural Networks*, 11, pp. 739-747.

El teorema de *Bayes*, que es parte fundamental del desarrollo de los modelos neuronales *probabilísticos*, posee la siguiente expresión[125],

$$P\left(y^i \middle| X\right) = \frac{P\left(X \middle| y^i\right) P\left(y^i\right)}{P\left(X\right)}$$

siendo,

$P\left(X \middle| y^i\right)$: probabilidad de que el vector de entrada X esté en la clase y^i,

$P\left(y^i\right)$: probabilidad que se produzca la clase "i",

$P\left(X\right)$: probabilidad de un vector de entrada con valor X,

$P\left(y^i \middle| X\right)$: probabilidad a posteriori de y^i.

En los modelos PNN el cálculo de la probabilidad condicionada, $P\left(X \middle| y^i\right)$, se aproxima mediante las ventanas *Parzen*, normalmente utilizando una función exponencial.

$$P\left(X \middle| y^i\right) = \left(\frac{1}{n^i \left(2\pi^{(p/2)} \sigma^p\right)}\right) \sum_{j=1}^{n^i} e^{\left(-D_j^{i2}/\left(2\sigma^2\right)\right)}$$

siendo,

$D_j^{i2} = \sum_{j=1}^{q^i} \left(x_i - \mu_j\right)^2$: distancia *Euclídea* al cuadrado en cada una de las i-clases definidas[126],

$k = \left(\frac{1}{\left(2\pi^{(p/2)} \sigma^p\right)}\right)$: factor de escala,

n^i: número de neuronas en la capa oculta para cada una de las i-clases definida siendo, n el total de las mismas,

p: número de *inputs* en el vector de entrada, $X = \left(x_1, x_2, \cdots, x_p\right)'$

[125] Véase un mayor desarrollo de modelos neuronales orientados a generar verosimilitudes en, Husmeier, D. (1999). **Neural networks for Conditional Probability Estimation. Forecasting Beyond Point Predictions**, Springer y para la especificación de redes neuronales bayesianas, Finn V. Jensen. (1996). **An introduction to Bayesian networks**, UCL Press.

[126] Dicha expresión es muy similar a la definida para el modelo neuronal *RBF*, apartado 2.5.5.1., excepto en los siguientes aspectos: en primer lugar, por el factor de escala,"k", en segundo lugar, las neuronas de la capa oculta se dividen en el número de clase predefinidas y en tercer lugar, todas las ponderaciones entre capas son unitarias.

Además, la probabilidad $P(y^i)=(n^i/n)$ es estimada como la frecuencia relativa del número de neuronas en la capa oculta segmentada para las i-clases. Si ahora aplicamos el teorema de Bayes para calcular la probabilidad condicional de " X " para cada i-clase tenemos,

$$P(X)=\sum_{i=1}^{n} P(X|y^i)P(y^i)$$

$$\downarrow$$

$$P(X)=\sum_{i=1}^{n}\left(\frac{1}{n^i(2\pi^{(p/2)}\sigma^p)}\right)\sum_{j=1}^{n^i} e^{\left(-D_i^2/(2\sigma^2)(n^i/n)\right)}$$

sustituyendo las expresiones anteriores en el teorema de *Bayes*, tenemos la probabilidad condicional,

$$P(y^i|X)=\frac{\sum_{j=1}^{n^i} e^{\left(-D_j^{i2}/(2\sigma^2)\right)}}{\sum_{j=1}^{n} e^{\left(-D_i^2/(2\sigma^2)\right)}}$$

donde, en primer lugar, cada valor que genera el modelo representa la probabilidad condicional de una i-clase, dado el vector de *inputs*. En segundo lugar, las neuronas de la capa oculta se segmentan según las clases definidas y en último lugar, el numerador de la expresión anterior,

$$\sum_{j=1}^{n^i} e^{\left(-D_j^{i2}/(2\sigma^2)\right)}$$

solo se adiciona en función de la clase que pertenece la neurona oculta respectiva, (véase ilustración 1.2.5.5.2.).

Un caso particular de los modelos neuronales probabilísticos es el modelo neuronal *Gram-Charlier* (GCNN), propuesto por Kim y Arozullah[127] (1992), ambos poseen el mismo objetivo pero su aproximación es diferente y por lo tanto también lo son sus propiedades. A primera vista ambos modelos son parecidos ya que estiman la función de densidad a través de los datos de aprendizaje para cada clase definida, pero en su implementación se encuentran las diferencias esenciales.

[127] Véase Kim, M.W.; Arozullah, M. (1992). **Generalized Probabilistic Neural Network-Based Classifiers**, *International Joint Conference on Neural Networks*, Baltimore. Referenciado en Masters, T. (1995). **Advanced Algorithms for Neural Networks. A C++ Sourcebook**, Wiley.

El modelo neuronal PNN utiliza su topología para aproximar la función de densidad a través de las ventanas de *Parzen*, en cambio el modelo GCNN la utiliza para aproximar las series de expansión de Gram-Charlier[128], su rapidez computacional es mucho mayor que el modelo PNN[129].

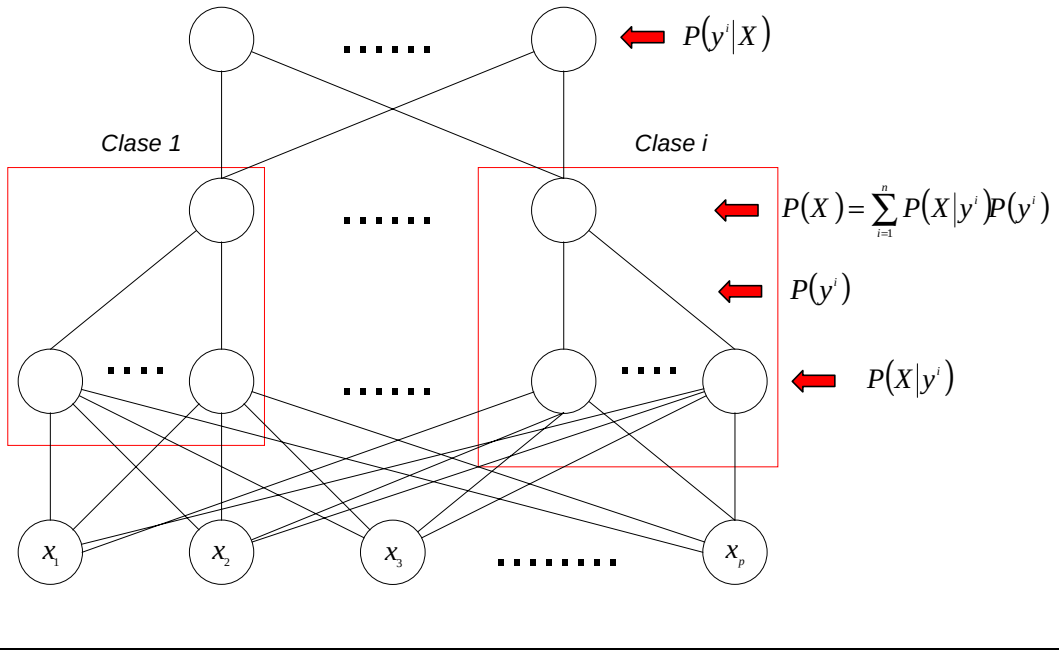


Ilustración 1.2.5.5.2.
Topología de un modelo neuronal PNN

En último lugar, existen avances desde la óptica de la construcción de algoritmos de aprendizaje para los modelos PNN, donde no es necesario definir a priori la topología necesaria, véase Berthold y Diamond[130] (1998).

[128] Véase para una aplicación en *finanzas* relacionado con los modelos no lineales de volatilidad GARCH en, Jondeau, E.; Rockinger, M. (2001). **Gram-Charlier densities**, *Journal of Economic Dynamics & Control*, 25, pp. 1457-1483 y aplicaciones en el entorno bursátil en términos de clasificación, Leung, M.T.; Daouk, H.; Chen, A-S. (2000). **Forecasting stock index: a comparison of classification and level estimation models**, *International Journal of Forecasting*, 16, pp. 173-190.

[129] Una alternativa para representar funciones de densidad y distribuciones con series de expansión, es la expansión de *Edgeworth*.

[130] Véase Berthold, M.R.; Diamond, J. (1998). **Constructive training of probablistics neural networks**, *Neurocomputing*, 19, pp. 167-183.

2.5.5.3. Redes neuronales de *Regresión Generalizada* (GRNN).

El modelo neuronal de regresión generalizada (GRNN) fue propuesto y desarrollado por Specht[131](1991). Posee la propiedad deseable de no requerir ningún entrenamiento iterativo, es decir, puede aproximar cualquier función arbitraria entre vectores de entrada (*inputs*) y salida (*outputs*), realizando la estimación de la función directamente a partir de los datos de entrenamiento[132]. Además, posee la propiedad de la consistencia, de forma que, a medida que el conjunto de entrenamiento se incrementa, el error de estimación tiende a cero, con sólo restricciones moderadas en la función.

El modelo GRNN descansa en la teoría de regresión no lineal[133], cuya expresión es la siguiente,

$$E[y/X]=\frac{\int_{-\infty}^{\infty} y\, f(X,y)dy}{\int_{-\infty}^{\infty} f(X,y)dy}$$

siendo,

X: vector de *input* $(x_1, x_2, \cdots, x_n)$,

y: escalar que representa el *output*,

$E[y|X]$: valor esperado de la salida (*output*), dado el vector de entrada X,

$f(X,y)$: función de densidad conjunta de probabilidad de X e y.

[131] Véase Specht, D. A. (1991). **General Regression Neural Network**, *IEEE Transactions on Neural Networks*, 2(6), pp. 568-576. Referenciado en Masters, T. (1995). **Advanced Algorithms for Neural Networks. A C++ Sourcebook**, Wiley.

[132] La familia de los modelos de regresión polinomial suelen ser introducidos con anterioridad a los modelos neuronales GRNN sobre todo por su gran capacidad de adaptación a los datos. Véase Wasserman, Philip D. (1993). **Advanced Methods in Neural Computing**, pp. 230-234, Van Nostrand ReinHold.

[133] Por definición, la regresión de una variable dependiente "*y*" respecto a una variable independiente "*x*" supone obtener el valor más probable para "*y*", dado un conjunto de entrenamiento o aprendizaje (*x*,*y*).

El modelo GRNN es, en esencia, un método para estimar $f(X, y)$sólo a través del conjunto de entrenamiento[134], de forma que la función de probabilidad conjunta, que se desconoce, se estima mediante el estimador de *Parzen*[135](1962). Para ello, debemos en primer lugar definir las siguientes distancias entre "*X*" e "*y*",

$$D_X(X, X_i) = \sum_{j=1}^{p} \left(\frac{x_j - x_{ij}}{\sigma_j} \right)^2$$

y

$$D_y(y, y_i) = \left(\frac{y - y_i}{\sigma_y} \right)^2$$

siendo, $D_X(X, X_i)$ la distancia entre el vector de *inputs*, (X) y el i-ésimo vector de referencia (normalmente el *centroide* (X_i)),"x_j" es el j-ésimo valor del vector de *inputs*, "x_{ij}" es j-ésimo valor para el caso i-ésimo del vector de referencia, finalmente,"σ_y" es el factor de *escala* o parámetro de normalización, que equivale al radio de influencia de la neurona en el espacio de las entradas. A mayor valor, la región que la neurona domina en torno al vector de referencia es más amplia. De la misma forma podríamos definir para $D_y(y, y_i)$.

Así la aproximación de la densidad de *Parzen* tendrá la siguiente expresión[136],

$$g(X, y) = \frac{1}{n\, c_X c_y} \sum_{i=1}^{n} e^{\left(-D_X(X, X_i) - D_y(y, y_i)\right)}$$
$$= \frac{1}{n c_X c_y} \sum_{i=1}^{n} e^{\left(-D_X(X, X_i)\right)} e^{\left(-D_y(y, y_i)\right)}$$

donde, "c_X, c_y" son constantes normalizadas que aseguran la condición unitaria de la integral de la función, $g(X, y)$ y "n" es el tamaño muestral utilizado.

---

[134] Debido a que la *función de densidad* se aproxima a partir de los datos, sin hipótesis a priori sobre su forma, el sistema es perfectamente general y además no hay problema si las funciones están compuestas de regiones no *gausianas* múltiples en cualquier número de dimensiones.

[135] *Parzen* (1962) desarrolló un método robusto para determinar la función de probabilidad de una población a partir de datos de una muestra, véase Masters, T. (1995). **Advanced Algorithms for Neural Networks. A C++ Sourcebook**, pp. 163-171, Wiley.

[136] La razón de la elección de la función exponencial es por sus buenas propiedades para la aproximación.

Mediante la expresión del estimador de *Parzen* podemos expresar de forma alternativa $E[y|X]$ como el ratio entre, $\hat{y}(X) = N(X)/D(X)$, siendo el denominador,

$$D(X) = \int_{-\infty}^{\infty} \frac{1}{n c_x c_y} \sum_{i=1}^{n} e^{(-D_X(X,X_i))} e^{(-D_y(y,y_i))} dy$$
$$= \frac{1}{n c_x c_y} \sum_{i=1}^{n} e^{(-D_X(X,X_i))} \int_{-\infty}^{\infty} e^{(-D_y(y,y_i))} dy = \frac{1}{n c_x} \sum_{i=1}^{n} e^{(-D_X(X,X_i))}$$

y el numerador,

$$N(X) = \int_{-\infty}^{\infty} \frac{y}{n c_x c_y} \sum_{i=1}^{n} e^{(-D_X(X,X_i))} e^{(-D_y(y,y_i))} dy$$
$$= \frac{1}{n c_x c_y} \sum_{i=1}^{n} e^{(-D_X(X,X_i))} \int_{-\infty}^{\infty} y \cdot e^{(-D_y(y,y_i))} dy = \frac{1}{n c_x} \sum_{i=1}^{n} y_i \cdot e^{(-D_X(X,X_i))}$$

donde la expresión fundamental del modelo neuronal de regresión generalizada (GRNN)[137],

$$\hat{y}(X) = \frac{N(X)}{D(X)} = \frac{\sum_{i=1}^{n} y_i \cdot e^{(-D(X,X_i))}}{\sum_{i=1}^{n} e^{(-D(X,X_i))}}$$

La topología del modelo GRNN posee en cuatro capas. La primera de ellas, representa el vector de entradas o *inputs*. La segunda, denominada pattern layer,[138] es igual al número de observaciones del vector de *inputs*. El valor de la neurona en esta capa no visible, se obtiene al aplicar la función de transferencia o activación, $h_i = e^{[-D_i^2/(2\sigma^2)]}$, que es una extensión de la función multivariante *gaussiana*. La tercera capa recoge dos tipos de *sumatorios*, el primero, $S_s = \sum_i h_i$, que es el denominador ($D(X)$), y el segundo de los sumatorios, que es el numerador, $S_w = \sum_i w_i h_i$ ($N(X)$), donde "w_{ij}" es el valor del output utilizado en la fase de aprendizaje y que actúan como pesos. En último lugar, en la cuarta capa se obtiene el output mediante la siguiente operación $y = S_w / S_S$, (véase ilustración 1.2.5.5.3.).

---

[137] En el proceso de sustitución reseñado se observa como el factor de escala, "σ_y", no esta presente en la expresión final, (véase Masters, T. (1995). **Advanced Algorithms for Neural Networks. A C++ Sourcebook**, pp. 234-237, Wiley).

[138] Para el caso Gaussiano, se denomina *Gaussianan Kernels*, (véase mayor detalle sobre la regresión de *kernel* y su relación con los modelos neuronales RBF en Bishop, C.M. (1995). **Neural Networks for Pattern Recognition**, pp. 177-179, Clarendon Press – Oxford).

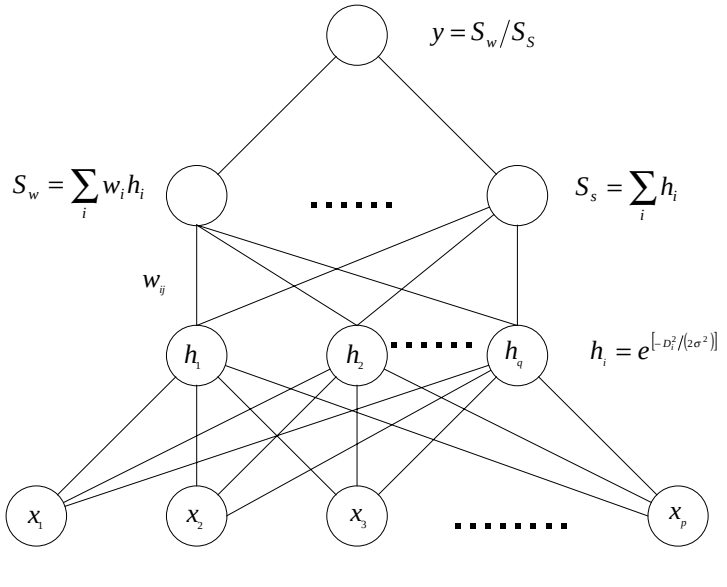


Ilustración 1.2.5.5.3.
Topología de un modelo neuronal GRNN

Una de las expresiones anteriores contiene un parámetro importante, σ[139], que satisface la siguiente condición, $\lim_{n\to\infty} \sigma(T) = 0$; $\lim_{n\to\infty} (T\sigma^n(T)) = \infty$, cuando, $\sigma = S/T^{(E/n)}\ 0 \le E < 1$. Siendo "*T*", el número de ventanas de *Parzen* utilizadas en el proceso de estimación que debe especificarse a priori y "*n*" representa la dimensión del espacio de los *inputs*, es decir, el número de neuronas de entrada en la topología del modelo neuronal[140]. La expresión obtenida es muy similar al modelo *radial basis function* (RBF), una vez normalizado. La diferencia esencial está en la forma que se estiman los pesos, "w_{ij}", para el caso de la GRNN en lugar de entrenar los pesos, simplemente se les asigna el valor directamente del conjunto de entrenamiento asociado[141]. Respecto a las aplicaciones de dicha modelización, existen ejemplos de una gran variedad de disciplinas científicas como por ejemplo en Ingeniería[142], Finanzas[143], etc.

[139] Nos indica el grado de interpolación entre los diferentes grupos formados de la base de entrenamiento.
[140] El valor de "S" se estima mediante *validación cruzada*, de forma que, se impone aquella "σ" que proporciona el error mínimo de la regresión.
[141] Existen trabajos donde se proponen topologías modificadas que permiten una mayor robustez, véase Tomandl, D. ; Schober A. (2001). **A Modified General Regression Neural Networks (MGRNN) with new efficient training algorithms as a robust 'black box' – tool for data analysis**, *Neural Networks*, 14, pp. 1023-1034.
[142] Véase Elsharkwy, A.M; Gharbi, R.B.C. (2001). **Comparing classical and neural regression techniques in modeling crude oil viscosity**, *Advances in Engineering Software*, 32, pp. 215-224.
[143] Véase Leung, Mark.T.; Chen, An-Sing; Daouk, Hazem. (2000). **Forecasting exchange rates using general regression neural networks**, *Computers & Operations Research*, 27, pp. 1093-1110.

2.5.6. Redes Neuronales *Polinomiales* (PoNN): *Algoritmo GMDH.*

Las redes neuronales *polinómicas* (PoNN) utilizan el algoritmo "*Group Method of Data Handling*" (GMDH). Las primeras investigaciones fueron a cargo de R. Shankar[144] (1972) el cual presentó el algoritmo GMDH como un método que permitía describir de forma sucesiva un sistema complejo de relaciones a partir de simples operaciones matemáticas. De hecho, es un buen método para solucionar problemas del estilo, identificación, predicción a corto y a largo plazo de procesos aleatorios, reconocimiento de patrones en entornos complejos, etc. La teoría matemática fue desarrollada de forma conjunta por muchos investigadores, siendo su máximo exponente A.G. Ivakhnenko[145], hacia los años sesenta. El contenido del algoritmo se desarrolló como vehículo para identificar relaciones no lineales entre *inputs* y *outputs*, generando una estructura óptima a partir de un proceso sucesivo de varias generaciones de descripciones parciales de los datos, mediante la incorporación de nuevas capas. En cada capa se inicia con un número máximo de neuronas (definido por combinatoria), de forma que, mediante un proceso de selección se determina el número de neuronas más idóneo en cada capa y así el proceso se diferencia claramente del *back-propagation* en donde todas las capas participan simultáneamente en el proceso de aprendizaje.

Los aspectos más importantes del proceso histórico son los siguientes:

- Se caracterizó por aplicaciones orientadas a establecer criterios de regularidad para solucionar problemas de identificación, predicción a corto plazo, reconocimiento de patrones, pero no se investigó la robustez frente al ruido de los datos (1968-1971).
- Se solucionó el problema de la modelización con datos incompletos o con ruido (1972-1975).
- Fue investigado el grado de convergencia del algoritmo GMDH (1976-1979).

[144] Para más información véase http://www.gmdh.net/.
[145] El método fue desarrollado por el grupo *Combined Control Systems* del instituto de Cibernética en Kiev (Ucranía) en 1968.

- Se obtuvieron importantes resultados teóricos, proponiendo modelos no físicos para predicciones a largo plazo, modelos en dos niveles, etc, (1980-1988).
- Aparición de nuevos algoritmos para la modelización no paramétrica, como por ejemplo, "*Twice-Multilayered Neural Nets*" (TMNN) (1988 hasta la actualidad).

Respecto a las aplicaciones, véase tabla 1.2.5.6., son cada vez más numerosas y relacionadas con diversos campos científicos[146].

Tabla 1.2.5.6. Áreas de *aplicación* de los modelos GMDH.

	Áreas de Aplicación	*Aplicaciones*
1	Sistemas Económicos	Identificación de procesos inflacionistas Evaluación de los principales factores de una economía Predicción Bursátil
2	Análisis de sistemas Ecológicos	Predicción de yacimientos de petróleo Predicción de flujos fluviales
3	Gestión de Sistemas	Predicción de la actividad solar Predicción de la contaminación de la atmósfera
4	Diagnóstico Médico	Diagnóstico de cáncer de pacientes Clasificación de las fases del sueño Clasificación del estado del cerebro humano
5	Modelización Meteorológica	Predicciones climáticas Influencia de los cambios climáticos a largo plazo en los embalses
6	Modelización Econométrica y marketing	Modelización de las relaciones coste-beneficio
7	Producción	Optimización y predicción de la producción de procesos Predicción de la calidad del cemento Modelización de procesos de fermentación
8	Procesamiento de señales	Sistema de seguridad
9	Sismología	Discriminación de seísmos Detección de bolsas de agua y petróleo
10	Sistemas militares	Clasificación de patrones de imágenes de radar Clasificación de ultrasonidos

El primer modelo neuronal diseñado fue "*Ivakhnenko Polynomial*" o "*Kolmogorov-Gabor Polynomial*", con una topología formada por dos *inputs* y un *output*. Dicha expresión es el resultado de una combinación cuadrática de los inputs generando un total de 6 ponderaciones, cuya expresión del *output* es, en este caso,

$$y = \alpha_0 + \alpha_1 x_1^2 + \alpha_2 x_1 x_2 + \alpha_3 x_2^2 + \alpha_4 x_1 + \alpha_5 x_2$$

[146] Cada vez más se entrelazan las diferentes técnicas en campos de investigación muy diversos, véase como ejemplo, Brusilovskiy. P.; Tilman, L.M. (1996). **Incorporating expert judgement into multivariate polynomial modelling Topic Department: Decision support systems foundations**, *Decision Support Systems*, 18, pp. 199-214.

Una topología más completa incorpora capas ocultas, donde el *output* puede ser expresado como un polinomio de grado $2(k-1)$, siendo "*k*" el número total de capas en el modelo neuronal.

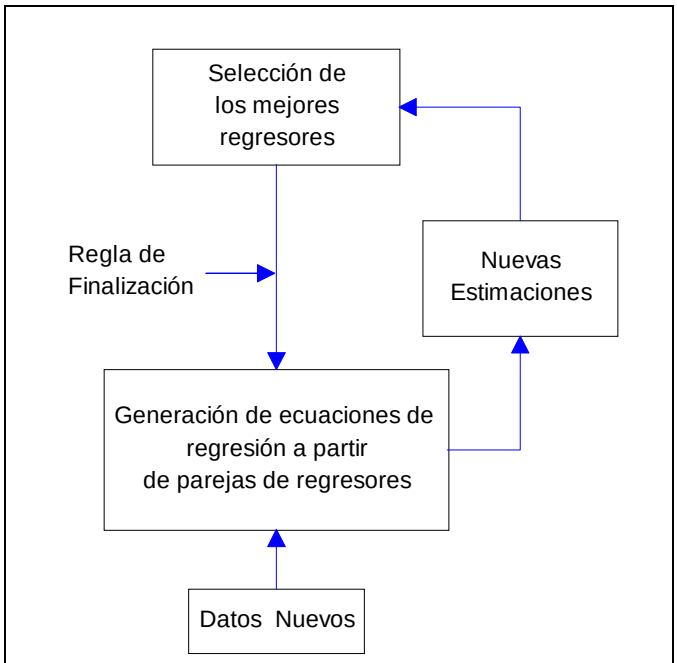


Ilustración 1.2.5.6.
Esquema de las fases del proceso *GMDH*

El proceso de estimación de los parámetros posee tres fases (véase ilustración 1.2.5.6.). La primera de ellas consiste en agrupar por parejas todas las variables independientes $(x_1, \cdots, x_n)$ de forma que,

$$\{(x_1, x_2), (x_1, x_3), \cdots, (x_1, x_n), \cdots, (x_{n-1}, x_n)\}$$

así con las variables anteriores se crean $n(n-1)/2$ ecuaciones de regresión,

$$y = a_0 + \sum_{i=1}^{n} a_i x_i + \sum_{i=1}^{n} \sum_{j=1}^{n} a_{ij} x_i x_j$$

Cada término de regresión se verifica con la muestra de datos de entrenamiento y de test, pero sólo las mejores variables se mantienen utilizando la muestra de test[147].

[147] El número de variables que se mantienen se decide de antemano.

Las variables nuevas se pueden considerar versiones mejoradas de las primeras variables generadas. Así por ejemplo, para un caso de 3 variables de entrada o *inputs* (x_1, x_2, x_3) obtenemos los siguientes modelos[148],

$$y = a_0 + a_1x_1 + a_2x_2 + a_3x_1x_2 + a_4x_1^2 + a_5x_2^2$$
$$y = a_0 + a_1x_2 + a_2x_3 + a_3x_2x_3 + a_4x_2^2 + a_5x_3^2$$
$$y = a_0 + a_1x_1 + a_2x_3 + a_3x_1x_3 + a_4x_1^2 + a_5x_3^2$$

En el segundo paso de la optimización, véase ilustración 2.2.5.6., las variables originales "x_i" se substituyen por las variables nuevas, "z_i", que son las que mejor describen la variable dependiente, "*y*".

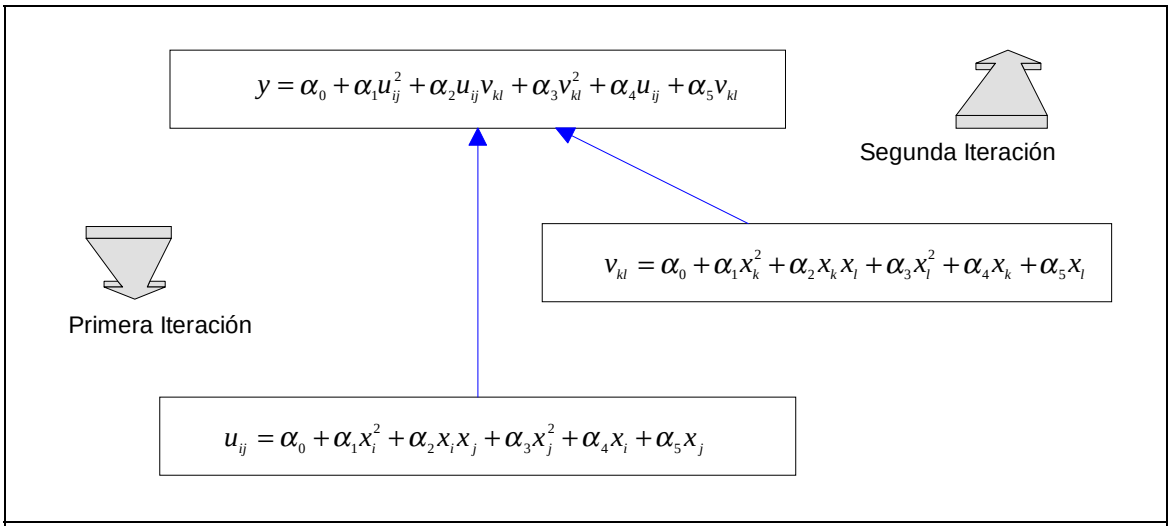


Ilustración 2.2.5.6.
Ejemplo de un proceso Iterativo del algoritmo *GMDH* (dos iteraciones)

Fuente: Bol, G. ; Nakhaeizadeh, G.; Vollmer, K-H. (1998).
Risk Measurement, Econometrics and Neural Networks, pp. 114-118, Springer.

Para cada combinación de las variables originales $(x_i; x_j)$, la raíz del error cuadrático medio se utiliza para calcular el *criterio de regularidad* (CR) como una medida de bondad del ajuste (mediante los datos de la base de test). Todas las variables "z_{ij}" con $r_j^2 < R$ se mantienen como nuevos regresores, donde "*R*" se especifica a priori y la expresión de "r_j^2"es,

[148] Véase algunas aplicaciones en Sforna, M. (1995). **Searching for the electric load-weather temperature function by using the group method of data handling**, *Electric Power Systems Research* 32, pp. 1-9 y Burger, CJSC; Dohnal, M; Kathrada, M; Law, R. (2001). **A practicioners guide to time-series methods for tourism demand forecasting- a case study of Durban, South Africa**, *Tourim Management*, 22, pp. 403-409.

$$r_j^2 = \frac{\sum_{i=1}^{m} \left(y_i - z_{ij}^2\right)^2}{\sum_{i=1}^{m} \left(y_i^2\right)} \qquad j = 1,2,\cdots,n$$

En la última fase, la bondad del modelo permite determinar la necesidad de más iteraciones. El valor de "r_j^2" más pequeño obtenido se compara con el generado en la última iteración, si no existe mejora el proceso ha terminado, (véase ilustraciones 3.2.5.6. y 4.2.5.6.).

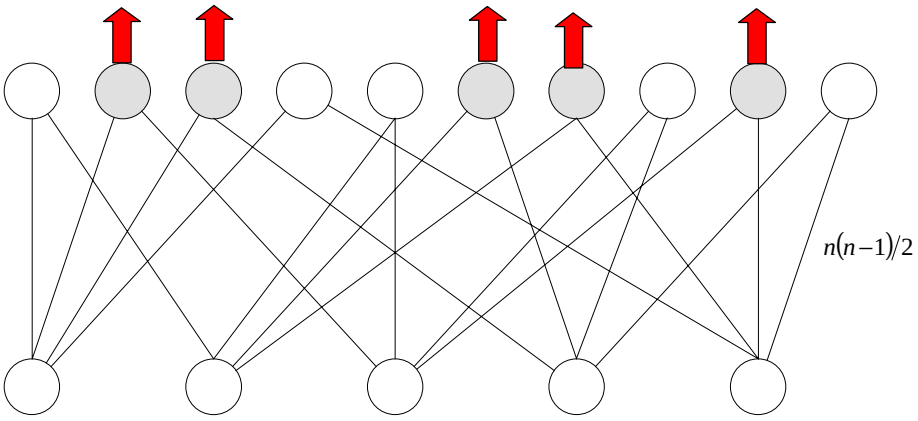


Ilustración 3.2.5.6.
Fases del proceso Iterativo algoritmo *GMDH* (1)

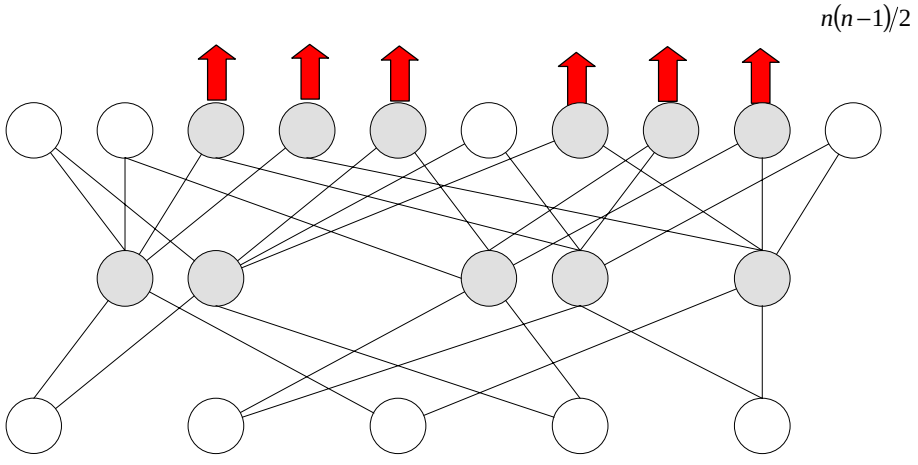


Ilustración 4.2.5.6.
Fases del proceso Iterativo algoritmo *GMDH* (2)

Al final del proceso anterior, se posee un modelo con los regresores que se crearon como una síntesis de diversas variables de generaciones anteriores. Estas variables pueden ser expresadas de manera recursiva en términos de las variables originales[149]. Como podemos observar, es un modelo que no está completamente interconectado, similar a un método *autoorganizativo inductivo* y con la capacidad de solucionar problemas complejos, (véase ilustración 5.2.5.6.).

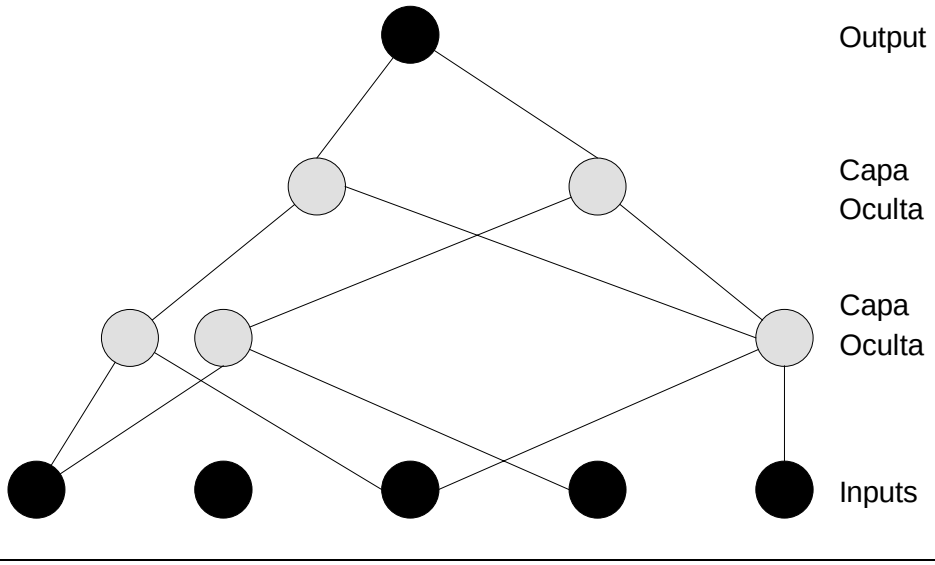


Ilustración 5.2.5.6.
Resultado final del proceso Iterativo algoritmo *GMDH*

Especialmente la última fase descrita es problemática. El *criterio de regularidad* (CR) descrito anteriormente es sensible a los datos con fuerte componente aleatoria, por esta razón se suelen utilizar otros criterios, como por ejemplo, el criterio *no sesgado* o el criterio *combinado*.

[149] Tal y como hemos comentado, la mayoría de los algoritmos GMDH utilizan funciones *polinomiales* que pueden expresar la conexión entre *inputs* y *outputs* mediante las series funcionales de *Volterra*, versión discreta del polinomio *Kolmogorov-Gabor* , $y = a_0 + \sum_{i=1}^{M} a_i x_i + \sum_{i=1}^{M}\sum_{j=1}^{M} a_{ij} x_i x_j + \sum_{i=1}^{M}\sum_{j=1}^{M}\sum_{k=1}^{M} a_{ijk} x_i x_j x_k$.

Para el primero de ellos, se subdividen los datos disponibles en dos muestras "A" y "B", donde para cada una de ellas, se aplica de manera independiente el algoritmo, siendo su expresión, la siguiente,

$$u_j^2 = \frac{\sum_{i=1}^{m} \left(z_{ij}^A - z_{ij}^B \right)^2}{\sum_{i=1}^{m} \left(y_i^2 \right)} \quad j = 1,2,\cdots,n$$

donde, la variable "z_{ij}" solo permanecerá en la próxima iteración si $u_j^2 < U$, siendo "*U*" un umbral predefinido. Los resultados empíricos no han sido muy buenos para este primer caso. En segundo lugar, el criterio combinado, intenta solventar los problemas del primero, combinando los dos con la siguiente expresión,

$$c_j = \sqrt{\left(r_j^2 - u_j^2 \right)} \quad j = 1,2,\cdots,n$$

donde, la variable "z_{ij}" solo permanecerá en la próxima iteración si $c_j^2 < C$, siendo "*C*" un umbral predefinido[150].

Los modelos neuronales polinomiales[151](PoNN) poseen una diferencia importante respecto a los modelos donde las neuronas están previamente definidas y es que, sus unidades de procesamiento poseen un papel activo, debido a que los algoritmos GMDH se ejecutan dentro de las propias unidades, representando una nueva variable, que es generada mediante la selección independiente de las entradas relevantes necesarias para encontrar la estructura óptima.

[150] Los algoritmos GMDH en PoNN se implementan mediante los procedimientos descritos y se combinan con algoritmos genéticos. Éstos aseguran que la combinación de regresores óptimos describa bien el modelo.

[151] Véase el trabajo de Sung-Kwun Oh; Witold Pedrycz. (2002). **The design of self-organizing Polynomial Neural Networks**, *Information Sciences*, Vol. 141, No. 3-4, pp. 237-258, donde se presenta la familia de dichos modelos neuronales.