Programación Orientada a Objetos: Simulación de una Red de Propagación Inversa

Rubén Peredo Valderrama
Profesor e Investigador del CINTEC-IPN.
Francisco F. Cordova Quiroz
Alumno de la Maestría del CINTEC-IPN.
Eduardo Rodríguez Escobar
Jefe del Departamento de Sistemas Digitales del CINTEC-IPN.

El presente artículo tiene la finalidad de mostrar el diseño de un simulador del modelo básico propuesto por Werbos [1974], de una red neuronal de propagación inversa (back-propagation) en lenguaje C++ orientado a objetos de Borland, versión 4.5 para Microsoft Windows.

Introducción

La red neuronal de propagación inversa es un modelo muy popular, dado que no tiene conexiones de retroalimentación y los errores son propagados inversamente durante el entrenamiento. Algunas aplicaciones pueden ser resueltas usando redes de propagación inversa y, dada su topología, es muy apropiada para multiestratos ("multilayer"). Los errores de la salida determinan la medida de los estratos ocultos, los cuales son usados como base para el ajuste de las conexiones de pesos entre la entrada y los estratos ocultos. El ajustar dos pesos entre parejas de estratos y recalcular la salida es un proceso interactivo que es acarreado hasta que el error cae por debajo del nivel de tolerancia. Los parámetros de la proporción de aprendizaje escalan el ajuste de los pesos. Un parámetro llamado momentum también puede ser usado en la escala de ajuste desde una interacción previa y adicionando el ajuste en la interacción presente.

Operación de la Red de Propagación Inversa

En este punto se explicara el funcionamiento de una red de propagación inversa. En principio la red aprende de una entrada-salida definida utilizando un ciclo de **propagación-adaptación**. Después de que el patrón ha sido aplicado como un estimulo al primer estrato de la red, este es propagado a través de cada estrato superior hasta que la salida es generada. La salida es entonces comparada con el valor deseado, y la señal de error es calculada para cada salida.

La señal de error se transmite desde el estrato de salida a cada nodo en el estrato intermedio, recibiendo únicamente una porción del error total y basado únicamente en la contribución relativa hecha a la salida original. Este proceso se repite, estrato por estrato, hasta que cada nodo en la red ha recibido una señal de error que describe las contribuciones relativas del error total. Basado en la señal de error que recibe, los pesos de las conexiones son entonces actualizados para cada unidad, ocasionando que la red converja hacia el estado que permite a todos los patrones de entrenamiento ser decodificados.

El significado de este proceso es que, durante el entrenamiento, los nodos en los estratos intermedios se organizan por si mismos tal como los diferentes nodos aprenden a reconocer diferentes características del espacio de entrada. Después del entrenamiento, cuando se presenta un patrón de entrada arbitrario (ruido o incompleto), las unidades en los estratos ocultos de la red responderán con una salida activa si la nueva entrada contiene un patrón semejante a las características de las unidades individuales aprendidas para reconocer durante el entrenamiento. Contrariamente, las unidades del estrato oculto tienen la tendencia a inhibir sus salidas si no tienen el patrón con él cual fueron entrenadas para reconocer. Como la señal se propaga a través de los diferentes estratos en la red, la actividad del patrón presente en cada estrato superior puede ser pensado como un patrón con características que pueden ser reconocidas por unidades en subsecuentes estratos. El patrón de salida generado puede verse como un mapa característico que proporciona la indicación de la presencia o ausencia de diversas combinaciones de características en la entrada. El efecto total de su funcionamiento es que la red de propagación inversa proporciona un

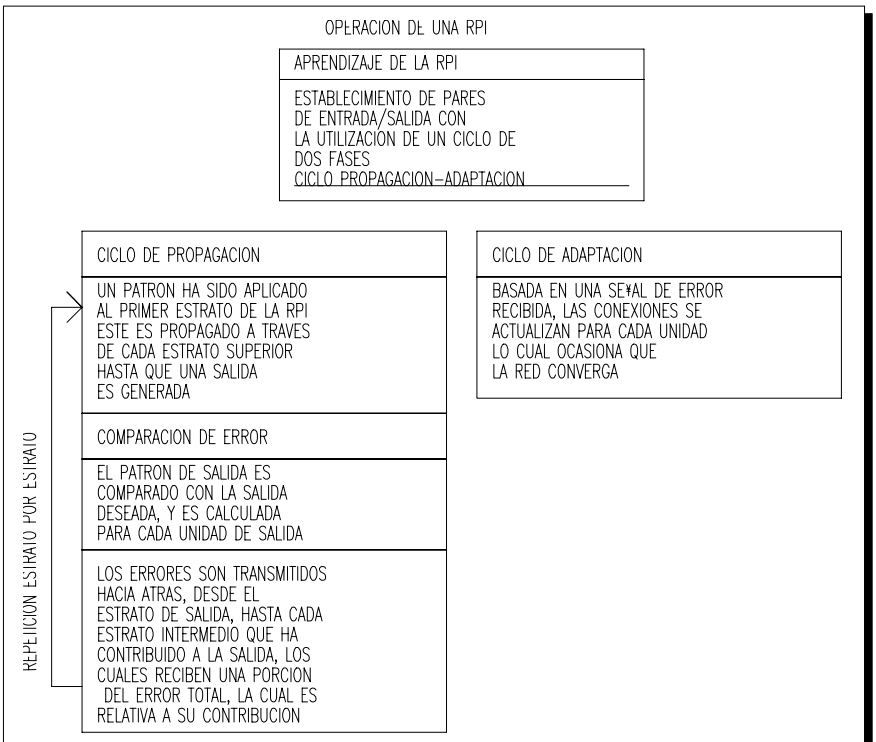

Figura 1. Esta figura muestra el funcionamiento de una red del tipo RPI.

efectivo significado al permitir a un sistema computarizado examinar patrones de datos que pueden ser incompletos o que pueden contener ruido, y reconocer patrones parciales de entrada. Esto puede apreciarse en la **figura 1**.

En muchas ocaciones se ha encontrado que durante el entrenamiento (**figura 3**), las redes de propagación inversa tienden a desarrollar relaciones internas entre nodos, así como a organizar el entrenamiento de datos dentro de ciertas clases de patrones. Esta tendencia puede ser extrapolada a la siguiente hipótesis: todas las unidades de los estratos ocultos en una red de propagación inversa están asociados con características especificas de los patrones de entrada como resultado de su entrenamiento. Esta asociación puede ser o no evidente a los observadores humanos. Lo importante es que la red ha encontrado una representación interna que posibilita las salidas deseadas cuando se dan las entradas de entrenamiento. Esta representación interna puede ser aplicada a las entradas que no fueron usadas durante el entrenamiento. La red de propagación inversa clasificará estas entradas de acuerdo a las características que comparten con las entradas de entrenamiento.

Construcción de una Red Neuronal de Propagación Inversa

La arquitectura de una red de propagación inversa se muestra en la **figura 2**. Pueden exisitir más estratos ocultos, pero en esta figura solo se representará uno. De igual forma, el número de neuronas en el estrato de entrada y en el estrato de salida están determinados por las dimensiones de los patrones de entrada y salida, respectivamente. Como en el

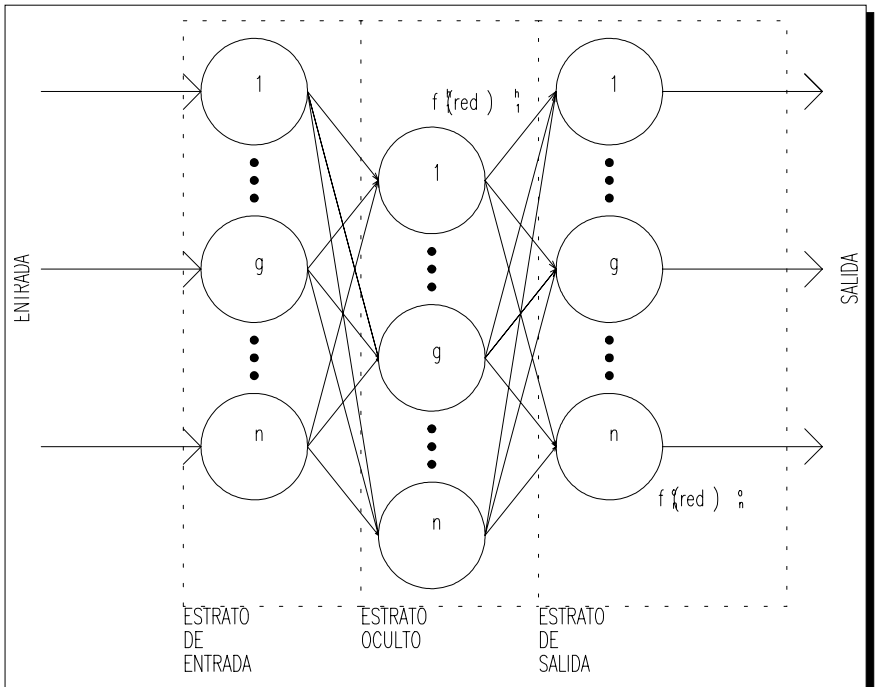

Figura 2. Esta figura muestra una arquitectura típica de una red del tipo RPI, y sus correspondientes estratos

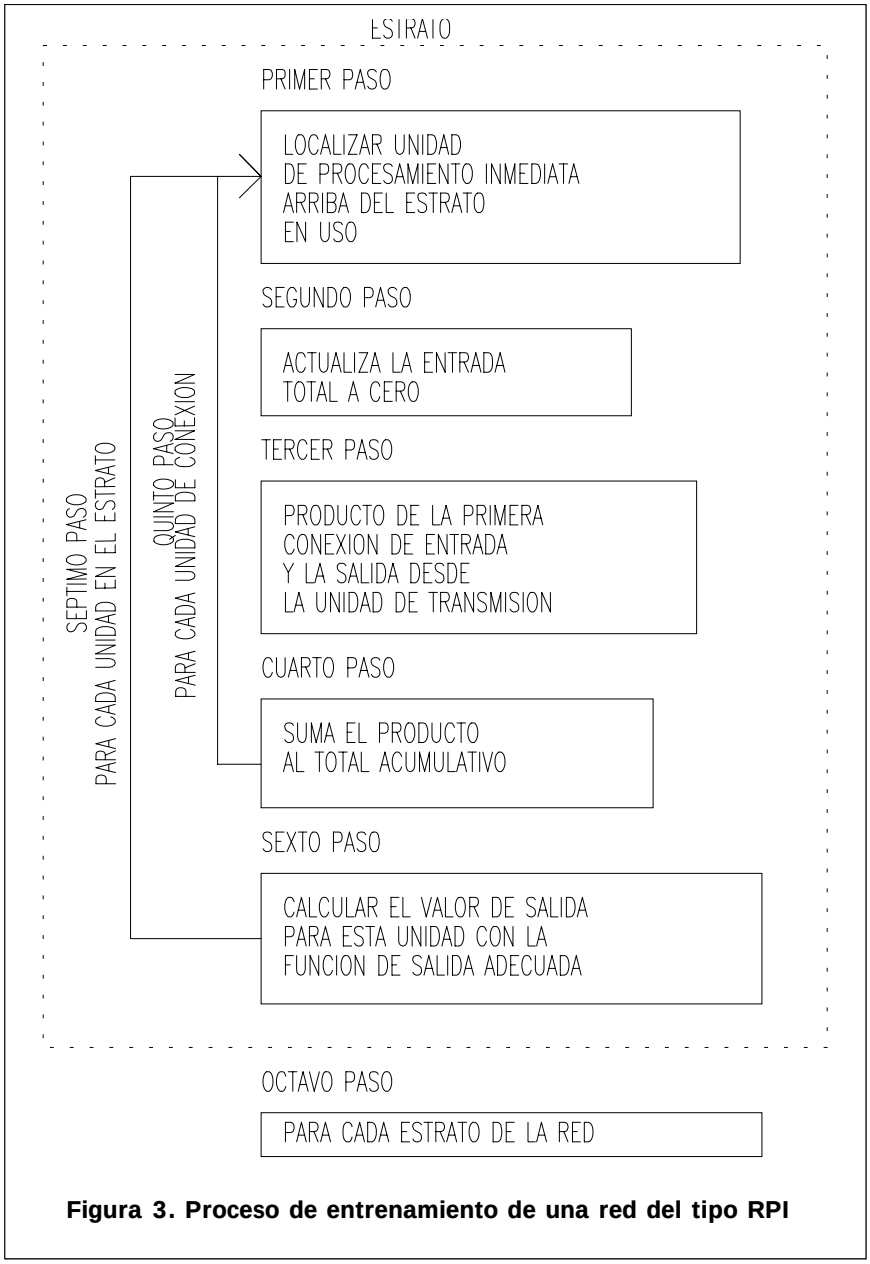

Figura 3. Proceso de entrenamiento de una red del tipo RPI

número de estratos ocultos, tampoco es fácil determinar cuantas neuronas son necesarias. Para evitar realizar una figura muy compleja se dibujará con cinco neuronas de entrada, tres neuronas en el estrato oculto, y cuatro neuronas de salida con algunas representaciones de conexiones.

Tamaño de la Red

¿Cuantos nodos son necesarios para resolver un problema particular? La respuesta involucra de manera considerable a los datos de entrenamiento, dado que no existe una respuesta estricta para resolver la pregunta. Por lo general 3 estratos son suficientes. Sin embargo algunas veces un problema podría ser más fácil de resolver con un solo estrato oculto. En este caso se entiende que más fácil significa que la red aprende más rápido.

El tamaño del estrato de entrada es usualmente dictado por la naturaleza de la aplicación. Frecuentemente se puede determinar el número de nodos de salida sabiendo si se buscan valores analógicos o valores binarios en las unidades de salida.

La determinación del número de unidades a usar en el estrato oculto comúnmente no es fácil como lo es en el caso de los estratos de entrada y salida. La idea principal es usar tan pocos estratos ocultos como sea posible, porque cada unidad adiciona una carga extra durante la simulación en el CPU de la computadora. De seguro, un sistema que es totalmente implementado en hardware (un procesador por elemento de procesamiento), adiciona cargas adicionales de trabajo al CPU.

Para una medida razonable, el tamaño del estrato oculto necesita ser relativamente una fracción más pequeña del estrato de entrada. Si la red falla al converger, se podría intentar adicionar más nodos ocultos para solucionar este problema. Si la red converge, se podría intentar reducir el número de nodos ocultos y asentar un tamaño básico para la ejecución de todo el sistema.

También es posible remover unidades ocultas que son superficiales. Si al examinar los valores de los pesos en los nodos ocultos periódicamente, cuando la red entrena, seguramente se observará que algunos pesos en ciertos nodos no cambian mucho de acuerdo a sus valores iniciales. Estos nodos puede que no participen del proceso de aprendizaje, y con ello es posible reducir el número de nodos ocultos, al mínimo necesario.

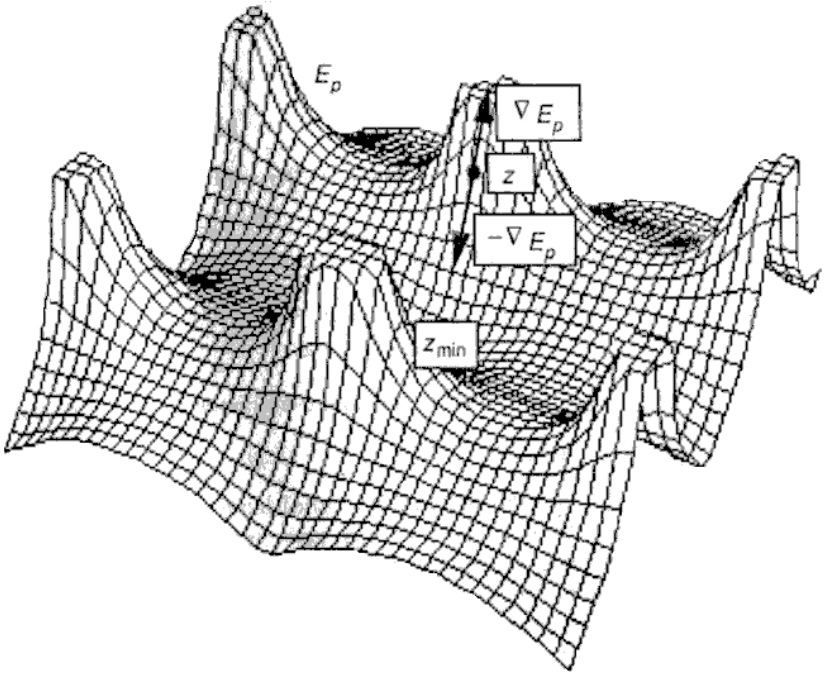

Figura 4. Una superficie hipotética de pesos en el espacio. El gradiente, ∇E_p , el punto z aparece a lo largo del gradiente negativo. Los cambios de pesos deberían ocurrir en la dirección del gradiente negativo, el cual es el punto más bajo de la superficie en el punto z. Además, los pesos cambiados deberían estar interactuando hasta que E_p alcance el punto mínimo que en este caso es z_{min}.

El factor de un ½ en la **ecuación 1** es por conveniencia para el calculo de la derivación. Con esto una constante arbitraria aparecerá en el resultado final, y la presencia de este factor no invalida la derivación.

Al determinar la dirección en la cuál cambian los pesos, debemos calcular el gradiente negativo de Ep, por lo tanto, ∇E_p con respecto a los pesos, w_{kj} . Por lo tanto, podemos ajustar los valores de los pesos tal que el error sea reducido. Esto sirve para pensar en Ep como una superficie de pesos en el espacio. La **figura 4** muestra un ejemplo en donde la red tiene únicamente pesos. La **figura 5** muestra un corte transversal de una superficie de pesos.

Para mantener de manera simple el desarrollo, se considerara cada componente de ∇E_p en forma separada. De la **ecuación (1)** y de la definición de δ_{pk}, obtenemos la siguiente ecuación

Actualización de Pesos en los Estratos

En la derivación de la regla de la delta, el error para el k ésimo elemento de entrada del vector es:

$$\delta_{pk} = (y_{pk} - d_{pk})$$

y donde la salida deseada es y_{pk} y la salida actual es o_{pk}. y los sufijos "p" se refiere al p-ésimo vector de entrenamiento, y "k" se refiere a la k-ésima unidad de salida. El error, que es minimizado por la regla generalizada de la delta, la cuál a su vez es el algoritmo de aprendizaje para la red, es la suma de los cuadrados de los errores de todas las unidades de salida:

$$E_p = \frac{1}{2}\sum_{k=1}^{M} \delta^2{}_{pk} \quad \text{.....(1)}$$

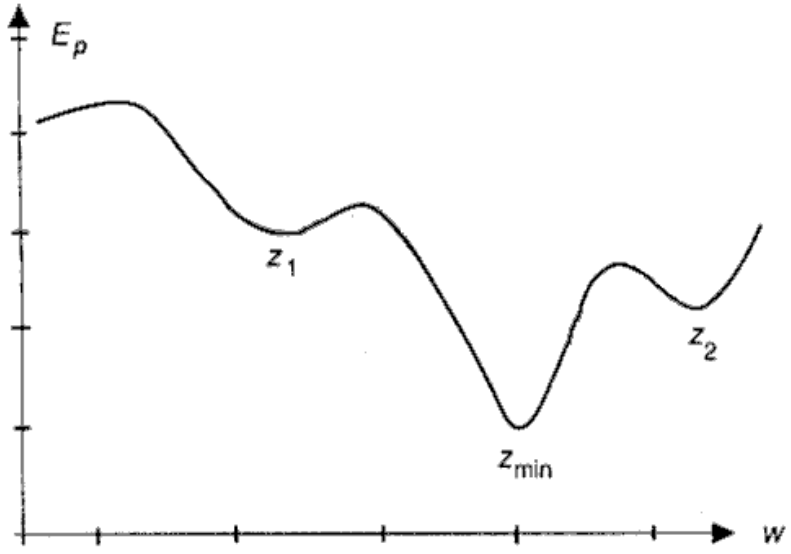

Figura 5 Esta gráfica muestra un corte de una superficie hipotética de error de pesos en el espacio. El punto, zmin, es llamado el mínimo global. Nota sin embargo, que hay otros puntos mínimos, z1 y z2. Un gradiente descendente busca el mínimo global lo que podría accidentalmente encontrar uno de estos mínimos locales en vez del mínimo global.

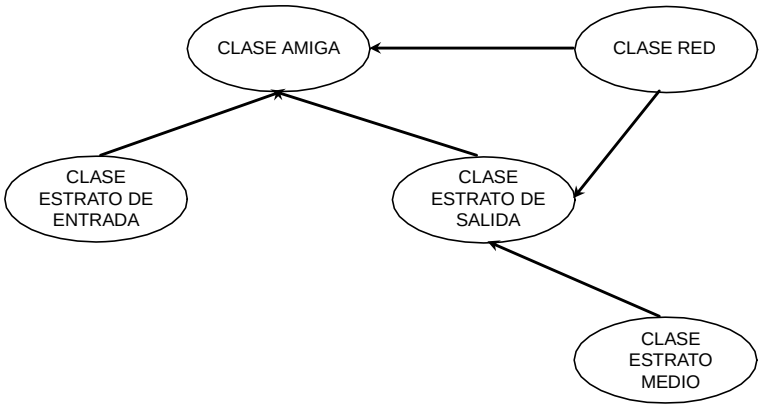

Figura 6. Muestra las clases de la red del tipo de RPI y su relación mutua en POO

$$E_p = \frac{1}{2}\sum_k (y_{pk} - o_{pk})^2 \quad \text{.....(2)}$$

y

$$o_{pk} = f_k^o(net_{pk}^o) \quad \text{.....(3)}$$

Sustituyendo (3) en (2) y derivando parcialmente

$$\frac{\partial E_p}{\partial w_{kj}^o} = -(y_{pk} - o_{pk})\frac{\partial f_k^o}{\partial (net_{pk}^o)}\frac{\partial (net_{pk}^o)}{\partial w_{kj}^o} \quad \text{.....(4)}$$

donde utilizamos la regla de la cadena para derivadas parciales. Por el momento, no intentaremos evaluar la derivada de f_k^o, en cambio escribiremos esto de manera más simple como:

$$f_k^{o'}(net_{pk}^o)$$

El último factor de la **ecuación (4)** es:

$$\frac{\partial (net_{pk}^o)}{\partial w_{kj}^o} = \left(\frac{\partial}{\partial w_{kj}^o}\sum_{j=1}^{L} w_{kj}^o i_{pj} + \theta_k^o\right) = i_{pj} \quad \text{.....(5)}$$

$$\frac{\partial E_p}{\partial w_{kj}^o} = -(y_{pk} - o_{pk}) f_k^{o'}(net_{pk}^o) i_{pj} \quad \text{.....(6)}$$

donde los pesos a la salida del estrato de salida se actualiza de acuerdo a la **ecuación (7)**.

$$w_{kj}^o(t+1) = w_{kj}^o(t) + \Delta_p w_{kj}^o(t) \quad \text{.....(7)}$$

$$\Delta w_{kj}^o = \eta (y_{pk} - o_{pk}) f_k^{o'}(net_{pk}^o) i_{pj} \quad \text{.....(8)}$$

donde h es el parámetro de proporción de aprendizaje en la ecuación de actualización de los pesos (8).

Implementación en Software de la Red Tipo RPI

En la **figura 6** se muestran las clases utilizadas en el programa y su relación entre ellas. La clase *estrato* se ramifica en dos. Una de estas ramificaciones es la clase *entrada_de_estrato* y la otra es la clase *salida_de_estrato*. La clase *estrato_medio* es muy parecida a la *salida_de_estrato* por lo cuál es derivada de esta clase.

La **figura 7** muestra a la red neuronal, con tres estratos para esta red. Un estrato contiene neuronas y pesos. El estrato es responsable de calcular su salida y almacenarla, y calcular y almacenar los errores para cada una de las neuronas. Notese que la clase de entrada no tiene pesos asociados a ella y por lo tanto es un caso especial, ya que el propósito del estrato de entrada es almacenar datos que han de ser propagados al siguiente estrato.

La **figura 8** es un esquema que trata de mostrar de manera clara una red de propagación inversa, con el cuál se pueden crear variaciones sobre la red y sus respectivas conexiones de retroalimentación.

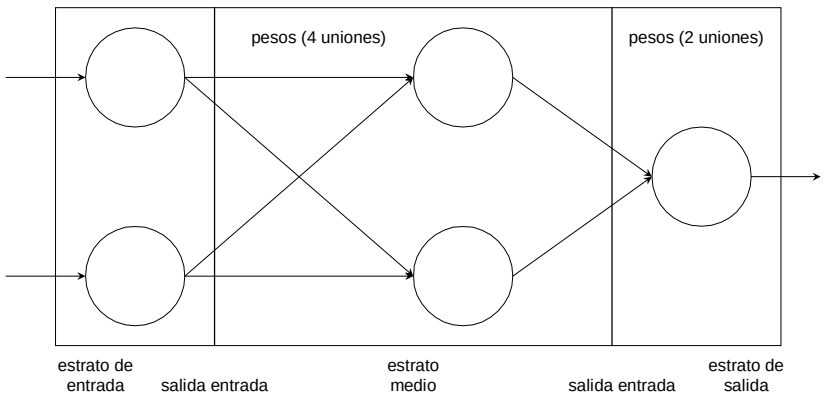

Figura 7. Organización de los estratos para la red de propagación inversa.

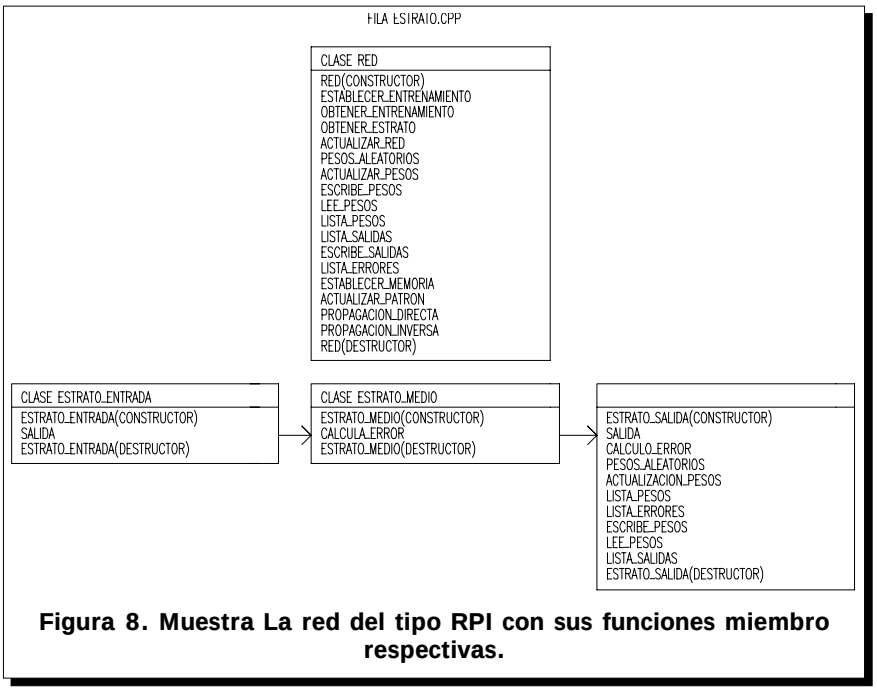

Figura 8. Muestra La red del tipo RPI con sus funciones miembro respectivas.

Las clase de una red de propagación inversa almacena arreglos de punteros a estratos y arreglos de tamaño de estratos para todos los estratos definidos. Estos objetos tipo estrato y arreglos son dinámicamente localizados en el heap con las palabras clave *new* y *delete* de C++. Como se puede ver, la red de propagación inversa puede rápidamente saturar la memoria y el CPU de una computadora, con redes largas y condiciones extensas de entrenamiento. El tamaño y arquitectura de la red por lo tanto estarán determinadas por la memoria y el CPU de la computadora.

La red del tipo RPI con sus clases y funciones miembros se muestra en la **figura 8**.

Conclusiones

Este modelo es uno de los más poderosos dentro de la teoría de redes neuronales, pero cuanta con un problema, y es que requiere de un gran poder de computo debido su sistema de entrenamiento, pero es excelente para el reconocimiento de patrones, por lo cuál se hace importante el encontrar un algoritmo adecuado para cada necesidad, y lograr disminuir su gran consumo de recursos computacionales.

Bibliografía

[1] García-Pelayo y Gross, Ramón; *"Diccionario Pequeño Larousse en Color"*, 1981.

[2] *"Standard Dictionary of the English Language"*; Funk & Wagnalls, 1967.

[3] Wienderhold, Gio; *"File Organization for Data Base Design"*; McGraw-Hill, 1967.

[4] Shaft, Adam; *"Historia y Verdad. Teoría y Praxis"*.

[5] Villee, Claude A.; *"Biología"*, 1986.

[6] van Gigch, Jonh P.; *"Teoría General de Sistemas"*, 1990.

[7] Hofstadter, Douglas R.; *"Godel, Escher, Bach: Una Eterna Trenza Dorada"*; CONACYT, 1982.

[8] Flores, Edmundo; *"La Creación de la Nueva Ciencia del Caos y el Ocaso de la Meteorología y de la Econometría"*; El BUHO, No. 253, EXCELSIOR, 1990.

[9] Freeman, James, *"Neural Networks, Algorithms, Applications, and Programming Techniques"*, Addison Wesley, 1991.

[10] Blum, Adam, *"Neural Networks in C++"*, Wiley, 1992.

[11] Kosko Bart, *"Neural Networks and Fuzzy Systems"*, Prentice Hall 1992.

[12] Freeman & Skapura, *"Neural Networks"*, Addison Wesley 1992.