Red Neuronal de base radial (*RBFN*)

Rafael Pérez Torres
Profesor: Dr. Wilfrido Gómez Flores,
LTI Cinvestav.

***Resumen*—Las redes neuronales de función de base radial (RBFN) son una alternativa al perceptrón multicapa. A diferencia del perceptrón multicapa, las redes neuronales de función de base radial no requieren realizar la retropropagación del error para realizar el ajuste de los pesos y además mantienen una estructura fija compuesta de una capa de entrada, una capa escondifa y una de salida.**

Este tipo de redes resultan útiles para aproximar funciones y además realizar la clasificación de patrones.

En este documento se presenta la implementación del entrenamiento híbrido y clasificación utilizando una red neuronal de función de base radial. El entrenamiento se ha realizado variando la cantidad de neuronas en la capa oculta, así como la función de media utilizada para realizar el cálculo de los radios. Se muestran las regiones particionadas generadas por las mejores configuraciones obtenidas durante el entrenamiento.

***Index Terms*—Reconocimiento de patrones, Redes neuronales de función de base radial, RBFN**

I. Introducción

Las redes funcionales de función de base radial surgen como un método alternativo al perceptrón multicapa, siendo propuestas por Broomhead y Lowe en el año de 1988.

Este tipo de redes definen una topología fija consistente en las siguientes capas:

- **Entrada**: recibe y distribuye los datos desde el exterior.
- **Oculta**: activada por funciones radiales *no lineales*.
- **Salida**: activada por funciones *lineales* continuas.

La salida de este tipo de redes está definida por una transformación no lineal en la capa oculta y una lineal en la capa de salida. Existen múltiples funciones de tansformación no lineal. Dentro de las más comunes se encuentran

- Función Gaussiana.
- Función multicuadrática.
- Función multicuadrática generalizada.
- Función multicuadrática inversa.
- Función multicuadrática inversa generalizada.

Las siguientes secciones describen de forma breve el marco teórico tras el entrenamiento y la clasificación de las RBFN.

II. Marco teórico

El entrenamiento y la clasificación de una RBFN conllevan procesos similares, cuyo común denominador, es la función de activación (transferencia) evaluada en cada una de las neuronas.

La función Gaussiana es la más utilizada con estos fines, siendo definida como:

$$\phi_i(x) = \exp(-\frac{||x - C_i||}{2\sigma_i^2})$$

donde $||-||$ es la distancia Euclideana entre el vector de entrada x y el centro de la i-ésima neurona oculta.

Las RBF definen zonas que pueden entenderse como regiones sensibles que se activarán de mayor forma cuando el patrón a clasificar esté más cerca de su centroide, es por ello que se les conoce como de *carácter local*. Este comportamiento puede ser modelado como:

$$\phi_i(x) \to 1 \text{ cuando} ||x - C_i|| \to 0$$

La salida de una RBFN es una combinación lineal de RBFs, en la que cada una se activa para una porción del espacio de características definidas por los patrones de entrada.

Así, la configuración de una RBFN es como se muestra en la Figura REFERENCIA-FIGURA, donde puede observarse la contribución de las entradas hacia la capa oculta, luego en cada neurona de la capa oculta se evalúa la entrada con la función de activación, para luego obtener una lista de pesos que ponderan los datos de entradas y que son acumulados en la capa de salida para obtener la respuesta de la red.

Las siguientes secciones describen los procesos de entrenamiento y clasificación.

II-A. Entrenamiento de una RBFN

El entrenamiento de una RBFN es realizado de forma híbrida, dividiéndose entre las capas oculta y de salida:

- **Capa oculta**: Se realiza de forma no supervisada tratando de encontrar los centros y radios de las neuronas de la RBFN.
- **Capa de salida**: Se realiza de forma supervisada, tratando de encontrar los pesos sinápticos.

Este entrenamiento híbrido puede realizarse combinando distintas técnicas; al final el objetivo es obtener una buena generalización y a la vez que se obtenga un nivel bajo de error.

Por ejemplo, para esta asignación se ha utilizado el algoritmo *k-means* para realizar la búsqueda de los centros de las neuronas.

Como es sabido, el algoritmo *k-means* permite, a partir de un valor de k grupos, agrupar a todos los datos dependiendo de su cercanía a uno de los k centroides. De forma iterativa, este algoritmo buscará mejorar la distribución de los datos respecto a los centroides, de tal forma que se obtenga la menor distancia entre ellos, finalizando cuando ya no se encuentren movimientos en la agrupación.

Para el cálculo de los radios de estas neuronas se pueden seguir los enfoques de:

- **Media uniforme**: En el que se miden las distancias euclideanas del centroide C_i hacia los p centroides más cercanos, calculándose el radio como

$$\sigma_i = \sqrt{\frac{1}{p}\sum_{i=1}^{p}(C_i - C_p)^2}$$

.

- **Media geométrica**: en el que el radio se calcula a partir de la distancia de un centroide C_i a dos centroides vecinos más cercanos:

$$\sigma_i = \sqrt{||C_i - C_a|| * ||C_i - C_b||}$$

Para el entrenamiento en la capa de salida, el cuál es supervisado , la intención es minimizar las diferencias entre la slaida de la red y las salidas deseadas.

Es posible utilizar el método de la matriz pseudoinversa para obtener una solución directa. La solución es dada por:

$$W = G^+ S = (G^T G)^{-1} G^T S$$

donde G contiene las activaciones de las neuronas de la capa oculta para los patrones de entrada x, S es la amtriz de salidas deseadas y W es la matriz de pesos.

Debido a la dificultad para encontrar el parámetro adecuado que indique la cantidad de neuronas en la capa oculta en una RBFN, es posible utilizar un enfoque para calcular este parámetro, así como para validar que cada una de las configuraciones no esté causando un sobreajuste o sobreentrenamiento en la red. Para ello, se construye un mecanismo como el descrito en el Pseudocódigo 1 que permite, dentro del entrenamiento, crear dos procesos que permitan, por un lado entrenar, y por otro validar que el error obtenido esté dentro de un umbral.

El proceso anterior se repite hasta encontrar un valor de error aceptable o hasta que se prueben todas las configuraciones posibles para luego elegir la de menor error.

Pseudocódigo 1 Metodología de entrenamiento

Entrada: x, k, umbralError
Salida: C_k, σ_k, W_k
1: **repetir**
2: Obtener los centros C_k y radios σ_k a partir de los datos de entrada x para k neuronas.
3: Obtener la salida de la capa oculta, evaluando con la función de transferencia. Por ejemplo, la función Gaussiana:

$$\phi_i(x) = \exp(-\frac{||x - C_i||}{2\sigma_i^2})$$

4: Calcular pesos, por ejemplo utilizando matriz pseudoinversa:

$$W_k = (\phi_k^T \phi_k)^{-1} \phi_k^T S$$

(S es la salida deseada)
5: Evaluar la salida de la red: $Y = \phi_k * W_k$
6: Calcular el $error$ de clasificación.
7: **si** error<umbralError **entonces**
8: **devolver** C_k, σ_k, W_k
9: **fin si**
10: **hasta que** No haya más configuraciones
11: **devolver** C_k, σ_k, W_k

II-B. Clasificación en una RBFN

Como se ha mencionado, la clasificación y el entrenamiento tienen como punto común la función de activación. En el Pseudocódigo 2 se muestra el algoritmo que permitiría realizar la clasificación de datos utilizando los pesos, radios y centroides calculados durante la etapa de entrenamiento. En este algoritmo, lo único que se realiza es la alimentación de la red neuronal con los parámetros mencionados anteriormente y calcular la salida de la red. La salida, un número, es entonces evaluada bajo un umbral para decidir la clase que se le asigna al dato

Pseudocódigo 2 Algoritmo de clasificacion

Entrada: X, Y, W_{ij}, W_{jk}
Salida: Y_p,
1: Salida en capa oculta:
2: $\phi_i(x) = \exp(-\frac{||x - C_i||}{2\sigma_i^2})$
3: Salida en capa de salida:
4: $Y = \phi_k * W_k$
5: $Y_{p_i} = Y > 0.5 ? 1 : 0, \ \forall i = 1, 2, \cdots, N$
6: **devolver** Y_p

III. METODOLOGÍA

La metodología seguida se sujetó por completo a las etapas de entrenamiento y clasificación mostradas en el marco teórico de la Sección II.

Se implementaron dos funciones correspondientes al entrenamiento y la clasificación utilizando *Matlab*. Con la intención de probar distintas alternativas en la configuración de parámetros se seleccionó el conjunto de parámetros mostrados en la Tabla I, realizando la ejecución de la combinación de todos ellos a manera de malla. La prueba de cada configuración de parámetros fue probada 31 veces, debido al componente aleatorio existente al realizar el algoritmo *k-means*.

El código para realizar la prueba de las configuraciones fue preparado para comportarse de forma similar al Pseudocódigo 1, desempeñando entonces las labores de entrenamiento y validación, con un enfoque híbrido mediante *k-means* y el método de la matriz pseudoinversa.

Parámetro	Valores
H (neuronas en capa oculta)	3 a 40
Tipos de media	Uniforme y geométrica

Tabla I: Configuración de parámetros para la ejecución de las RBFN

IV. RESULTADOS

La experimentación fue realizada en un equipo de cómputo con un procesador Intel core i7 de 8 núcleos a 2.00 GHz con 6 GB de memoria en RAM.

La Tabla II muestra un resumen de las mejores ejecuciones para cada uno de los datasets. Es importante hacer notar, que estas mejores configuraciones se refieren a los mejores valores obtenidos en el proceso de entrenamiento.

La serie de Figuras figs. 1 to 4 muestra el espacio particionado obtenido por las mejores configuraciones de parámetros con las que se lanzó la ejecución del entrenamiento y clasificación de las RBFN. Se entiende por *mejor* configuración a aquella que obtuvo el menor error. Es importante mencionar también, que los porcentajes de error reportados en la Tabla II no coinciden exactamente con los mostrados en las figuras ya que éstas últimas utilizan datos distintos.

Dataset	Error mín.	H (cantidad neuronas)
complex	0.0520	39
linear	0	9
ring	0	10
xor	0	4

Tabla II: Mejores RBFN obtenidas

V. Discusión de resultados

Las RBFN implementadas obtienen valores de error muy bajos para los datasets *linear*, *ring* y *xor*. Para el caso del dataset *complex*, los resultados de error aumentan de forma considerable, tal como en el caso de las RNA MLP implementadas en la asignación anterior.

Un aspecto importante a destacar es la rapidez con la que la combinación de métodos entregan los resultados. Fueron implementadas 31 ejecuciones para cada una de las configuraciones de parámetros. Si bien es cierto que son menos que en la implementación de las RNA MLP de la asignación anterior, es evidente que la rapidez de las RBFN les posicionan como un mejor candidato (de acuerdo a los resultados de los datasets utilizados en esta práctica).

Adicionalmente, en el aspecto técnico este tipo de redes resultan más fáciles de implementar, considerando las facilidades provistas por *Matlab*.

VI. Conclusiones

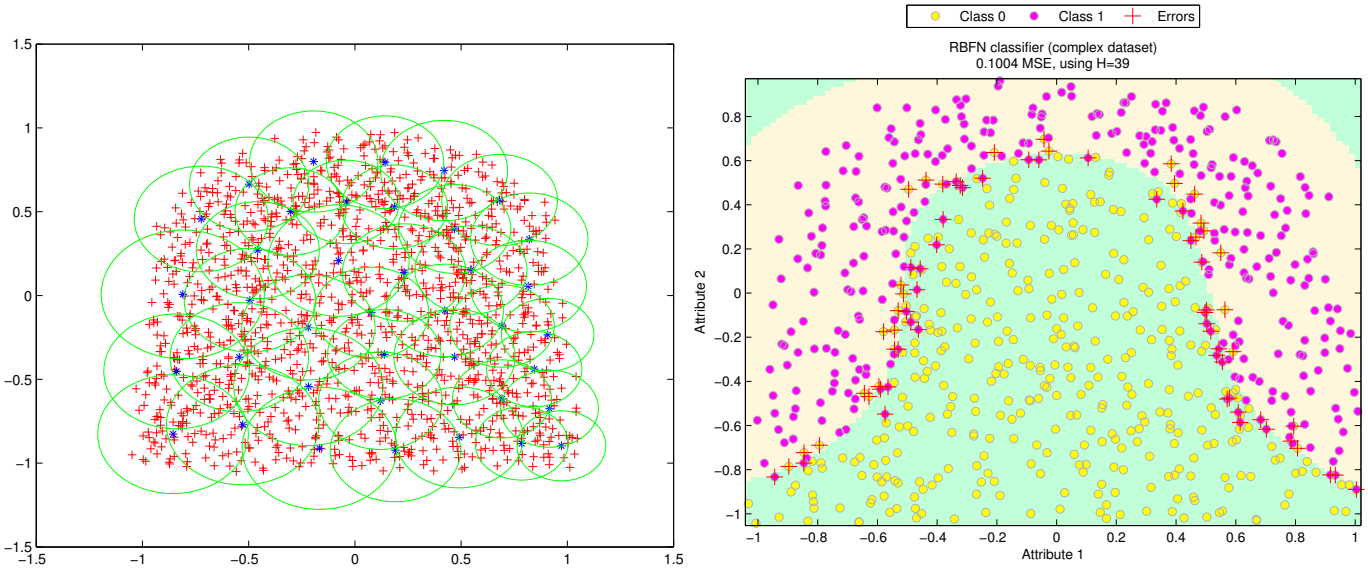
Las RBFN se presentan como una alternativa al perceptrón multicapa para labores de aproximación de funciones y clasificación de patrones. Este tipo de redes definen una estructra fija de capas de entrada, oculta y de salida, siendo activadas las neuronas por una función no lineal, como por ejemplo la función Gaussiana.

Debido a su entrenamiento híbrido, algunas de las variantes de entrenamiento de las RBFN no requieren una iteración excesiva ni la retropropagación del error, por lo que su ejecución puede resultar más rápida.

Este documento ha descrito la implementación y resultados de la ejecución de RBFN con variaciones en la cantidad de neuronas y cálculos de radios, utilizando un entrenamiento híbrido con *k-means* en la etapa no supervisada y el método de la matriz pseudoinversa en la etapa supervisada sobre cuatro datasets. Los resultados indican que este tipo de redes pueden desempeñarse de forma más rápida y obtener mejores resultados (para los datasets considerados) en la clasificación que las redes de tipo MLP.

Referencias

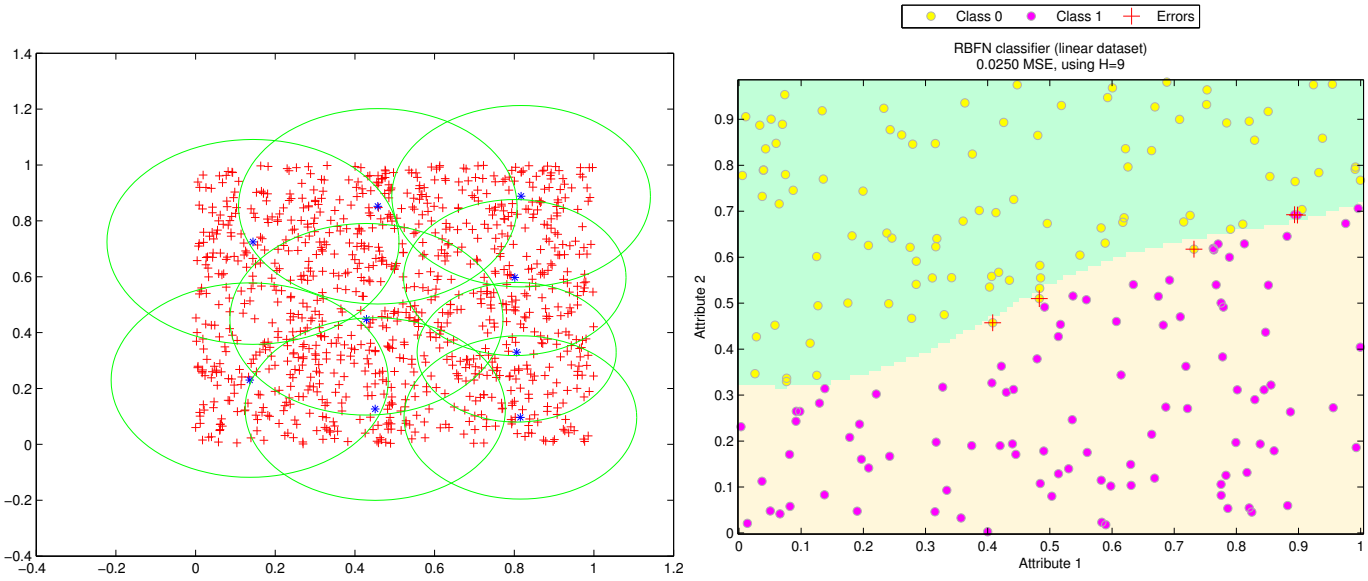
[1] R. Duda, P. Hart, and D. Stork, *Pattern Classification*, 2002.
[2] W. Gomez Flores, "Diapositivas de clase Redes neuronales artificiales: Red neuronal de función de base radial," 2015.
[3] ——, "Diapositivas de clase Redes neuronales artificiales: Perceptrón multicapa," 2015.



(a) Vista de los radios y centroides obtenidos

(b) Espacio particionado de la mejor RBFN

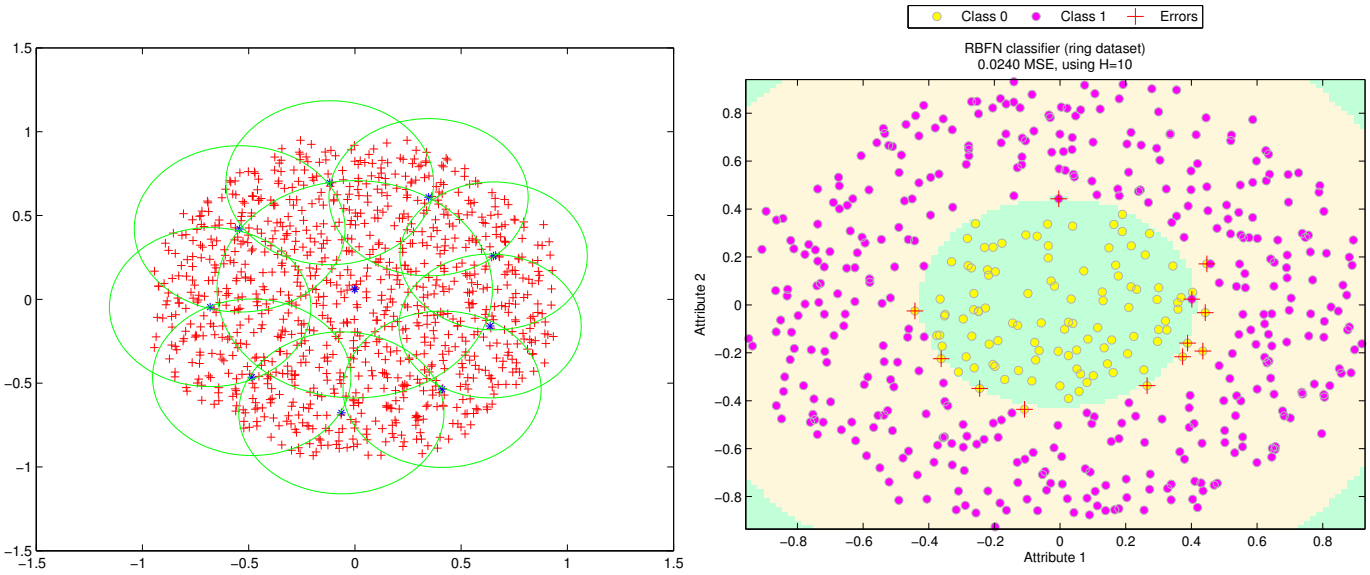
Figura 1: Resultados obtenidos para el dataset *complex*



(a) Vista de los radios y centroides obtenidos

(b) Espacio particionado de la mejor RBFN

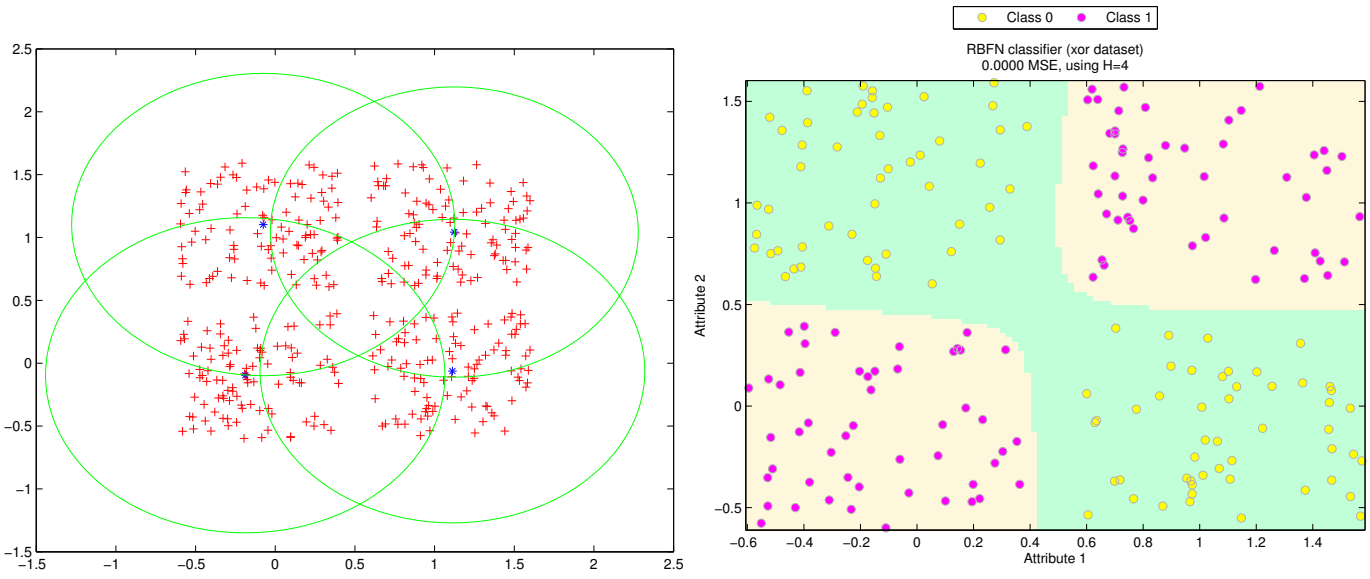
Figura 2: Resultados obtenidos para el dataset *linear*



(a) Vista de los radios y centroides obtenidos

(b) Espacio particionado de la mejor RBFN

Figura 3: Resultados obtenidos para el dataset *ring*



(a) Vista de los radios y centroides obtenidos

(b) Espacio particionado de la mejor RBFN

Figura 4: Resultados obtenidos para el dataset *xor*