Introduccion a las Redes de Neuronas Artificiales

Article · August 2013

CITATIONS
4

READS
350

1 author:

Marcos Gestal
University of A Coruña
87 PUBLICATIONS **262** CITATIONS

SEE PROFILE

Introducción a las Redes de Neuronas Artificiales

Marcos Gestal Pose
Depto. Tecnologías de la Información y las Comunicaciones
Universidade da Coruña
http://sabia.tic.udc.es/~mgestal
mgestal@udc.es

Índice

Índice de figuras

El siguiente tutorial no pretende ser un documento exhaustivo acerca de las Redes de Neuronas Artificiales. Más bien una referencia avanzada que sirva para permita al lector conocer la terminología, los conceptos claves y una bibliografía de base.

1. Introducción

Desde la primera mitad del siglo XX se han empezado a desarrollar modelos computacionales que han intentado emular el comportamiento del cerebro humano [16]. Aunque se han propuesto una gran cantidad de ellos, todos usan una estructura en red en la cual los nodos o neuronas son procesos numéricos que involucran estados de otros nodos según sus uniones. Una clase de estos modelos computacionales son las Redes de Neuronas Artificiales [10].

Las Redes de Neuronas Artificiales (RNA) se han hecho muy populares debido a la facilidad en su uso (ver Figura 1) e implementación y la habilidad para aproximar cualquier función matemática. Las Redes de Neuronas Artificiales, con su marcada habilidad para obtener resultados de datos complicados e imprecisos, pueden utilizarse para extraer patrones y detectar tramas que son muy difíciles de apreciar por humanos u otras técnicas computacionales.

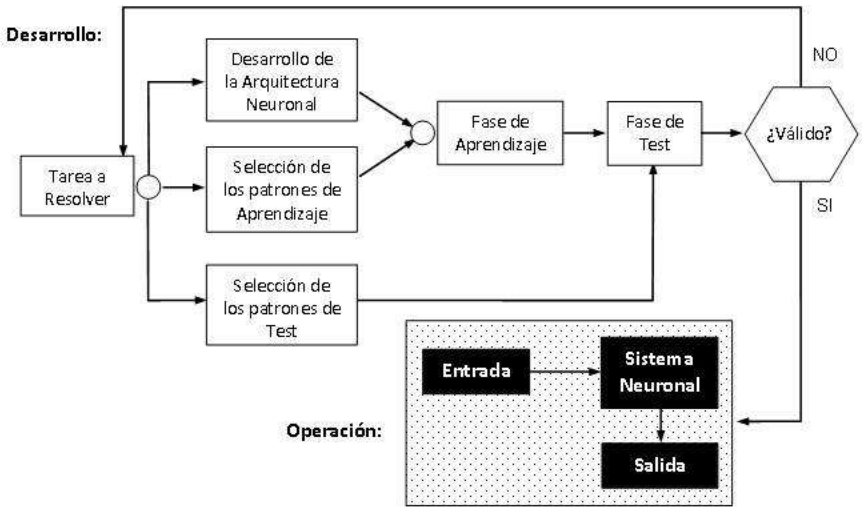

Figura 1: Esquema básico del trabajo con RNA

El primer modelo de neurona artificial fue propuesto por McCulloch y Pitts, donde se modelizaba una estructura y un funcionamiento simplificado de las neuronas del cerebro, considerándolas como dispositivos con n entradas, una única salida y sólo dos estados posibles: activa o inactiva [16].

Una red era, en ese planteamiento inicial, una colección de neuronas de McCulloch y Pitts, todas sincronizadas, donde las salidas de unas neuronas estaban conectadas a las entradas de otras. Algunos de los planteamientos de McCulloch y Pitts se han mantenido desde 1943 sin modificaciones, otros por el

contrario han ido evolucionando, pero todas las formalizaciones matemáticas que se han realizado desde entonces, sobre las Redes de Neuronas Artificiales, aún sin pretender ser una modelización exacta de las redes de neuronas biológicas, sí han resultado un punto de partida útil para el estudio de las mismas.

Una de las definiciones que se estima más certera de Red de Neuronas Artificiales es la siguiente: "Las redes neuronales son conjuntos de elementos de cálculo simples, usualmente adaptativos, interconectados masivamente en paralelo y con una organización jerárquica que le permite interactuar con algún sistema del mismo modo que lo hace el sistema nervioso biológico" [13].

Su aprendizaje adaptativo, auto-organización, tolerancia a fallos, operación en tiempo real y fácil inserción dentro de la tecnología existente, han hecho que su utilización se haya extendido en áreas como la biológica, financiera, industrial, medio ambiental, militar, salud, etc. [11]. Están funcionando en aplicaciones que incluyen identificación de procesos [9], detección de fallos en sistemas de control [1], modelación de dinámicas no lineales [17], control de sistemas no lineales [18][22] y optimización de procesos [19].

En general, se puede encontrar que una Red de Neuronas Artificiales se suele caracterizar por tres partes fundamentales: la topología de la red, la regla de aprendizaje y el tipo de entrenamiento.

En este afán de emular el cerebro, esto es simular tanto su estructura como su funcionamiento, se han desarrollado numerosos modelos de Redes de Neuronas Artificiales [8], entre los que se pueden mencionar: Perceptron (1957), Adeline y Madeline (1960), Avalancha (1967), Retropropagación (1974), Hopfield y SOM (1980), ART (1986), etc. De los modelos anteriores se puede apreciar que esta idea tiene más de 40 años, sin embargo, sólo en las últimas décadas se ha desarrollado la tecnología que permita su aplicación de manera eficiente.

Cabe destacar, para concluir esta breve introducción, que las Redes de Neuronas Artificiales, gracias al masivo paralelismo de su estructura, gozan de una serie de ventajas:

- *Aprendizaje adaptativo*. Capacidad de aprender a realizar tareas basadas en un entrenamiento o una experiencia inicial.
- *Autoorganización*. Una red neuronal puede crear su propia organización o representación de la información que recibe durante la etapa de aprendizaje.
- *Tolerancia a fallos*. Gracias a poseer la información distribuída o vía información redundante la destrucción parcial de una red puede conducir a una degradación de su estructura; sin embargo, algunas capacidades de la red se pueden retener, incluso sufriendo daños considerables.
- *Capacidad de generalización*. Ante la entrada de datos nuevos es capaz de producir resultados coherentes de acuerdo con la naturaleza del problema para el cual han sido entrenadas.
- *Operación en tiempo real*. El cómputo neuronal puede realizarse en paralelo, bien vía software o mediante máquinas especiales para obtener esta ventaja (hardware conexionista o masivamente paralelo).

Pero los sistemas neuronales no están exentos de ciertos inconvenientes. Uno importante es que habitualmente realizan un complejo procesamiento que

supone millones de operaciones, por lo que no es posible seguir paso a paso el razonamiento que les ha llevado a extraer sus conclusiones. Sin embargo, en redes pequeñas, mediante simulación o por el estudio de los pesos sinápticos sí es posible saber, al menos, qué variables de las introducidas han sido relevantes para tomar la decisión.

2. Fundamentos y Conceptos Básicos

Warren McCulloch y Walter Pitts pueden ser considerados como los padres de las Redes de Neuronas Artificiales, ya que fueron los primeros en diseñar una neurona artificial. En los años 40 suponían que las neuronas biológicas eran de carácter binario, lo cual resulta ser bastante inexacto, pero sirvió de base para posteriores estudios sobre el sistema nervioso.

Así, McCulloch y Pitts proponen el modelo de neurona que lleva su nombre, el cual es un dispositivo binario con un umbral fijo que hay que superar para que cambie de estado. Recibe sinapsis excitadoras de otros elementos, los cuales tienen la característica de ser del mismo valor. Puede recibir sinapsis inhibidoras que son de acción total, lo cual supone que la recepción de una impide el cambio de estado del elemento, sin importar la cantidad de sinapsis excitadoras que hubiese.

La neurona artificial o elemento formal está conceptualmente inspirada en la neurona biológica. Esto es, los investigadores están en su inmensa mayoría pensando en la organización cerebral cuando consideran configuraciones y algoritmos de Redes de Neuronas.

Se va a considerar una neurona como un elemento formal o módulo o unidad básica de la red que recibe información de otros módulos o del entorno; la integra, la computa y emite una única salida que se va a transmitir idéntica a múltiples neuronas posteriores [24].

Considérese a los pesos sinápticos como referencia a la noción biológica de la fuerza de unión entre los elementos; es decir, a la fuerza de la sinapsis. Se considera que una sinapsis es fuerte; es decir, tiene un alto grado de conexión, cuando la información que transmite contribuye, en gran medida, a un nuevo estado o a la alteración que se produzca en la neurona receptora y, por tanto, en la respuesta que ésta elabora.

En una red de neuronas existe un peso o fuerza sináptica que va a ser un valor numérico que pondera las señales que se reciben por sus entradas. Este peso será un valor que determina la fuerza de conexión entre 2 neuronas. Cuando se evalúa una neurona se debe calcular el conjunto de todas las fuerzas o valores (denominado NET) que se reciben por sus entradas. Una vez calculado el valor conjunto de todas las entradas se aplica una función de activación (FA) que determinará el valor del estado interno de la neurona y que será lo que se transmita a su salida.

La combinación de las señales que recibe una neurona se puede calcular como muestra en la ecuacion 1:

$$NET_i(t) = \sum_{j=1}^{N-1} [W_{ij} \cdot O_j \cdot (t-1)] \tag{1}$$

dónde Wij representa el peso de la conexión entre una neurona emisora j y neurona receptora i.

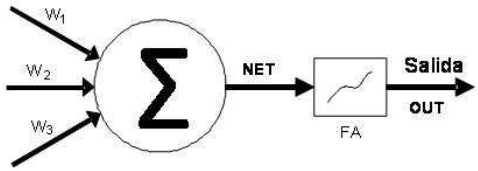

Figura 2: Neurona Artificial

La función de activación y la función de transferencia son las encargadas de definir el nuevo estado de activación Ai y la respuesta Oi de la neurona. Tanto el estado de activación como la salida de la neurona van a variar en función de las entradas que recibe en un determinado momento y del estado de activación previo que tenga esa neurona (ver figura 2).

3. Función de activación

La función de activación de una neurona es la encargada de relacionar la información de entrada de la neurona con el siguiente estado de activación que tenga esa neurona.

Existen dos modelos de función de activación:

- *Modelos acotados:* El valor de la activación de la neurona puede ser cualquiera dentro de un rango continuo de valores.
- *Modelos No acotados:* No existe ningún límite para los valores de activación.

Cuando se diseña una red debe establecerse cómo van a ser los valores de activación de cada neurona y se debe decidir la función de activación (FA) con la que cada neurona procesará las entradas. Por lo tanto, tal y como se muestra en la ecuación 2 la función de activación va a actuar sobre las señales de entrada, sobre los pesos sinápticos asociados con cada entrada y sobre el valor de activación que tenía la neurona en el momento de recibir las señales.

$$A_i(t) = FA(A_i(t-1), NET_i(t-1)) \tag{2}$$

FA = Función de activación.

4. Tipos de neuronas

La linealidad de las funciones que definen a los elementos de la red es quizás lo que va a proporcionar la característica más definitoria. Así, se pueden clasificar las neuronas en lineales y no lineales.

4.1. Neuronas lineales

Una neurona es lineal cuando su salida es linealmente dependiente de sus entradas, es decir, proporcional a las funciones de transferencia y de activación (ver Figura 3.

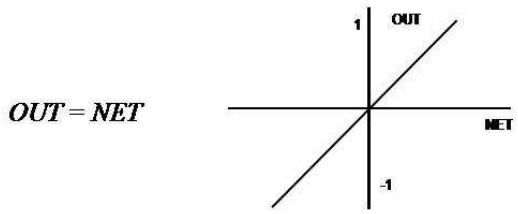

Figura 3: Función de Transferencia Lineal

Esto conlleva ciertos problemas como la falta de persistencia en las respuestas, de modo que cambios muy pequeños en las entradas pueden producir fluctuaciones bastante grandes en las respuestas, o la falta de adecuación simultánea, pues es imposible que con neuronas lineales la respuesta de una neurona se adapte tanto a señales grandes como a pequeñas.

4.2. Neuronas no lineales

En estas neuronas, o bien la función de activación, o bien la función de transferencia (o ambas) son funciones no lineales, dando lugar a que la respuesta de la neurona no sea función lineal de sus entradas.

Este tipo de neuronas va a producir respuestas acotadas, desapareciendo los problemas de fluctuación y la falta de adecuación a señales pequeñas y grandes.

Como ejemplo de funciones no lineales se pueden destacar la función umbral (Figura 4), la función sigmoide (Figura5) o la función hiperbólica tangente (Figura6).

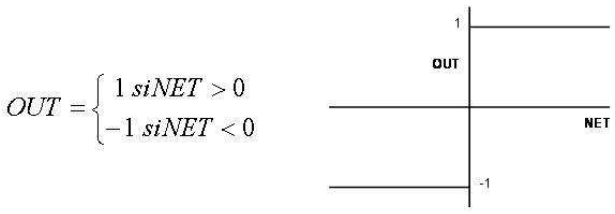

Figura 4: Función de Transferencia Umbral

$$OUT = \frac{1}{1 + e^{-NET}}$$

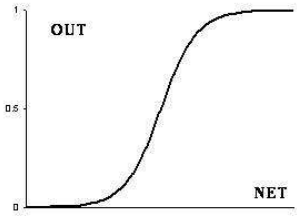

Figura 5: Función de Transferencia Sigmoide

$$OUT = \frac{1 - e^{-NET}}{1 + e^{-NET}}$$

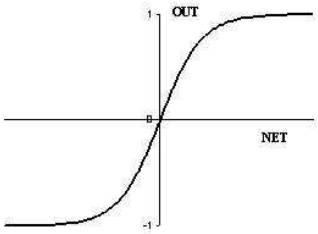

Figura 6: Función de Transferencia Hiperbólica-Tangente

5. Flujo de datos en las Redes de Neuronas Artificiales

5.1. Redes alimentadas hacia delante

Las redes alimentadas hacia delante –generalmente conocidas como redes feedforward– son aquellas en las que, como su nombre indica, la información se mueve en un único sentido, desde la entrada hacia la salida (Figura 7). Estas redes están clásicamente organizadas en "capas". Cada capa agrupa a un conjunto de neuronas que reciben sinapsis de las neuronas de la capa anterior y emiten salidas hacia las neuronas de la capa siguiente. Entre las neuronas de una misma capa no hay sinapsis.

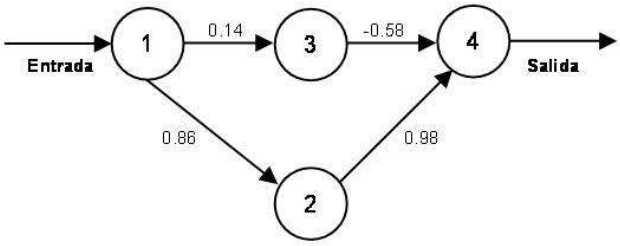

Figura 7: RNA alimentada hacia delante

En este tipo de redes existe al menos una capa de entrada, formada por las neuronas que reciben las señales de entrada a la red y una capa de salida,

formada por una o más neuronas que emiten la respuesta de la red al exterior. Entre la capa de entrada y la de salida existen una o más capas intermedias.

En redes así construidas es evidente que la información sólo puede moverse en un sentido: desde la capa de entrada hasta la capa de salida, atravesando todas y cada una de las capas intermedias una sóla vez.

El hecho de que no haya conexión entre las neuronas de una misma capa hace que no haya tiempos de espera en los que las neuronas estén interactuando unas sobre otras hasta que toda la capa adquiera un estado estable. Se trata por tanto de redes rápidas en sus cálculos.

5.2. Redes con Retroalimentación total o parcial

En este tipo de redes los elementos pueden enviar estímulos a neuronas de capas anteriores, de su propia capa o a ellos mismos, por lo que desaparece el concepto de agrupamiento de las neuronas en capas (Figura 8). Cada neurona puede estar conectada a todas las demás; de este modo, cuando se recibe información de entrada a la red, cada neurona tendrá que calcular y recalcular su estado varias veces, hasta que todas las neuronas de la red alcancen un estado estable. Un estado estable es aquel en el que no ocurren cambios en la salida de ninguna neurona. No habiendo cambios en las salidas, las entradas de todas las neuronas serán también constantes, por lo que no tendrán que modificar su estado de activación ni su respuesta, manteniéndose así un estado global estable [21].

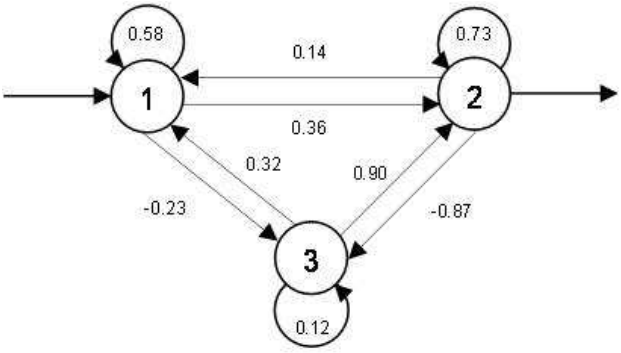

Figura 8: RNA con retroalimentación

Las redes retroalimentadas emulan más fielmente la estructura del cerebro humano, en donde los fenómenos de retroalimentación son fundamentales.

6. Información en las Redes de Neuronas Artificiales

En el cerebro todas las neuronas actúan del mismo modo. Una neurona concreta no puede distinguir características como “rojo” o “frío”. Lo único que una neurona biológica distingue es el potencial de membrana, su estado de

activación y sus secuencias de activación. Estos tres últimos elementos pueden ser expresados numéricamente.

La información de entrada en números tiene que ser adecuada para las funciones de activación y transferencia de la red y se deben distribuir esas informaciones O_j entre las neuronas de la capa de entrada de forma adecuada.

Se observa, por tanto, que a la hora de diseñar una red neuronal es necesario realizar un análisis tanto de la información de entrada que hay que suministrarle para presentarle el problema, como de la información de salida que la red proporcionará como solución a dicho problema.

Si los datos de entrada, es decir, la información del problema que se desea que la red de neuronas resuelva, no se pueden representar mediante valores dicotómicos, se tendrán que modificar las funciones de la red para hacerla así capaz de trabajar con valores O_j con los que se puedan representar los datos.

Si, por ejemplo, se trabaja con información incierta, en donde sólo se pueda dar un valor de la probabilidad de que un determinado elemento pertenezca a un conjunto, se necesitará que la función sea continua para poder expresar valores de probabilidad. De esta manera el rango de los O_j variará de 0 a 1.

Si los valores de O_j son continuos y los datos también, se tendrá que volver a normalizar los datos de entrada para hacerlos corresponder con el rango de valores que los O_j pueden tomar. Es decir, es necesario reajustar la escala de los datos para que se adapte a la escala de los O_j. Así, si por ejemplo un dato de entrada es una longitud en metros (desde 0 a 1.000 m.) y los O_j se mueven desde -1 a 1 se tendrá que establecer una correspondencia en la que a la longitud de 0 m. le corresponda -1, a 1.000 m. le corresponda +1, a 500 m. le corresponda 0, etc.

En cuanto al proceso de normalización se pueden realizar tres clases de transformación de los datos: transformación lineal, estandarización estadística y funciones matemáticas. La transformación lineal es la técnica más empleada, en este caso, se escalan los valores al rango [0, 1] o [-1, 1] de forma lineal. El objetivo es asegurar que todas las variables (entradas de la RNA) reciben igual atención en el proceso de entrenamiento y son escaladas de acuerdo a los límites de las funciones de activación utilizadas en las neuronas de la capa de salida de la red.

En la estandarización estadística, el escalado involucra la resta de una cierta medida, como por ejemplo la media aritmética, y dividir por un cierto valor, como por ejemplo la desviación estándar. Cualquier escalado que establezca una tendencia central próxima a cero suele ser beneficioso en el proceso de inicialización de los pesos de la red.

La última transformación se basa en funciones matemáticas como por ejemplo aplicar la función logaritmo sobre los datos, con el objetivo de estabilizar la variancia [7]. El aplicar el logaritmo o la raíz cuadrada, se usa principalmente para aproximar los datos a distribuciones Gausianas con el objetivo de minimizar el efecto de los valores extremos del rango. Se ha demostrado [4][23] que transformar las entradas de la red para que sigan una distribución uniforme puede ayudar a un mejor proceso de conversión y transformación en los valores de salida por la red de neuronas y, en consecuencia, un mejor proceso de entrenamiento de la red. Una forma de conversión a una distribución uniforme cuando los datos no siguen ninguna distribución probabilística conocida es la ecualización del histograma [15].

7. Tipos de problemas

Una vez decidido cómo representar la información, el siguiente paso es considerar qué tipo de problema va a resolver la red. La mayoría de los problemas se dividen en dos grandes grupos y, consecuentemente, existen dos grandes tipos de redes de neuronas: redes de clasificación y redes de evaluación. Las redes del primer tipo, asocian una determinada configuración de entrada, o patrón de entrada, con una salida. Se denomina a estas redes, redes asociativas, clasificadoras o reconocedoras de patrones [2][3]. El segundo tipo de redes, maneja problemas en los que lo que se pide es un "juicio", por lo que se denominan redes evaluadoras. Otro grupo de Redes de Neuronas Artificiales lo constituyen modelos específicos para tareas especiales como: restauración de patrones, predicción, etc. Funcionamiento

El funcionamiento básico de las redes neuronales abarca dos grandes etapas. La primera comprende todas las fases de creación y desarrollo de la misma, y posteriormente viene la fase de funcionamiento real o fase de ejecución, durante la cual la RNA ya es operativa y, tanto su estructura interna como los valores de los pesos de las conexiones no vuelven a ser modificados, a pesar de la existencia de varios tipos de redes en los que esto ocurre. Durante esta fase se usa la RNA como si se tratara de cualquier otro programa informático convencional, y es cuando se utiliza de forma efectiva para resolver los problemas para los que ha sido diseñada.

7.1. Fase de creación y desarrollo

Esta etapa comprende todas las fases necesarias para el desarrollo de una RNA, y comprende las siguientes etapas:

- Diseño de la arquitectura.
- Entrenamiento de la red.
- Validación de la red.

En los siguientes epígrafes se detallan cada una de las referidas fases.

7.1.1. Diseño de la arquitectura

En la fase de diseño de la arquitectura, también denominada diseño de la topología, se determina el número de neuronas que tendrá la red, así como su disposición en capas y la conectividad entre las mismas. A partir de un análisis de los datos del problema, se determinan cuántas entradas y salidas tiene la red, así como el número de neuronas y cómo estas están distribuidas en capas e interconectadas entre sí. Esta etapa es crítica, puesto que la topología de la red determina la capacidad de representatividad de la misma, y, por lo tanto, la cantidad de conocimiento que puede albergar. La topología de la red debe adecuarse al problema a resolver, y la no existencia de técnicas que realicen esta función hace que haya que recurrir a la experiencia y a la técnica de *ensayo y error*, probando varias topologías distintas, para finalmente conseguir una que se adapte de forma satisfactoria al problema.

Esta etapa también alberga el determinar las funciones de activación y transferencia que se usarán (habitualmente, las mismas para todas las neuronas).

7.1.2. Fase de Entrenamiento

Norbert Wiener en su libro "Dios y el Golem" [25] proporciona tal vez una de las definiciones más neutrales de aprendizaje de todas las conocidas. Allí definió un sistema que aprende como: "Un sistema organizado puede definirse como aquel que transforma un cierto mensaje de entrada en uno de salida, de acuerdo con algún principio de transformación. Si tal principio está sujeto a cierto criterio de validez de funcionamiento, y si el método de transformación se ajusta a fin de que tienda a mejorar el funcionamiento del sistema de acuerdo con ese criterio, se dice que el sistema aprende".

Esta definición es válida para definir el aprendizaje bajo los dos puntos de vista en que éste puede entenderse: el ontogenético, que es el aprendizaje de un individuo humano o no, y el filogenético, aprendizaje que afecta a la especie.

Uno de los principales objetivos de los sistemas autónomos es emular la habilidad que posee el hombre para interactuar con el ambiente y aprender de dichas interacciones. Es así como necesitan de una estructura flexible, capaz de desempeñarse en ambientes de operación dinámicos sujetos a diversas incertezas y perturbaciones. Dentro de las aproximaciones existentes en la teoría de Inteligencia Artificial, surgen las Redes de Neuronas Artificiales como elementos capaces de proveer de dicha estructura flexible, mediante la integración con diversos sistemas de aprendizaje. Tales sistemas están orientados hacia diferentes operaciones y pueden ser clasificados en dos tipos [5][14]: aprendizaje supervisado y no supervisado, y dentro de este, aprendizaje auto-organizativo y aprendizaje por reforzamiento[12].

Una vez diseñada la arquitectura de la red (capas y número de neuronas por capa) y las funciones que la regirán, se tiene que proceder a entrenar a la red para que *aprenda* el comportamiento que debe tener; es decir, para que aprenda a dar la respuesta adecuada a la configuración de estímulos o patrones de entrada que se le presenten [6][20].

Una excepción a esta regla general la constituyen las redes de Hopfield, que no son entrenadas sino construidas, de modo que tengan ya inicialmente el comportamiento deseado. Por este motivo, se ha dicho que las redes de Hopfield simulan el comportamiento *instintivo* mientras que las demás redes simulan el comportamiento *aprendido*.

En la vida de las redes con comportamiento aprendido se distinguen dos periodos o fases claramente diferenciados. Durante la fase de aprendizaje se entrena a la red para que vaya modificando sus pesos sinápticos, adaptándolos paulatinamente para que la respuesta de la red sea la correcta. Después, viene la fase de funcionamiento real o fase de ejecución, durante la cual la red ya es operativa y sus pesos sinápticos no volverán a ser modificados. Durante esta fase se usa la red como si se tratara de cualquier otro programa informático convencional.

7.1.3. Aprendizaje supervisado

Con esta técnica de aprendizaje el entrenamiento consiste en presentarle a la red repetitivamente patrones de estímulos de entrada pertenecientes a un juego de ensayo. El juego de ensayo está formado por parejas "patrón de estímulos - respuesta correcta" y debe de ser elegido cuidadosamente. Cada pareja se denomina hecho. En el juego de ensayo debe estar representada equilibradamente

toda la información que la red necesite aprender.

Al realizar el entrenamiento la respuesta que da la red a cada patrón se compara con la respuesta correcta ante dicho patrón y, en virtud de esa comparación, se reajustan los pesos sinápticos. El reajuste de los pesos sinápticos está orientado a que, ante el patrón de entrada, la red se acerque cada vez más a la respuesta correcta.

Cuando ante un patrón de entrada la red de neuronas ya responde correctamente, se pasa al siguiente patrón del juego de ensayo y se procede de la misma manera.

Cuando se termina con el último patrón del juego de ensayo, se tiene que volver a empezar con el primero, ya que los pesos se han seguido modificando.

En casos sencillos, al cabo de unos pocos pasos de entrenamiento completos, con todos los elementos del juego de ensayo, los pesos sinápticos de todas las neuronas se estabilizan en torno a unos valores óptimos. Se dice entonces que el algoritmo de aprendizaje converge. Es decir, después de sucesivas presentaciones de todos los patrones estimulares del juego de ensayo, la red, responderá correctamente a todos ellos y se puede considerar entrenada y dar por terminada la fase de aprendizaje.

7.1.4. Aprendizaje no supervisado

En este tipo de aprendizaje, no se especifica a la red cuál debe ser la respuesta correcta; es decir, no hay una comparación entre la respuesta de la red y la respuesta deseada. Además, en este modelo de aprendizaje no existe ninguna influencia externa a la red, puesto que no se le informa de si un resultado fue correcto o no; tan sólo se le suministran grandes cantidades de datos con los que la red pueda construir sus propias asociaciones. Se necesita, por tanto, una cantidad mucho mayor de patrones de entrada durante el entrenamiento para que la red pueda ajustar correctamente sus pesos sinápticos. Por supuesto, los procedimientos de aprendizaje son diferentes a los utilizados con el modelo de entrenamiento supervisado.

En este tipo de aprendizaje, lo que de hecho se está haciendo es exigirle a la red que capte por sí misma alguna de las características presentes en los datos de entrada.

Evidentemente, muchos de los aprendizajes básicos que realizan los sistemas biológicos son de este tipo. Los recién nacidos (al igual que los ciegos de nacimiento que recuperan la visión en edad adulta) aprenden a organizar los datos visuales sin ayuda de *profesor* que les indique para cada patrón de estímulos de entrada, cual es la organización - interpretación correcta de dichos estímulos; es decir, la respuesta del subsistema neurológico de visión que servirá a su vez de entrada a otros subsistemas.

De hecho, en el aprendizaje no supervisado se pretende que las neuronas se auto-organicen aprendiendo a captar las *regularidades* de los datos de entrada sin suministrarles ningún tipo de criterio o ayuda externa que dirija dicha autoorganización.

7.1.5. Problemas comunes en el entrenamiento

Uno de los problemas más comunes al entrenar una RNA es que la red no se entrene con precisión suficiente; es decir, que tenga un alto porcentaje de fallos

que no se reduzca por más veces que se le pase el juego de ensayo o que la red tarde mucho tiempo en entrenarse. Cuando esto sucede, se deberá analizar el diseño de la red y del juego de ensayo. El hecho de cambiar el número de capas ocultas aumentará o disminuirá el tiempo de aprendizaje, pero probablemente no afectará en gran medida a la precisión o proporción de respuestas acertadas de la red. Si los resultados son pobres al presentarle una colección de patrones de entrada, es preciso comprobar el juego de ensayo. Tal vez se haya olvidado representar en el juego de ensayo alguno de los problemas tipo con los que la red debe enfrentarse, o la información sea engañosa o contradictoria. Se tendrá en cuenta que, cuando se modifica el juego de ensayo añadiendo nuevos hechos o corrigiendo alguno de los existentes, la red debe de ser entrenada de nuevo con la totalidad de los hechos y no tan sólo con los nuevos o con los corregidos. Una forma de determinar qué hechos están produciendo problemas consiste en entrenar la red hasta que responda con un nivel de aciertos aproximadamente del 90 %. Luego, al probar la red, se observan los hechos a los cuales la red responde mal. A continuación, se le deberá proporcionar a la red una mayor información similar a la de los hechos no aprendidos y se entrenará de nuevo. Si se llega a la conclusión de que se le han dado suficientes hechos a la red; es decir, que el juego de ensayo es completo y cubre todos los problemas a resolver y ésta sigue comportándose poco satisfactoriamente, habrá que considerar la posibilidad de cambiar el modo de presentar la información de entrada a la red. Si se están utilizando entradas no distribuidas, se deberá cambiar a distribuidas lo cual requerirá una estructuración completamente nueva de la red. Existen, por supuesto, problemas tan complejos que una red de neuronas no puede resolver en un tiempo y un espacio razonables. Si este es el caso, se podrá montar una cadena de redes de neuronas de modo que las salidas de una red alimenten las entradas de otras. Las primeras redes deberán simplemente identificar los objetos y sus salidas se utilizarán para alimentar a otra red que realice la evaluación. El añadir más capas a una única red no produce el mismo efecto. Otro de los problemas que pueden ocurrir, y de hecho es bastante común, es el del sobreentrenamiento. Este problema puede observarse al hacer un test de una red que acaba de ser entrenada con un alto porcentaje de acierto en dicho entrenamiento. Si el comportamiento que ofrece la red en el test no es todo lo bueno que puede hacer suponer por los resultados obtenidos en el entrenamiento, la red obtenida está sobreentrenada. Esto quiere decir que la red ha aprendido los patrones existentes en el juego de ensayo, pero realmente no ha sido capaz de abstraer y generalizar las relaciones entre los mismos, con lo que esta red no es aplicable en el mundo real, en el que se le presentarán entradas que no están presentes en el juego de ensayo. Este problema puede estar provocado por dos motivos fundamentales:

- La topología de la red tiene una complejidad muy alta. Es decir, que tiene un número de capas y/o neuronas demasiado alto, con lo que tiene un nivel de representatividad demasiado elevado para el problema que se está intentando resolver. No existe ningún método que determine la arquitectura de una red adecuada al problema a resolver, sino que esta tarea se basa en la experiencia del experto que diseñe la red.
- La red se ha entrenado durante demasiados ciclos.

Una situación muy común es entrenar una RNA con un juego de ensayo que contiene ruido, puesto que en el mundo real en muchas ocasiones no se puede eliminar totalmente el ruido de las mediciones que se realizan. Cualquier algoritmo de aprendizaje máquina –y las redes neuronales no son una excepción– que sea capaz de aprender y ofrecer una precisión del 100 % (o, en general, un porcentaje superior a la precisión de los patrones sin el ruido) ha aprendido los patrones de entrenamiento y el ruido de los mismos, sin llegar a discriminarlo, es decir, ha perdido la capacidad de generalización.

El proceso de entrenamiento tiende a minimizar el error que se consigue en el conjunto de entrenamiento durante el mismo, (Figura 9.a). Sin embargo, si durante el entrenamiento se realizan tests a la red con patrones que no se presentan durante el entrenamiento, el error observado en estos tests no tiende siempre a minimizarse, como puede verse en la Figura 9.b, sino que llega a un punto en el que este empieza a crecer. Este es el punto en el que la red empieza a sobreentrenarse y perder la capacidad de generalización, por eso se comporta mal en los test.

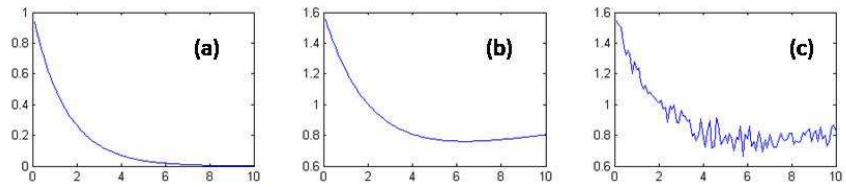

Figura 9: Evolución del error durante la fase de entrenamiento

Una solución a este problema podría ser el realizar test durante el entrenamiento, y detectar el punto en el que el error en el test comienza a crecer. (valor 6 del eje de abscisas en la Figura 9.c) Esto no se puede realizar con el conjunto de test, puesto que, de hacerlo así, se está condicionando la red de alguna forma a este conjunto de test, lo cual no es deseable, porque este conjunto de test está pensado para ser presentado a una red final, no a una que todavía está en el proceso de entrenamiento, y con patrones que nunca haya visto.

Para solucionar esto, se cuenta con otro conjunto de patrones, denominado conjunto de validación, que tendrá patrones distintos a los del entrenamiento y test. Al ser evaluada la red durante el entrenamiento con este nuevo conjunto de patrones, se ofrece una estimación de cómo se comportará la red cuando se realice el test, es decir, en su ejecución real con patrones que no ha visto. Este conjunto de validación también sirve para tomar decisiones acerca de qué red es mejor tomar, entre varias redes distintas –posiblemente con topologías distintas– ya entrenadas.

Sin embargo, este proceso no es tan sencillo. El error de validación mostrado en la Figura 3.8.b es un error teórico. En la práctica se tiene uno similar al mostrado en la Figura 9.c, con lo que no resulta sencillo determinar cuándo hay que parar el entrenamiento. Existen diversas técnicas que, utilizando el conjunto de validación, dictan cuándo es razonablemente aconsejable parar el entrenamiento. Sin embargo, ninguna de ellas consigue determinar con exactitud cuándo es el mejor momento.

7.2. Fase de validación o test

Tras la fase de entrenamiento viene la fase de ejecución, durante la que se le pedirá a la red que responda a estímulos diferentes a los presentados durante la fase de entrenamiento. Gracias a los ejemplos aprendidos del juego de ensayo, la red deberá ser capaz de generalizar y dar respuestas correctas ante patrones de estímulos nuevos.

En otras palabras, una vez terminado el aprendizaje, una red puede generalizar; es decir, ante entradas similares a las de su juego de ensayo, producirá salidas correctas. Hay que tener en cuenta que es muy difícil conseguir la capacidad de generalización de una red sin utilizar grandes cantidades de datos y que estos sean muy variados.

Para operar con una red entrenada, el proceso es el mismo que cuando se realizaba el entrenamiento. Por lo tanto, se le sigue suministrando información de entrada a la red, sólo que ahora no se realizará ningún ajuste en los pesos sinápticos. La red reconocerá o evaluará y dará una respuesta a partir de los pesos establecidos durante el entrenamiento.

Para conseguir el mejor rendimiento de generalización, los datos usados para el entrenamiento deben cubrir un rango de hechos suficientemente amplio. En general, cuando aumenta el tamaño y variedad del juego de ensayo disminuye la necesidad de que los datos de entrada durante la fase de trabajo normal se parezcan mucho a los patrones del juego de ensayo; es decir, la red generalizará mejor. Si los datos de un problema se diferencian demasiado de todos los patrones del juego de ensayo, la red tendrá dificultades para encontrar la respuesta correcta. Selección del juego de ensayo En vista de la gran cantidad de información que se le puede proporcionar a una red, se ha de buscar un criterio de selección para crear el juego de ensayo. En el juego de ensayo debe de haber suficientes hechos, es decir, parejas "patrones de estímulos - respuesta correcta". Además, los hechos del juego de ensayo deberán cubrir ampliamente la totalidad de las características a los que la red debe de enfrentarse. El problema de decidir cuántos hechos se han de incluir en el juego de ensayo es un problema de Teoría de Muestras al que se está buscando solución. Un problema muy similar se plantea al establecer, en base a casos de prueba, el grado de experiencia de un experto.

Por otra parte, a una red evaluadora es importante mostrarle tanto los patrones de entrada que llevan a evaluaciones positivas, como los patrones de entrada que llevan a evaluaciones negativas. Es decir, el entrenamiento de la red debe incluir situaciones que se evalúen negativamente, pues de lo contrario la red simplemente aprenderá que todo está correcto siempre. También se debe incluir en el juego de ensayo cada uno de los casos en los cuales el valor de una entrada es causa de un *mal* resultado.

Además, no se puede incluir en el juego de ensayo una colección exageradamente grande de hechos. Es necesario seleccionar aquellos hechos que reflejen claramente cada uno de los patrones a reconocer y las situaciones extremas de evaluación en una red evaluadora.

Lo ideal es preparar una colección amplia de hechos de entrenamiento que cubran todos los problemas a los que se pueda tener que enfrentar la red. A continuación, se seleccionarán algunos de ellos para el juego de ensayo, procurando que todos los problemas queden bien representados.

8. Redes recurrentes

Cuando se trabaja con patrones dinámicos; es decir, con patrones de secuencias en las que aparece el concepto tiempo, las RNA alimentadas sólo hacia adelante se encuentran bastante limitadas ya que no permiten conexiones que unan neuronas creando bucles. En las redes recurrentes no se impone ninguna restricción en su conectividad, con lo que se gana un número mayor de pesos por neurona y por lo tanto una mayor representatividad, dado que las RNA representan la información de forma distribuida en sus pesos. De esta forma, la principal característica de este tipo de redes es la de realimentar su salida a su entrada, evolucionando hasta un estado de equilibrio donde proporciona la salida final de la red (Demuth, Beale, 1994). Esta característica las hace útiles cuando se quiere simular sistemas dinámicos; sin embargo, su entrenamiento es más lento que el de una red alimentada sólo hacia delante, y a la vez mucho más complejo. El primer algoritmo de entrenamiento de este tipo de redes aparece en 1987, cuando se adapta el algoritmo de retropropagación del error de las RNA alimentadas sólo hacia delante a las redes recurrentes aplicadas a patrones estáticos ("Recurrent Backpropagation") y se pudieron aplicar estas redes a los mismos problemas a los que se aplicaban las multicapa alimentadas hacia delante. Además, otros investigadores se centran en desarrollar aproximaciones del algoritmo de aprendizaje que lo hagan más práctico surgiendo el algoritmo llamado "Real-Time Recurrent Learning" o RTRL indicado para tareas de tiempo real.

A partir de entonces, las redes recurrentes se han venido aplicando en un buen número de tareas, desde reconocimiento del habla hasta la simulación de autómatas finitos. Sin embargo, la aplicación de redes recurrentes presenta un mayor número de problemas. En el caso de patrones estáticos, una red recurrente funciona presentándole un patrón, haciendo después evolucionar la red hasta que sus salidas se estabilizan. Sin embargo, esto no está asegurado, pudiéndose dar comportamientos oscilatorios o caóticos y aunque existen estudios para establecer las condiciones para que esto no ocurra, se limitan a ciertas arquitecturas muy concretas como las Hopfield. E1 caso de los patrones dinámicos es todavía más complicado, ya que, si se sabe poco del comportamiento de una red recurrente (por ejemplo la dificultad de estabilizarse), se sabe aún menos de su comportamiento dinámico. El poco conocimiento es empírico y no existen estudios formales ni de la red recurrente más simple: una neurona aislada con una conexión a sí misma. Tampoco existen estudios teóricos que avalen utilizar un algoritmo basado en el descenso del gradiente para tareas de tratamiento de patrones dinámicos. Un problema sencillo, como es enseñar a oscilar a una neurona aislada con realimentación, da muchos problemas del tipo de mínimos locales y hasta ahora no se conoce su justificación teórica. Además, en redes multicapa se conoce más o menos bien qué arquitectura hay que utilizar en la mayoría de los problemas, gracias a conocimientos basados fundamentalmente en la experiencia. Sin embargo, por una parte, la variedad arquitectónica en redes recurrentes es infinitamente superior, por lo que su diseño es más complicado y, por otra, la gran variedad de este tipo de patrones hace difícil su categorización.

8.1. Aprendizaje por épocas

El aprendizaje en este tipo de redes se realiza por épocas. La red se hace evolucionar durante un periodo de tiempo (época) en el que se le ha introducido una secuencia de ejemplo y ha debido dar la respuesta adecuada. Una vez llegado al final de una época se reinicializa la red para que el estado inicial no dependa del estado final de la época anterior y se entra en una nueva época. La variación de los pesos sólo se realiza al acabar cada época y, de la misma forma que en las multicapa, se podrá llevar a cabo *por lotes* (calcular todos los incrementos de los patrones-épocas) o de forma incremental (después de cada patrón-época), siendo ésta última, la más utilizada en redes recurrentes.

8.2. Aprendizaje en modo continuo

Aparte de *por épocas*, existe el modo de operación continuo, donde en ningún momento se inicializa la red. Este tipo es el más apropiado para aplicaciones en tiempo real, donde no se sabe *a priori* la salida deseada. El aprendizaje suele ser distinto ya que, en el caso de operación por épocas, normalmente se pueden aplicar las secuencias un cierto número de veces, mientras en el caso de la operación continua, esto normalmente no se puede hacer. Además, no está nada claro el momento en que se deberían actualizar los pesos de la red. La elección de un modo u otro dependerá de los problemas a resolver, e incluso existen técnicas que aprovechan las características de ambos modos. La principal diferencia entre ellos radica en que, en el modo *por épocas*, la transición del estado final de una época al estado inicial de la siguiente se produce externamente mientras que, en el continuo, esta transición la debe aprender a realizar la propia red, por lo que se le está exigiendo más. Otra cuestión problemática es la asignación de la salida deseada. En ciertos problemas, como la predicción de una serie temporal o la simulación de un autómata finito, la salida deseada se exige en todo momento. Sin embargo, en otros problemas como los de reconocimiento, no se puede establecer cuál es la salida deseada en cada instante. Esto plantea preguntas como: ¿Qué salida se exige a la red cuando se le introduce el patrón?, ¿en qué momento se exige la salida deseada?, ¿al acabar de introducir el patrón o cuánto tiempo después?, ¿la salida será puntual o continua?, etc. Las respuestas dependen siempre del problema a tratar; por ejemplo, podría emplearse una salida neutra 0 durante la entrada del patrón si se emplea un intervalo [-1,1]. En el caso de las otras preguntas, se relacionan de forma que, si se ha escogido no forzar un comportamiento de la salida mientras se está realizando la entrada, no es conveniente utilizar una salida puntual ya que será difícil determinar si la salida en un determinado momento corresponde a un estado transitorio o al reconocimiento de una secuencia. Otro problema que puede surgir está relacionado con la forma de la salida. Si la salida es de tipo escalón donde debe pasar de un punto extremo a otro (por ejemplo, de -1 a 1, o de 0 a 1) en un paso de tiempo, es bastante normal que la red no sea capaz de realizar una transición tan brusca. En este caso es necesario proponer una salida suavizada o bien dejar un intervalo de transición entre los dos estados sin salida deseada.

Referencias

[1] C. Aldrich and J. S. J. van Deventer. Comparison of different artificial neural nets for the detection and location of gross errors in process systems. *Industrial & Engineering Chemistry Research*, 34(1):216–224, 1995.

[2] C. M. Bishop. *Neural Networks for Pattern Recognition*. Oxford University Press, USA, 1995.

[3] C. M. Bishop. *Pattern recognition and machine learning*. Springer, 2006. (Online service).

[4] G. J. Bowden, G. C. Dandy, and H. R. Maier. Data transformation for neural network models in water resources applications. *Journal of Hydroinformatics*, 5(4):245–258, 2003.

[5] M. Brown and C. Harris. *Neurofuzzy adaptive modelling and control*. Prentice Hall, 1995.

[6] Y. Chauvin and D. E. Rumelhart. *Backpropagation: Theory, Architectures, and Applications*. Lawrence Erlbaum Associates, 1995.

[7] J. Faraway and C. Chatfield. Time series forecasting with neural networks: a comparative study using the air line data. *Journal of the Royal Statistical Society: Series C: Applied Statistics*, 47(2):231–250, 1998.

[8] J. A. Freeman and D. M. Skapura. *Neural Networks: Algorithms, Applications, and Programming Techniques*. Addison-Wesley, 1991.

[9] R. González-García, R. Rico Martínez, and I. G. Kevrekidis. Identification of distributed parameter systems: A neural net based approach. *Computers and Chemical Engineering*, 22:965–968, 1998.

[10] S. Haykin. *Neural Networks: A Comprehensive Foundation*. Prentice Hall, NJ, USA, 2008.

[11] J. R. Hilera and V. J. Martinez. *Redes neuronales artificiales. Fundamentos, modelos y aplicaciones*. Addison-Wesley Iberoamericana S.A, Madrid, 1995.

[12] J. C. Hoskins and D. M. Himmelblau. Process control via artificial neural networks and reinforcement learning. *Computers & chemical engineering*, 16(4):241–251, 1992.

[13] T. Kohonen. *Self-organization and associative memory*. Springer Verlag, New York, 1989.

[14] C. T. Lin and C. S. G. Lee. *Neural fuzzy systems: a neuro-fuzzy synergism to intelligent systems*. Prentice-Hall, Inc. Upper Saddle River, NJ, USA, 1996.

[15] C. G. Looney. *Pattern recognition using neural networks: theory and algorithms for engineers and scientists*. Oxford University Press, Inc. New York, NY, USA, 1997.

[16] W. S. McCulloch and W. Pitts. A logical calculus of the ideas immanent in nervous activity. *Bulletin of Mathematical Biology*, 5(4):115–133, 1943.

[17] K. Meert and M. Rijckaert. Intelligent modelling in the chemical process industry with neural networks: a case study. *Computers and Chemical Engineering*, 22:587–593, 1998.

[18] K. S. Narendra, M. J. Feiler, and Z. Tian. Control of complex systems using neural networks. *Modeling and Control of Complex Systems*, 2008.

[19] C. A. O. Nascimento, R. Giudici, and R. Guardani. Neural network based approach for optimization of industrial chemical processes. *Computers and Chemical Engineering*, 24(9-10):2303–2314, 2000.

[20] Alejandro Pazos. *Redes de neuronas artificiales y algoritmos genéticos*. Servicio de Publicaciones Universidade da Coruña, 1996.

[21] B. A. Pearlmutter. Dynamic recurrent neural networks. Technical report, Technical Report CMU-CS. School of Computer Science, Carnegie Mellon University, 1990., 1990.

[22] I. Rivals and L. Personnaz. Nonlinear internal model control using neural networks: application to processes with delay and design issues. *IEEE Transactions on Neural Networks*, 11(1):80–90, 2000.

[23] J. J. Shi. Reducing prediction error by transforming input data for neural networks. *Journal of Computing in Civil Engineering*, 14:109, 2000.

[24] P. D. Wasserman. *Neural computing: theory and practice*. Van Nostrand Reinhold Co. New York, NY, USA, 1989.

[25] N. Wiener. *God and Golem: a Comment on Certain Points where Cybernetics Impinges on Religion*. The MIT Press, 1964.