# ESTUDIO DE LA CAPACIDAD DE CLASIFICACIÓN DE NEURONAS WAVELET SOBRE FUNCIONES BOOLEANAS

## *STUDY OF CLASSIFICATION PERFORMANCE OF WAVELET NEURONS ON BOOLEAN FUNCTIONS*

***Oscar Herrera Alcántara***
Universidad Autónoma Metropolitana, unidad Azcapotzalco, México
*oha@azc.uam.mx*

***Josué Rubén Castelán Aguilar***
Universidad Autónoma Metropolitana, unidad Azcapotzalco, México
*al209206744@azc.uam.mx*

**Recepción:** 29/noviembre/2022 **Aceptación:** 22/diciembre/2022

## Resumen

En este artículo se estudia la capacidad de clasificación de neuronas artificiales tipo perceptrón modificadas con funciones de activación wavelet. El modelo original del perceptrón contempla una función Heaviside y posteriormente se consideraron funciones de activación continuas, crecientes y acotadas como las funciones tangente hiperbólico y sigmoidal. El perceptrón puede clasificar datos binarios linealmente separables, y como contraejemplo suele presentarse el problema de la compuerta XOR, que requiere conectar más de un perceptrón para obtener una solución. La capacidad de clasificación de una sola neurona, y por consiguiente de redes de neuronas, puede mejorarse si se usa otro tipo de funciones de activación. En este trabajo se usan funciones wavelet, y los experimentos muestran mejoras en la capacidad de clasificación para funciones booleanas con dos y tres entradas.

**Palabras Clave:** Clasificación binaria, perceptrón, wavelets.

## Abstract

*This paper studies the classification capacity of artificial neurons modified from the original perceptron with wavelet activation functions. The original perceptron model contemplates a Heaviside function, and other increasing and bounded*

*activation functions were considered subsequently such as hyperbolic tangent and sigmoidal. The perceptron classifies linearly separable binary data, and as a counterexample the XOR gate problem is often presented, which requires connecting more than one perceptron to obtain a solution. The classification capacity of a single neuron, and therefore of networks of neurons, can be improved by using other types of activation functions. In this work wavelet functions are used, and the experiments show improvements in the classification performance for boolean functions with two and three inputs.*

**Keywords:** Binary classification, perceptron, wavelets.

## 1. Introducción

Una neurona artificial tipo perceptrón es un modelo matemático inspirado en la neurona biológica que actúa como clasificador binario [Rosenblatt, 1961]. El modelo original del perceptrón contempla una función Heaviside. Posteriormente, se consideraron funciones de activación continuas, crecientes y acotadas [Rumelhart, 1986] tales como las funciones tangente hiperbólico y sigmoidal. Las componentes originales de un perceptrón son: entradas X, pesos sinápticos W, potencial de activación $v =< X, W >$ como el producto interno de esos vectores, función de activación $f$, y una salida $y = f(v)$. El Teorema de Convergencia del Perceptrón [Lippmann, 1987] asegura que modificar los pesos W, en forma proporcional al error entre la salida de la neurona y el valor deseado, minimizará el error global ante todas las posibles entradas X, siempre que sean linealmente separables [Haykin, 2008].

En la figura 1, se ilustra una función lógica OR como problema linealmente separable en la izquierda, y una función lógica XOR en la derecha como problema no linealmente separable. En el caso separable, el vector de entrada X= {0,0} está debajo de la línea como clase 0, y los demás datos quedan arriba de la línea como clase 1.

Históricamente, se han explorado algunas alternativas para lidiar con la clasificación de datos no linealmente separables, entre ellas se tiene el “truco del kernel” que consiste en transformar el vector $X^n$ en otro vector $S^{n+k}$, con k>=0 tal que $S^{n+k}$ sea linealmente separable [Aizerman, 1964]. También, se han estudiado las redes en

multicapa [Cybenko, 1989] [Rumelhart, 1986] o de aprendizaje profundo [Bengio, 2016] que usan varias capas de neuronas para transformar los datos y tratarlos como problema linealmente separable. Otra alternativa es mediante autocodificadores divergentes, que planean resolver problemas no linealmente separables como el XOR con una sola capa [Kurtz, 2007].

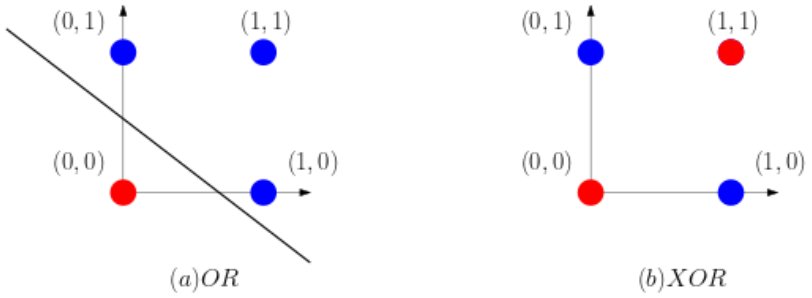



Figura 1 Problema linealmente separable y no linealmente separable.

Cabe resaltar que las técnicas previamente mencionadas no consideran, como eje fundamental, ampliar el conjunto de funciones de activación. En contraste, en [Piazza, 1992] se plantea usar una transformación para construir funciones de activación con polinomios adaptivos capaces de resolver el problema no linealmente separable con un menor número de capas. Tal es el caso de la XOR resuelta con un solo perceptrón. En este artículo, se retoma un tópico que puede considerarse "olvidado", "minimalista", "mítico" o hasta "esotérico" ya que se ha asumido errónea e históricamente que, con una sola capa o neurona, sólo es posible clasificar datos linealmente separables. Además, asumir que la única manera de resolver un problema no linealmente separable es con más y más capas de neuronas. También, se puede recordar la década en donde se redujo el interés en las redes neuronales al propagarse la idea de que un solo perceptrón no podía resolver el problema de la función booleana XOR. En este artículo se plantea incluir como parte del conjunto de funciones de activación a las funciones wavelet, y estudiar su capacidad de clasificación mediante funciones booleanas. El problema XOR de la tabla 1 puede resolverse con dos neuronas tipo perceptrón $N_1$ y $N_2$ en una capa oculta, combinadas mediante una tercera neurona $N_3$ en la capa de salida [Haykin, 2008].

Tabla 1 Tabla de verdad de la función lógica XOR de dos entradas.

| $X_0$ | $X_1$ | $X_2$ | d |
|---|---|---|---|
| 1 | 0 | 0 | 0 |
| 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 0 |

La figura 2 muestra una solución con pesos sinápticos b1=-3/2, b2=-1/2, b3=-1/2, w11 = w12=1, w21=w22=1, w31=-2 y w32=1, en donde el vector X tiene componentes X1 y X2, además de X0=1 que corresponde al BIAS. La salida deseada es la columna d. La salida $y= H(v)$ es obtenida con una función Heaviside.

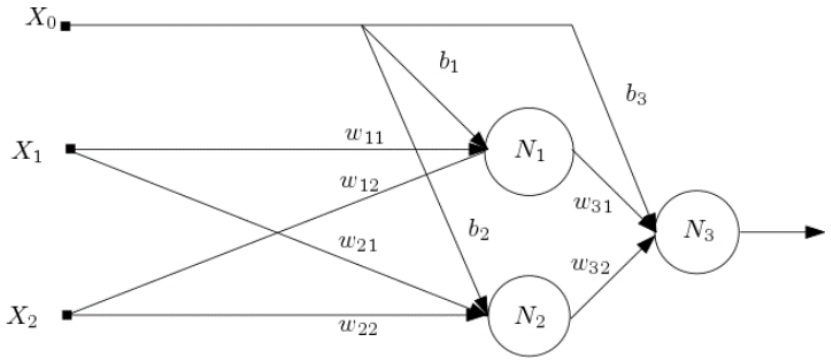


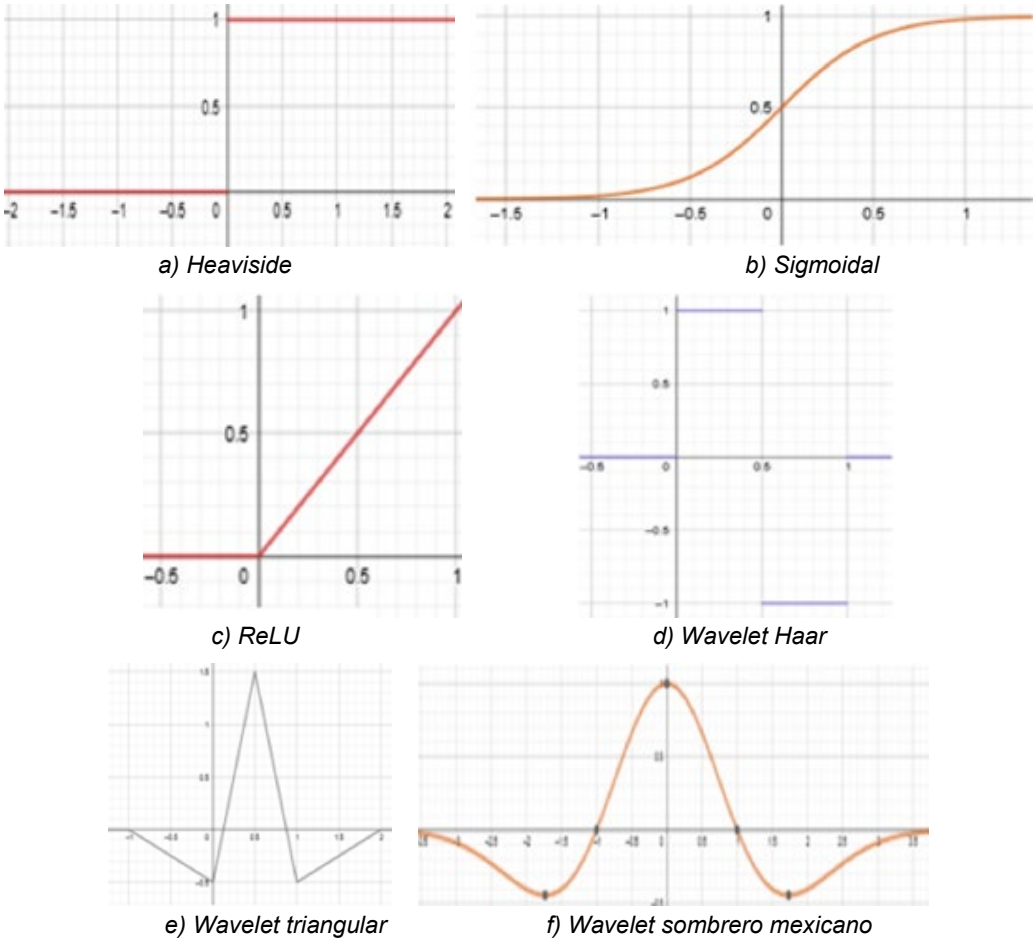

Figura 2 Solución a la función XOR con 3 neuronas.

De las K=16 compuertas o funciones booleanas de 2 entradas (sin contar el BIAS) existen 2 casos no linealmente separables conocidos como XOR y XNOR. Para 3 entradas se tienen K=256 funciones booleanas. En los experimentos se explorarán todas las compuertas para verificar cuáles son posibles de resolver usando una o dos neuronas con funciones de activación Heaviside, sigmoidal y ReLU. Además, se comparará su desempeño con el de funciones wavelet Haar, triangular y sombrero mexicano.

En la figura 3 se ilustran las gráficas de estas funciones de activación. Las ecuaciones de estas funciones son:

- Heaviside, ecuación 1.

$$y = \begin{cases} 0, & si\ v < 0 \\ 1, & si\ v \geq 0 \end{cases} \quad (1)$$



*a) Heaviside* *b) Sigmoidal*

*c) ReLU* *d) Wavelet Haar*

*e) Wavelet triangular* *f) Wavelet sombrero mexicano*

Figura 3 Gráfica de funciones de activación.

- Sigmoidal, ecuación 2.

$$y = \frac{1}{1 + e^{-v}} \tag{2}$$

- ReLU, ecuación 3.

$$y = \begin{cases} 0, & si\ v < 0 \\ v, & si\ v \geq 0 \end{cases} \tag{3}$$

- Wavelet Haar, ecuación 4.

$$y = \begin{cases} 1, & si\ 0 < v \leq 0.5 \\ -1, & si\ 0.5 < v \leq 1 \\ 0, & caso\ contrario \end{cases} \tag{4}$$

- Wavelet sombrero mexicano, ecuación 5.

$$y = (1 - v^2)e^{-\frac{v^2}{2}}. \tag{5}$$

- Wavelet triangular, ecuación 6.

$$y = \begin{cases} -0.5v - 0.5, & si\ -1 < v \leq 0 \\ 4v - 0.5, & si\ 0 < v \leq 0.5 \\ -4v + 3.5, & si\ 0.5 < v \leq 1 \\ 0.5v - 1, & si\ 1 < v \leq 2 \\ 0, & caso\ contrario \end{cases} \quad (6)$$

Aunque existen más funciones de activación [Herrera, 2022] para propósitos de esta investigación se consideran suficientes las descritas en las ecuaciones 1 a 6.

## 2. Métodos

Para estudiar la capacidad de clasificación de las neuronas se seguirán los siguientes pasos:

- Crear archivos con todas las funciones booleanas de 2 y 3 entradas: 00.txt, 01.txt, …, 15.txt para el caso de 2 entradas y 000.txt, 001.txt, …, 255.txt para el caso de 3 entradas.
- Usar funciones de activación “clásicas” y wavelets. Las clásicas serán: Heaviside, sigmoidal y ReLU. Las wavelets serán: Haar, triangular, y sombrero mexicano.
- Usar un algoritmo genético para optimizar los pesos sinápticos. La función de aptitud es el error promedio sobre los D vectores de entrada, D=4 para el caso de dos entradas y D=8 para el caso de tres entradas. También, se usará el algoritmo de optimización de gradiente ADAM.
- Medir el error de clasificación de cada caso. Para 2 entradas se tienen K=16 compuertas, y para 3 entradas se tienen K=216 compuertas. Se contabilizará el número de fracasos y, por ende, los éxitos que miden la capacidad de clasificación.
- Comparar los resultados de clasificación de las funciones clásicas y wavelets.

Conviene mencionar que algunas de las compuertas tienen nombres conocidos, por ejemplo, la función 06.txt corresponde al patrón de salida “0110“ XOR. La función 09.txt tiene el patrón de salida “1001” XNOR. La función AND es 01.txt con salidas “0001”, y la función OR es 07.txt con salidas “0111”. En el caso de 3 entradas 001.txt

corresponde a la función AND con salidas “0001”, mientras que 007.txt corresponde a la función OR con salidas “0111”.

La neurona modificada a utilizar considera traslaciones y dilataciones en las funciones de activación, que en el caso de wavelets está fundamentado en la fórmula de reconstrucción [Daubechies, 1992] de una función $f(x)$ mostrada en la ecuación 7.

$$f(x) = \frac{1}{C_h}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} < f(x), h_{ab} > \frac{1}{\sqrt{|a|}} h_{ab}(x)\frac{1}{a^2}\,dadb \qquad (7)$$

Donde:

- $f(x)$ es la función por aproximar
- $h_{ab}(x) = h(\frac{x-b}{a})$ es la función wavelet $h$ trasladada con b y dilatada con a
- $C_h > 0$ es la constante de admisibilidad
- $< f(x), h_{ab} >$ es el coeficiente de transformación wavelet obtenido con el producto interno entre la función por aproximar y la función $h_{ab}(x)$

La neurona wavelet propuesta se muestra en la figura 4 para el caso de dos entradas. Se ilustra el vector de pesos sinápticos $W=\{W_0, W_1 \text{ y } W_2\}$ que conecta las componentes del vector de entrada $X=\{X_0, X_1, X_2\}$. El potencial de activación es el producto interno $v=<X, W>$ el cual se usa en la función de activación para obtener $u=h_{ab}(v)$. Es importante mencionar que se usa una función tipo Heaviside $H$ para calcular la salida de la neurona, de esta forma $y=H(u\text{-}0.5)$ permite empatar el rango de valores de u con los valores binarios 0 y 1 de las tablas de verdad (valores deseados d). El uso de la función de salida $H$, además facilita el conteo de los casos de fracaso y éxito en la aproximación de las funciones booleanas.

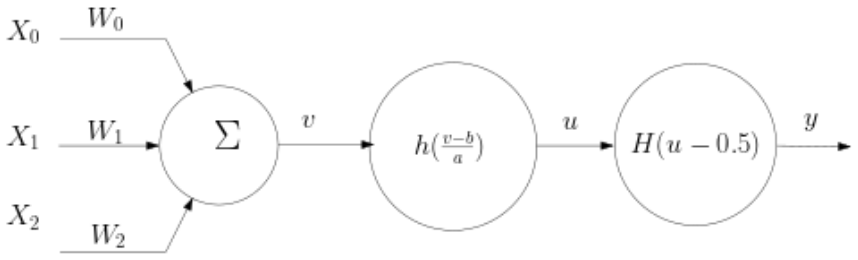



Figura 4 Neurona con función de activación wavelet para dos entradas.

El error de aproximación para un vector de entrada está dado por $\mathrm{e} = \mathrm{d} - \mathrm{y}$, y el error promedio $\mathrm{E_{av}}$ sobre las $\mathrm{D}$ muestras de entrada se define en la ecuación 8.

$$E_{av} = \frac{1}{D}\sum \frac{1}{2} e^2 = \frac{1}{D}\sum \frac{1}{2}(d-y)^2 \qquad (8)$$

En la ecuación 8, cuando el error $E_{av}$ es cero indica que la función booleana ha sido resuelta satisfactoriamente. Por tanto, la capacidad de aproximación es máxima al resolver, sin error, el 100% de las $\mathrm{K}$ funciones booleanas. Nótese que $\mathrm{K}=8$ para 2 entradas y $\mathrm{K}=256$ para 3 entradas.

## 3. Resultados

En esta sección, se describen dos experimentos. En el Experimento 1 se aplica un algoritmo genético, y en el Experimento 2 se aplica un método de gradiente. En ambos experimentos se busca optimizar los parámetros de las neuronas a efecto de minimizar el error de clasificación $E_{av}$.

### Experimento 1

Se usa un algoritmo genético con cruzamiento en un punto con probabilidad $\mathrm{P_c}=0.97$. La mutación es aplicada solo a los descendientes con probabilidad $\mathrm{P_m}=0.2$. Se usa selección directa conservando $\mathrm{P}$ individuos con la mayor aptitud de una población temporal de tamaño $2\mathrm{P}$. Se crean $\mathrm{P}$ individuos descendientes a partir de $\mathrm{P}$ progenitores ordenados según su aptitud y con apareamiento $(1,\mathrm{P})$, $(2,\ \mathrm{P}\text{-}1)$, ... $(\mathrm{P}/2, \mathrm{P}/2+1)$. El tamaño de la población es $\mathrm{P}=40$. La función de aptitud es minimizar el error promedio $\mathrm{E_{av}}$. El número de generaciones es $\mathrm{G}=2000$, con la idea de proveer oportunidad suficiente para la convergencia.

Se contemplan 3 casos:

- **Una sola neurona, 2 entradas y algoritmo genético**. En la tabla 2, se muestra el número de fracasos y los tiempos de ejecución para una sola neurona, con traslaciones y dilataciones en las funciones de activación clásicas y wavelets. Las funciones de activación se listan en la primera columna. La capacidad de aproximación es máxima para todas las wavelets

con cero fracasos de las $K=16$ compuertas. Las funciones clásicas no pueden resolver 2 casos, como se muestra en la segunda columna.

Tabla 2 Capacidad de clasificación de una neurona para compuertas de dos entradas.

| Función de activación | Fracasos (máximo K=16) | Tiempo de ejecución |
|---|---|---|
| Heaviside | 2 | 17 s |
| Sigmoidal | 2 | 86 s |
| ReLU | 2 | 14 s |
| Haar | 0 | 1 s |
| Triangular | 0 | 1 s |
| Sombrero mexicano | 0 | 1 s |

- **Una neurona, 3 entradas y algoritmo genético.** En la tabla 3 se reportan los resultados para una neurona con las mismas funciones de activación del caso de dos entradas. En la segunda columna, se aprecia que el mejor desempeño lo logra la función wavelet sombrero mexicano (con 46 fracasos de las $K=256$ compuertas) en comparación con las funciones clásicas que tienen 152 fracasos. Las wavelets Haar y triangular alcanzan una capacidad cercana a la del sombrero mexicano y, en un tiempo menor, reducen casi 3 veces el número de fracasos con respecto a las funciones clásicas.

  El tiempo de ejecución tan alto en las funciones clásicas, se explica por el número máximo de generaciones $G=2000$ del algoritmo genético que tiene la finalidad de explorar todo el conjunto de soluciones, y en particular porque se deben ejecutar todas las generaciones cuando no se resuelve la función booleana. Los tiempos de ejecución se reportan en la tercera columna. Nótese que, aunque la función sombrero mexicano tarda más, logra resolver un mayor número de funciones booleanas.

Tabla 3 Capacidad de aproximación para una neurona y funciones de tres entradas.

| Función de activación | Fracasos (máximo K=256) | Tiempo de ejecución |
|---|---|---|
| Heaviside | 152 | 22 minutos |
| Sigmoidal | 152 | 33 minutos |
| ReLU | 152 | 17 minutos |
| Haar | 52 | 5 minutos |
| Triangular | 52 | 7 minutos |
| Sombrero mexicano | 46 | 34 minutos |

- **Dos neuronas, 3 entradas y algoritmo genético.** Dado que existen casos de 3 entradas que no se pudieron resolver con una sola neurona, ya sea con funciones clásicas o wavelets, se propone usar 2 neuronas combinadas mediante la función $H$. Los resultados se reportan en la tabla 4. Nótese que las funciones wavelet resuelven las $K=256$ funciones booleanas de 3 entradas. Sin embargo, no sucede lo mismo con las funciones clásicas, que tienen 24 fracasos con Heaviside y ReLU, y 37 fracasos con la función sigmoidal. Inclusive, tardan casi media hora más de ejecución. ¿Por qué el tiempo de ejecución requerido es sumamente elevado? Porque termina todas las generaciones el algoritmo genético. Esto no es necesario con las funciones wavelet que alcanzan la solución en un menor número de generaciones, y por ende, el tiempo de ejecución es mucho menor, del orden de segundos, como se muestra en la tercera columna de la tabla 4.

Tabla 4 Capacidad de aproximación con 2 neuronas.

| Función de activación | Fracasos (máximo K=256) | Tiempo de ejecución |
|---|---|---|
| Heaviside | 24 | 287 s |
| Sigmoidal | 37 | 1,980 s |
| ReLU | 24 | 223 s |
| Haar | 0 | 8 s |
| Triangular | 0 | 12 s |
| Sombrero mexicano | 0 | 37 s |

## Experimento 2

En un segundo experimento se usó el optimizador ADAM [Kingma, 2015] de las bibliotecas Keras [Chollet, 2015] y Tensorflow [Abadi, 2015]. Estas bibliotecas han sido diseñadas para optimizar redes neuronales profundas con muchos parámetros, inclusive con uso de hardware GPU de alto rendimiento. En este experimento se probó con una o dos neuronas y una GPU NVidia GEForce RTC 3070.

Las funciones de activación fueron: sigmoidal, ReLU, y wavelet sombrero mexicano. Cabe mencionar que tanto Keras como Tensorflow, no implementan en sus APIs funciones de activación wavelet. Sin embargo, en este experimento se ha implementado la función sombrero mexicano tomando como referencia un trabajo

previo [Herrera, 2022]. Aunque, no se contempla la optimización de parámetros de dilatación ni escalamiento. Se usó Python para leer los archivos de las funciones booleanas con la rutina loadtxt() de numpy. Una vez leídos los datos, se crearon los datasets $X$, $Y$ para 2 o 3 entradas. Los datasets se introdujeron a un modelo secuencial con 1 capa densa de una o dos neuronas. Cada neurona usa alguna de las 3 funciones de activación mencionadas previamente. Como función de salida $H$, se considera una función de salida sigmoidal disponible en Keras-Tensorflow (en lugar de una Heaviside, como se ilustró en la figura 4) para empatar el rango de valores de las tablas de verdad. La función de pérdida (loss function) es MSE (Mean Squared Error) con un número máximo de 1500 épocas. El conjunto de datos de prueba (test dataset) en realidad es el mismo conjunto de datos de entrenamiento (train dataset) porque al tratarse de un problema que mide el desempeño de clasificación, y al resolver las funciones booleanas, se deja fuera la medición de la capacidad de generalización sobre un conjunto de prueba.

Se contemplan 3 casos:

- **Una neurona, 2 entradas y optimizador ADAM.** En la tabla 5 se muestran los resultados de las funciones sigmoidal, ReLU y sombrero mexicano, listadas en la primera columna. Se puede ver que la función ReLU, ampliamente utilizada en redes neuronales profundas, tiene en este experimento el peor desempeño con 7 fracasos sobre las 16 compuertas. Le sigue la función Sigmoidal con 6 fracasos. En el Experimento 1 estas funciones solo tenían 2 fracasos, pero se contemplaban parámetros de dilatación y escalamiento, optimizados con un algoritmo genético.

Tabla 5 Capacidad de aproximación con 1 neurona, 2 entradas con Keras-Tensorflow.

| Función de activación | Fracasos (máximo K=16) | Tiempo de ejecución |
|---|---|---|
| Sigmoidal | 6 | 57 s |
| ReLU | 7 | 56 s |
| Sombrero mexicano | 0 | 34 s |

La máxima capacidad de clasificación la logra la wavelet sombrero mexicano sin fracasos, es decir, resuelve satisfactoriamente las K=16 compuertas.

Además, lo hace en casi la mitad del tiempo al requerir un menor número de épocas para encontrar las soluciones, como se muestra en la tercera columna de la tabla 5.

- **Una neurona, 3 entradas y optimizador ADAM.** Como se muestra en la tabla 6, el mismo comportamiento del caso anterior se observa con funciones de 3 entradas. El peor desempeño lo tiene la función ReLU con 198 fracasos, seguida de la sigmoidal con 166 fracasos. El menor número de fracasos es igual a 76 con la función sombrero mexicano. El tiempo de ejecución de la tercera columna de la tabla 6, se explica por la relación que existe entre el número de fracasos y el número de épocas, es decir en la búsqueda de la solución se explora todo el conjunto de soluciones hasta agotar el número máximo de épocas permitidas. En caso de fracaso se ejecutará el máximo número de épocas. Si se encuentra una solución a una compuerta, el algoritmo termina la búsqueda y avanza con la siguiente de las K=256. Esto justifica por qué la función sombrero mexicano requiere menos tiempo.

Tabla 6 Capacidad de aproximación con 1 neurona, 3 entradas con Keras-Tensorflow.

| Función de activación | Fracasos (máximo K=256) | Tiempo de ejecución |
|---|---|---|
| Sigmoidal | 166 | 2,050 s |
| ReLU | 198 | 2,161 s |
| Wavelet sombrero mexicano | 76 | 1,477 s |

- **Dos neuronas, 3 entradas y optimizador ADAM.** Al igual que en el Experimento 1, al tener varias compuertas sin resolver con una sola neurona, se probó con 2 neuronas combinadas con una función $H$ de salida. Los resultados se reportan en la tabla 7, en donde la función sigmoidal tiene el peor desempeño, tanto en el número de fracasos igual a 153 como en un mayor tiempo de ejecución de 1915 segundos.

Tabla 7 Capacidad de aproximación con 2 neuronas, ADAM y Keras-Tensorflow.

| Función de activación | Fracasos (máximo K=256) | Tiempo de ejecución |
|---|---|---|
| Sigmoidal | 153 | 1,915 s |
| ReLU | 129 | 1,692 s |
| Wavelet sombrero mexicano | 3 | 686 s |

La función ReLU tiene un menor número de fracasos que la sigmoidal, que son 129. Esto representa casi el 50% de las K=256 compuertas en un tiempo ligeramente menor. En este caso, al usar dos neuronas con wavelet sombrero mexicano, se tienen 3 fracasos que es mucho menor en comparación con los de las funciones sigmoidal y ReLU. Y en cuanto a tiempos de ejecución se refiere, la función wavelet sombrero mexicano requiere casi tres veces menos tiempo. Entonces, no solo hay una mejora en cuanto a capacidad de clasificación sino también en el tiempo de ejecución.

## 4. Discusión

De los experimentos realizados, se pueden hacer los siguientes comentarios.

Todas las funciones de 2 entradas pueden resolverse con un solo perceptrón modificado con funciones de activación wavelet.

Dotar a las funciones de activación con parámetros de traslación y dilatación justificados en la teoría de wavelets, y en particular en la fórmula de reconstrucción de la transformada wavelet, ofrece un mayor grado de libertad en el entrenamiento de las neuronas. Aunque este concepto no aplica directamente al caso de las funciones clásicas, se puede usar el mismo concepto para medir la capacidad de aproximación de las funciones de activación en igualdad de condiciones.

Aún con parámetros de traslación y dilatación, las funciones clásicas no muestran mejoría en el caso de compuertas de 2 entradas, ya que hay 2 de las 16 funciones booleanas XOR y XNOR que no se resuelven. En contraste, con funciones wavelets sí fue posible resolver la totalidad de las 16 compuertas, por lo que no se consideró necesario probar con 2 o más neuronas al resolver las compuertas de 2 entradas.

Para el caso de compuertas de 3 entradas, fue suficiente usar 2 neuronas seguidas de una función Heaviside o sigmoidal, para acoplar los valores con el rango esperado de las 256 tablas de verdad.

Se ha usado un algoritmo genético y el optimizador de gradiente ADAM para explorar el conjunto de soluciones. El algoritmo genético minimiza de mejor manera la probabilidad de quedarse en un mínimo local mediante el cruzamiento, la mutación y la selección, así como con un alto número de generaciones por lo que

logra mejor desempeño de clasificación que ADAM. Además, el algoritmo genético permitió optimizar parámetros de traslación y escalamiento, algo que aún no se contempla en versiones de Keras y Tensorflow.

Al usar las bibliotecas de Keras y Tensorflow, se optó por aplicar una función sigmoidal de salida en lugar de una función Heaviside, principalmente porque no está disponible en el listado estándar de funciones de activación.

El haber trabajado con problemas aparentemente pequeños deja ver que la discusión que originalmente se planteó en los inicios de las redes neuronales, y que está relacionada con los problemas no linealmente separables, sigue vigente, y mantiene un área de investigación abierta para proponer nuevas funciones de activación, arquitecturas de conexiones entre neuronas, y algoritmos de optimización, entre otros desafíos para futuras investigaciones.

## 5. Conclusiones

En base a los experimentos realizados, y considerando diferentes funciones de activación se puede concluir que:

- Ha sido posible optimizar los parámetros de neuronas con función wavelet usando algoritmos genéticos para resolver todas las funciones booleanas de 2 y 3 entradas.
- Es evidente la mejora en el desempeño de las funciones wavelet comparadas con funciones clásicas, en donde las primeras tienen mejor capacidad de clasificación toda vez que permiten resolver todas las funciones booleanas de 2 y 3 entradas en menor tiempo y con menos neuronas.
- Las neuronas wavelet proveen una mayor capacidad de clasificación como neuronas individuales y en redes de neuronas.
- El desempeño de neuronas con funciones de activación wavelet es superior al de funciones clásicas al pretender resolver las funciones booleanas de dos y tres entradas.
- A pesar de los resultados alentadores obtenidos, quedan preguntas abiertas, por ejemplo: ¿Cómo optimizar con eficiencia redes profundas con funciones de activación wavelet? ¿Se requieren redes demasiado profundas aún con

neuronas wavelet? ¿Cómo aplicar métodos de gradiente que permitan optimizar parámetros de traslación y escalamiento, requeridos por las funciones wavelet? ¿Existen otras funciones de activación que maximicen la capacidad de clasificación de las neuronas individuales y de redes de neuronas?

En trabajos futuros se espera abordar estas preguntas y obtener aportaciones al respecto.

## 6. Bibliografía y Referencias

[1] Abadi, M.; Agarwal, A.; Barham, P.; Brevdo, E.; Chen, Z.; Citro, C.; Corrado, G.S.; Davis, A.; Dean, J.; Devin, M.; et al. TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems, 2015.

[2] Aizerman, M., Braverman, E., Rozonoer, L. Automation and Remote Control, No. 25, 821-837, 1964.

[3] Bengio, Y. Deep Learning, MIT Press, 2016.

[4] Chollet, F.; Zhu, Q.S.; Rahman, F.; Lee, T.; Marmiesse, G.; Zabluda, O.; Qian, C.; Jin, H.; Watson, M.; Chao, R.; et al. Keras. 2015. https://keras.io/ (Consultado el 29 de septiembre de 2022).

[5] Cybenko, G. Approximation by superpositions of a sigmoidal function, Mathematics of Control, Signals, and Systems, No. 2, vol. 4, 303-314, 1989.

[6] Daubechies, I. Ten Lectues on Wavelets, Society for Industrial and Applied Mathematics, Philadelphia, PA, 1992.

[7] Kurtz, K.J. The divergent autoencoder (DIVA) model of category learning, Psychonomic Bulletin & Review No. 14, 560-576, 2007.

[8] Haykin, S. Neural networks and learning machines, Pearson Prentice Hall, 2008.

[9] Herrera, O., Priego, B. Wavelets as activation functions in Neural Networks, 2022. Journal of Intelligent & Fuzzy Systems, No. 42, 2022, 4345-4355.

[10] Kingma, D. P., B. J. Adam: A Method for Stochastic Optimization, 2014. 3rd International Conference for Learning Representations, San Diego, 2015.

[11] Lippmann, R. P. An Introduction to Computing with Neural Nets, ASSP Magazine, IEEE, 1987.

[12] Piazza, F., Uncini, A., Zenobi, M. Artificial Neural Networks with Adaptive Polynomial Activation Function, 1992.

[13] Rosenblatt, F. Principles of Neurodynamics. Spartan, Washington, DC, 1961.

[14] Rumelhart, D. E., Hinton, G.E., Williams, R. J. Learning representations by back-propagation errors. Nature, No. 323, 1986.