Un Sistema de Patrones de Software para Redes Neuronales Artificiales

Beatriz Peralta Cortés *
Jorge L. Ortega Arjona **
Departamento de Matemáticas
Facultad de Ciencias, UNAM.

Héctor Benítez Pérez ***
Departamento de Ingeniería
de Sistemas Computacionales
y Automatización, IIMAS-UNAM.

Resumen

El presente artículo describe un Sistema de Patrones de Software para Redes Neuronales Artificiales, como un esfuerzo inicial en la búsqueda de un criterio de selección de redes neuronales artificiales. Las redes que inicialmente forman parte de este sistema y que se describen como patrones de software en este artículo son la red de Mapas Auto-Organizados, el Perceptrón Multicapa, las redes de Funciones de Base Radial y las redes de Hopfield.

1. Introducción

Las Redes Neuronales Artificiales son un campo de investigación que estudia el desarrollo de modelos matemáticos a partir de elementos simples y no lineales [Corchado et al., 2000, Hilera and Martínez, 1995, Fausett, 1994]. Una Red Neuronal Artificial es un sistema de procesamiento de información que está formado por elementos no-lineales llamados *neuronas*. Se trata de modelos matemáticos viables para resolver una amplia variedad de problemas, tales como clasificación de patrones, reconocimiento y síntesis de lenguaje, funciones de aproximación, modelado de sistemas no lineales y de control, compresión de imagen, memoria asociativa, agrupamientos, etc. [Luo and Unbehauen, 1997, Hassoon, 1995, Hilera and Martínez, 1995].

Existen diferentes formas de clasificar las Redes Neuronales Artificiales. Estas formas toman en cuenta criterios básicos como la *topología* de la red, el *tipo de asociación* de las señales de entrada y salida, así como los *mecanismos de aprendizaje* [Corchado et al., 2000, Hassoon, 1995]. Sin embargo, estas clasificaciones no establecen criterios para seleccionar el tipo de red a utilizar en un problema dado.

* bety08@yahoo.com.mx
** jloa@fciencias.unam.mx
*** hector@uxdea4.iimas.unam.mx

Este artículo pretende proponer un Sistema de Patrones de Software como una heurística para la selección de Redes Neuronales Artificiales. Es decir, partiendo del análisis de los problemas que se resuelven con Redes Neuronales, se busca identificar las características más relevantes que impliquen su uso como solución. En este sistema, cada patrón toma en cuenta las características del problema y la red neuronal que se emplea para resolverlo. El sistema está compuesto por un conjunto representativo de redes neuronales, cada una de las cuales se describe utilizando la forma POSA [Buschmann et al., 1996]. Las redes que se describen como patrones de software en este artículo son las siguientes:

- **La red de Mapas Auto-Organizados o SOM.** Este tipo de redes se utiliza para la clasificación de datos.
- **El Perceptrón Multicapa o MLP.** Estas redes utilizan la salida deseada para establecer una relación entre los datos de entrada, cuando es importante la precisión de la respuesta.
- **Las redes de Funciones de Base Radial o RBF.** Estas redes utilizan la salida esperada para establecer una relación entre los datos de entrada, cuando no es importante el tiempo de respuesta.
- **Las redes de Hopfield.** Este tipo de redes se utiliza para la clasificación o reconstrucción de patrones a partir de la información almacenada.

---

2. El Patrón *Mapas Auto-organizados o SOM*

El Patrón SOM describe una red neuronal artificial que se utiliza para la clasificación de patrones de datos [1].

2.1. También conocido como

Redes de Kohonen o Mapas de Características Autorganizados.

2.2. Ejemplo

En sistemas de reconocimiento de caracteres, se requiere identificar patrones de entrada que, agrupados, representen letras con diferentesfuentes. El problema es ¿cómo detectar automáticamente cada letra por un conjunto de puntos, de tal manera que en su conjunto se reconozcan como la letra que se trata (sobre una superficie bidimiensional)? Además, puede darse que algunos puntos dentro del conjunto no se encuentren propiamente en un sitio adecuado, o estén fuera del conjunto. Los puntos se presentan en diferentes cantidades y densidades [Fausett, 1994].

[1] Estos patrones representan las características de los datos a través de vectores

2.3. Contexto

El patrón SOM se utiliza cuando los datos no están bien definidos o se tienen muchos datos de entrada.

2.4. Problema

Encontrar características similares entre los datos de entrada sin supervisión. Las *Fuerzas* asociadas a este problema son:

- Los datos representan la información obtenida de varias dimensiones.
- Los datos no están bien definidos.
- Se tiene una gran cantidad de datos en los que pueden existir dependencias entre ellos.
- Se requiere obtener grupos de datos que representen toda la información.
- No se requiere supervisión para llevar a cabo la clasificación [2].

2.5. Solución

La Red SOM forma mapas de características de manera auto-organizada. El objetivo es establecer una correspondencia entre los datosde entrada y un espacio (normalmente de 1 ó 2 dimensiones) a través de una función de semejanza, de tal manera que ante vectores de entrada con características comunes se deben activar neuronas situadas en zonas próximas.El entrenamiento de la red consiste en la adaptación y modificación de los pesos entre las conexiones adyacentes a las *neuronas ganadoras*, es decir, las neuronas que hayan tenido la distancia mínima. De esta manera, vectoresde datos de entrada similares activan y generan un orden global.

El aprendizaje de la red termina cuando se reducen las zonas de neuronas activadas por vectores de datos de entrada parecidos. Es necesario repetir el proceso de entrenamiento hasta refinar el mapa topólogico.

2.6. Estructura

Las redes SOM poseen dos capas de neuronas con conexiones entre ellas: n neuronas en la capa de entrada y m neuronas en la capa de salida, con $n << m$ y $n * m$ conexiones. La figura 1 muestra la estructura de una red SOM.

[2] Esta red también trabaja con aprendizaje supervisado. Sin embargo, en estatesis no se considera tal caso. Para mayor referencia consultar el libro de Kohonen [?]

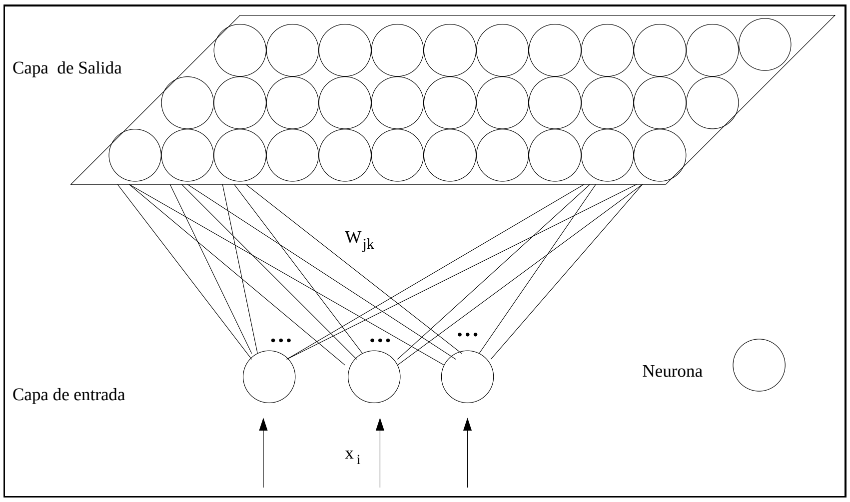

Figura 1: Estructura de la red SOM

2.7. Participantes

- Neuronas de entrada: Estas neuronas pertenecen a la capa de entrada y sólo reciben los datos para el entrenamiento de la red. Después pasan los datos a las neuronas de la capa de salida.
- Neuronas de salida: Estas neuronas se encargan de llevar a cabo (a) el entrenamiento de la red, identificando las diferencias entre las distancias euclidianas y modificando los pesos de las conexiones adyacentes a las neuronas ganadoras y (b) el aprendizaje de la red, que consiste en establecer las diferentes categorías entre los datos en función de su semejanza.

2.8. Dinámica

Suponemos un vector de entrada o patrón de entrenamiento con n características y representado por el vector $\mathbf{x} = (x_1, x_2, \ldots, x_n)$ en el espacio de dimensión n. La red proyecta el espacio de entrada sobre la salida, y en el entrenamiento realiza la proyección conservan do el orden topológico. Para ello se utiliza la regla de aprendizaje competitivo con respecto al conjunto de neuronas vecinas, de tal manera que el proceso de aprendizaje no sea global sino local.

Cada neurona k se caracteriza por un vector de pesos $\mathbf{w}_k = (w_{k1}, w_{k2}, \ldots, w_{kn})$ de dimensión n, y recibe la misma entrada $\mathbf{x} \in \mathbb{R}^n$ en la capa de entrada. Este vector pasa a la capa de salida en donde se lleva a cabo el entrenamiento de la red.

En la capa de salida, las neuronas compiten por activarse comparando los datos de entrada con su vector de pesos asociado. Así, el vector de pesos más parecido al vector de datos de entrada determina las neuronas ganadoras y

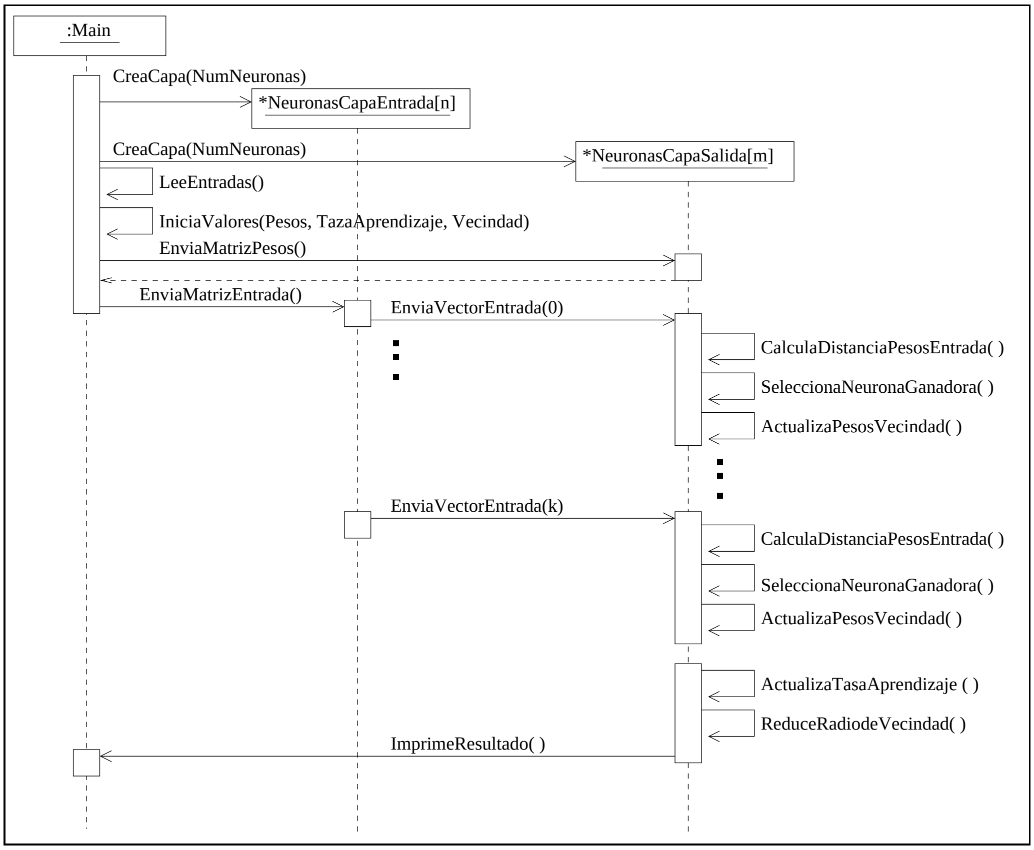

Figura 2: Diagrama de secuencia del algoritmo SOM

son las únicas que modifican los pesos asociados a sus conexiones. La figura 2 describe el diagrama de secuencia que emplea el algoritmo SOM.

El algoritmo de aprendizaje que utiliza la red SOM se describe a continuación:

1. Crear el vector de pesos $\mathbf{w}_k = (w_{k1}, w_{k2} \ldots, w_{kn})$ para cada conexión entre las neuronas.

2. Fijar la zona inicial de vecindad entre las neuronas de la capa de salida.

3. Presentar el vector de entrada $\mathbf{x} = (x_1, x_2, \ldots, x_n)^T$ en la capa de entrada.

    a) Para cada neurona de salida k caracterizada por su vector de pesos $\mathbf{w}_k$, determinar su estado de activación Y_k, definido por la distancia euclidiana entre $\mathbf{x}$ y $\mathbf{w}_k$: $Y_k = \mathbf{w}_k\mathbf{x}$.

b) Seleccionar como neurona ganadora, aquélla neurona cuyo resultado entre la distancia del vector de datos de entrada y el vector de pesos haya sido el menor.

c) Una vez seleccionada la neurona ganadora i, se actualizan los pesos de las conexiones entre la neurona ganadora y la neurona de entrada con la siguiente regla de aprendizaje.

$$\mathbf{w}_k(t+1) = \mathbf{w}_k(t) + \eta g(k,i)(\mathbf{x}(t) - \mathbf{w}_k(t)) \tag{1}$$

donde η es la tasa de aprendizaje y $g(k,i)$ es una función entre las neuronas k e i tal que $g(i,i) = 1$.

Generalmente se utiliza la función gaussina para $g(k,i)$ de forma que:

$$g(k,i) = e^{-||k-i||^2} \tag{2}$$

A esta función también se le conoce como malla, ya que define cómo quedan agrupados los datos de entrenamiento.

4. Actualizar la tasa de aprendizaje.

5. Reducir el radio de vecindad.

6. El proceso se repite, a partir del paso 3, volviendo a presentar todo el conjunto de entrenamiento t veces.

Las propiedades más importantes de esta red son: (a) el concepto de aprendizaje en un vecindario cercano a la neurona ganadora, y (b) la forma en que se modifican los pesos para que destaquen propiedades topológicas en la proyección de características.

2.9. Ejemplo resuelto

En el problema de reconocimiento de caracteres se requiere identificar patrones de entrada que, agrupados, representan letras con diferentes fuentes. Este problema consiste en detectar automáticamente cada letra por un conjunto de puntos, de tal manera que en su conjunto se reconozcan como la letra que se trate.

Se seleccionan siete letras (A, B, C, D, E, J y K) con 3 fuentes diferentes. Cada letra se representa por una matriz de 7×9 puntos y está formada por valores binarios. Dentro de esta matriz se puede dar que algunos puntos no estén en un sitio adecuado, o estén fuera del conjunto. Para resolver el problema, se utiliza una red SOM que agrupa los datos de entrada sin supervisión [Fausett, 1994].

La red SOM está formada por: *(a)* 63 neuronas de entrada, que reciben los datos de cada letra; *(b)* 63 neuronas de salida, tomando en cuenta que la red SOM tiene una estructura lineal. Es decir, que la vecindad de la neurona ganadora k consiste en actualizar los pesos de sus vecinas $k-1$ y $k+1$. Para este problema se utiliza una tasa de aprendizaje igual a 0.6. Este valor va decreciendo durante el entrenamiento de la red hasta obtener un valor final de 0.01 [Fausett, 1994].

Al iniciar el proceso de entrenamiento, únicamente las neuronas vecinas a la neurona ganadora actualizan sus pesos. Sin embargo, es muy posible que estas neuronas no tengan vectores de pesos cercanos a los valores del vector de entrada. Es por esto que durante el proceso de entrenamiento los pesos se van ajustando, de tal manera que identifiquen 21 grupos. Se considera que cada grupo representa una letra [Fausett, 1994].

2.10. Usos conocidos

En general, la red SOM se utiliza en agrupamientos (*clustering*) cuando se desea determinar a qué grupo pertenecen los datos de entrada. Comúnmente se usa en la extracción o identificación de características relevantes de una señal. También en la reducción de dimensiones para proyectar y visualizar espacios de señales de n dimensiones a espacios de uno o dos dimensiones [Corchado et al., 2000].

- Las redes SOM se han empleado en el **procesamiento de lenguaje**. El objetivo es convertir el habla a texto escrito en tiempo real, independientemente de la persona que hable y del ruido del ambiente. El sistema inicia con una señal procesada a través de un micrófono y un preprocesador analógico/digital. La señal se convierte en la representación de 15 canales que cubren un rango de frecuencia de 200 Hz a 5 kHz. Estos canales constituyen un vector de entrada $\mathbf{x}(t)$ de dimensión 15 y representa los diferentes fonemas del lenguaje, para un mapa de características. Se entrena a la red de tal manera que para fonemas muy parecidos se activen neuronas de salida próximas creando un *mapa fonotópico* [Hassoon, 1995].

- Las redes SOM se han utilizado en la **codificación de datos**. El objetivo es diseñar códigos para reducir o comprimir el tamaño de la información. Por ejemplo, en la compresión de imágenes en color se trata de averiguar qué pixeles de la imagen pueden representarse por el mismo color, con el fin de utilizar menos bits por pixel. Generalmente esta aplicación se utiliza en la conversión de imágenes originales con pixeles de 2^{24} diferentes colores a imágenes para monitores VGA con pixeles de 2^8 colores (8 bits por pixel y el color de un pixel se codifica con 24 bits). En este problema se usa una red SOM para establecer los 2^8 colores más frecuentes que están presentes en la imagen original. Se divide cada imagen en tres imágenes, de tipo monocolor que representan las versiones en rojo, verde y azul. Estas tres combinaciones dan lugar a la imagen original. Cada pixel de estas versiones se codifica con 8 bits, por lo que existen 256 posibles grados de intensidad de rojo, verde y azul.

  Se usa una red SOM con tres neuronas en la capa de entrada y 256 neuronas en la capa de salida. Estas neuronas corresponden a los 256 colores que deben identificar la red. En la capa de entrada, la red recibe los valores correspondientes al grado de intensidad de los colores básicos: rojo, verde y azul de cada píxel de la imagen. Después del entrenamiento, los pesos de las conexiones de cada neurona en la capa de salida representan la combinación de los colores verde, rojo y azul para cada uno de los 256 colores, de manera que se obtiene un *mapa cromato-tópico* de la imagen aprendida [Hilera and Martínez, 1995].

- Las redes SOM se han utilizado para simular la formación de *mapas somatosensoriales* entre los receptores tácticos de la superficie de una mano y una corteza artificial formada por un capa de 30×30 neuronas. El conjunto de entrenamiento consiste en la actividad de los patrones de un conjunto de receptores tácticos cubriendo la superficie de una mano. El mapa de características se inicia de manera aleatoria y los puntos se seleccionan de acuerdo a su distribución de probabilidad definida por cinco regiones que corresponden a los dedos [Hassoon, 1995].

2.11. Consecuencias

Ventajas

- La red SOM aprende rápidamente, es decir, ante un dato de entrada nuevo, la red puede clasificarlo en un tiempo relativamente corto.
- Cuando no se cuenta con la información suficiente de entrada, la red SOM es todavía capaz de llevar a cabo una clasificación.
- La red SOM agrupa la información de tal manera que reduce la dimensión de los datos de entrada.
- La red SOM preserva la topología de los datos de entrada, es decir, no modifica la informacion que recibe, solo la etiqueta.

Desventajas

- La red SOM que utiliza aprendizaje sin supervisión necesita cada vez repetir todo el proceso de aprendizaje para clasificar un nuevo dato de entrada.

---

3. Patrón *Perceptrón Multicapa o MLP*

El Patrón MLP describe una red neuronal artificial que utiliza la salida deseada para establecer una relación entre los datos de entrada.

3.1. También conocido

Retropropagación (*Backpropagation*).

3.2. Ejemplo

En medicina, se requiere detectar la oclusión coronaria o infarto al miocardio agudo, la cual es una enfermedad difícil de diagnosticar. El problema consiste en averiguar si un paciente puede sufrir esta enfermedad comparando su historial clínico con el historial clínico de pacientes que sí la han padecido. El resultado debe ser lo más preciso posible, considerando a menudo historiales muy similares que no se relacionan con la enfermedad del paciente [Hassoon, 1995].

3.3. Contexto

El patrón MLP se utiliza cuando se tiene grandes cantidades de datos que están relacionados y se conoce de antemano la salida. Además es importante la precisión de la respuesta pero no el tiempo de ejecución.

3.4. Problema

Establecer la relación entre los datos de entrada y la salida esperada.
Las *Fuerzas* asociadas a este problema son:

- Existe una gran cantidad de datos con dependecias entre ellos.
- Los datos pueden tener información redundante.
- Es necesario conocer la salida esperada para llevar a cabo el entrenamiento de la red.
- Se requiere que la solución sea precisa.
- No es importante el tiempo de respuesta.

3.5. Solución

La Red MLP establece una relación entre los datos de entrada y los datos de salida a partir de ciertos valores esperados. Esta red está formada por neuronas artificiales que calculan el producto punto de los pesos por las entradas más unos pesos extras llamados umbrales o polarización (*bias*). Con el resultado que se obtiene del producto punto se evalúa la función de activación. Las redes MLP están formadas por varias capas de neuronas, así que la salida de las neuronas en una capa forma la entrada de la capa siguiente. Durante el aprendizaje, la red calcula los pesos usando el algoritmo de propagación hacia atrás. El objetivo es utilizar el error que existe entre la salida de la red y el valor esperado para ajustar la matriz de pesos y minimizar el error.

3.6. Estructura

Las redes MLP están formadas por tres tipos de neuronas: n neuronas de entrada, m neuronas ocultas o intermedias y k neuronas de salida, $n \leq m \leq k$. En la figura 3 se muestra la estructura general de las redes MLP.

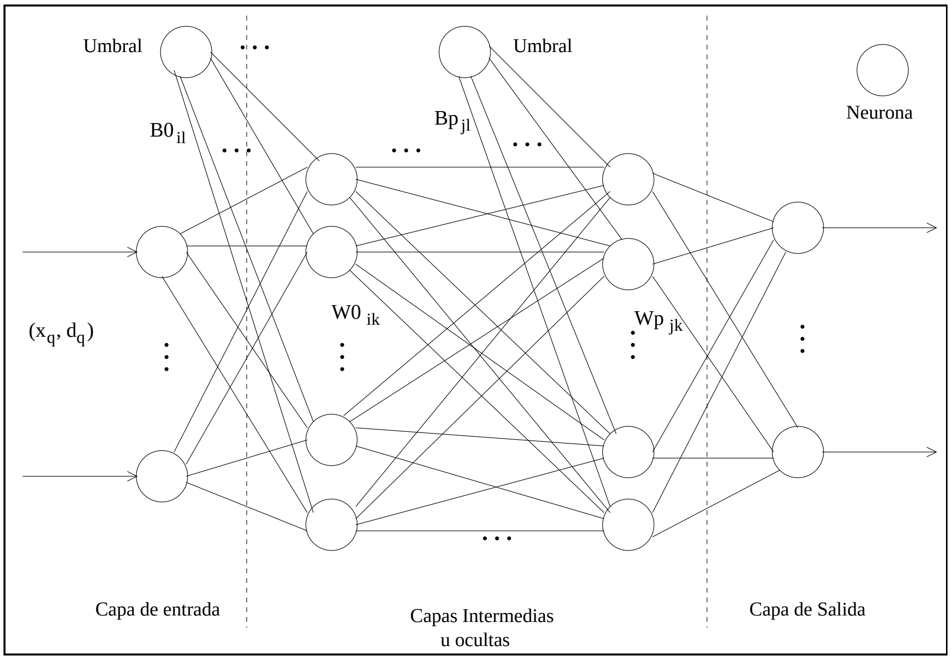

Figura 3: Estructura general de las redes MLP. Los pares (x_q, d_q) son los valores de entrada que reciben las redes MLP. x_q representa el vector de entrada q y d_q es la salida esperada para el vector de entrada x_q. $w0_{ik}$ es el peso de la conexión entre las neuronas i y k en la capa oculta 0 y wp_{jk} es el peso de la conexión entre las neuronas j y k en la capa oculta p. Así $b0_{il}$ es el umbral de la conexión entre las neuronas i y l en la capa oculta 0 y bp_{jl} es el umbral de la conexión entre las neuronas j y l en la capa oculta p.

3.7. Participantes

- Neuronas de entrada: Estas neuronas pertenecen a la capa de entrada y se encargan de recibir los datos para el entrenamiento de la red. Es decir, los valores de entrada y los valores de salida deseados, además de enviar estos datos a las neuronas de la capa oculta y de la capa de salida respectivamente.

- Neuronas ocultas: Estas neuronas calculan el producto punto de las entradas y los pesos asociados a sus conexiones, y utilizan ese valor para obtener la función de similitud o función de activación. Su resultado se envía a a siguiente capa oculta o a la capa de salida.

- Neuronas de salida: Estas neuronas reciben el valor de la salida esperada el resultado de las capas ocultas y proporcionan el resultado de la red. El resultado se obtiene de compar el valor obtenido por la red con un valor esperado. Si el resultado es diferente del valor esperado, se calcula el error que comete la red y se envía ese valor a las neuronas ocultas para modificar los pesos asociados a cada neurona, así como el valor de los umbrales.

3.8. Dinámica

Suponemos que un vector de entrada o patrón de entrenamiento tiene n características y está representado por el vector $x = (x_1, x_2, \ldots, x_n)$ en el espacio de dimensión n. La red proyecta el espacio de entrada sobre la salida a través de las neuronas ocultas. En el entrenamiento se lleva a cabo la modificación de los pesos entre las conexiones. El objetivo es ajustar la matriz de pesos para minimizar el error entre la salida de la red y los valores esperados. Existen varios métodos para resolver este problema. Por ejemplo: el método del gradiente descendiente por pasos, el método de Newton y el método del gradiente conjugado [Hu and Hwang, 2002]. En este trabajose considera el método del gradiente descendiente por pasos, que consiste en utilizar el error entre la salida de la red y la salida esperada para modificar los pesos de las capas anteriores. Después del entrenamiento, la red MLP solo utiliza la fase de propagación hacia adelante para realizar la prueba del algoritmo y utilizar los valores obtenidos en el entrenamiento.

Una neurona k se caracteriza por un vector de pesos $\mathbf{w}_k = (w_{k1}, w_{k2}, \ldots, w_{kn})$ de dimensión n, recibe el vector de entrada $\mathbf{x}$ y tiene la misma función de activación, la cual generalmente es una función sigma o una función tangente hiperbólica. La salida de las neuronas de una capa forma la entrada de la capa siguiente.

En la capa de salida, las neuronas calculan el resultado de la red considerando la diferencia entre la salida de las neuronas y el valor esperado. De ello depende que se modifiquen los pesos, de tal manera que cuando los valores son diferentes se obtiene el error que comete la red, regresándolo a las neuronas ocultas para actualizar los pesos y los umbrales. Esto se muestra en la figura 4.

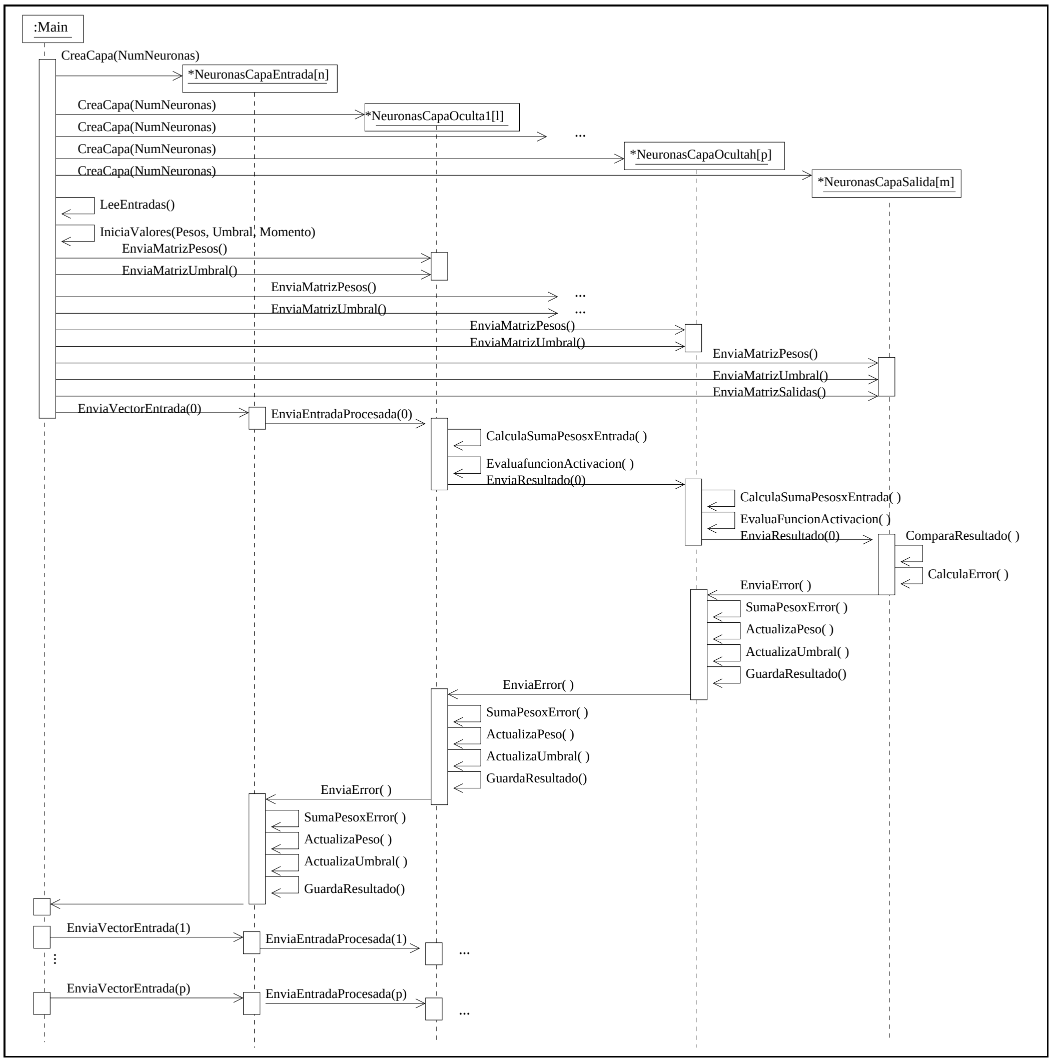

Figura 4: Diagrama de secuencia para fase de entrenamiento del MLP

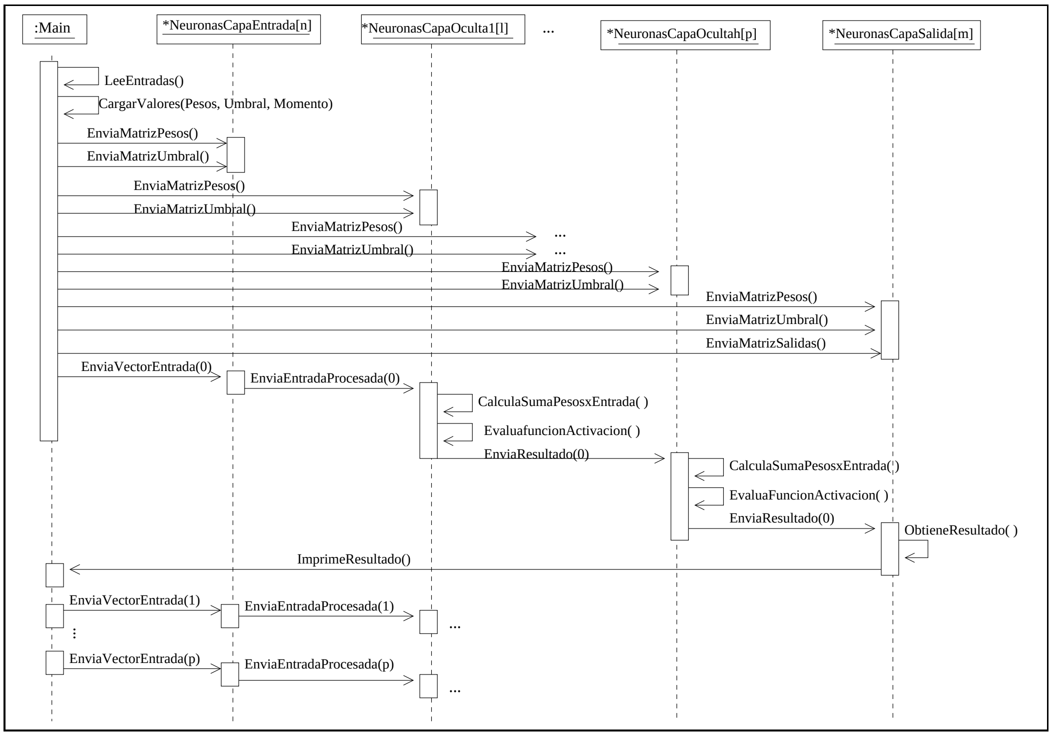

Figura 5: Diagrama de secuencia para la fase de prueba del MLP

El algoritmo de aprendizaje que utiliza la red MLP para el entrenamiento se describe a continuación:

1. Crear el vector de pesos $\mathbf{w}_k = (w_{k1}, w_{k2}, \ldots, w_{kn})^T$ para cada conexión entre las neuronas.
2. Presentar el vector de entrada $\mathbf{x}_q = (x_1, x_2, \ldots, x_n)$ y la salida deseada $\mathbf{d}_q = (d_1, d_2, \ldots, d_n)$ en la capa de entrada.
3. Cada neurona en la capa de entrada recibe los valores de entrada y los pasa a las neuronas de la capa oculta.
   a) Las neuronas de la capa oculta calculan la suma del producto del vector de entrada por su vector de pesos y evalúa la función de activación para calcular la salida, que se envía a la siguiente capa.
   b) El paso anterior se repite para todas las neuronas de las capas ocultas. La última capa oculta envía el resultado a las neuronas de la capa de salida.
   c) Las neuronas en la capa de salida reciben el valor esperado correspondiente al vector de entrada y calcula el error que comete la red. La diferencia entre la salida de la red y el valor esperado se denomina *delta* y se utiliza para actualizar los pesos y los umbral de la conexiones en las capas ocultas.
   d) Las neuronas en las capas ocultas reciben el valor delta de la capa de salida, obtienen la suma de delta multiplicado por la derivada de su función de activación para calcular el error que comente la red y ajustar los pesos y los umbrales para cada conexión.
   e) Las neuronas de salida actualizan los pesos y los umbrales de sus conexiones.
4. El proceso se repite t veces.

Para la fase de prueba, que se lleva a cabo después del entrenamiento de la red MLP, sólo se utiliza la propagación hacia adelante del algoritmo de entrenamiento. Esto se muestra en la figura 5:

1. Iniciar los pesos y los umbrales con el resultado del algoritmo de entrenamiento.
2. Para cada vector de entrada:
   a) Obtener la suma de los pesos por las entrada más los umbrales.
   b) Enviar el resultado a la siguiente capa oculta.
   c) Aplicar la función de activación.
3. Repertit el paso 2 en cada capa hasta la capa de salida.

3.9. Ejemplo resuelto

El problema consiste en diagnosticar si un paciente que ha ingresado a la unidad de cuidado coronario puede sufrir un infarto. Para ello se compara el historial clínico de pacientes que han sufrido esta enfemedad con el historial del paciente que no ha padecido un infarto. El resultado debe ser lo más preciso posible, considerando que a menudo historiales muy similares no se relacionan con la enfermedad que puede sufrir un paciente [Hassoon, 1995].

Para este problema se utiliza una red MLP con el objetivo de obtener mejores resultados que los diagnósticos clínicos o las aproximaciones de sistemas expertos tradicionales. Esta red está formada por cuatro capas de neuronas conectadas completamente. La red se implementa mediante 10 neuronas en la capa de entrada, que se conectan con 10 neuronas de la primer capa oculta. Las neuronas de la primer capa oculta se conectan con 10 neuronas de la segunda capa oculta, y las neuronas de la segunda capa oculta se conectan con las neuronas de la capa de salida. Para el aprendizaje de la red, todas la neuronas utilizan una función de activación sigmoide y el algoritmo de propagación hacia atrás [Hassoon, 1995].

El conjunto de entrenamiento contiene la información de 356 pacientes que han sido admitidos en la unidad de cuidado coronario. De los 356 pacientes, 236 no han sifrido un infarto y 120 lo han padecido. Para el entrenamiento de la red se selecciona aleatoriamente un conjunto de datos tomado de la mitad de los pacientes que han sufrido un infarto (118 pacientes) y la mitad de los pacientes que no lo han sufrido (60 pacientes). El resto de la información se utiliza para verificar el funcionamento de la red en la fase de prueba [Hassoon, 1995].

Los datos de cada paciente se integran por 20 variables que son utilizadas para predecir un infarto. Algunas de estas variables son edad, sexo, náuseas y vómito, respiración agitada, diabetes, hipertensión, angina, entre otras. Estas variables se eligen de un conjunto de 41 variables registradas en los pacientes que ingresaron a la unidad de emergencia de cuidado coronario. La mayoría de la variables pueden ser codificadas en binario, de tal manera que 1 representa la presencia de un síntoma y 0 representa su ausencia. Otras variables, como la edad del paciente, se pueden codificar con valores entre 0.0 y 1.0. La salida de la red consiste de valores binarios: 1 confirma la presencia de un infarto y 0 la ausencia del infarto [Hassoon, 1995].

Después de entrenar a la red, se utiliza la información de los 178 pacientes restantes del conjunto de entrenamiento (118 pacientes no han sufrido un infarto y 60 si lo han padecido). Algunos resultados de la red identificaron correctamente la presencia de un infarto con una tasa de 92 % y su ausencia con un tasa de 96 %. Este resultado mejora el 88 % de certidumbre que proporcionan los diagnósticos tradicionales; el 26 % restante se considera de falsa alarma [Hassoon, 1995].

3.10. Usos conocidos

Generalmente las redes MLP se utilizan en el reconocimiento de patrones, procesamiento de señales, reconocimiento de lenguaje, diagnósticos médicos, modelado de sistemas no lineales, etc.[Hassoon, 1995].

- Una de las primeras aplicaciones de la red MLP es en el **procesamiento del lenguaje** para convertir texto en inglés a habla. NETtalk es un sistema que consiste de 2 módulos: un módulo para el mapeo de la red y el otro módulo está formado por un sintetizador de habla comercial [Hassoon, 1995]. Para este problema se utiliza una red neuronal MLP que tiene 80 neuronas en la capa oculta y 26 neuronas en la capa de salida. Las neuronas de la capa de salida forman un código de uno de los 26 fonemas y la salida de la red maneja un sintetizador de lenguaje que transforma el sonido generado asociado a los fonemas de entrada.

  La entrada de la red es un vector binario de dimensión 203 (que codifica una ventana de 7 caracteres consecutivos, 29 bits para cada 7 caracteres, incluyendo la puntuación; cada caracter es codificado usando un código binario de uno de los 29 fonemas). La salida deseada es el código de un fonema dado para la pronunciación de las letras de entrada.

  El conjunto de entrenamiento para la red MLP está formado por 1024 palabras de un conjunto de fonemas en inglés. NETtalk es capaz de identificar un lenguaje ilegible después de 10 ciclos de entrenamiento con una exactitud de 95 %. Cuando se prueba con un texto nuevo, la red puede distinguir entre vocales y consonantes con una exactitud de 78 %.

- Otra aplicación de la red MLP se lleva a cabo en el **reconocimiento de escritura**. En Estados Unidos, el servicio postal está interesado en reconocer la escritura de los dígitos que aparecen en los correos. Para este problema se diseña una red MLP que reconoce segmentos de números digitalizados en la escritura de los códigos ZIP [Hassoon, 1995].

  La red MLP se entrena con un conjuto de 7291 ejemplares, los cuales contienen números ambiguos, inclasificables, o con algún error de clasificación. Los ejemplos se presentan a través de una transformación lineal que hace que el segmento de un dígito se ajuste en una imagen de 16×16 niveles de grises (los niveles de grises en cada imagen son tranformados y escalados a un rango de 1 a -1).

  La red se entrena durante 23 ciclos a través de un conjunto de entrenamiento que require de 3 días de tiempo de CPU sobre una Sun SparcStarion 1. El porcentaje de patrones mal clasificado es de 0.14 % sobre el conjunto de entrenamiento y 5 % sobre el conjunto de prueba (con 2,007 nuevas muestras).

- La red MLP se emplea en la **compresión de imágenes**. El objetivo es explotar la redundancia que existe en las imágenes para almacenarlas o transmitirlas, de tal manera que se utilice un número menor de bits [Hassoon, 1995].

  La compresión de imágenes es un problema de codificación y decodificación para optimizar la calidad de una imagen codificada, de tal forma que sea posible recuperarla de tamaño original. Para este problema se emplea la red MLP formada por el mismo número de neuronas en la capa de entrada como en la capa de salida y cuenta con una capa de neuronas ocultas. Las neuronas de la capa oculta evalúan una función sigmoide bipolar y las neuronas de la capa de salida tienen una función lineal. Esta red se entrena con un conjunto de vectores de dimensión n formado por números reales y actúa como un codificador.

La red recibe como entrada regiones de 8×8 pixeles de una imagen ($n = 64$) y cuenta con 16 neuronas ocultas. La propagación del error hacia atrás se usa para que la red autoasocie aleatoriamente ventanas seleccionadas de 8×8 pixeles de la imagen dada. Después del entrenamiento, la red se utiliza para descomprimir la imagen de tal manera que se pueda recuperar la imagen total, parte por parte.

3.11. Consecuencias

Ventajas

- La red MLP proporciona resultados más precisos si se utiliza un número grande de neuronas en sus capas ocultas.
- Esta red es poco sensitiva al ruido, ya que los datos para el entrenamiento son procesados muchas veces.
- La red MLP es capaz de proporcionar un resultado cuando la información es redundante, o no se cuenta con la información completa.

Desventajas

- La velocidad para el entrenamiento de la red es muy lenta debido a que tiene que propagarse el error hacia las capas ocultas.
- A pesar de que la red ya ha sido entrenada, la evaluación para un dato nuevo es muy lenta.
- En ocasiones las red no converge a una solución determinada.

---

4. Patrón *Funciones de Base Radial o RBF*

El Patrón RBF describe una red neuronal artificial que utiliza la salida esperada para establecer una relación entre los datos de entrada cuando es importante el tiempo de respuesta.

4.1. También conocido

Aproximador Universal.

4.2. Ejemplo

En la comunicación entre sistemas de cómputo se requiere reconstruir señales binarias, de tal manera que apartir de una señal distorsionada o con ruido, pueda recuperarse la señal original en tiempo real [Luo and Unbehauen, 1997].

4.3. Contexto

El patrón RBF se utiliza cuando se tienen pocos datos relacionados, se conoce de antemano la salida, y es importante el tiempo de respuesta. Sin embargo no se requiere mucha precisión en la respuesta.

4.4. Problema

Establecer la relación entre los datos de entrada y la salida esperada.
Las *Fuerzas* asociadas a este problema son:

- Se cuenta con poca información de entrada.
- No siempre hay dependencia clara entre los datos.
- Se conoce la salida esperada.
- Es importante el tiempo de ejecución.
- Se requiere poca precisión en la respuesta.

4.5. Solución

La Red RBF establece una relación entre los datos de entrada y los datos de salida a partir de ciertos valores esperados. Esta red está formada por neuronas artificiales que calculan la distancia euclidiana entre el vector de entrada y los centros de las funciones base en las neuronas ocultas. En el entrenamiento se lleva a cabo la modificación de los pesos entre las conexiones y se actualizan los parámetros necesarios para las funciones base. El objetivo es establecer la relación que existe entre los datos de entrada y salida esperada por medio de la combinación lineal de funciones base [Hu and Hwang, 2002] y limitar la respuesta de la red en una región local dentro del espacio de entrada para cada función base.

4.6. Estructura

Las redes RBF están formadas por tres tipos de neuronas: n neuronas en la capa de entrada, k neuronas en la capa oculta o intermedia y m neuronas en la capa de salida, $n \leq k \leq m$, con $(n \times k) + (k \times m)$ conexiones. La figura 6 muestra la estructura general de las redes RBF.

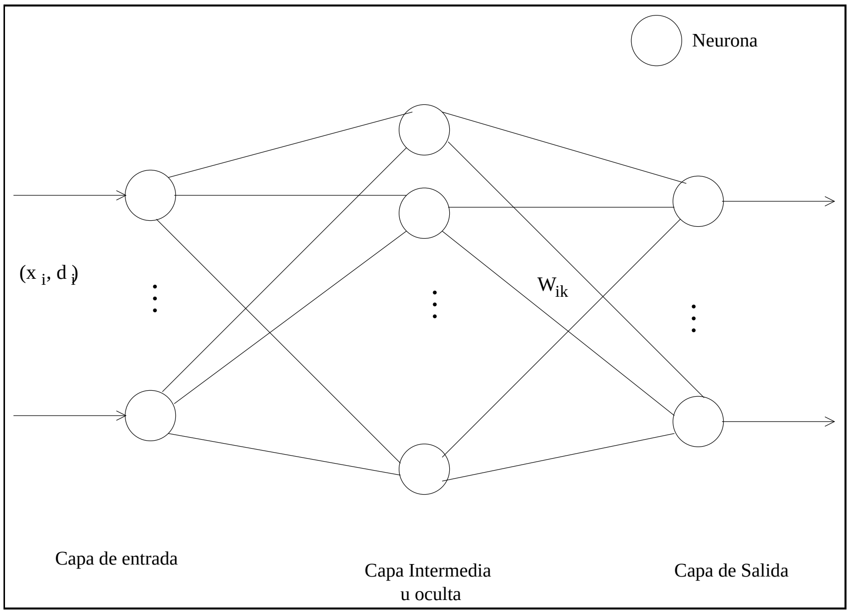

Figura 6: Estructura de la red RBF

4.7. Participantes

- Neuronas de entrada: Estas neuronas reciben los datos para el entrenamiento de la red, es decir, los vectores de entrada, así como los valores de salida esperados. Se encargan de comunicar estos datos a las neuronas de la capa oculta y a las neuronas de la capa de salida.
- Neuronas ocultas: Estas neuronas contienen los parámetros necesarios para las funciones base, como son los centros y la amplitud de dichas funciones. Reciben el vector de entrada y calculan la distancia de este vector a los centros asociados a cada neurona. Evalúan las funciones de activación para actualizar los parámentros necesarios (centros, amplitud de las funciones base y los pesos) y envían el resultado a las neuronas de salida.
- Neuronas de salida: Estas neuronas reciben el resultado de la capa oculta y lo comparan con la salida esperada, obteniendo el error y tratando de mínimizarlo usando el algoritmo de mínimos cuadrados.

4.8. Dinámica

Suponemos que un vector de entrada o patrón de entrenamiento tiene n características y está representado por el vector $\mathbf{x} = (x_1, x_2, \ldots, x_n)$ en el espacio de dimensión n. Sea $\mathbf{c} = (c_1, c_2, \ldots, c_n)$ el vector que contiene los centros de las funciones base en el espacio de dimensión n. La red proyecta el espacio de entrada sobre la salida a través de la capa oculta.

Cada neurona k en la capa oculta se caracteriza por un vector de pesos $\mathbf{w}_k = (w_{k1}, w_{k2}, \ldots, w_{kn})$ de dimensión n, recibe la misma entrada $\mathbf{x} \in \mathbb{R}^n$ y tiene diferente función de activación. En la capa oculta, la red calcula la distancia euclidiana entre el vector de entrada y el vector de centros, evalúa la función de activación y manda tal resultado a la capa de salida.

Durante el aprendizaje, la red calcula los pesos usando el algoritmo de mínimos cuadrados (*Least Mean Square, LMS*). Es decir, considera el error que existe entre la salida de la red y el valor esperado para ajustar la matriz de pesos, los centros y la amplitud de las funciones base minimizando, el error para modelar los datos en un sentido local.

En la capa de salida, las neuronas calculan el resultado utilizando el algoritmo de mínimos cuadrados. De ello depende que se modifique los parámentros de la red. La figura 7 muestra el diagrama de secuencia para la fase de entrenamiento de la red RBF.

El algoritmo de entrenamiento que utiliza la red RBF se describe a continuación:

1. Crear el vector de pesos $\mathbf{w}_k = (w_{k1}, w_{k2}, \ldots, w_{kn})^T$ para cada conexión entre las neuronas de la capa oculta y la capa de salida.

2. Determinar el vector de centros $\mathbf{c}_k = (c_1, c_2, \ldots, c_n)^T$ para cada función base. Existen diferentes métodos que emplean los datos de entrada para escoger los centros de las funciones base [Hu and Hwang, 2002].

3. Presentar el vector de entrada $\mathbf{x} = (x_1, x_2, \ldots, x_n)$ en la capa de entrada. Y la salida deseada $\mathbf{d}_i = (d_1, d_2, \ldots, d_n)$ en la capa de salida.

4. Cada neurona en la capa de entrada recibe los vectores del conjunto de entrenamiento y los envía a las neuronas de la capa oculta. En la capa oculta:

    a) Se calcula el producto punto entre el vector de entrada $\mathbf{x}$ y los centro $\mathbf{c}$ de las funciones base.

    b) Se evalúa la función base y el resultado se envía a las neuronas de la capa de salida.

    En la capa de salida:

    a) Se calcula el error que comete la red comparando el resultado de la red y la salida esperada. El error se minimiza utilizando el algoritmo de mínimos cuadrados.

    b) Se actualizan los centros, la amplitud de las funciones base y los pesos.

5. El proceso se repite t veces.

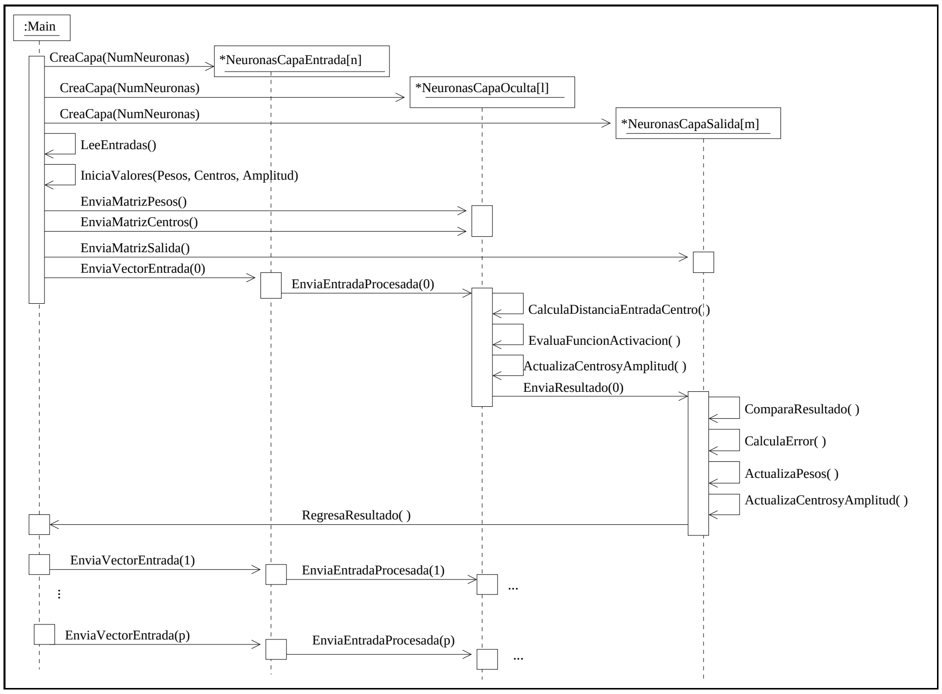

Figura 7: Diagrama de secuencia para el entrenamiento del algortimmo RBF

4.9. Ejemplo resuelto

El problema de reconstruir señales binarias en los sistemas de comunicación en tiempo real se resuelve con un ecualizador no lineal que se basa en redes RBF [Luo and Unbehauen, 1997].

Este ecualizador está formado por tres capas de neuronas. Las neuronas de la capa de entrada evalúan una función lineal y el resultado alimenta la entrada de las neuronas en la capa oculta. Para este problema las conexiones entre la capa de entrada y la capa oculta no tienen pesos. Sin embargo en la capa oculta, cada neurona recibe el valor de entrada correspondiente sin alterar y evalúa las funciones de base radial [Luo and Unbehauen, 1997].

Para este ejemplo se utiliza la función Gausiana y el resultado se denota por $h_i(n)$.

$$h_i(n) = f_i(||Y(n) - \mathbf{c_i}||) \tag{3}$$

La capa de salida está formada por una neurona que calcula la siguiente función de activación:

$$g(x) = \frac{1 - e^{\alpha x}}{1 + e^{\alpha x}} \tag{4}$$

El resultado de la red es el equalizador no lineal que funciona como un umbral. Las conexiones entre las neuronas de la capa oculta y la neurona de la capa de salida sí tienen pesos, denotados por el vector $\mathbf{w}_k$ que representa el pesos de la conexión entre la $k-$ésima neurona en la capa oculta y la neurona de salida [Luo and Unbehauen, 1997].

El parámetro de escalamiento σ, el vector de centros $\mathbf{c}$, y el vector de pesos $\mathbf{w}_k$, $(k = 1, 2, \ldots, n)$ se calculan antes que la salida de la red. Después se estiman los valores de la señal que se transmite. Esto depende del canal de comunicación, que se determina con los datos disponibles y algunos ejemplares conocidos de la señal transmitida [Luo and Unbehauen, 1997].

4.10. Usos conocidos

En general las redes RBF se utilizan en funciones de aproximación, clasificación y modelado de sistemas dinámicos, en series de tiempo, en procesamiento de señales, etc.

- Una aplicación de las redes RBF se lleva a cabo en la **clasificación de sonidos**. El problema consiste en clasificar el sonido de 10 palabras pronunciadas por 67 personas (hombres, mujeres y niños) [Hassoon, 1995].

  Los datos se obtienen haciendo el análisis espectrográfico de las vocales contenidas en las palabras pronunciadas. Para ello se considera la primera y segunda frecuencia de la región de resonancia. Las palabras utilizadas en esta aplicación son: *heed, hid, head, had, hud, hod, heard, hood, who'd,* y *hawed.* La información obtenida se divide aleatoriamente en dos conjuntos, uno con 338 datos para el entrenamiento de la red y el otro con 333 para la etapa de prueba.

La red RBF que se utiliza está formada por 100 neuronas ocultas, con una función de activación gaussiana. Después de la fase de entrenamiento, esta red RBF clasifica correctamente alrededor del 87.1 % de los datos del conjunto de pruba (333 ejemplares).

- En el **Mercado Finaciero**, se usan las redes RBF como una solución alternativa para la opción de precios. La fórmula Black-Scholes se emplea para derivar valores iniciales bajo condiciones muy especificas, pero en circunstancias donde el modelo es no lineal y no existe una solución análitica. Entonces se utilizan las redes RBF. El objetivo es que la red RBF aprenda la fórmula de Black-Scholes y simule los datos. Se supone que esta fórmula determina la opción de precios y emplea el método de Monte Carlo para generar datos de precios artificiales. El resultado de este trabajo muestra que la red RBF es capaz de aprender la opción de precios con una precisión adecuada [Hu and Hwang, 2002].

- Las redes RBF se utilizan para modelar **series de tiempo**. Las series de tiempo requieren estimaciones de un modelo de la forma $y(t) = F(x(t))$ donde F es la función que se desea mapear $F : \mathbb{R}^m \to \mathbb{R}$.

  Un problema típico en las series de tiempo consiste en predecir la serie de Mackey-Glass, observando los datos generados por el retraso de la ecuación diferencial Mackey-Glass:

  $$\frac{\delta x(t)}{\delta t} = -bx(t) + a\frac{x(t-\tau)}{1 + x(t-\tau)^{10}} \tag{5}$$

  con $\tau = 17$, $a = 0{,}2$, y $b = 0{,}1$.

  En este problema se emplea las redes RBF para predecir series de tiempo generadas por una aproximación de la ecuación 5. La red RBF que se utiliza contiene entre 100 y 1000 neuronas. Para seleccionar los centros de las funciones base se consideran los vecinos más cercanos usando un centro como punto base. Sin embargo sólo se aplica para un conjunto finito de datos. Otra forma de seleccionar los centros consiste en utilizar el algoritmo de agrupamiento k-means.

  Computacionalmente, la red RBF para este problema es 16 veces más eficiente que la red MLP, pero requiere alrededor de 27 veces más datos para obtener una precisión similar [Hu and Hwang, 2002].

4.11. Consecuencias

Ventajas

- La velocidad en el entrenamiento de la red es rápida con respecto a la del red Perceptrón Multicapa (MLP).

- La red RBF no requiere muchas neuronas para proporcionar el resultado.
- Se puede controlar la precisión a través de los parámetros de las funciones de activación.
- Requiere poca memoria para la ejecución del programa.

Desventajas

- Necesita un mayor número de información con respecto a MLP para obtener la misma precisión.
- Para obtener resultados más precisos se requiere calcular un número elevado de centros para las funciones de base.
- En ocasiones no converge a una solución y puede caer en algún mínimo local.

5. Patrón *Redes de Hopfield*

El Patrón Redes de Hopfield describe una red neuronal artificial que se utiliza para la clasificación o reconstrucción de patrones a partir de la información almacenada.

5.1. También conocido

Redes de Memoria Asociativa.

5.2. Ejemplo

En reconocimiento de imágenes, se requiere reconstruir imágenes que representan cuatro letras. Las letras están distorsionadas o incompletas. En este caso los datos no definen completamente la imagen y la clasificación debe ser automática [Hilera and Martínez, 1995].

5.3. Contexto

El patrón Redes de Hopfield se utiliza cuando se tiene una gran cantidad de datos con mucho ruido.

5.4. Problema

Encontrar características similares entre los datos de entrada sin supervisión. Las *Fuerzas* asociadas a este problema son:

- Los datos representan un gran número de características.
- Los datos pueden tener dependencias entre ellos.

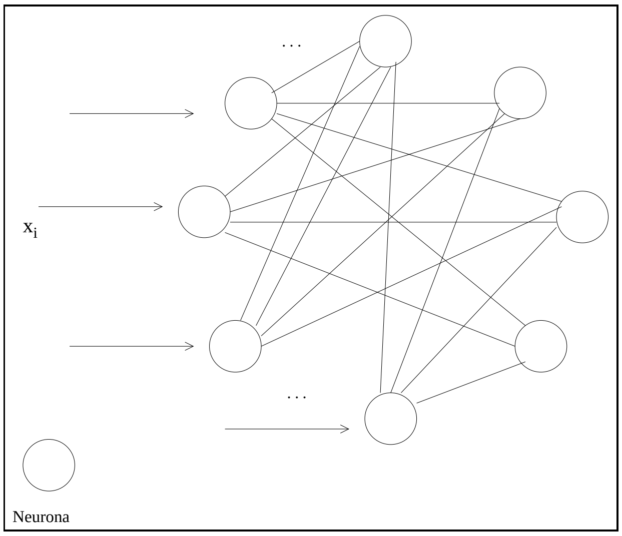

Figura 8: Topología de la red de Hopfield.

- Los datos puede representar solo una parte de la información a clasificar.
- No se requiere supervisión para llevar a cabo la clasificación.

5.5. Solución

La red de Hopfield encuentra características similares entre los datos de entrada. El objetivo es clasificar o reconstruir la información de entrada a partir de la información ya almacenada. El entrenamiento de la red consiste en almacenar un conjuto de información mediante la presentación repetida de los datos de entrada y la adaptación de los pesos. En la fase de prueba, la red es capaz de recuperar la información original, si se presenta un dato ya almacenado. Sin embargo, si la información de entrada no coincide con ningún dato almacenado por estar incompleto o deformado, la red genera la salida más parecida.

5.6. Estructura

Las redes de Hopfield poseen una capa formada por n neuronas. Estas neuronas se conectan entre sí, de tal manera que la red tiene $(n \times n)$ conexiones. En la figura 8 se muestra la estructura general de las redes de Hopfield.

5.7. Participantes

- Neuronas de entrada y salida: En las redes de Hopfield todas las neuronas tienen la misma función. Reciben el mismo vector de entrada, calculan los pesos entre sus conexiones por medio de la multiplicación de matrices y se actualizan de manera asíncrona dependiendo del resultado de la función de activación.

5.8. Dinámica

Suponemos que un vector de entrada o patrón de entrenamiento tiene n características y está representado por el vector $\mathbf{x} = (x_1, x_2, \ldots, x_n)$ en el espacio de dimensión n. Durante el entrenamiento, la red almacena un conjunto de vectores de entrada y proyecta el espacio de entrada sobre la salida. El objetivo es reconstruir la información de entrada a partir de la información almacenada.

Las redes de Hopfield están formadas por una capa con n neuronas conectadas completamente. Cada neurona k se caracteriza por un vector de pesos $\mathbf{w}_k = (w_{1k}, w_{2k}, \ldots, w_{kn})$ de dimensión n. Los pesos de las conexiones son simétricos, es decir, el peso de la conexión entre la neurona k a la neurona j tienen el mismo valor $w_{kj} = w_{jk}$ para $k \neq j$ en otro caso $w_{jk} = 0$.

Cada neurona recibe la misma entrada $\mathbf{x} \in \mathbb{R}^n$, evalúa la misma función de activación y su salida retroalimenta al resto de las neuronas.

La red de Hopfield está diseñada para trabajar con valores binarios -1 y 1. Sin embargo, para el ajuste de los pesos también se consideran los valores de 0 y 1.

Al iniciar la fase de entrenamiento se presenta el vector de entrada $\mathbf{x}$ a la única capa que tiene esta red. Cada neurona recibe como entrada la salida de cada una de las otras neuronas. Estos valores inicialmente coinciden con los datos de entrada, multiplicados por los pesos de sus conexiones correspondientes. La suma de todos estos valores corresponde a la entrada de otra neurona, la cual evalúa la función de transferencia. Este procedimiento es el primer paso del algoritmo de entrenamiento y se repite hasta que la salida de las neuronas se estabilicen, es decir, la salida de las neuronas no cambie su valor. Cuando esto sucede se genera la salida de la red.

El funcionamiento de la red de Hopfield se basa en la regla de Hebb (referencia). Esta regla es un método para determinar los pesos de la red utilizando la información de entrada [Fausett, 1994]. La figura 9 muestra el diagrama de secuencia para el entrenamiento de la red de Hopfield. El objetivo es establecer el cambio de los pesos entre dos neuronas de manera proporcional. A esta proporción se le denomina tasa de aprendizaje. Para obtenerla se localizan los minímos locales de la función de activación, de tal manera que al presentarse un nuevo dato de entrada, este valor sea asociado a un dato ya almacendado como un minímo local.

El algoritmo de aprendizaje que utiliza la red de Hopfield se describe a continuación:

1. Iniciar el vector de pesos $\mathbf{w}_k = (w_{k1}, w_{k2} \ldots, w_{kn})^T$ para las conexiones entre las neuronas.

2. Presentar el vector de entrada $\mathbf{x} = (x_1, x_2, \ldots, x_n)^T$ a cada neurona.

3. El estado inicial de la red, en el tiempo $t = 0$, es igual al vector de entrada $y(0) = x(0)$.

   a) Calcular el nivel de activación de la neurona k en el tiempo $t + 1$ como la suma del producto punto de los pesos por el estado anterior de la red. Este valor se denota por $h_k(t + 1)$

   b) Determinar la activación de cada neurona en el tiempo $t+1$ evaluando la función de activación. Generalmente se utliliza la función signo (sgn) o escalón, sobre el nivel de excitación $h_k(t + 1)$ determinado por:

$$y_k(t+1) = sgn(h_k(t+1)) = \begin{cases} +1 & \text{si} \quad h_k(t+1) > 0 \\ -1 & \text{si} \quad h_k(t+1) < 0 \\ y_k(t) & \text{si} \quad h_k(t+1) = 0 \end{cases} \qquad (6)$$

   c) Difundir el resultado a todas las neuronas y actualizar su estado.

   d) El proceso se repite hasta alcanzar un estado estable. Es decir, que el estado de todas las neuronas no se modifique.

$$y_k(t) = y_k(t+1) \qquad (7)$$

5.9. Ejemplo resuelto

En el sistema de reconocimiento de imágenes se consideran únicamente la reconstrucción de cuatro letras (A, B, C y D). Estas letras se codifican como matrices de 7×6 pixeles y cada letra puede estar distorcionada o incompleta [Hilera and Martínez, 1995].

Para este problema se emplea una red de Hopfield capaz de recuperar la información de entrada apartir de la información parcial o deformada de la misma. Para ello se emplea un red formada por una capa con 42 neuronas, considerando una neurona por pixel, con $42 \times 42 = 1722$ conexiones. Se utilizan valores binarios: negro = +1 y el blanco =-1 [Hilera and Martínez, 1995].

Durante la fase de entrenamiento, se almacenan los datos correspondientes a las cuatro letras. Después se calculan los 1,722 pesos que tiene la red utilizando los datos de entrada. Posteriormente en la fase de prueba, la red es capaz de reconocer una imagen de entrada distorsionada y genera la salida más parecida a los datos de entrada [Hilera and Martínez, 1995].

5.10. Usos conocidos

Las redes de Hopfield generalmente se emplean en reconocimiento de imágenes y resolución de problemas de optimización.

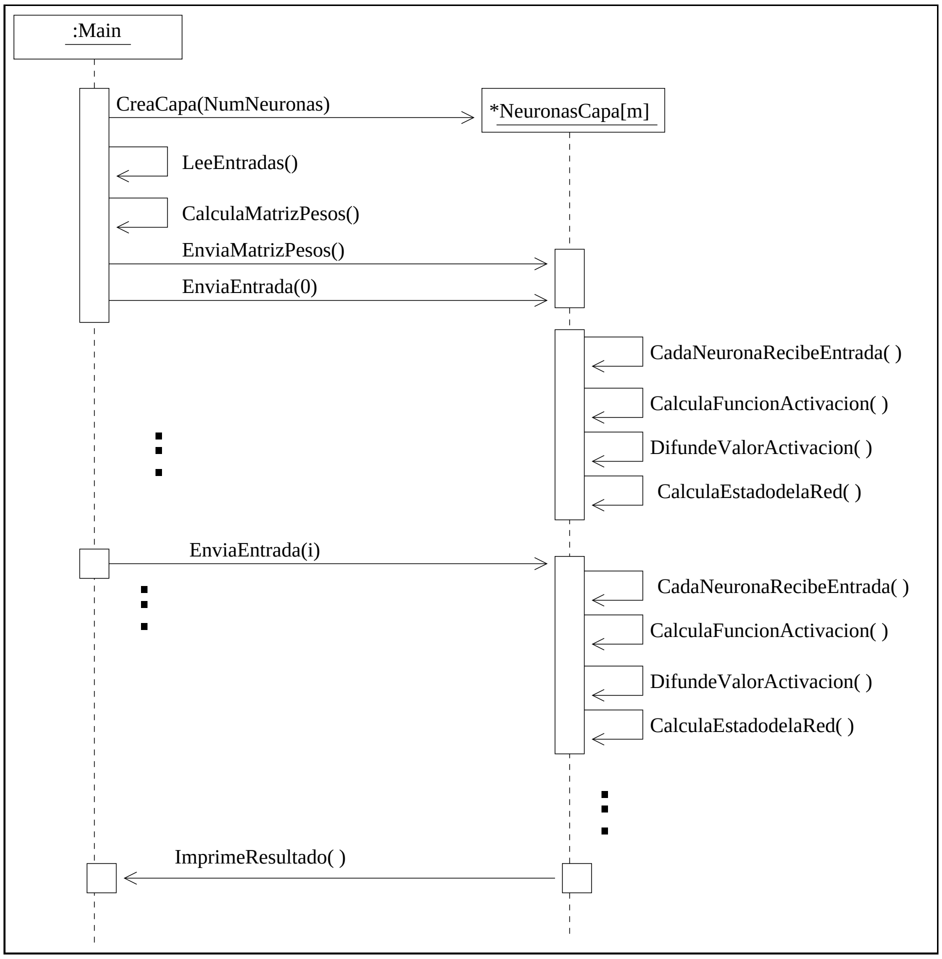

Figura 9: Diagrama de secuencia para el entrenamiento de la red de Hopfield

- En el **reconocimiento de imágenes** se utiliza la red de Hopfiel para reconstruir versiones distorcionadas o parciales de imágenes almacenadas. Se diseña una red con 234000 neuronas para reconocer imágenes de 130×180 pixeles. Durante el entrenamiento, la red almacena siete imágenes. En la etapa de prueba, se presentan imágenes distorsionadas e incompletas y la red las asocia al patrón más parecido que haya almacenado antes [Hilera and Martínez, 1995].

- La red de Hopfield se ha empleado para resolver **problemas de optimización**. El objetivo es expresar un problema mediante una expresión matemática, llamada función de costo, de tal manera que se busca minimizarla. Por ejemplo para desarrollar conversores analógicos-digitales (A/D), el diseño se plantea como la solución a un problema de optimización en términos de una función de energía. Los conversores A/D son circuitos electrónicos que reciben como entrada una señal analógica (por ejemplo, una señal de audio) y genera una serie de configuraciones binarias o palabras con un cierto número de bits. Mientras mayor sea el número de bits, mayor es la precisión en la conversión. Un conversor A/D se caracateriza por el número de dígitos binarios que componen cada palabra generada: es posible tener 2^n palabras diferentes, donde n es el número de bits de cada palabra. El objetivo es reconstruir una señal analógica a partir de los valores de las entradas y obtener una señal semejante a la original. Para el diseño del conversor A/D se utiliza la red de Hopfield: primero se define la función que se quiere minimizar y se ajusta mediante un término extra que garantice la estabilidad; segundo, se obtiene los pesos; finalmente, la implementación física se hace usando una red con 4 neuronas [Hilera and Martínez, 1995].

- Otra aplicación de las redes de Hopfield se lleva a cabo en la solución del problema del **agente viajero**. Un viajero tiene que visitar n ciudades, de tal manera que saliendo de una ciudad visite todas las ciudades sin pasar más de una vez por cada una y además recorra la distancia menor entre cada ciudad. Al final debe regresar a la ciudad de partida [Corchado et al., 2000, Hilera and Martínez, 1995]. Para este problema se utiliza una red de Hopfield con $n \times n$ neuronas. Si se visitan cinco ciudades, la red de Hopfield está formada por 25 neuronas conectadas completamente. Se inicializan los pesos y se establece una función de optimización de tal manera que después de entrenar a la red, es capaz de presentar la ruta con la distancia mínima.

5.11. Consecuencias

Ventajas

- La red Hopfield reconstruye imágenes deformadas a partir de la información almacenada.

- La red asocia datos de entrada a patrones ya identificados.

Desventajas

- El número máximo de patrones que puede almacenar la red es limitado.
- Requiere mucho tiempo de procesamiento para converger a una solución estable.
- Puede caer en mínimos locales y no converger a una solución.
- La red no puede recuperar correctamente un patrón ya almacenado si ha sufrido una rotación o traslación [Hilera and Martínez, 1995].

---

6. Resumen

El Sistema de Patrones de Software para Redes Neuronales Artificiales proporciona criterios de selección a partir de las caractísticas del problema y la experiencia obtenida en problemas ya conocidos, de tal manera que se propone una red neuronal artificial como solución.

Este sistema esta formado por cuatro redes neuronales artificiales, las cuales se describen mediante la forma POSA: *(a)* la red de Mapas Auto-Organizados, *(b)* el Perceptrón Multicapa, *(c)* las redes de Funciones de Base Radial y *(d)* las redes de Hopfield. Estas redes se utilizan principalmente en la clasificación de los datos de entrada. Como lo considera la teoría de Patrones de Software, este no es un conjunto exahustivo, sino que el presente sistema se considera más bien como un conjunto inicial de patrones de software para redes neuronales, que puede ser complementado y modificado a futuro.

Referencias

[Buschmann et al., 1996] Buschmann, F., Meunier, R., Rohnert, H., Sommerlad, P., and Stal, M. (1996). *Pattern-Oriented Software ArchitectureA System of Patterns.* John Wiley and Sons Ltd, Chichester.

[Caudill and Butler, 1992] Caudill, M. and Butler, C. (1992). *Understanding Neural Networks: Computer Explorations*, volume 1: Basic Networks. The MIT Press, London, England.

[Corchado et al., 2000] Corchado, J. M., Díaz, F., Borrajo, L., and Fernández, F. (2000). *Redes Neuronales Artificiales, Un enfoque práctico.* Servicio de Publicacións de Univerdidade de Vigo, España.

[Fausett, 1994] Fausett, L. (1994). *Fundamentals of neural networks Architectures, algorithms, and applications.* Prentice-Hall, United States of America.

[Freeman and Skapura, 1992] Freeman, J. A. and Skapura, D. M. (1992). *Neural Networks. Algorithms Applications and Progamming Techniques.* Addison Wesley, United States of American.

[Hassoon, 1995] Hassoon, M. H. (1995). *Fundamentals of Artificial neural network*. The MIT Press, London England.

[Hilera and Martínez, 1995] Hilera, J. R. and Martínez, V. J. (1995). *Redes Neuronales Artificiales Fundamentos, modelos y aplicaciones*. RA-MA, Madrid, España.

[Hu and Hwang, 2002] Hu, Y. H. and Hwang, J. (2002). *Handbook of Neural Network Signal Processing*. CRC Press, New York Washington, D.C.

[Luo and Unbehauen, 1997] Luo, F. and Unbehauen, R. (1997). *Applied Neural Networks for Signal Processing*. Cambridge University Press, United States of America.

[Pal and Mitra, 1999] Pal, S. K. and Mitra, S. (1999). *Neuro-Fuzzy Pattern Recognition, Methods in Soft Computing*. Wiley Series on Intelligent Systems. John Wiley & Sons inc., United States of America.