1 Introducción

1.1. ¿Pueden las computadoras aprender a resolver problemas a partir de ejemplos de forma autónoma?

El cerebro es el sistema de cálculo más complejo que conoce el hombre. El ordenador y el cerebro están especializados en hacer tareas completamente diferentes; así la operación de reconocer el rostro de una persona resulta una tarea relativamente sencilla para el cerebro y difícil para el ordenador, mientras que la contabilidad de una empresa es tarea costosa para un experto contable y una sencilla rutina para un ordenador básico. Esta capacidad de raciocinio nos ha permitido desarrollar una tecnología propia de tal manera que, en estos momentos, esta tecnología se orienta a descubrir su origen. ¿Como funciona el cerebro?, ¿se pueden construir modelos artificiales que lo emulen?, ¿se pueden desarrollar máquinas inteligentes? Todas estas cuestiones que rozaban no hace mucho tiempo la frontera de la ciencia ficción es actualmente objeto de profundos estudios en un campo multidisciplinar del conocimiento conocido como Inteligencia Artificial (IA). Este campo se podría dividir en dos clases que podríamos definir como "macroscópico" y "microscópico".

El primero de ellos intenta modelizar el funcionamiento del cerebro basándose en reglas definidas por distintas lógicas. El nombre de macroscópico se debe a que no se toma en cuenta en ningún momento la estructura interna del cerebro sino que modeliza su comportamiento en base a un funcionamiento que podríamos definir como global.

En el caso del segundo enfoque se parte de la estructura que presenta el cerebro de tal forma que se construyen modelos que tienen en cuenta dicha estructura. El resultado ha sido una nueva tecnología llamada Computación Neuronal o también Redes Neuronales Artificiales (RNA).

Las Redes Neuronales Artificiales están inspiradas en las redes neuronales biológicas del cerebro humano, y han sido propuestas como modelos extremadamente simplificados del funcionamiento de este, donde sus elementos de proceso más básicos, conocidos como neuronas artificiales, se comportan de forma similar a la neurona biológica en sus funciones más comunes: i) las neuronas artificiales no pueden encontrarse más que en dos estados posibles, activas o en reposo; ii) están interconectadas mediante sinapsis que pueden ser modificadas por aprendizaje y iii) el estado de una neurona a cada instante es determinado por el de otras neuronas, información que es transmitida por las sinapsis.

Aunque las RNA sean un modelo muy esquemático, presentan una riqueza sorprendente de estados y de comportamientos ya que el sistema global generado por una red neuronal presenta propiedades complejas que no pueden predecirse a partir del estudio individual de sus neuronas. Así, el todo es mucho más que la suma de sus partes.

La Inteligencia Artificial macroscópica clásica o simbólica (IAC) ha generado una

cierta decepción al tratar de explicar los procesos cognitivos debido a que, por una parte, la representación usando reglas se aleja mucho de cualquier inspiración biológica, y por otra, conduce a la creación de sistemas que son demasiado rígidos y al mismo tiempo extremadamente frágiles. El reconocimiento, el aprendizaje y la memoria son mecanismos cognoscitivos que no pueden ser explicados por medio del simbolismo de la IA, sino más bien en función de unidades simples pero altamente interconectadas de las cuales emerge un comportamiento complejo, paralelo y autoorganizado, sin necesidad de tener reglas explícitas de decisión ni de un procesador maestro o motor de inferencia. La secuenciación y la rigidez de reglas son dos aspectos que han sido duramente criticados en las técnicas de IA. La inmensa ventaja de los métodos conexionistas comparado con los métodos tradicionales de IA es la siguiente: no es necesario conocer ni una expresión ni una construcción de la función a modelar, tan sólo se requiere disponer de un conjunto de aprendizaje satisfactorio para que la red pueda aproximar esta función aplicando una regla de aprendizaje. Así, muchos fenómenos cognoscitivos han logrado ser modelados a través de sistemas conexionistas.

1.2. Características de las redes neuronales artificiales

Las RNA intentan imitar algunas características propias del cerebro. Por ejemplo las RNA aprenden de la experiencia, generalizan de ejemplos previos a ejemplos nuevos y abstraen las características principales de una serie de datos.

Aprender: "Adquirir el conocimiento de una cosa por medio del estudio, ejercicio o experiencia" (RAE). Las RNA pueden cambiar su comportamiento en función del entorno, es decir, a partir de un conjunto de entradas pueden ajustar sus conexiones sinápticas para producir la salida deseada.

Generalizar: "Sacar una conclusión general de algo particular" (RAE). Las RNA generalizan automáticamente debido a su propia estructura y naturaleza. Estas redes pueden ofrecer, dentro de un margen, respuestas correctas a patrones de entrada no experimentados con anterioridad.

Abstraer: "Separar por medio de una operación intelectual las cualidades de un objeto para considerarlas aisladamente o para considerar el mismo objeto en su pura esencia o noción" (RAE). Algunas RNA son capaces de abstraer la esencia de un conjunto de entradas que aparentemente no presentan aspectos comunes o relativos.

La potencia computacional de una red neuronal deriva, principalmente, de su estructura de calculo distribuido paralelo. Esta estructura le permite la resolución de problemas que necesitarían gran cantidad de tiempo en ordenadores "clásicos". Aparte de este hecho, aparecen otras propiedades que la hacen especialmente atractivas para ser usadas en una gran cantidad de problemas prácticos [**?**, **?**]:

Sistemas distribuidos no lineales: Una neurona es un elemento no lineal, por lo que la interconexión entre ellas (red neuronal) también será un dispositivo no lineal. Esta propiedad permitirá la simulación de sistemas no lineales y caóticos.

Tolerante a fallos: Una red neuronal, al ser un sistema distribuido, permite el fallo de algunos elementos individuales (neuronas) sin alterar significativamente la respuesta total del sistema. Este hecho las hace especialmente atractivas frente a las computadoras actuales, que por lo general son sistemas secuenciales tal que un fallo en uno de sus componentes conduce al fallo total del sistema.

Adaptabilidad: Una red neuronal tiene capacidad de modificar los parámetros de los que depende su funcionamiento de acuerdo con los cambios que se produzcan en su entorno de trabajo (cambios en las entradas, presencia de ruido, etc...). Con respecto a la capacidad de adaptación hay que tener en cuenta que ésta no puede ser tampoco excesivamente grande, ya que conduciría a tener un sistema inestable respondiendo a pequeñas perturbaciones. Este problema es conocido como el dilema plasticidad-estabilidad.

Establecen relaciones no lineales de los datos: Las redes neuronales son capaces de establecer asociaciones entre los datos sin las restricciones habitualmente impuestas por métodos estadísticos tradicionales, como la linealidad, gausianidad, etc. [**?**]

Posibilidad de implementación en VLSI: La arquitectura de estos sistemas favorecen su implementación en sistemas en tiempo real simulando sistemas biológicos mediante elementos de silicio [**?**, **?**, **?**].

Todas estas ventajas hacen el uso de las redes neuronales especialmente atractivo en un gran número de aplicaciones, las cuales se describen en detalle en la sec. 1.6

1.3. Diferentes modelos conexionistas

En el campo de las redes neuronales, se conoce con el nombre de arquitectura la forma en la que se unen los diferentes elementos, neuronas, mediante una serie de conexiones, llamadas pesos sinápticos. Podemos definir tres niveles en cuanto a arquitectura se refiere:

Microestructura: Este nivel hace referencia al elemento más pequeño que nos podemos encontrar en un modelo conexionista: la neurona. Este nivel es el más pequeño pero no por ello es el menos importante; aquí se fijan características tan importantes como la función de activación.

Mesoestructura: Una vez sobrepasado el nivel neuronal, se fija el tipo de conexión y la disposición de los elementos para conseguir la arquitectura de red neuronal.

Macroestructura: Las diferentes redes planteadas en el nivel anterior se pueden combinar entre si para crear estructuras más complejas con el objetivo de obtener mejores prestaciones en su aplicación a diferentes tareas.

1.4. Estructura básica de una red neuronal

1.4.1. Analogía con el cerebro

La neurona es la unidad fundamental del sistema nervioso y en particular del cerebro que consta de un gran número (aproximadamente 10^{11}) de ellas altamente interconectadas (aproximadamente 10^4 conexiones por neurona). Cada neurona es una simple unidad procesadora que recibe y combina señales desde y hacia otras neuronas. Cada neurona tiene tres componentes principales, las dendritas, el cuerpo de la célula o soma, y el axón. Las dendritas, son el árbol receptor de la red, son como fibras nerviosas que cargan de señales eléctricas el cuerpo de la célula. El cuerpo de la célula, realiza la suma de esas señales de entrada. Cuando esta suma supera un determinado umbral, la neurona se activa y el axón lleva la señal desde el cuerpo de la célula hacia otras neuronas a través de uniones llamadas sinapsis. La eficacia de la sinapsis es modificable durante el proceso de aprendizaje de la red. La Fig. 1.1 muestra las partes que constituyen una neurona.

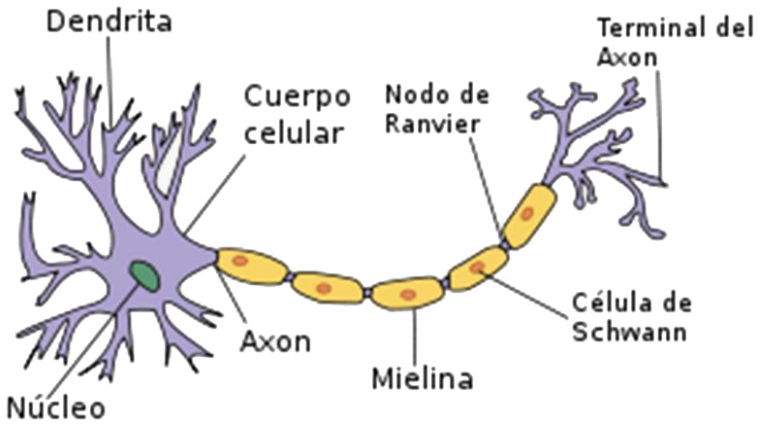


Figura 1.1: Principales componentes de una neurona biológica.

1.4.2. Redes Neuronales Artificiales

En las Redes Neuronales Artificiales (RNA) la unidad análoga a la neurona biológica es el elemento procesador (EP). Un elemento procesador tiene varias entradas que habitualmente integra mediante un sumador básico. La suma de las entradas es modificada por una función de transferencia, y el valor de la salida de esta función de transferencia pasa directamente a la salida del elemento procesador.

La salida del EP se puede conectar a las entradas de otras neuronas artificiales mediante conexiones ponderadas correspondientes a la eficacia de la sinapsis de las conexiones neuronales.

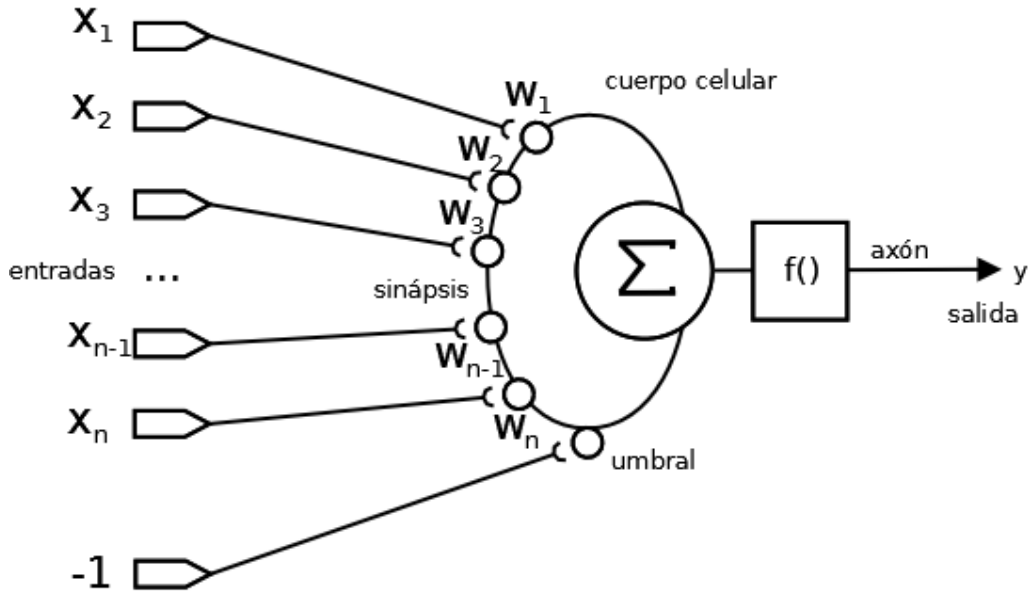


Figura 1.2: Diagrama esquemático de una neurona artificial (PE).

La Fig. 1.2 representa un elemento procesador de una red neuronal artificial implementada en un ordenador.

Una red neuronal consiste en un conjunto de unidades elementales EP conectadas de una forma concreta. El interés de las RNA no reside solamente en el modelo del elemento EP sino en las formas en que se conectan estos elementos procesadores. Generalmente los elementos PE están organizados en grupos llamados niveles o capas. Existen diferentes tipos de redes neuronales. Una red típica conocida como *feedforward* o de procesamiento hacia adelante, consiste en una secuencia de capas con conexiones entre capas adyacentes consecutivas.

Existen dos capas con conexiones hacia el mundo exterior. Una capa de entrada, *buffer* de entrada, donde se presentan los datos a la red, y una capa *o buffer* de salida que mantiene la respuesta de la red a una entrada. El resto de las capas reciben el nombre de capas ocultas. La Fig. 1.3 muestra las partes de una Red Neuronal Artificial *feedforward*.

1.5. Historia de la computación neuronal

En 1943 el neurobíologo Warren McCulloch y el estadístico Walter Pitts propusieron un modelo de neurona del cerebro humano y animal [**?**]. Estas neuronas nerviosas

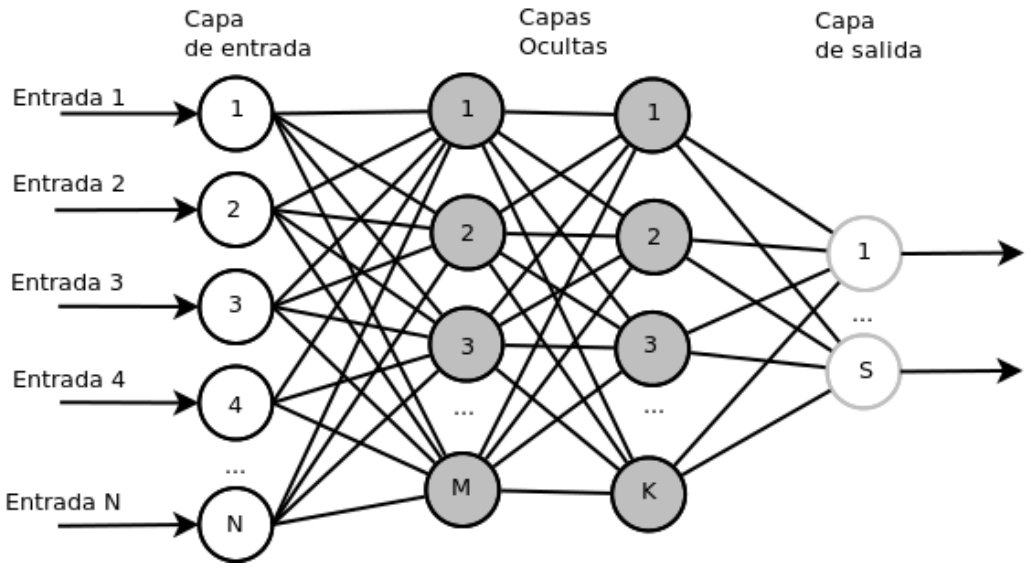


Figura 1.3: Arquitectura de una red neuronal simple.

abstractas proporcionaron una representación simbólica de la actividad cerebral. Más adelante, Norbert Wiener elaboró estas ideas junto con otras, dentro del mismo campo, conocido por aquel entonces como "cibernética" creando las bases de lo que hoy conocemos como Inteligencia Artificial [**?**].

En 1956, los pioneros de la Inteligencia Artificial Minsky, McCarthy, Rochester, y Shannon, organizaron la primera conferencia de Inteligencia Artificial (patrocinada por la Fundación Rochester). Esta conferencia se celebró en el verano de 1956 en la localidad inglesa de Darmouth, y en muchos libros se hace referencia al verano de este año como la primera toma de contacto seria con las redes neuronales artificiales.

Nathaural Rochester del equipo de investigación de IBM presentó el modelo de una red neuronal diseñado por él mismo, hito considerado como el primer software de simulación de redes neuronales artificiales. En 1957, Frank Rosenblatt publicó el mayor trabajo de investigación en computación neuronal realizado hasta esas fechas, que consistía en el desarrollo de un elemento llamado "perceptrón" [**?**].

El perceptrón es un sistema clasificador de patrones que puede identificar patrones geométricos y abstractos. El primer perceptrón era capaz de aprender y además era robusto, comportándose correctamente después de que algunas celdas fueran destruidas, y sólo su comportamiento variaba si resultaban dañados muchos componentes del sistema. El perceptrón fue originalmente diseñado para el reconocimiento óptico de patrones en el que una rejilla de 400 fotocélulas sensibles a la luz, correspondientes a las neuronas de la retina, recibía un estímulo óptico. Estas fotocélulas emitían impulsos eléctricos y estaban conectadas a elementos asociativos que los recogían. Las conexiones entre los elementos asociativos y las fotocélulas se realizaban de forma aleatoria, tal que si las células presentan un valor de entrada superior a un umbral predeterminado entonces el elemento asociativo produce una salida. La

Fig. 1.4 ilustra la estructura de la red perceptrón.

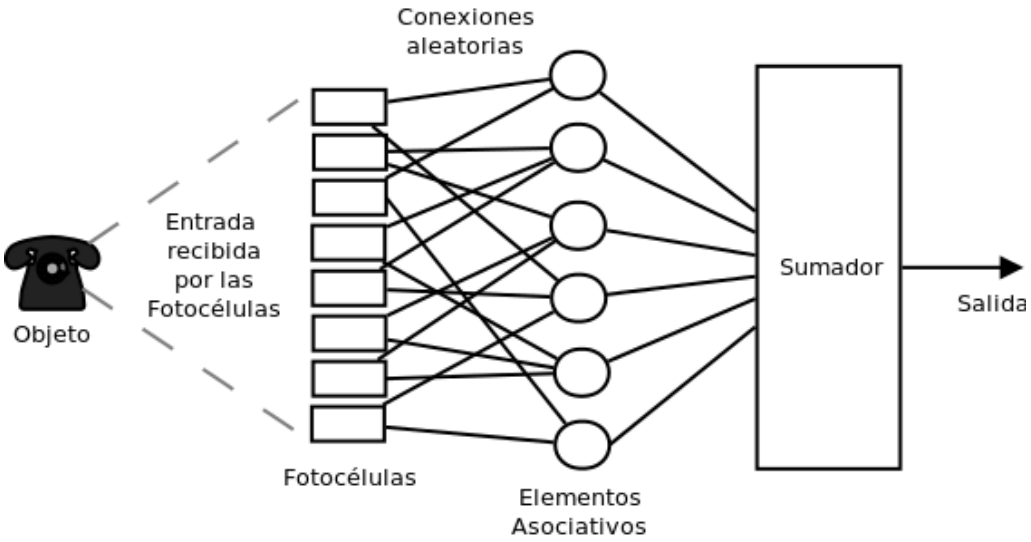


Figura 1.4: Aplicación de la red perceptrón.

El perceptrón presentó algunas limitaciones debido a que se trataba de un dispositivo en desarrollo. La mayor limitación la reflejaron Minsky y Papert [**?**] años más tarde, y ponían de manifiesto la incapacidad del perceptrón en resolver algunas tareas o problemas sencillos como por ejemplo la función lógica OR exclusivo.

Uno de los mayores cambios realizados en el perceptrón de Rossenblatt a lo largo de la década de los 60 ha sido el desarrollo de sistemas multicapa que pueden aprender y categorizar datos complejos. En 1959, Bernard Widrow en Stanford desarrolló un elemento adaptativo lineal llamado "Adaline" (*Adaptive Linear Neuron*). La Adaline y una versión posterior de dos capas, llamada "Madaline", fueron utilizadas en distintas aplicaciones como reconocimiento de voz y caracteres, predicción del tiempo, control adaptativo y sobre todo en el desarrollo de filtros adaptativos que eliminen los ecos de las líneas telefónicas.

A mediados de los años 60, Minsky y Papert, pertenecientes al Laboratorio de Investigación de Electrónica del MIT, realizaron un gran trabajo que criticaba al perceptrón. El resultado de este trabajo, el libro *perceptróns* [**?**], era un análisis matemático del concepto del perceptrón. La conclusión de este trabajo, que se transmitió a la comunidad científica del mundo entero, es que el perceptrón y la Computación Neuronal no eran temas interesantes que estudiar y desarrollar. A partir de este momento descendieron drásticamente las inversiones en la investigación de la computación neuronal.

Uno de los pocos investigadores que continuaron con su trabajo en la computación neuronal tras la publicación del libro de Minsky [**?**] fue James Anderson. Su trabajo se basó en el desarrollo de un modelo lineal consistente en un modelo asociativo distribuido basado en el principio de Hebb (las conexiones son reforzadas cada vez

que son activadas las neuronas). Una versión extendida de este modelo lineal es el llamado modelo *Brain-State-in-a-Box* (BSB).

Teuvo Kohonen, de la Universidad de Helsinki, es uno de los mayores impulsores de la computación neuronal de la década de los 70. De su trabajo de investigación destacan dos aportaciones: la primera es la descripción y análisis de una clase grande de reglas adaptativas, reglas en las que las conexiones ponderadas se modifican de una forma dependiente de los valores anteriores y posteriores de las sinapsis. Y la segunda aportación es el principio de aprendizaje competitivo en el que los elementos compiten por responder a un estímulo de entrada, y el ganador se adapta él mismo para responder con mayor efecto al estímulo.

Otro investigador que continuó en la línea de la computación neuronal, a pesar del mal presagio de Minsky y Papert, fue Stephen Grossberg. Grossberg estaba especialmente interesado en la utilización de datos de la neurología para construir modelos de computación neuronal. La mayoría de sus reglas y postulados derivaron de estudios fisiológicos. Su trabajo ha constituido un gran impulso en la investigación del diseño y construcción de modelos neuronales. Una de estas clases de redes es la *Adaptive Resonance Theory* (ART).

En 1982 John Hopfield consiguió devolver el interés y la confianza en el fascinante campo de la computación neuronal con la publicación del artículo *Hopfield Model o Crossbar Associative Network*, junto con la invención del algoritmo *Backpropagation* tras dos décadas de casi absoluta inactividad y desinterés. Hopfield presenta un sistema de computación neuronal consistente en elementos procesadores interconectados que buscan y tienden a un mínimo de energía.

1.5.1. Investigación de hoy en día

Diversos grupos de investigación a nivel internacional realizan trabajos de investigación en el área de las redes neuronales artificiales. La motivación de cada uno de estos grupos es claramente heterogénea, dada la interdisciplinaridad del área, en la cual trabajan neurólogos, psicólogos del conocimiento, físicos, ingenieros y matemáticos. Todos ellos aportando diferentes puntos de vista e intuiciones en esta área de la técnica.

Dentro de los principales grupos del área, destacamos los siguientes:

- El grupo del Profesor Stephen Grossberg en la Universidad de Boston, que junto con G. Carpenter, continúan trabajando en modelos de redes neuronales con motivación biológica a partir del modelo ART propuesto por ellos mismos [**?**].
- El grupo del Profesor G. Hinton en la Universidad de Toronto. El gran poder de cómputo existente hoy en día ha permitido estudiar y aplicar nuevas redes constituídas por varias capas llamadas "Deep Neural Nets" [**?**], y construídas a partir de la idea de la Máquina de Boltzmann propuesta por Hinton y colegas

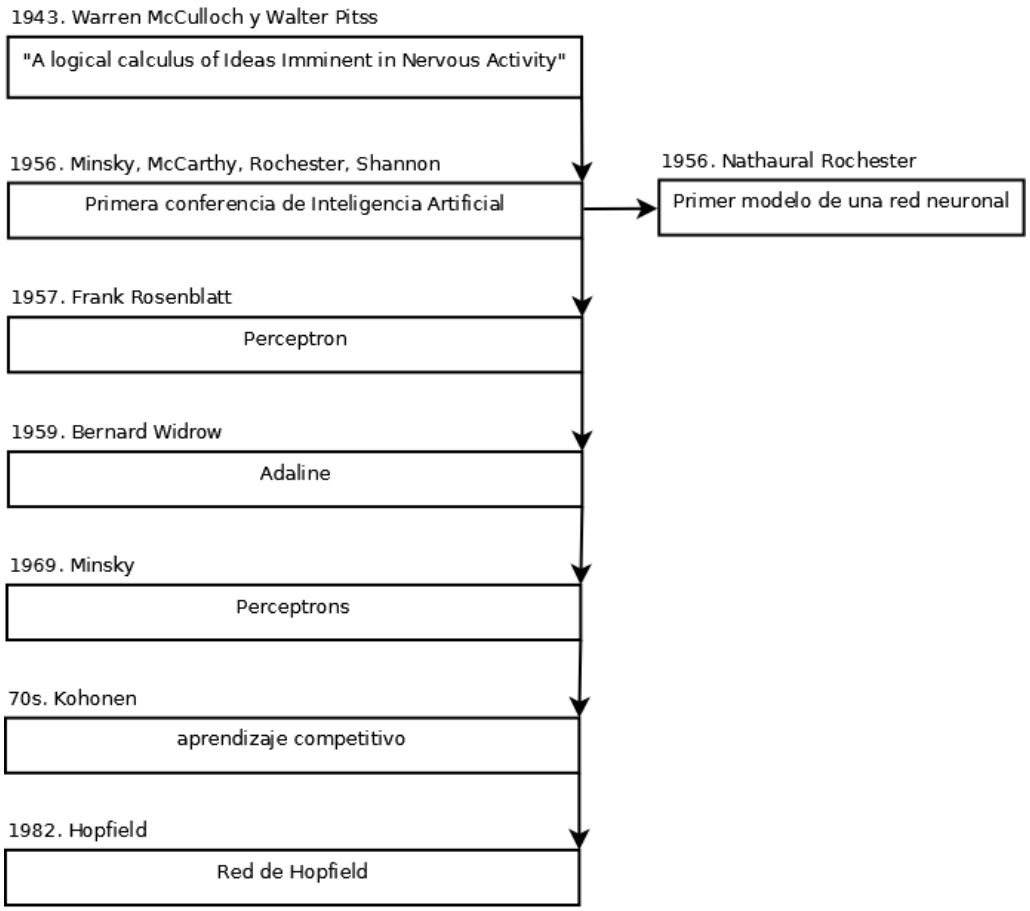


Figura 1.5: Cronograma histórico de la computación neuronal.

en 1985 [**?**]. Dentro de una similar línea de investigación, también destacamos el trabajo de J. Schmidhuber en Lugano, Suiza [**?**].

- Otro grupo de investigación muy activo en estos últimos años lo constituye el grupo del Prof. Huang en Singapur, quienes están llevando adelante la idea del aprendizaje extremo ("Extreme Learning Machines") [**?**]. Por último destacamos las redes neuronales construidas para reconocimiento de objetos en base a arquitecturas con motivación biológica propuestas por el grupo del Prof. T Poggio en el MIT [**?**].

1.6. Aplicaciones de las redes neuronales artificiales

Las características especiales de los sistemas de computación neuronal permiten que sea utilizada esta técnica de cálculo en una extensa variedad de aplicaciones. La computación neuronal provee un acercamiento mayor al reconocimiento y percepción humana que los métodos tradicionales de cálculo. Las redes neuronales artificiales

presentan resultados razonables en aplicaciones donde las entradas presentan ruido o están incompletas. Algunas de las áreas de aplicación de las RNA son las siguientes:

Conversión texto a voz: uno de los principales promotores de la computación neuronal en esta área es Terrence Sejnowski. La conversión texto-voz consiste en cambiar los símbolos gráficos de un texto en lenguaje hablado. El sistema de computación neuronal presentado por Sejnowski y Rosemberg, el sistema llamado NetTalk, convierte texto en fonemas y con la ayuda de un sintetizador de voz (Dectalk) genera voz a partir de un texto escrito. La ventaja que ofrece la computación neuronal frente a las tecnologías tradicionales en la conversión texto-voz es la propiedad de eliminar la necesidad de programar un complejo conjunto de reglas de pronunciación en el ordenador. A pesar de que el sistema NetTalk ofrece un buen comportamiento, la computación neuronal para este tipo de aplicación abre posibilidades de investigación y expectativas de desarrollo comercial [**?**, **?**, **?**, **?**].

Procesado natural del lenguaje: incluye el estudio de cómo se construyen las reglas del lenguaje [**?**, **?**, **?**]. Los científicos del conocimiento Rumelhart y McClelland han integrado una red neuronal de proceso natural del lenguaje. El sistema desarrollado ha aprendido el tiempo verbal *"pass tense"* de los verbos en inglés. Las características propias de la computación neuronal, como la capacidad de generalizar a partir de datos incompletos y la capacidad de abstraer, permiten al sistema generar buenos pronósticos para verbos nuevos o verbos desconocidos .

Compresión de imágenes: la compresión de imágenes es la transformación de los datos de una imagen a una representación diferente que requiera menos memoria o que se pueda reconstruir una imagen imperceptible. Cottrel, Munro y Zisper de la Universidad de San Diego y Pisttburgh han diseñado un sistema de compresión de imágenes utilizando una red neuronal con un factor de compresión de 8:1.

Reconocimiento de patrones en imágenes: algunas aplicaciones aplicación típicas son la clasificación de objetivos detectados por un sonar [**?**, **?**, **?**, **?**] o el reconocimiento de rostros [**?**, **?**, **?**, **?**, **?**].

Problemas de combinatoria: en este tipo de problemas la solución mediante cálculo tradicional requiere un tiempo de proceso (CPU) que es exponencial con el número de entradas. Un ejemplo es el problema del viajante de comercio; el objetivo es elegir el camino más corto posible que debe realizar el vendedor para cubrir un número limitado de ciudades en una área geográfica específica. Este tipo de problema ha sido abordado con éxito por Hopfield y el resultado de su trabajo ha sido el desarrollo de una RNA que ofrece buenos resultados para este problema de combinatoria.

Procesado de señal: el procesado de señales es la disciplina que desarrolla y estudia las técnicas de tratamiento (filtrado, amplificación,...), el análisis y la

clasificación de las señales. Se basa en los resultados de la teoría de la información, de la estadística y la matemática aplicada. En este tipo de aplicaciones, existen tres clases diferentes de problemas de la señal que han sido objeto de las RNA como son la predicción, el modelado de un sistema y el filtrado de ruido.

Predicción: en el mundo real existen muchos fenómenos de los que conocemos su comportamiento a través de una serie temporal de datos o valores. Lapedes y Farber, del Laboratorio de Investigación de los Álamos, han demostrado que la red backpropagation supera en un orden de magnitud a los métodos de predicción polinómicos y lineales convencionales para las series temporales caóticas.

Modelado de sistemas: los sistemas lineales son caracterizados por la función de transferencia, que no es más que una expresión analítica entre la variable de salida y una variable independiente y sus derivadas. Las RNA también son capaces de aprender una función de transferencia y comportarse correctamente como el sistema lineal que está modelando.

Filtro de ruido: las RNA también pueden ser utilizadas para eliminar el ruido de una señal. Estas redes son capaces de mantener en un alto grado las estructuras y valores de los filtros tradicionales.

Modelos económicos y financieros: una de las aplicaciones más importantes del modelado y pronóstico es la creación de pronósticos económicos como por ejemplo los precios de existencias, la producción de las cosechas, el interés de las cuentas, el volumen de las ventas etc. Las redes neuronales están actualmente ofreciendo mejores resultados en los pronósticos financieros que los métodos convencionales.

Servo control: un problema difícil en el control de un complejo sistema de servomecanismo es encontrar un método de cálculo computacional aceptable para compensar las variaciones físicas que se producen en el sistema. Entre los inconvenientes destaca la imposibilidad en algunos casos de medir con exactitud las variaciones producidas y el excesivo tiempo de cálculo requerido para la obtención de la solución matemática. Existen diferentes redes neuronales que han sido entrenadas para reproducir o predecir el error que se produce en la posición final de un robot. Este error se combina con la posición deseada para proveer una posición adaptativa de corrección y mejorar la exactitud de la posición final.

1.7. Implementación y tecnologías emergentes

El resurgimiento de la computación neuronal en los últimos años se ha producido por el desarrollo teórico de nuevos modelos matemáticos del comportamiento del

cerebro, y por el desarrollo de nuevas tecnologías que ya están siendo utilizadas en una gran variedad de aplicaciones comerciales.

Entre los avances o desarrollos tecnológicos que permiten la realización de la computación neuronal destacan los programas software de simulación, los aceleradores hardware, los chips de silicio y los procesadores ópticos.

Simuladores Software: Constituyen una de las formas más versátiles con las que se pueden implementar redes neuronales. Estos programas constituyen todo un sistema de diseño e implementación de prototipos de redes neuronales. Estos programas se utilizan para diseñar, construir, entrenar y probar redes neuronales artificiales para resolver problemas complejos y problemas del mundo real. Los primeros simuladores software se ejecutaban en ordenadores de grandes prestaciones, mientras que el avance en la tecnología de los ordenadores personales, en capacidad de procesado y capacidad de memoria, hace posible que exista una serie de simuladores software de grandes prestaciones se ejecutan sobre ordenadores personales. Entre otros paquetes software, se pueden destacar *Neural Works*, *Neuralyst*, *Explore Net* y *Kwowledge Net*.

Aceleradores Hardware: La naturaleza paralela de la computación neuronal se presta a realizar diseños concretos y a medida de dispositivos físicos, aceleradores Hardware, que aceleren la ejecución de los cálculos. Los aceleradores Hardware para los sistemas de computación neuronal son dispositivos físicos constituidos por diferentes procesadores interconectados que ayudan a la realización y ejecución del comportamiento de las RNA. Una de las ventajas de los aceleradores Hardware diseñados específicamente para la computación neuronal es el aumento de la velocidad de procesado. Esta característica permite la utilización de las RNA en aplicaciones de tiempo real. Robert Hecht-Nielsen desarrolló el acelerador Hardware Mark III que constaba de 8100 procesadores y trabajaba como un periférico de un VAX. La mayoría de las casas comerciales dedicadas al diseño de las RNA han desarrollado diferentes tarjetas basadas en los diferentes procesadores existentes, diseñadas para trabajar en el entorno de un ordenador personal y presentando un progresivo ratio de actualizaciones de interconexiones por segundo.

Chips de silicio: Otro de los campos de la investigación en el mundo de las RNA al margen de los simuladores software y aceleradores hardware, es la integración de todos los componentes de computación neuronal en un chip de silicio. Un ejemplo concreto es el chip EEN de la compañía AT&T, que contiene 256 transistores-neuronas y más de 100.000 resistencias-sinapsis. Actualmente este chip está siendo utilizado para aplicaciones de compresión del ancho de banda de imágenes de vídeo para poder ser transmitidas por una línea telefónica. Existen muchas compañías y centros de investigación que están trabajando en el desarrollo de circuitos integrados que realizan computación neuronal. La mayoría de las aplicaciones de estos chips está siendo la simulación de procesos sensitivos como la visión de imágenes y la audición de sonidos.

1.8. Computación Tradicional frente a Computación Neuronal

1.8.1. Programación frente a entrenamiento

Las técnicas tradicionales de programación utilizadas para la solución de un problema requieren la creación de un algoritmo. Un algoritmo consiste en una secuencia de instrucciones que indica el modo en el que debe proceder el sistema, basado en un ordenador, para lograr el fin perseguido, que es la resolución del problema.

El diseño de una secuencia de instrucciones para resolver un problema de contabilidad es relativamente sencillo, pero existen muchos problemas del mundo real en los que resulta difícil realizar un algoritmo que los resuelva. Por ejemplo, imaginemos desarrollar un programa de reconocimiento del rostro de una persona. Hay muchas variaciones de la imagen de una persona, como que presente un rostro serio o un rostro alegre, variaciones en general que deben tenerse en cuenta a la hora de diseñar el algoritmo.

Las RNA, a diferencia de los algoritmos que son instrucciones previamente programadas, deben ser previamente entrenadas. Esto significa que a la red se le muestra en su capa de entrada unos ejemplos y ella misma se ajusta en función de alguna regla de aprendizaje.

1.8.2. Arquitectura

Las RNA presentan una arquitectura totalmente diferente a los ordenadores tradicionales con un único procesador. Las máquinas tradicionales basadas en el modelo de Von Neuman tienen un único elemento procesador, la CPU que realiza todos los cálculos ejecutando todas las instrucciones de la secuencia programada en el algoritmo. Cualquier CPU, realiza más de cien comandos básicos, incluyendo sumas, restas, y desplazamientos entre otros.

Los comandos o instrucciones se ejecutan secuencialmente y sincronizadas con el reloj del sistema. Sin embargo, en los sistemas de computación neuronal cada elemento PE sólo puede realizar uno, o como mucho, varios cálculos. La potencia del procesado de las RNA se mide principalmente por el número de interconexiones actualizadas por segundo durante el proceso de entrenamiento o aprendizaje. Sin embargo, las máquinas de Von Neuman se miden por el número de instrucciones que ejecuta por segundo el procesador central CPU.

La arquitectura de las RNA parte de la organización de los sistemas de procesado en paralelo, es decir, sistemas en los que distintos procesadores están interconectados. No obstante, los procesadores son unidades de computo simples, diseñadas para la integración mediante un sumatorio de los valores de las entradas y con un ajuste automático de las conexiones ponderadas.

1.8.3. Sistemas Expertos

Los sistemas expertos difieren de la programación tradicional en que la base del conocimiento está separada del motor de inferencia (el método del procesado del conocimiento). Esta característica permite que todo el conocimiento adicional pueda ser añadido al sistema sin necesidad de una reprogramación completa. Esta técnica requiere que exista una persona experta en un área y que se puedan crear reglas que codifiquen el conocimiento. En el desarrollo de una red neuronal no hay que programar ni el conocimiento ni las reglas del procesamiento del conocimiento. La red neuronal aprende las reglas del procesamiento del conocimiento mediante el ajuste de las conexiones ponderadas entre las neuronas de distintas capas de la red.

Ademas, mientras que en los Sistemas Expertos el conocimiento se hace explícito en forma de reglas, en la computación neuronal las RNA generan sus propias reglas aprendiendo de los ejemplos que se les muestran en la fase de entrenamiento. El aprendizaje se consigue a través de una regla de aprendizaje que adapta o cambia los pesos de las conexiones en respuesta a los ejemplos de entrada, y opcionalmente también en respuesta a las salidas deseadas. Esta característica de las RNA es lo que permite decir que las redes neuronales aprenden de la experiencia.

Otra característica importante de las RNA es la forma o el modo en que se almacena la información. La memoria o el conocimiento de estas redes está distribuida a lo largo de todas las conexiones ponderadas de la red. Algunas RNA presentan la característica de ser "asociativas" que significa que para una entrada parcial la red elegirá la entrada más parecida en memoria y generará una salida que corresponda a la entrada completa. La naturaleza de la memoria de las RNA permite que la red responda adecuadamente cuando se le presenta una entrada incompleta o con ruido. Esta propiedad suele ser referida como la capacidad de "generalización".

Finalmente, las RNA son tolerantes a los fallos (*Fault Tolerance*). Este término se refiere al hecho de que en muchas RNA el comportamiento de la red seria minimamente modificado si resultaran destruidos varios elementos procesadores PE o se alteraran las conexiones, es decir, el comportamiento varía pero el sistema no se descompone o deja de funcionar. Esta característica se debe a que las RNA tienen la información distribuida a lo largo de toda la red y no está contenida en un único lugar.

1.9. Arquitecturas clásicas frente a Arquitecturas constructivas

Se han propuesto muchos modelos de redes neuronales para aproximar una función en problemas de clasificación y regresión. De todos ellos, las redes neuronales con múltiples capas (FFNNs) son las más populares debido a la flexibilidad de su estruc-

tura, su buena capacidad de representación (aproximadores universales), y el gran número de algoritmos de entrenamiento disponibles.

En general, en el aprendizaje supervisado, la capacidad de generalización y el tiempo de entrenamiento de las redes neuronales artificiales (FFNN) depende en gran medida de factores tales como la arquitectura de la red elegida (número de neuronas ocultas y la topología de conexión entre las neuronas), la elección de la función de activación para cada neurona, la elección el método de optimización y otros (tasa de aprendizaje, pesos iniciales, etc.). Si la elección de la arquitectura de red es muy conservadora, con una arquitectura de muy pocas neuronas, la red podría no ser capaz de ajustarse correctamente al problema a clasificar (*underfitting*). Por otro lado, una arquitectura con muchas neuronas en su arquitectura, aparte de incrementarse el tiempo de entrenamiento, la red podría sobre-ajustarse al problema (*overffiting*), situación nada deseable en la cual se deja de aprender el problema para pasar a un fase de memorización de este.

Las redes neuronales artificiales (FFNNs) con una sola capa oculta en su arquitectura (SLFNNs) son aproximadores universales (UAP) con número suficiente de neuronas en esa capa, es decir, estos modelos son capaces de aproximar cualquier función continua a cualquier grado de precisión deseado [**?**, **?**]. No hay métodos conocidos eficientes para determinar la arquitectura de red óptima para un problema en cuestión. La selección de la arquitectura de red óptima sigue siendo un problema abierto [**?**, **?**, **?**, **?**]. Aunque existen algunas propuestas para solucionar el problema [**?**], no hay un acuerdo general sobre la estrategia a seguir a fin de seleccionar una arquitectura de red neuronal óptima. El método computacionalmente ineficiente de "prueba y error" sigue siendo todavía muy utilizado en las aplicaciones que utilizan redes neuronales artificiales.

Las redes neuronales constructivas o adaptativas son una colección de un grupo de técnicas en las que se construye una red de neuronas adaptada a cada problema particular, usando la información contenida en el conjunto de aprendizaje y evitando pruebas y errores en el diseño de la arquitectura. La adaptación de la estructura de la red puede ser aplicable a tres niveles: la adaptación de la arquitectura, la adaptación funcional y la adaptación de los parámetros de entrenamiento. Estos enfoques se pueden clasificar en dos grupos distintos: evolutivos y no evolutivos.

Muchos algoritmos evolutivos han sido propuestos basándose en técnicas de optimización global, tales como los algoritmos genéticos, la programación genética y estrategias evolutivas[**?**, **?**]. Los métodos de búsqueda globales, como la optimización de colonias de hormigas y optimización de enjambre de partículas, son muy utilizados en la actualidad para determinar la arquitectura óptima durante el aprendizaje [**?**, **?**]. Sin embargo, el enfoque evolutivo es bastante exigente en cuestión de tiempo de ejecución y de parámetros definidos por el usuario [**?**].

1.9.1. Redes neuronales constructivas no evolutivas

A diferencia de los algoritmos de redes neuronales que requieren la definición de la arquitectura antes de que comience el entrenamiento, las redes neuronales constructivas generan la arquitectura a la vez que se realiza el entrenamiento. Se han propuesto muchos métodos para determinar la arquitectura óptima de la red durante el entrenamiento, tales como algoritmos constructivos, con poda, constructivos con poda, y los algoritmos de regularización.

Un algoritmo constructivo agrega neuronas, capas ocultas, y conexiones intentando minimizar la arquitectura de la red durante el entrenamiento. Un algoritmo de poda hace lo contrario, es decir, elimina neuronas, capas ocultas redundantes, y conexiones a partir de redes neuronales muy grandes durante el entrenamiento. Un algoritmo constructivo con poda es un enfoque híbrido en el que la red neuronal puede ser podada después de la finalización del proceso constructivo. También la poda puede ser intercalada con el proceso constructivo. Un método de regularización añade (resta) un término de penalización a la función de error que ha de ser minimizada (maximizada), de manera que el efecto produce una disminución en los pesos sinápticos que no son importantes en la red entrenada. La función de error modificada tiene un forma similar a $E'(W) = E(W) \pm \lambda\, R(W)$, donde, $E(W)$ es la función de error del entrenamiento, $R(W)$ es el término de regularización y λ es un parámetro de regularización que controla la influencia de la función de regularización. La dificultad de utilizar la función de regularización que modifica la función del error está en la elección de un parámetro de regularización adecuado, que a menudo requiere de prueba y error. La regularización puede asimismo utilizarse con algoritmos constructivos y de poda [**?**, **?**].

Las principales ventajas de los algoritmos constructivos frente a los algoritmos de poda son los siguientes:

1. Es relativamente fácil especificar una arquitectura de red inicial en los algoritmos constructivos empezando normalmente con una arquitectura de red compuesta únicamente por una neurona, mientras que en los algoritmos de poda por lo general se desconoce a priori cual debe ser el tamaño de la red inicial. Por lo tanto, se suele elegir una red inicial mucho más grande que realmente se necesita por el problema subyacente en los algoritmos de poda, conduciendo a un proceso de cálculo de la red costoso.

2. Los algoritmos constructivos tienden a construir redes pequeñas debido a su naturaleza de aprendizaje incremental conduciendo a la generación de una red que se corresponde con la complejidad del problema dado, mientras que los algoritmos de poda gastan mucho esfuerzo en podar pesos y neuronas redundantes. Por lo tanto, los algoritmos constructivos son generalmente más económicos (en términos de tiempo de entrenamiento y complejidad de la red / estructura) que los algoritmos de poda.

3. En los algoritmos constructivos, para conseguir una buena generalización, des-

de el comienzo del entrenamiento hasta la finalización de éste, se han de actualizar menos pesos sinápticos que en un algoritmo de poda.

4. Una característica común en los algoritmos constructivos es asumir que las neuronas ocultas ya entrenadas en la red han aprendido parte del modelo, considerándose útil la información que contienen. Por lo tanto, los pesos sinápticos de las neuronas ocultas ya entrenadas son congelados cuando una nueva neurona es añadida a la red, reduciéndose el número de pesos a optimizar a la vez que se reducen el tiempo y la memoria durante la ejecución del algoritmo.
5. En los algoritmos de poda y métodos de regularización se deben ajustar adecuadamente varios parámetros con el fin de obtener una red aceptable con un rendimiento satisfactorio. Este requisito hace que estos algoritmos sean más difícil de utilizar en aplicaciones de la vida real.

Las redes neuronales constructivas (CoNN) son por lo tanto una colección de algoritmos que alteran la estructura de la red en el aprendizaje produciendo automáticamente una red con un tamaño adecuado para un problema concreto [**?**]. Los algoritmos constructivos comienzan con una arquitectura de red mínima, y van añadiendo neuronas, conexiones y capas durante el entrenamiento según necesidad. El proceso de adaptación de la arquitectura se continúa hasta que el algoritmo de entrenamiento genera una solución satisfactoria del problema. Los principales motivos para usar algoritmos constructivos frente a las redes neuronales clásicas son:

1. Flexibilidad para explorar diferentes topologías de redes neuronales durante el aprendizaje.
2. Capacidad de adaptación a la complejidad intrínseca del problema.
3. Buen estimador de la complejidad de un problema.

Nicoletti y Bertini [**?**] evaluaron empíricamente varios algoritmos CoNN de clasificación biclase y multi-clase en la generación de arquitecturas *feedforward*, clasificándolos en dos grupos: 1) Algoritmos que generan una arquitectura adecuada en función del error de clasificación 2) Algoritmos basados en un modelo de aprendizaje secuencial, tal que añaden diferentes neuronas ocultas a la arquitectura de la red que aprenderán diferentes partes del conjunto de entrenamiento.

1.9.1.1. Algoritmos que se basan en la minimización del error de clasificación

Se han propuesto diversos algoritmos constructivos en la literatura siendo los algoritmos de clasificación binaria los más conocidos: *Tower* y *Pyramid* [**?**] de Gallant, *Tiling* [**?**] de Mézard y Nadal, *Upstart* [**?**] de Frean, *RDP* [**?**] de Tajine y Elizondo*, y perceptrón cascade* [**?**] de Burgess, de los que posteriormente se han generado diferentes versiones multi-clase *MTower*, *MPyramid*, *MUpstart, Mtiling* y *Cascade Mperceptrón.*

Tower La idea básica es muy simple: el algoritmo comienza con una arquitectura de red de n neuronas en la capa de entrada y una única neurona en la capa de salida, e intenta clasificar el problema. Si no lo consigue, el algoritmo añade una nueva neurona como neurona de salida congelando los pesos del resto de la red. La nueva neurona añadida en una etapa k tendrá como entrada la neurona generada en la etapa $k-1$ y las n neuronas de la capa de entrada. En la figura 1.6a se muestra un esquema típico de la arquitectura generada por este algoritmo.

Pyramid El algoritmo de la pirámide es muy similar al algoritmo de la torre, con la única diferencia de que cada neurona añadida tiene como entrada cada una de las neuronas generadas ya por la red, incluida la capa de entrada tal y como muestra la figura 1.6b.

Tiling El algoritmo de Tiling genera redes de varias capas ocultas. En un primer paso, el algoritmo genera una arquitectura de red inicial de n neuronas en la capa de entrada y una única neurona en la capa de salida. La primera neurona de cada capa es conocida como neurona maestra y es la responsable inicial de clasificar correctamente el problema. Si no lo consigue, el algoritmo añade neuronas en la misma capa hasta que consigue que todos los datos de entrada estén fielmente representados por la capa, esto es, que no existan dos patrones de entrada con distinta clase de salida que generen los mismos valores de salida para las neuronas de la capa que se esta creando. Tiling genera así sucesivas capas, de tal manera que cada capa es más pequeña que la anterior y cada capa sólo tiene como entrada el resultado de la capa anterior. El algoritmo finalizará su aprendizaje cuando: la red converge, la neurona añadida en la nueva capa (neurona maestra de la capa) genera un resultado peor que la neurona maestra de la capa anterior, se alcanza un valor máximo de capas o un valor máximo de neuronas definido antes de comenzar el algoritmo.

Upstart El algoritmo Upstart construye redes con una topología en forma de árbol binario de puertas umbrales. Cada neurona de la red intentará clasificar diferentes funciones binarias objetivo con los mismos patrones de entrada iniciales, ya que la primera neurona intenta clasificar el problema inicial, y cada una de las neuronas siguientes intentaran clasificar el error generado por su neurona padre. La red comienza su aprendizaje con una arquitectura inicial de n neuronas en la capa de entrada y una única neurona en la capa de salida, la cual tiene como entrada las n neuronas de la capa de entrada. Si la neurona no consigue clasificar el problema correctamente, es porque ha clasificado mal uno o varios ejemplos de entrada. Estos ejemplos mal clasificados pueden ser divididos en dos grupos: ejemplos con salida binaria "verdadero" que han sido clasificados erróneamente por la neurona con un valor "falso", o el caso inverso, ejemplos con salida binaria "falso" que han sido clasificados erróneamente por la neurona con un valor "verdadero". Denominaremos función de error "positivo" / "negativo" a la función generada por todos los patrones

de entrada del conjunto original a los que se les modifica la salida a "true" cuando se produce un error "positivo" / "negativo" y "false" en caso contrario. Cuando se produce un error "positivo" / "negativo", la red añade neuronas "hijas" a la neurona padre para corregir únicamente el error de su "padre. Estas neuronas tienen como entrada la capa de entrada de la red y como función objetivo a clasificar, el error "positivo" y "negativo" generado por su neurona padre, quedando conectáda su salida a su neurona "padre" añadiendo dos entradas más a esta (ver Figura 1.6c) con valores de pesos sinápticos lo suficientemente "altos" / "bajos" como para corregir los errores "positivos" / "negativos" originales. Este procedimiento se ejecutara recursivamente hasta la convergencia final del algoritmo.

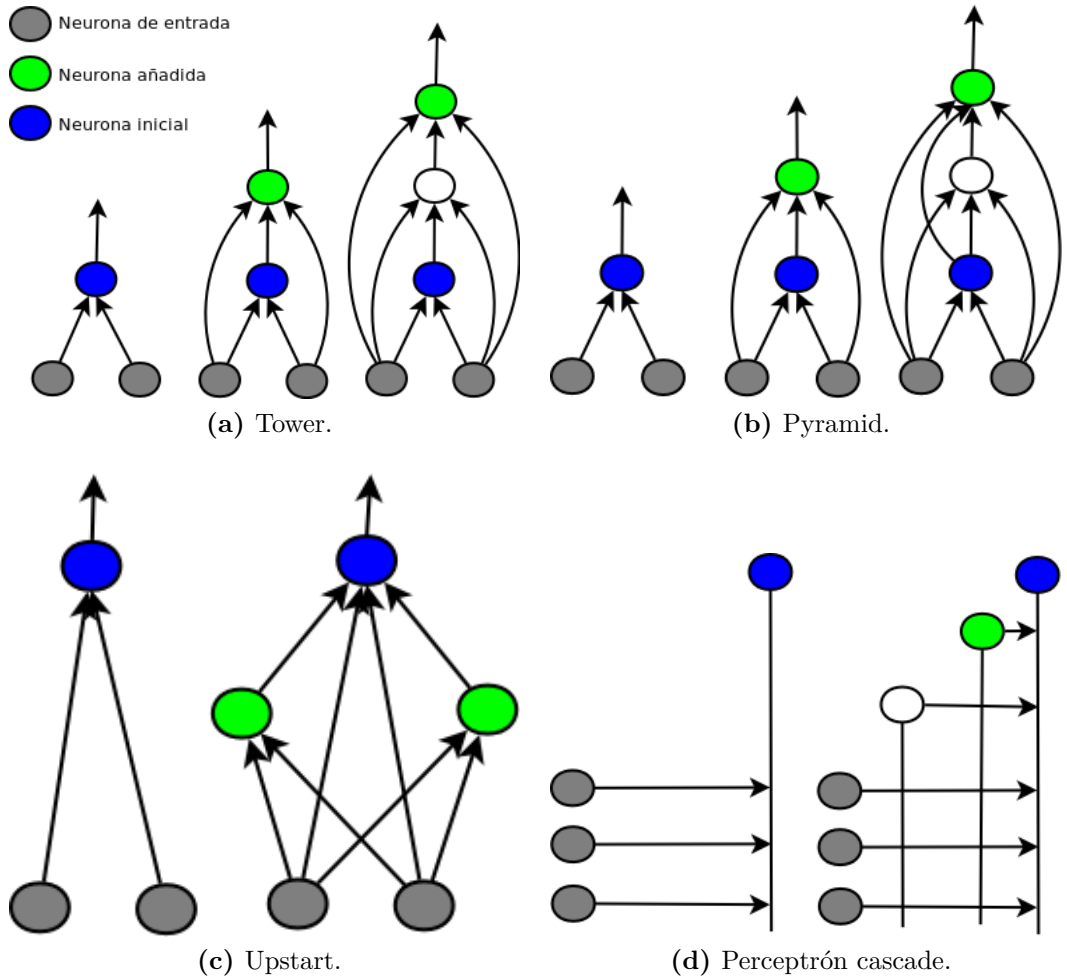
Perceptrón cascade: El algoritmo genera arquitecturas de red con una topología similar a la generada por el algoritmo de *cascade correlation* [**?**] con la única diferencia de que cada nodo está formado por una puerta umbral. En la figura 1.6d se muestra un esquema típico de la arquitectura generada por este algoritmo.

Perceptrón recursivo determinista: El perceptrón recursivo determinista (RDP) genera arquitecturas de red *feedforward* similares a las arquitecturas generadas por el algoritmo *Pyramid* asegurándose de que en cada ciclo del aprendizaje, se genera una nueva neurona que separa un conjunto de ejemplos pertenecientes a una misma clase del resto de ejemplos del conjunto de aprendizaje.

1.9.1.2. Algoritmos que se basan en un modelo de aprendizaje secuencial

Los algoritmos CoNN basados en un modelo de aprendizaje secuencial añaden neuronas a la red que clasifican parcialmente el conjunto de entrenamiento. Los algoritmos más conocidos de redes neuronales constructivas basados en un modelo de aprendizaje secuencial son: *Irregular Partitioning Algorithm* [**?**] de Marchand, *Carve [?]* de Young y Downs, y el algoritmo de descomposición en restricciones [**?**] de Draghici.

Carve: Este algoritmo construye una red *feedforward* con una sola capa oculta de neuronas umbrales que clasifica ejemplos de entrenamiento con entrada real, ampliando la idea propuesta por Marchand en el algoritmo *Irregular Partitioning Algorithm* [**?**]. Para ello, en cada ciclo del aprendizaje se genera una nueva neurona en la capa oculta que separa un conjunto de ejemplos pertenecientes a una misma clase del resto de ejemplos del conjunto de aprendizaje, eliminándose estos ejemplos del conjunto de aprendizaje para posteriores iteraciones. Esta fase del entrenamiento termina cuando no quedan ejemplos en el conjunto de aprendizaje con diferente clase de salida, tras lo cual, en una segunda etapa de la fase de entrenamiento, se entrena la neurona de salida de la capa oculta recibiendo como entrada la salida de la anterior capa recientemente creada. La Fig. 1.7 muestra un ejemplo de como sería la construcción de la capa oculta durante el entrenamiento.



(a) Tower. **(b)** Pyramid.

(c) Upstart. **(d)** Perceptrón cascade.

Figura 1.6: Arquitecturas generadas por los algoritmos de redes neuronales constructivos más conocidos.

Descomposición en restricciones (CDB): La descomposición en restricciones es una técnica de red neuronal constructiva que clasifica ejemplos de problemas biclase con entrada real construyendo una red de cuatro capas: una capa de entrada, una capa oculta que implementa los hiperplanos separadores, una capa AND, y finalmente una capa OR, aunque ésta ultima podría no ser necesaria. El algoritmo, en una primera fase del aprendizaje, genera las neuronas de la capa oculta de los hiperplanos separadores. Para ello, CBD comienza escogiendo aleatoriamente un patrón de cada clase. Si llamamos x_A al patrón de la clase C_A y x_B al patrón de la clase C_B, el algoritmo inicialmente entrena cada neurona con sólo estos dos patrones generando un hiperplano h_i que los clasifica correctamente, ya que este es un problema linealmente separable. Posteriormente, se irán añadiendo patrones de forma aleatoria de cada clase al conjunto de aprendizaje de la neurona hasta que el problema deje de ser linealmente separable, terminando el aprendizaje con la última configuración

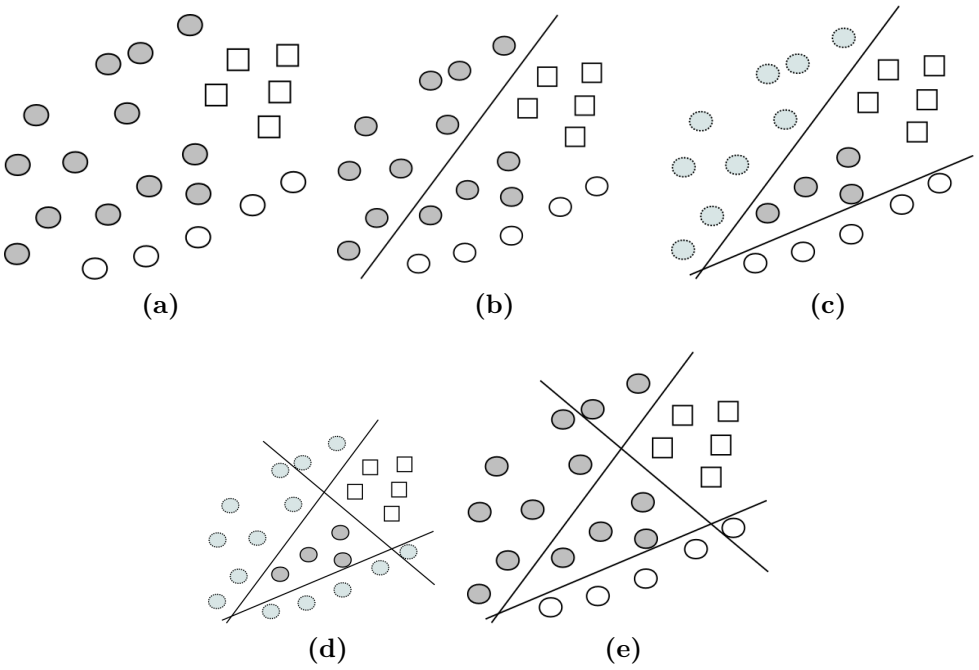


Figura 1.7: Construcción de una red CARVE.

linealmente separable que obtuvo la neurona.

Cada neurona creada genera un hiperplano separador h_i que divide el espacio de entrada en dos regiones R^i_+ y R^i_-. Este proceso se repite recursivamente con cada región generada (R^i_+ y R^i_-) generando nuevas neuronas que continúan dividiendo el espacio de entrada en más regiones hasta que las regiones generadas solo contengan patrones de una clase o no contengan patrones, finalizando así la creación de la capa oculta de hiperplanos separadores.

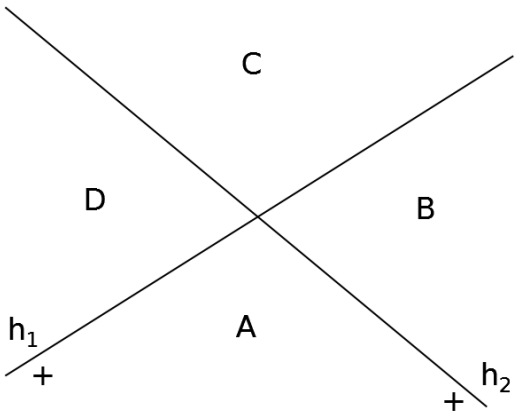


Figura 1.8: Representación interna de la capa oculta de la red CDB.

En este punto, cada región contiene sólo patrones de una única clase C_i, y esta definida por la disyunción de hiperplanos separadores h_i que la forman. Por ejemplo,

La Fig. 1.8 muestra un ejemplo en el que dos hiperplanos están generando 4 regiones. El carácter "+" indica la región positiva generada por cada uno de los hiperplanos. En este caso, las regiones generadas por los hiperplanos pueden ser descritas de la siguiente forma: $A = h_1 h_2$, $B = h_1 \overline{h}_2$, $C = \overline{h}_1 \overline{h}_2$, $D = \overline{h}_1 h_2$. Por lo que si la clase C_A está formada por las regiones B y D, se podría describir la clase como $C_A = h_1 \overline{h}_2 + \overline{h}_1 h_2$, y análogamente, la clase $C_B = h_1 h_2 + \overline{h}_1 \overline{h}_2$ generando una arquitectura de red de cuatro capas en este caso tal y como se ve en la Fig. 1.9

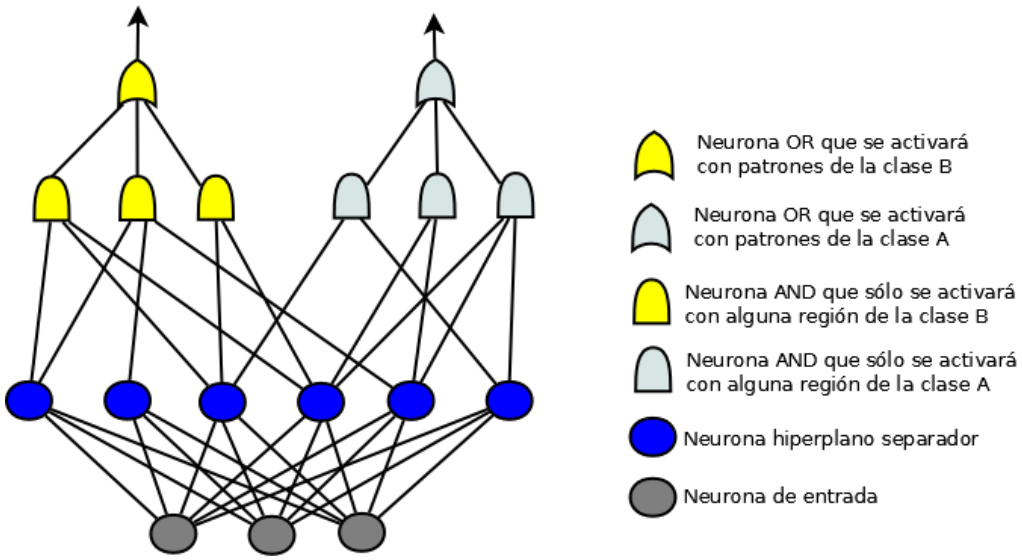


Figura 1.9: Arquitectura generada por el algoritmo CDB.

1.10. Objetivos

El principal objetivo de esta Tesis Doctoral consiste en:

- Proponer, desarrollar y evaluar nuevos algoritmos de redes neuronales constructivas que incorporen alguna mejora o variación significativa sobre las técnicas existentes en la bibliografía.
- Su aplicación a problemas reales del ámbito de la ingeniería.

Con este objetivo se han desarrollado dos nuevos algoritmos constructivos: DASG y C-Mantec. DASG es un algoritmo de redes neuronales constructivo basado en un enfoque secuencial que genera redes con una capa oculta de puertas umbrales, siendo capaz de clasificar y sintetizar conjuntos de datos binarios con salida biclase (**??**). C-Mantec es un algoritmo con un enfoque basado en reducir el error de clasificación que incorpora el concepto biológico de competición neuronal en el proceso de aprendizaje. Esta competencia evita la congelación de los pesos sinápticos de la red durante el proceso constructivo, procedimiento estándar en la mayoría de los algoritmos constructivos existentes. Los resultados obtenidos muestran que el efecto

de la aplicación de la competencia en el algoritmo es beneficiosa y relevante para la construcción de arquitecturas más compactas y para la obtención de buenos valores de generalización.

1.11. Estructura de la tesis

Con posterioridad a este capítulo introductorio, el presente documento ha sido estructurado en los siguiente capítulos: El Capítulo 2 introduce el algoritmo constructivo de descomposición de funciones booleanas para síntesis y generalización, DASG. Se presentan los conceptos matemáticos necesarios para explicar con detalle el algoritmo, y se muestra los resultados obtenidos en la síntesis de funciones booleanas. En el Capítulo 3 se introduce el algoritmo constructivo C-Mantec, algoritmo de redes neuronales basado en un aprendizaje competitivo que incorpora un novedoso sistema de filtrado de ejemplos para evitar el sobre-entrenamiento. Posteriormente, en el capítulo 4 se analizan en detalle dos aplicaciones prácticas de los algoritmos propuestos: 1) predicción de recidiva temprana en cáncer de mama y extracción de conocimiento usando el algoritmo constructivo DASG, y 2) uso del algoritmo constructivo C-Mantec para la reprogramación eficiente de redes de sensores inalámbricas. Finalmente, en el capítulo 5, se presentan y discuten las conclusiones de esta tesis doctoral.