Aproximación de funciones con EPWavenets

Alejandro Romero Herrera, Oscar Herrera Alcántara, Victor Manuel Landassuri Moreno

Universidad Autónoma del Estado de México,
Estado de México, México

aromeroh688@alumno.uaemex.mx
{oherreraa,vmlandassurim}@uaemex.mx

Resumen. En este trabajo presentamos las redes neuronales *EPWavenets* como una arquitectura de red neuronal de funciones de base radial modificada para incluir funciones wavelets que se obtienen con parametrizaciones de filtros de reconstrucción perfecta. Los parámetros son optimizados con un algoritmo evolutivo a fin de mejorar la aproximación de funciones $f(x)$. Los experimentos comparan el desempeño de las *EPWavenets* con redes neuronales que usan la función gaussiana y la función sigmoidal. Se concluye que las *EPWavenets* logran un alto grado de adaptabilidad y un desempeño globalmente competitivo respecto a las demás redes neuronales.

Palabras clave: Wavelets, redes neuronales, funciones de base radial, algoritmos evolutivos.

1. Introducción

Una de las motivaciones que dieron origen a las redes neuronales artificiales (RNA) fue la inspiración en el modelo conexionista del cerebro humano [11]. Posteriormente, ante los cuestionamientos del cómo y por qué funcionaban las redes neuronales y en vista a hacer un mejor uso de ellas, se estableció la conexión con otras áreas del conocimiento, en particular con el Análisis Funcional que es una rama del Análisis Matemático orientada a estudiar las propiedades de funciones que generan espacios vectoriales. El Teorema de Aproximación Universal (TAU), por ejemplo, permite abordar desde esta perspectiva la capacidad de aproximar funciones a partir de funciones no constantes, acotadas, monótonas crecientes y continuas [1]. El TAU guarda una estrecha relación con las redes neuronales de perceptrones.

Otro tipo de redes neuronales son las de funciones de base radial (RNFBR) cuya expresión como aproximadores de funciones está dada por:

$$f(x) \approx \sum_{i=1}^{N} w_i \varphi(||x - x_i||) + w_0 \tag{1}$$

en donde $x \in R^n$ es el conjunto de datos de entrada, N es el número de unidades de procesamiento o neuronas, $x_i \in R^n$ son los centroides de las funciones de base

radial, w_i son los pesos sinápticos, w_0 es el umbral de desplazamiento, $||x - x_i||$ es la métrica de distancia, y $\varphi(\cdot)$ es la función de base radial.

Se han propuesto varias funciones de base radial entre las que se encuentran las funciones: gaussiana, multicuadrática, multicuadrática inversa, y *splines* [4].

Otras funciones que también se han usado como funciones base son las *wavelets*, que dan lugar a las redes neuronales *wavenets* [14][3].

Para aproximar una función $f(x)$ mediante funciones wavelets $\psi(x)$ se puede usar la expresión:

$$f(x) \approx \sum_{i=1}^{N} w_i \psi_{a_i,b_i}(x) + w_0 = \sum_{i=1}^{N} w_i \psi(\frac{x - b_i}{a_i}) + w_0 \tag{2}$$

en donde los $a_i \neq 0$ son parámetros de escalamiento, y los $b_i \in R$ son parámetros de desplazamiento de la función wavelet principal $\psi(x)$.

El problema de entrenamiento de la RNFBR descrita en la ec. (1) radica en determinar el conjunto óptimo de parámetros libres que incluyen: el conjunto de valores a_i, b_i, los pesos sinápticos w_0 y w_i con $i = 1, 2, \ldots, N$, donde N es el número de neuronas.

En éste artículo presentamos las redes neuronales *EPWavenets* que usan bases wavelets adaptables obtenidas a partir de filtros de reconstrucción perfecta. En las *EPWavenets* el ajuste de parámetros libres se realiza aplicando un algoritmo evolutivo que minimiza el error al aproximar una función objetivo $f(x)$.

El resto del artículo está organizado de la siguiente manera: En la Sección 2 revisamos conceptos de wavelets, en la Sección 3 se comenta cómo generar múltiples funciones wavelets a partir de filtros paramétricos de reconstrucción perfecta. En la Sección 4 comentamos la relación entre wavelets y redes neuronales. En la Sección 5 describimos las *EPWavenets* y cómo aplicar un algoritmo genético para optimizar sus parámetros libres para aproximar funciones. En la Sección 6 presentamos resultados experimentales y, finalmente, en la Sección 7 presentamos las conclusiones obtenidas además de algunas ideas para continuar en trabajos futuros.

2. Wavelets

Las wavelets son familias de funciones $\psi_{a,b}(x) = a^{\frac{1}{2}} \psi(\frac{x-b}{a})$ con $a \neq 0$, que surgen de una función principal $\psi(x)$ que cumple las condiciones $\int_{-\infty}^{\infty} \psi(x)dx = 0$ y $\int_{-\infty}^{\infty} |\psi(x)|^2 dx = 1$. Las funciones wavelets además cumplen la condición de admisibilidad dada por $\int_{-\infty}^{\infty} \frac{|\hat{\psi}(w)|^2}{|w|} dw < \infty$ lo cual les da la propiedad de localización, que significa que su transformada de Fourier $\hat{\psi}(w)$ se desvanece rápidamente. También se pueden generar wavelets con soporte compacto cuyo valor es cero fuera del intervalo de soporte en el dominio temporal. Una forma de generar funciones wavelets es a través del análisis multiresolución (AMR) [8] en donde se hace uso de funciones de escalamiento $\phi(x)$ ortogonales a las funciones

wavelets $\psi(x)$ y que en conjunto conforman una base del espacio vectorial $L^2(R)$, que es el espacio de todas las funciones f para las cuales $\int_{-\infty}^{\infty} |f(x)|^2 dx < \infty$.

3. Filtros ortogonales de reconstrucción perfecta

De acuerdo con el AMR es posible generar wavelets ortogonales con soporte compacto a través de la parametrización de los coeficientes de un banco de filtros ortogonales h y g de reconstrucción perfecta (FORP). Los FORP constan de un filtro pasabajas h y un filtro pasaaltas g. Un filtro $h = [h_0, h_1, h_2, h_3]$ de longitud $L = 4$, por ejemplo, genera wavelets con soporte compacto en el intervalo $[0, 1]$, y sus coeficientes dependen de un parámetro α de la siguiente manera [10]:

$$\begin{array}{ll} h_0 = \frac{1}{4} + \frac{1}{2\sqrt{2}} cos\alpha & h_1 = \frac{1}{4} + \frac{1}{2\sqrt{2}} sin\alpha \\ h_2 = \frac{1}{4} - \frac{1}{2\sqrt{2}} cos\alpha & h_3 = \frac{1}{4} - \frac{1}{2\sqrt{2}} sin\alpha \end{array} \tag{3}$$

en donde $\alpha \in [0, 2\pi)$. Los correspondientes coeficientes del filtro pasaaltas se calculan a partir de los coeficientes h_i mediante la ecuación $g_i = (-1)^i h_{N-i-1}$. En el caso de filtros de longitud $L = 6$ los coeficientes dependen de dos parámetros α y β de la siguiente manera [10]:

$$\begin{array}{lll} h_0 = \frac{1}{8} + \frac{1}{4\sqrt{2}} cos\alpha + \frac{p}{2} cos\beta \, , & h_1 = \frac{1}{8} + \frac{1}{4\sqrt{2}} sin\alpha + \frac{p}{2} sin\beta \\ h_2 = \frac{1}{4} - \frac{1}{2\sqrt{2}} cos\alpha \quad , & h_3 = \frac{1}{4} - \frac{1}{2\sqrt{2}} sin\alpha \\ h_4 = \frac{1}{8} + \frac{1}{4\sqrt{2}} cos\alpha - \frac{p}{2} cos\beta \, , & h_5 = \frac{1}{8} + \frac{1}{4\sqrt{2}} sin\alpha - \frac{p}{2} sin\beta \end{array} \tag{4}$$

en donde $p = \frac{1}{2}\sqrt{1 + sin(\alpha + \frac{\pi}{4})}$, $\alpha, \beta \in [0, 2\pi)$.

Se puede demostrar que los FORP deben tener una longitud L par, y que el soporte compacto del wavelet es $[0, \frac{L}{2} - 1]$.

Cabe mencionar que existen pocas funciones wavelet con soporte compacto que tengan una expresión analítica. Uno de los casos en los que sí se tiene la expresión analítica es para la función wavelet Haar descrita como:

$$f(x) = \begin{cases} 1 & \text{si} \quad x \in [0, \frac{1}{2}) \\ -1 & \text{si} \quad x \in [\frac{1}{2}, 1) \\ 0 & \text{caso contrario} \end{cases} \tag{5}$$

En otros trabajos con *wavenets*, a diferencia de las *EPWavenets*, se usan wavelets sin soporte compacto o con una $\psi(x)$ fija. En [9] se usa el wavelet $\psi(x) = -e^{-\frac{x^2}{2}} cos(5x)$ con parámetros de escalamiento y dilatación fijos, y en [14] y [13] se usa la "Derivada-Gaussiana" $\psi(x) = -xe^{-\frac{1}{2}x^2}$.

En el caso de la *EPWavenets* hemos optado por aproximar las funciones wavelets en forma discreta usando el algoritmo en cascada [7] y filtros ortogonales paramétricos, lo cual nos da una amplia gama de posibilidades. A manera de ejemplo considérese la aproximación del wavelet Haar con las ecuaciones dadas en (3) y $\alpha = \frac{\pi}{4}$. La Figura 1 muestra el wavelet Haar aproximado con $A = 16$ valores discretos, en donde el soporte es $[0, 1]$.

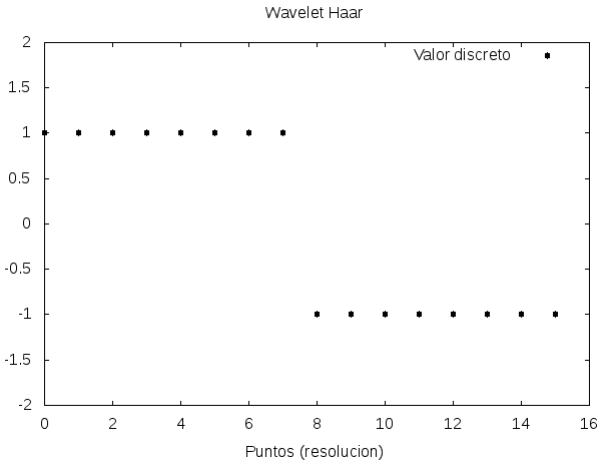


Fig. 1: Aproximación del wavelet de Haar con $\alpha = \frac{\pi}{4}$ y una resolución de $A = 16$ puntos

4. Redes neuronales de funciones de base radial

La arquitectura de las RNFBR define tres capas: la capa de entrada, la capa oculta, y la capa de salida. La capa oculta aplica una transformación no lineal a los datos provenientes de la capa de entrada para llevarlos a otro espacio vectorial que suele tener una mayor dimensión. Al pasar de un espacio a otro de mayor dimensión se explota de mejor manera la capacidad de las redes neuronales en tareas de clasificación y de aproximación de funciones [4]. Esta última tarea es de interés en el presente artículo.

Una RNFBR como aproximador de funciones descompone cada uno de los datos de entrada en varias componentes que pasan por N neuronas, mismas que se suman en la capa de salida. La Figura 2 ilustra la arquitectura de una RNFBR acorde con la ecuación (1), en donde cada muestra de entrada corresponde a un vector $X \in R^n$. Cada neurona incluye una función de base radial (FBR), entre las que se encuentran: la función gaussiana, la función cuadrática, la función cuadrática inversa, la función multicuadrática, y las funciones *splines*. La capa de salida realiza una combinación lineal de las activaciones de las neuronas de la capa oculta considerando los pesos sinápticos w_i. El peso sináptico w_0 corresponde al valor de umbral ($BIAS$=1) que modifica el valor de la salida Y.

De manera análoga al funcionamiento de las RNFBR, las *EPWavenets* utilizan las funciones wavelets como una opción a las FBR. Las funciones wavelets tienen propiedades que no tienen las FBR. Por ejemplo, el soporte compacto es una propiedad relevante toda vez que permite capturar información local de una función, y no interferir con la aproximación global. Las FBR sí permiten identificar información local, sin embargo el número de opciones se restringe a

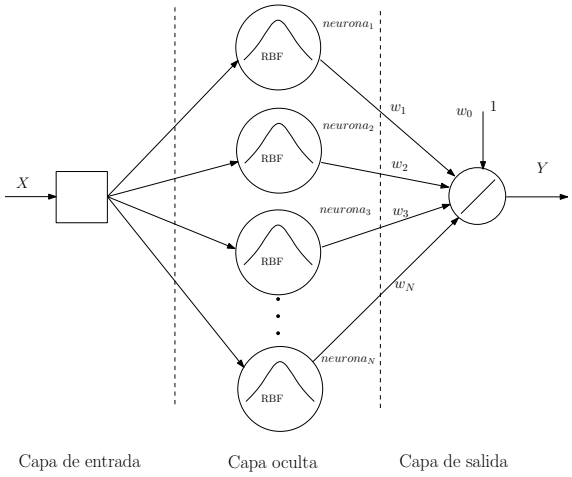


Fig. 2: Una red neuronal de función de base radial

algunas de ellas como son: la función gaussiana, la función cuadrática, la función cuadrática inversa, y los *splines*. Otra propiedad relevante de las wavelets obtenidas con FORP's es la posibilidad de controlar los momentos de desvanecimiento definidos como:

$$\int x^k \psi(x) dx = 0, \text{ para } k = 1, 2, \ldots, p \tag{6}$$

y que cuanto mayor sea el valor de p los wavelets son "más suaves", lo que facilita que funcionen apropiadamente en la aproximación de funciones que tienen varias derivadas. Los wavelets que maximizan los momentos de desvanecimiento para un soporte compacto dado corresponden a las funciones propuestas por Daubechies [2], y que están incluidos dentro de las parametrizaciones de los FORP (ver [5]).

Nótese ciertamente que el uso de wavelets junto con redes neuronales es un tema ya abordado [14,9,13], sin embargo la principal aportación de las *EPWavenets* incluye el uso de wavelets paramétricos optimizados evolutivamente para adaptarlas a un contexto particular y a su vez mantener la adaptabilidad para aproximar diferentes funciones.

5. EPWavenets

La idea fundamental de las redes neuronales *EPWavenets* consiste en incluir funciones wavelets generadas con filtros paramétricos (ver ecuaciones 3 y 4), usar la ecuación de aproximación (2), y aplicar algoritmos evolutivos para determinar el conjunto óptimo de parámetros libres. El uso de wavelets y redes neuronales ya ha sido abordado en otros trabajos. En [9] por ejemplo, se optimizan los pesos w_i y se dejan fijos los parámetros de traslación y escalamiento, lo cual restringe

demasiado la capacidad de aproximación con wavelets. En [14] se propone usar un método de gradiente para optimizar los parámetros w_i junto con los parámetros a_i y b_i. Sin embargo, el problema inherente al uso de métodos de gradiente es que se requiere conocer la función analítica de $\psi(x)$ misma que además debe ser derivable, algo que ciertamente restringe la variedad de funciones wavelets, y conlleva a usar un mayor número de neuronas cuando la función a aproximar no es suave.

También es conocido que determinar el conjunto de parámetros optimos de los filtros paramétricos para aproximar una función es un problema no convexo [12]. Entonces, dada la complejidad de calcular los parámetros óptimos de los filtros y de los parámetros a_i, b_i, w_i de la red neuronal, se propone el uso de algoritmos evolutivos, por lo que podemos decir que las *EPWavenets* son la combinación de algoritmos evolutivos, wavelets y redes neuronales:

$$EPWavenets = Algoritmos\ Evolutivos + Wavelets + Redes\ Neuronales$$

La Figura 3 ilustra una red neuronal *EPWavenet* con los diferentes parámetros involucrados. La función que minimiza el algoritmo evolutivo es el error de

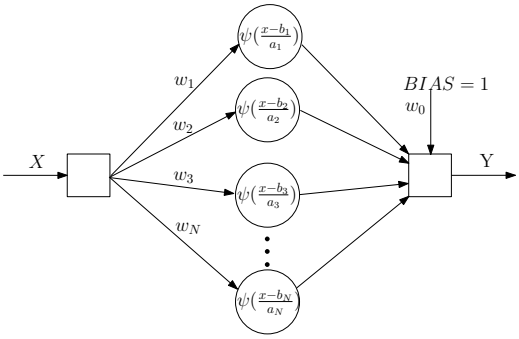


Fig. 3: Red neuronal *EPWavenet*

aproximación entre la salida Y de la *EPWavenet* y los valores de la función objetivo $f(x)$ obtenidos con la entrada X. Tanto X como Y están en la misma dimensión n, así que el error de aproximación (cuadrático medio) puede ser calculado así:

$$Error = \frac{1}{M} \sum_{m=1}^{m=M} \sum_{i=1}^{n} (x_i(m) - y_i(m))^2 \tag{7}$$

en donde M es el número de muestras de los datos de entrada, $x_i(m)$ es la i-ésima componente del dato de entrada en el instante m, y $y_i(m)$ es la i-ésima componente de la salida de la *EPWavenet* en el instante m. La aproximación con wavelets puede ser suficientemente pequeña de acuerdo con el número de funciones involucradas, y con los valores apropiados de los parámetros de escalamiento

y dilatación. Matemáticamente se expresa así:

$$|(\sum_{i=1}^{N} w_i \psi(\frac{x - b_i}{a_i}) + w_0) - f(x)| < \epsilon \tag{8}$$

para $\epsilon > 0$. En particular para un filtro de longitud 4 se tiene un parámetro α, y con N neuronas se generan $3N + 1$ parámetros adicionales que corresponden a los valores de a_i, b_i, w_i, con $i = 1, 2, \ldots, N$. Si se usa un algoritmo genético, la codificación de un individuo podría ser así:

$$\alpha|a_1 a_2 ... a_N|b_1 b_2 ... b_N|w_0 w_1 w_2 ... w_N$$

Para un filtro de longitud 6 se tienen dos parámetros α y β, y la codificación en un individuo podría ser así:

$$\alpha\beta|a_1 a_2 ... a_N|b_1 b_2 ... b_N|w_0 w_1 w_2 ... w_N$$

En la siguiente sección ponemos en práctica los conceptos explicados en las secciones anteriores.

6. Experimentos

Con los siguientes experimentos se persigue:

- Explorar la capacidad de aproximación de funciones wavelets generadas con FORP.
- Comparar el desempeño entre las wavelets paramétricas en una arquitectura de *EPWavenets* y funciones gaussianas en una arquitecura de RNFBR al aproximar y predecir funciones.

Aproximación de funciones wavelets paramétricas. En los experimentos se realiza la aproximación de funciones wavelets a partir de parameterizaciones de filtros de longitud $L = 4$ y $L = 6$. Con A muestras de aproximación para la función wavelet $\psi(x)$ con un intervalo para el soporte compacto $[0, \frac{L}{2} - 1]$, y para valores de x dentro del soporte compacto, la mejor aproximación para $\psi(x_r)$ está dada por la $r-$ésima muestra discreta, donde $r = \lfloor x_r * A/(N - 1) \rfloor$. Obviamente, para valores fuera del soporte $\psi(x) = 0$. Nótese que el valor aproximado del wavelet es devuelto como si $\psi(x)$ fuese una función analítica continua.

Configuración del AG. Existen varios algoritmos evolutivos que pueden usarse con las *EPWavenets*. En los experimentos realizados se optó por usar un algoritmo genético no tradicional [5] caracterizado por:

- Una población con P individuos, en donde P es par, y una poblacion temporal de tamaño $2P$.

- Cruzamiento en parejas entre el i-ésimo y el $(P-1-i)$-ésimo individuos, en donde $i \in [0, \frac{P}{2} - 1]$. Los individuos se ordenan según la medida de aptitud.
- Selección determinista, en donde los P individuos más aptos de la poblacion temporal pasan a la siguiente generación.
- Cruzamiento anular en un solo punto.

Para codificar los parámetros α y β en cada uno de los individuos se utilizó una codificación en binario pesado, con $q = 29$ bits de precisión por parámetro. Para generar valores $V_{\alpha\beta}$ para α y β en un rango de $[0, 2\pi)$ dado el valor en binario pesado $B_{\alpha\beta}$ se usó la ecuación (9):

$$V_{\alpha\beta} = 2\pi \frac{B_{\alpha\beta}}{2^q - 1} \tag{9}$$

Para codificar los valores de w, a y b se usaron $q = 28$ bits de precisión, y un bit de signo. A partir de su valor en binario pesado con signo B_{wab} se obtuvieron valores V_{wab} en el intervalo $(-8192, 8192)$ mediante la ecuación (10):

$$V_{wab} = 8192 \frac{B_{wab}}{2^q - 1} \tag{10}$$

El número de generaciones del algoritmo genético fue $G = 2000$, una población de $P = 50$ individuos, una probabilidad de cruzamiento $P_c = 0.97$ y una probabilidad de mutación $P_m = 0.02$. El tiempo promedio de ejecución del algoritmo genético con los parámetros anteriores para un conjunto de datos de entrada de $M = 1000$ es de 11 minutos, para $M = 5000$ es de 13 minutos y para $M = 9000$ es de 16 minutos, corriendo sobre servidores Sun Fire $X2270$ con procesadores Xeon de 16 núcleos y 32 GB de RAM con GNU/Linux Debian. En los experimentos se consideraron 13 funciones $f(x)$ cuyos valores se normalizaron (valores en el intervalo $[0, 1]$). La Tabla 1 resume la información de cada función $f(x)$, en donde en la primera columna se muestra el nombre de la función, en la segunda se muestra un idenficador $f\#$, y en la tercera columna se indica el número de muestras o puntos para la función $f(x)$ discretizada. El origen de nuestras funciones es el repositorio del software Visual Recurrence Analysis (VRA) [6]. Con las funciones de la Tabla 1 se realizaron los siguientes experimentos:

Tabla 1: Funciones $f(x)$ del repositorio y sus características

Nombre	Brownian Motion	Dow Jones	ECG	Henon	Ikeda	Laser	Logistic	Lorenz	Rossler	Seno	Seno con Ruido	Sun Spots	White Noise
f#	f1	f2	f3	f4	f5	f6	f7	f8	f9	f10	f11	f12	f13
Número de muestras (M)	1000	1000	9000	5000	5000	5000	1000	3500	5000	1000	1000	280	1000

6.1. Experimento 1

Determinar la mejor *EPWavenet* para una función $f\#$, con datos de entrada en una dimensión ($n = 1$), variando el número de neuronas $N = 1, 2, \ldots, 10$, y el tipo de filtro (wavelet) con $L = 4$ para un filtro de longitud 4, y $L = 6$ para un filtro de longitud 6, ver ecuaciones (3) y (4). El criterio para elegir la mejor arquitectura fue minimizar el error cuadrático medio (ver ec. (7)), sobre las M muestras. Los resultados se muestran en la Tabla 2 en donde en la primera columna se indica el valor de L y N, y en las demás columnas se indica el tipo de función aproximada idenfificada por $f\#$. La mejor arquitectura por columna se ha remarcado en negritas para cada caso. El número promedio de neuronas

Tabla 2: Arquitecturas de *EPWavenets* para la aproximación de funciones: $f1$. Brownian Motion, $f2$. Dow Jones, $f3$. ECG, $f4$. Henon, $f5$. Ikeda, $f6$. Laser, $f7$. Logistic, $f8$. Lorenz, $f9$. Rossler, $f10$. Seno, $f11$. Seno con Ruido, $f12$.Sun Spots, $f13$.White Noise

Func	$f1$	$f2$	$f3$	$f4$	$f5$	$f6$	$f7$	$f8$	$f9$	$f10$	$f11$	$f12$	$f13$
L, N	NC	NC	NC	NC	NC	NC	NC	NC	NC	NC	NC	NC	NC
4, 1	$1.84E-6$	0.00434	0.09305	0.19952	0.17513	**2.91E-9**	$1.21E-5$	0.24306	**1.18E-5**	$4.62E-6$	**1.28E-16**	$7.23E-4$	0.00398
4, 2	0.00345	0.00903	0.07690	0.21114	0.04699	0.03520	0.00183	0.07842	0.02476	0.00285	0.00229	$3.98E-5$	$1.99E-4$
4, 3	0.00703	0.00869	0.02102	0.00541	0.06779	0.01058	$3.12E-4$	0.01952	0.09505	0.00107	$1.22E-4$	$9.59E-7$	$1.47E-7$
4, 4	$4.06E-4$	0.00594	0.02525	0.02009	0.01489	0.03456	$1.65E-4$	0.01685	0.01646	0.00204	$5.65E-4$	$1.56E-5$	$1.17E-4$
4, 5	$2.87E-4$	$5.30E-4$	0.02084	0.06891	0.00753	0.00669	$2.39E-4$	0.01551	0.08655	$3.58E-5$	$4.49E-4$	$9.48E-5$	$3.63E-4$
4, 6	$1.27E-5$	$8.12E-4$	0.02941	0.04353	0.01084	0.00444	$4.26E-5$	0.01015	0.00648	$2.47E-4$	$1.53E-4$	$7.31E-7$	$5.97E-6$
4, 7	$2.28E-5$	0.00152	0.03750	0.07567	0.01000	0.01069	**2.20E-10**	0.01065	0.01655	$1.13E-5$	$3.10E-9$	$1.16E-5$	$1.17E-4$
4, 8	**1.68E-8**	0.00649	0.01166	$8.23E-5$	**1.63E-4**	0.00436	$2.97E-5$	**8.87E-6**	0.02839	**1.79E-8**	$3.64E-4$	$1.27E-5$	**5.31E-8**
4, 9	$1.05E-5$	0.00315	0.01743	0.05384	0.05532	0.00418	$6.67E-5$	0.04234	$4.72E-5$	$2.25E-5$	$8.31E-5$	$9.05E-7$	$2.36E-5$
4, 10	$3.50E-5$	0.00133	0.03197	0.08349	0.01160	0.01280	$4.18E-4$	0.00818	0.01212	$6.54E-4$	$1.08E-5$	$9.22E-6$	$3.08E-5$
6, 1	0.00519	0.00587	0.08905	0.12375	0.19189	0.02997	$9.47E-4$	0.10162	0.06970	$9.76E-4$	$2.11E-5$	$3.21E-4$	0.01368
6, 2	0.02138	0.00246	0.02127	0.11266	0.00378	0.02125	$2.22E-4$	0.00426	0.03568	0.01482	0.00128	$1.31E-4$	0.00266
6, 3	0.00258	0.01542	0.03566	0.07052	0.00833	0.02580	$2.16E-5$	0.03497	0.05618	$5.82E-5$	0.00743	0.00109	$2.02E-4$
6, 4	$2.69E-5$	0.00432	0.03510	0.08667	0.01482	0.00532	$1.09E-4$	0.01528	0.04018	$5.06E-5$	$1.99E-4$	$4.19E-4$	$1.78E-4$
6, 5	$5.94E-5$	$6.13E-4$	0.01794	0.02906	0.04455	$7.26E-4$	$4.59E-5$	$6.33E-4$	$4.11E-4$	$2.84E-5$	$1.39E-4$	$2.10E-4$	$5.51E-4$
6, 6	$2.91E-4$	**1.53E-4**	0.02701	**3.17E-5**	0.00511	0.00200	$2.47E-4$	0.01911	0.00886	0.00622	$3.80E-5$	0.00100	$2.23E-4$
6, 7	0.01714	0.01390	**1.57E-4**	0.03620	$3.12E-4$	0.00145	0.00552	0.03061	$7.14E-4$	0.01605	0.00945	$2.96E-4$	$1.62E-5$
6, 8	$4.34E-5$	0.00172	0.06659	0.00166	0.01947	0.00430	0.00331	$7.84E-4$	0.00771	0.00101	$1.34E-4$	**4.42E-7**	$3.69E-4$
6, 9	$5.77E-5$	0.03970	0.01903	$2.92E-4$	0.04756	0.03121	$2.07E-6$	$3.68E-5$	0.00212	$3.27E-4$	$1.20E-4$	$1.90E-6$	$6.36E-6$
6, 10	$2.06E-4$	0.00434	0.06997	0.00445	0.05882	0.00138	0.00303	$3.41E-4$	0.00229	0.00157	$1.95E-4$	$5.31E-6$	$9.08E-4$

requeridas en las *EPWavenets* es de $\overline{N} = 5.85$ con una desviación estándar de $\sigma_N = 2.85$. El error promedio de aproximación para los 13 casos fue $4.05E-5$ con una desviación estándar de $\sigma = 6.74E-5$.

Nótese que en el 69 % de los casos la mejor aproximación se obtuvo con filtros de longitud $L = 4$ y en el restante 31 % se obtuvo la mejor aproximación con filtros de longitud $L = 6$.

Esto se atribuye a la dificultad inherente de optimizar un parámetro β adicional, de hecho se puede demostrar que la parametrización con $L = 4$ es un subcaso de la parametrización de $L = 6$.

6.2. Experimento 2

Se comparó el resultado de la aproximación entre la mejor arquitectura *EPWavenet* del Experimento 1 y la aproximación de una RNFBR con función gaussiana (RNFBRG). Esto significa haber usado el mismo número de neuronas para cada $f\#$ pero diferente función base. Los resultados se muestran en la Tabla 3 en donde en la primera columna se muestra el tipo de red neuronal (*EPWavenet* o RNFBR) y en las columnas restantes se muestra el error de aproximación de cada una de las funciones $f1$ a $f13$. En la Tabla 3 se puede observar que en todos

Tabla 3: Comparación de *EPWavenets* y RNFBR con función gaussiana

Red neuronal	$f1$	$f2$	$f3$	$f4$	$f5$	$f6$	$f7$	$f8$	$f9$	$f10$	$f11$	$f12$	$f13$
EPWavenet	**1.68E-8**	**1.53E-4**	**1.57E-4**	**3.17E-5**	**1.63E-4**	**2.91E-9**	**2.20E-10**	**8.87E-6**	**1.18E-5**	**1.79E-8**	**1.28E-16**	**4.42E-7**	**5.31E-8**
RNFBRG	$2.86E-5$	$7.86E-4$	$5.92E-4$	$1.96E-4$	0.04887	$9.01E-5$	0.01233	$2.17E-4$	$1.50E-4$	$1.85E-4$	0.06237	$8.45E-4$	$1.80E-6$

los casos la *EPWavenet* mejora la aproximación de la RNFBRG. Se hace notar que el error promedio de la aproximación con *EPWavenets* es $4.04E{-}005$ con una desviación estándar de $6.74E{-}5$, en tanto que el error promedio de aproximación de la RNFBRG es $9.74E-3$ con una desviación estándar de $2.08E-2$.

6.3. Experimento 3

Se generalizó la ecuación (7) al introducir un desplazamiento $k >= 0$ entre el i-ésimo dato de entrada en el momento m, y el i-ésimo dato de salida en el momento $m + k$ que matemáticamente se expresa así:

$$Error(k) = \frac{1}{M} \sum_{m=1}^{m=M} \sum_{i=1}^{n} (x_i(m) - y_i(m+k))^2 \tag{11}$$

Nótese que la ecuación (11) permite realizar la predicción de la función al establecer la dependencia $y(m + k) = f(x(m))$.

Se compararon las predicciones para $k = 1, \ldots, 100$[1] para la mejor arquitectura *EPWavenet* (ver Experimento 1) y las RNFBRG con el mismo número de neuronas, con las funciones $f1$ a $f13$. Adicionalmente se consideró incluir una función sigmoidal (que no es de base radial) para explorar su comportamiento como predictor usando también una capa oculta.

[1] A excepción del caso de $f12$ se usaron el 10 % de las muestras, esto es $k = 1, 2, \ldots, 28$.

Los resultados se muestran en la Figuras 4 y 5 mismas que en cada subfigura comparan a las *EPWavenets* con RNRBFG y redes neuronales con funciones sigmoidales.

En las Figuras 4 y 5 se puede apreciar que para $f1$, $f2$, y $f8$ (Brownian Motion, Dow Jones y Lorenz) las *EPWavenets* tienen un mejor desempeño que el resto de las redes neuronales. También se observa que al incrementarse el valor de k el error de aproximación se incrementa paulatinamente.

Algo similar ocurre con la función $f9$ (Rossler) en donde el error se incrementa conforme el valor de k, sin embargo la aproximación con *EPWavenets* se ve superada por el resto de las redes neuronales.

Los peores resultados con *EPWavenets* se dan con la función $f3$ (EGC) que si bien es cierto ofrece la mejor aproximación con $k = 0$ falla con $k > 0$. En tanto que la RNFBRG y la red con función sigmoidal siguen un comportamiento suave e incremental.

Para los casos $f4$, $f5$, $f6$, $f7$ y $f13$ (Henon, Ikeda, Laser, Logistic, y White Noise) el comportamiento es prácticamente el mismo para cualquier $k > 1$, es decir, sólo se puede aproximar para $k = 0$, y para $k > 0$ el error de aproximación se dispara de inmediato con cualquier tipo de red neuronal. De hecho la peor predicción se obtiene con la función sigmoidal que presenta saltos abruptos para diferentes valores de k, esto se atribuye a que no puede captar singularidades en las funciones objetivo y toda vez que este tipo de funciones siguen un comportamiento caótico.

Para los casos $f10$, $f11$ y $f12$ (Seno, Seno con Ruido y Sun Spots) se observa que todas las redes neuronales se comportan de forma similar y que el valor de predicción fluctua dependiendo de la periodicidad de las funciones. La peor predicción se logra con funciones sigmoidales, en tanto que la *EPWavenet* tiene un desempeño competitivo respecto a la RNFBRG.

7. Conclusiones

Se presentaron las *EPWavenets* que combinan técnicas de cómputo evolutivo, redes neuronales y wavelets paramétricos.

Los experimentos permiten afirmar que es posible aproximar funciones con *EPWavenets* inspirados en la arquitectura de redes neuronales de funciones de base radial con una sola capa oculta, y usar wavelets paramétricos sin requerir de su expresión analítica.

Las *EPWavenets* logran un alto grado de adaptabilidad y un desempeño competitivo respecto a otras redes neuronales que involucran funciones de base radial. Los experimentos nos permitieron comparar el desempeño de las *EPWavenets* con RNFBRG, y notamos que en cuanto a aproximación se refiere las *EPWavenets* son mejores en todos los casos.

En cuanto a predicción se refiere, al comparar la predicción de *EPWavenets* con RNFBR con funciones gaussianas y funciones sigmoidales (que no son de base radial) notamos que estás últimas tienen un peor desempeño, lo cual se

atribuye a la no localización de la función sigmoidal, y solo en un caso (uno de trece) las *EPWavenets* se ven consistentemente superadas.

Adicionalmente notamos que se requiere un número reducido de neuronas con wavelets para realizar la tarea de aproximación ($\overline{N} = 5.85$) lo cual se atribuye al uso de *wavenets* paramétricas ortogonales que minimizan el número de funciones wavelet requeridas al ser bases no redundantes.

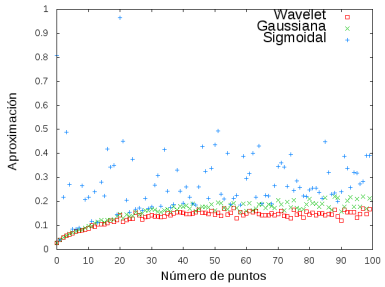
8. Trabajo futuro

Se tiene contemplado extender los experimentos para hacer comparaciones con otras FBR como son la función cuadrática, la función multicuadrática, la función inversa cuadrática, y las funciones *splines*. Además de usar parametrizaciones de mayor longitud, y plantear el problema de aproximación con *EPWavenets* como un problema multiobjetivo. También se contempla incrementar la dimensión n de los datos de entrada, toda vez que en este artículo se trabajó con vectores de entrada unidimensionales. Otros trabajos futuros también incluyen aplicar *EPWavenets* en problemas de clasificación y reconocimiento de patrones.

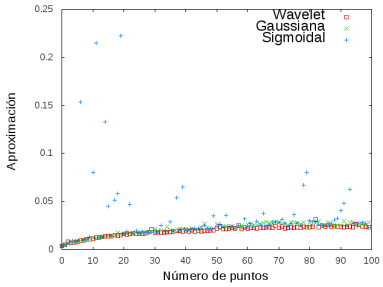
Referencias

1. Cybenko, G.: Approximation by Superpositions of a Sigmoidal Function. Mathematics of Control, Signals, and Systems pp. 303–314 (1989)
2. Daubechies, I.: Ten lectures on wavelets. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA (1992)
3. Delyon, B., Juditsky, A., Benveniste, A.: Accuracy analysis for wavelet approximations. Neural Networks, IEEE Transactions on 6(2), 332–348 (Mar 1995)
4. Haykin, S.: By Simon Haykin Neural Networks: A Comprehensive Foundation. Prentice Hall, 2 edn. (Jul 1998)
5. Herrera, O., Mora, R.: Aplicación de Algoritmos Genéticos a la Compresión de Imágenes con Evolets. Sociedad Mexicana de Inteligencia Artificial (2011)
6. Kononov, E.: Visual Recurrence Analysis (Apr 2015), `http://www2.informatik.uni-osnabrueck.de/marc/lectures/zra\_ss03/prgdat/vra4v2.zip`
7. Mallat, S.G.: A theory for multiresolution signal decomposition: The wavelet representation. IEEE Trans. on Patt. Anal. Mach. Intell. 11(7), 674–693 (1989)
8. Mallat, S.G.: Multiresolution approximations and wavelet orthonormal bases of $\mathbb{L}^2(\mathbb{R})$. Trans. Amer. Math. Soc. 315(1), 69–87 (1989)
9. Pourtaghi, A.: Wavelet Neural Network and Wavenet Performance Evaluation in Hydrodynamic Force Prediction due to Waves on Vertical Cylinders. International Journal of Information and Computer Science 11(9), 187–213 (dec 2012)
10. Roach, D.W., Lai, M.: Parameterizations of univariate orthogonal wavelets with short support, pp. 369–384. Vanderbilt Univ. Press (2002)
11. Sotelo, C.: The chemotactic hypothesis of Cajal: a century behind. Progress in brain research 136, 11–20 (2002)
12. Tewfik, A., Sinha, D., Jorgensen, P.: On the Optimal Choice of a Wavelet for Signal Representation. IEEE Transactions on information theory 38(2), 747–765 (march 1992)

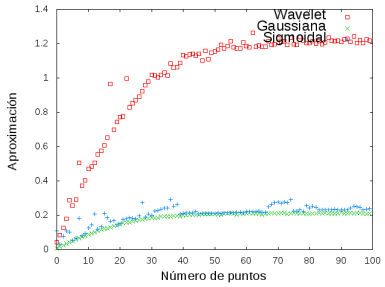
13. Ypma, A., Duin, R.P.: Using the Wavenet for function approximation pp. 23–6 (1997)
14. Zhang, Q., Benveniste, A.: Wavelet Networks. IEEE Trans. Neural Networks 3(6), 889–898 (1992)



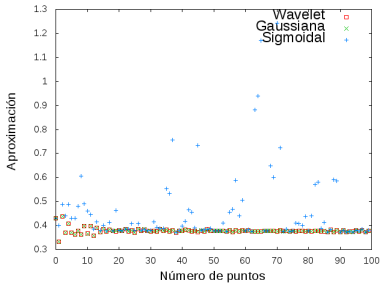
(a) Predicción de $f1$ con 8 neuronas



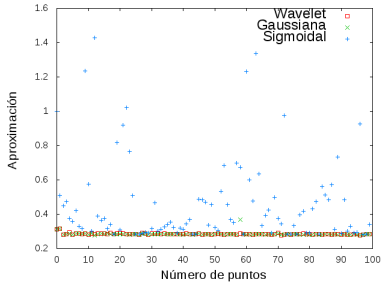
(b) Predicción de $f2$ con 6 neuronas



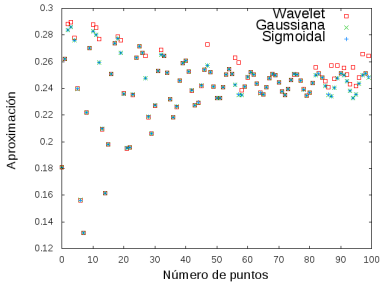
(c) Predicción de $f3$ con 7 neuronas



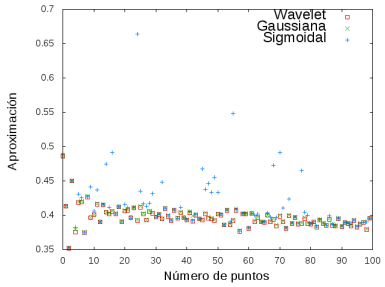
(d) Predicción de $f4$ con 6 neruonas



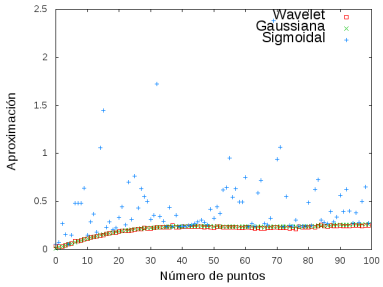
(e) Predicción de $f5$ con 8 neuronas



(f) Predicción de $f6$ con 1 neurona

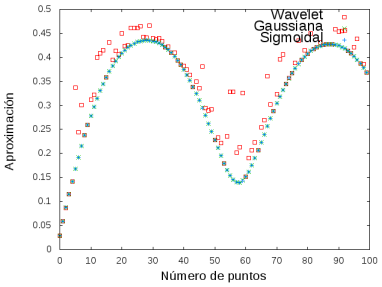


(g) Predicción de $f7$ con 7 neuronas

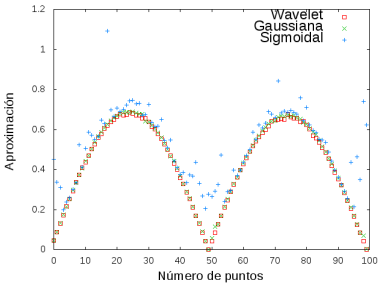


(h) Predicción de $f8$ con 8 neuronas

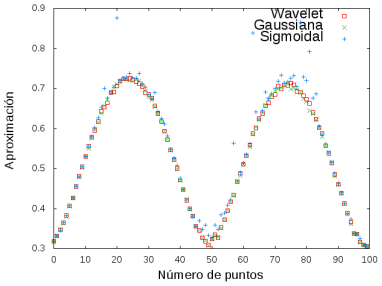
Fig. 4: Predicción de funciones $f1$ a $f6$



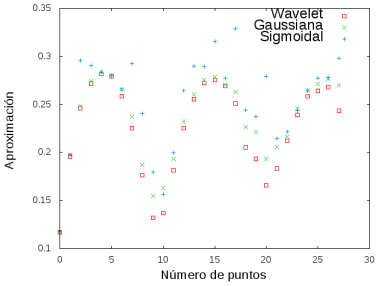
(a) Predicción de $f9$ con 1 neurona



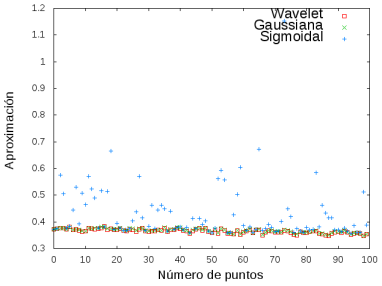
(b) Predicción de $f10$ con 8 neuronas



(c) Predicción de $f11$ con 1 neurona



(d) Predicción de $f12$ con 8 neuronas



(e) Predicción de $f13$ con 8 neuronas

Fig. 5: Predicción de funciones $f7$ a $f13$