POLYSIGMOID: UNA FUNCIÓN DE ACTIVACIÓN POLINOMIAL PARA REDES NEURONALES

POLYSIGMOID: A POLYNOMIAL ACTIVATION FUNCTION FOR NEURAL NETWORKS

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

Jorge Miguel Jaimes Ponce
Universidad Autónoma Metropolitana, Unidad Azcapotzalco, México
jjp@azc.uam.mx

Victor Alberto Cruz Barriguete
Universidad Autónoma Metropolitana, Unidad Azcapotzalco, México
vacb@azc.uam.mx

Recepción: 14/noviembre/2024 **Aceptación:** 16/abril/2025

Resumen

En este artículo se propone una nueva función de activación que aproxima la función de activación $Sigmoid$, la cual es ampliamente usada en redes neuronales artificiales. El interés de aproximar a la función $Sigmoid$ es para reducir la evaluación de un gran número de términos polinomiales (teóricamente infinito) lo cual genera un alto costo computacional. El enfoque es aproximarla mediante un polinomio a trozos de orden finito llamado $PolySigmoid$. La función $PolySigmoid$ contempla dos parábolas para aproximar la parte no lineal de la función Sigmoid que es continua, creciente y acotada. Adicionalmente, dada la relación que hay entre las funciones $Sigmoid$ y $Tanh$ también es posible proponer una función $PolyTanh$ que aproxima la función $Tanh$, y esto da pauta a aproximar otras funciones de activación. Experimentalmente, se valora la aproximación de ambas funciones $PolySigmoid$ y $PolyTanh$ mediante resultados de clasificación con la base de datos MNIST.

Palabras Clave: Función de activación, Redes neuronales artificiales, Clasificación.

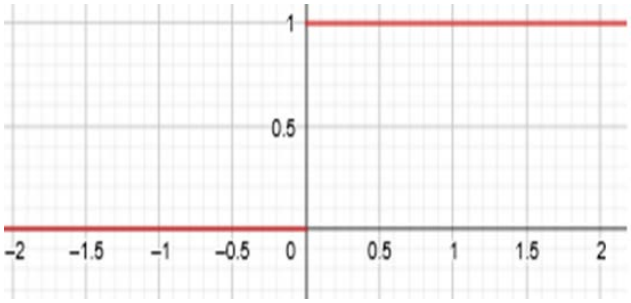
Abstract

In this paper, a novel activation function is proposed as an approximation to the Sigmoid activation function. The $Sigmoid$ function is widely used in neural networks, and its approximation is to reduce the evaluation of a large number (theoretically infinite) of polynomial terms which generates a high computational cost. The $Sigmoid$ activation function is approximated with a piecewise polynomial called $PolySigmoid$. The $PolySigmoid$ function considers two parabolas to approximate the non-linear sections since it is a continuous, monotone, and non-decreasing function. Additionally, there is a relationship between the $Sigmoid$ and $Tanh$ functions, then it is possible to propose another function $PolyTanh$ to approximate the $Tanh$ function, which opens the possibility to approximate other activation functions. Experimentally, the approximations of $PolySigmoid$ and $PolyTanh$ *are valuated through the classification results of a neural network trained with the MNIST dataset.*

Keywords**:** *Activation function, Artificial neural networks, Classification.*

1. Introducción

Históricamente, el Perceptrón [Rosenblatt, 1961] ha fungido de piedra angular de modernas redes neuronales artificial (RNA) incluidas las redes de perceptrones en multicapa (MLP) y las redes neuronales profundas (DNN) [Bengio, 2016]. En el primer modelo del perceptrón se incluye la función de activación $Heaviside$, mostrada en la Figura 1.



Fuente: elaboración propia

Figura 1 Gráfica de la función de activación $Heaviside$.

La expresión matemática de la función $Heaviside$ es la Ecuación 1 [Davies, 2002].

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

En donde, la variable v alude al potencial de activación que se calcula con el producto punto de la Ecuación 2.

$$v =< W, X > \quad (2)$$

En la Ecuación 2, X representa el vector de datos de entrada y W el vector de pesos sinápticos. En el caso del Perceptrón, la salida y se obtiene con la Ecuación 3, evaluando la función de activación.

$$y = H(v). \quad (3)$$

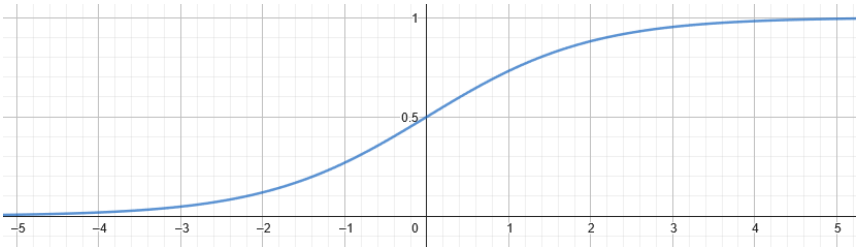
El Teorema de Convergencia del Perceptrón (TCP) [Lippmann, 1987] establece que al proveerle datos linealmente separables pertenecientes a dos clases 0 o 1, el Perceptrón generará un hiperplano de separación en un número finito de iteraciones, siempre que la modificación de los pesos W se realice multiplicando la entrada correspondiente por el error entre el valor deseado y la salida de la neurona, lo cual minimizará el error global ante las entradas X [Haykin, 2008].

Sin embargo, a pesar de que la función Heaviside es consistente con el TCP, ésta ha dado pauta a controversias al no tener derivada en el origen. Ante ello, se han propuesto otras funciones de activación incluidas la $Sigmoid$ y $Tanh$, en gran parte justificadas por el artículo de Cybenko “Approximation by superpositions of a sigmoidal function” [Cybenko, 1989] en el cual se demuestra que se pueden aproximar funciones en intervalos acotados mediante combinaciones lineales de funciones continuas, crecientes y acotadas, que son características que cumplen ambas funciones $Sigmoid$ y $Tanh$. La Ecuación 4 define a la función $Sigmoid$ [Narayan, 1997] y su gráfica se muestra en la Figura 2.

$$s(v) = \frac{1}{1 + e^{-v}} \quad (4)$$

Existe una relación entre la función tangente hiperbólica y la función $Sigmoid$ dada por la Ecuación 5.

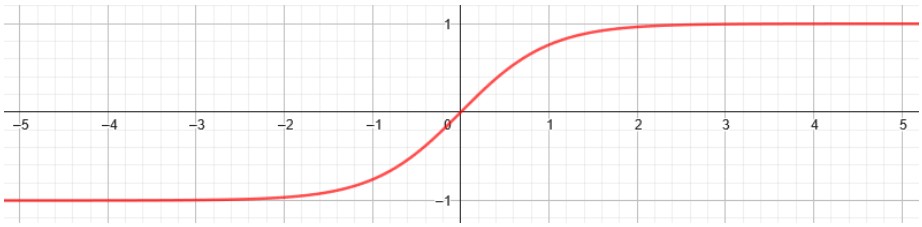
$$Tanh(v) = 2s(2v) - 1 \quad (5)$$



Fuente: elaboración propia

Figura 2 Gráfica de la función de activación $Sigmoid$.

La Figura 3 muestra la gráfica de la función $Tanh$. Como puede apreciarse, al comparar las Figuras 2 y 3, ambas funciones de activación, $Sigmoid$ y $Tanh$, comparten la misma forma, lo que cambia es el rango que en el caso de la función $Sigmoid$ es de 0 a 1, y en el caso de $Tanh$ es de -1 a 1.



Fuente: elaboración propia

Figura 3 Gráfica de la función de activación $Tanh$.

Ambas funciones $Sigmoid$ y $Tanh$ involucran la función exponencial, que puede aproximarse con la serie infinita de la Ecuación 6.

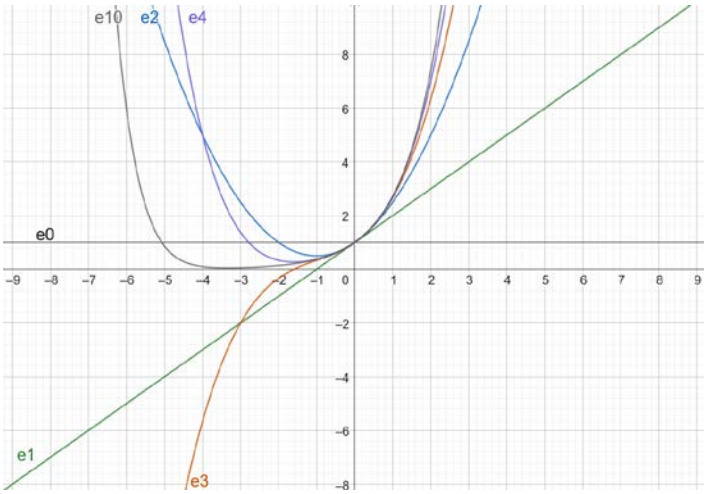
$$e^x = 1 + x + \frac{x^2}{2} + \frac{x^3}{3!} + \ldots \tag{6}$$

Por lo que, al combinar las Ecuaciones 4 y 6, se tiene una aproximación inmediata para la función Sigmoid en la Ecuación 7.

$$s(x) \approx \frac{1}{2 - x + \frac{x^2}{2} - \frac{x^3}{3!} + \cdots} \tag{7}$$

Lo cual indica que se requiere un número infinito de términos en el denominador de la aproximación (Ecuación 7). En la Figura 4, se muestran aproximaciones a la función $exponencial$, es decir con $\mathrm{e}_n(\mathrm{x})$ en donde n es el número de términos de la serie de la Ecuación 6. Así, por ejemplo, $\mathrm{e}_0(\mathrm{x}) = 1$, $\mathrm{e}_1(\mathrm{x}) = 1 + \mathrm{x}, \ldots, \mathrm{e}_n(\mathrm{x}) =$

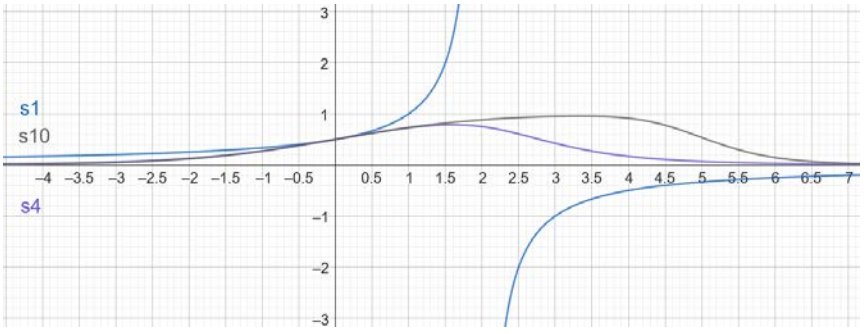
$\mathrm{e}_{n-1}(\mathrm{x}) + \frac{x^n}{n!}$ con lo que se pretende ilustrar que la serie $\mathrm{e}_n(\mathrm{x})$ se aproxima cada vez más a la función $exponencial\ e^x$. Sin embargo, con $\mathrm{n} = 10$, se puede apreciar la falta de comportamiento asintótico a cero a partir de $\mathrm{x} < -4$.



Fuente: elaboración propia

Figura 4 Gráfica de la función exponencial aproximada con series finitas.

En este mismo sentido, en la Figura 5 se presentan aproximaciones de la función $Sigmoid$ mediante $s_n(x)$ para los casos $n = 0$, $n = 4$ y $n = 10$. Obsérvese que la gráfica de $s_{10}(x)$ pierde el comportamiento asintótico a 1, aproximadamente para $x > 3.33$.



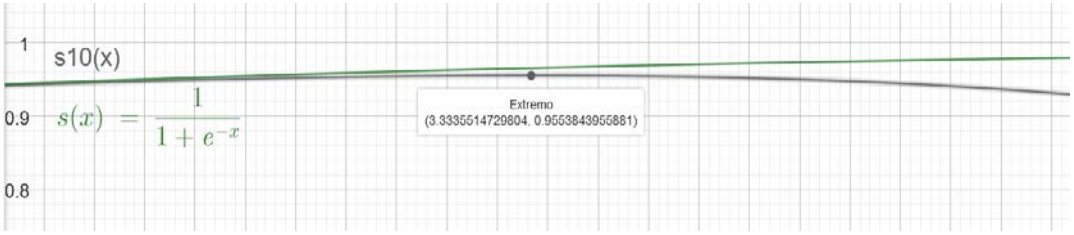
Fuente: elaboración propia

Figura 5 Gráfica de las funciones Sigmoid y exponencial aproximada con series finitas.

Así que surgen las preguntas ¿es posible tener otras aproximaciones para la función $Sigmoid$? ¿Es posible reducir el número de términos a evaluar para aproximar

aceptablemente a la función $Sigmoid$? Encontrar la respuesta a estas preguntas es una de las motivaciones para el presente artículo.

Dado que la Figura 5 muestra que la función $s_{10}(x)$ pierde la capacidad de aproximación asintótica hacia 1.0 en el eje positivo, se plantea la posibilidad de aproximar la función a trozos (o por intervalos, en inglés piecewise). De tal manera que puedan juntarse diferentes expresiones matemáticas, por ejemplo, para $x < 3.3335$ usar $s_{10}(x)$ y para $x > 3.3335$ simplemente fijarlo a 1.0. El problema que subyace es que el valor de $s_{10}(x)$ para $x = 3.3335$ es casi 0.955384, y entonces tendría "un salto" para pasar al valor 1, lo cual puede apreciarse en la Figura 6 al comparar la función $s_{10}(x)$ con $s(x)$ en la vecindad de $x = 3.3335$. Inclusive, aun cuando la función $Sigmoid$ no llega a 1 exactamente, sino que el máximo está en $s(3.3335) = 0.9655$, existe una diferencia que generaría la discontinuidad.



Fuente: elaboración propia

Figura 6 Diferencia entre $s_{10}(x)$ y la función Sigmoid alrededor de $x = 3.3335$.

De la discusión anterior se justifica la necesidad de explorar otra aproximación para la función $Sigmoid$ que, además de reducir el error de aproximación, también pueda ser más eficiente computacionalmente, en otras palabras, se buscan aproximaciones que involucren un número finito de términos, y a la vez logren una aproximación aceptable.

Cabe mencionar que se han propuesto otras funciones de activación tales como Swish [Ramachandran, 2017], GeLU [Hendrycks, 2016], Mish [Misra, 2019], SAAF [Shou, 2020], FELU [Qiumei, 2019], SELU [Sakketou, 2019], DELU [Burak, 2023], SPLASH [Agostinelli, 2015] y APTx [Kumar, 2022] entre otras. Sin embargo, todas ellas involucran funciones exponenciales o sus expresiones son más complejas, y por ende elevan el costo computacional.

En este trabajo se retoma un trabajo previo [Herrera, 2024] en donde se presenta la función de activación $PolySigmoid$, la cual aproxima la función $Sigmoid$ en cuatro intervalos. En este artículo, se plantea mejorar la aproximación al ajustar un parámetro de escalamiento α mediante un algoritmo genético. Luego, se comparará el desempeño de $PolySigmoid$ y la función $Sigmoid$ en un problema de clasificación de datos con redes neuronales artificiales.

2. Métodos

Pasos para obtener la expresión mejorada de la función $PolySigmoid$:

- Revisar la expresión de la función $PolySigmoid$ e introducir un factor de escalamiento $s(k)$, es decir, usar αx en lugar de x.
- Hacer un muestreo $s(k)$ de K puntos de la función $Sigmoid$ $s(k)$ en el intervalo $[-\frac{1}{\alpha}, 0]$, aprovechando la simetría de la función, y toda vez que podría hacerse también sobre el intervalo $[0, \frac{1}{\alpha}]$.
- Comparar el error sobre K puntos $ps(k)$ que dependen de α y dado el muestreo de la función $PolySigmoid$ y la función $Sigmoid$ con las muestras $s(k)$, donde $k = 1, 2, \dots, K$.
- Minimizar el error de aproximación optimizando el valor de α. Conforme el valor de α cambia, se generan K puntos de muestreo $ps(k)$ a partir de la función $PolySigmoid$, los cuales se alejan o acercan a los correspondientes K puntos $s(k)$, de la función $Sigmoid$, con lo que se calcula el error de aproximación mismo que se busca minimizar. El caso ideal se lograría si los K puntos de $Sigmoid$ y $PolySigmoid$ se traslaparan completamente, y el error minimizado fuera cero. Es importante mencionar que el error depende del valor de α, el cual debe ser optimizado.
- Aproximar la función $Tanh$ mediante la función $PolyTanh$ a partir de la función $PolySigmoid$ usando la Ecuación 2.
- Entrenar una red neuronal con la base de datos MNIST con 60,000 muestras divididas en 50,000 de entrenamiento y 10,000 de prueba usando las funciones de activación $Sigmoid$ y $PolySigmoid$, así como $Tanh$ y $PolyTanh$.

- Comparar la precisión de clasificación, así como la correlación de los resultados de clasificación de la red neuronal entrenada con las funciones de activación $Sigmoid\ vs\ PolySigmoid$, y $Tanh\ vs\ PolyTanh$.

3. Resultados

En esta sección, se presentan los resultados de tres experimentos. En el primer experimento se optimiza el parámetro α de la función $PolySigmoid(\alpha x)$ para aproximar de la mejor manera la función $Sigmoid\ s(x)$.

El objetivo del experimento 1 es minimizar el error entre K puntos de muestreo sobre la función $Sigmoid$, y los correspondientes K puntos de muestreo de la función $Polysigmoid(\alpha x)$. Al ir variando el valor de α, se acercan o alejan los puntos de ambas funciones, así que el caso ideal se alcanza cuando los K puntos de $Sigmoid$ y $PolySigmoid(\alpha x)$ coinciden, en cuyo caso el error minimizado es cero. Si no se logra el caso ideal, después de un número máximo de iteraciones, se obtiene el mejor valor de α que minimiza el error y se considera valor óptimo de α.

Luego, en el experimento 2, se entrena una red neuronal con la base de datos MNIST para validar la clasificación usando la función de activación $PolySigmoid(\alpha x)$ y comparando los resultados de precisión con los de la función $Sigmoid$.

El objetivo del experimento 2 es comparar la precisión de clasificación sobre el conjunto de prueba. La base de datos MNIST contiene 60,000 imágenes de dígitos del 0 al 9 (imágenes clasificadas en 10 clases) y se divide en dos conjuntos: el de entrenamiento y el de prueba. Con los datos de entrenamiento se obtiene el error de entrenamiento ($Train\ Error$) que se usa para ajustar los parámetros de la red neuronal usando separadamente las funciones de activación: $Sigmoid$ o $PolySigmoid$ con el valor optimizado de α. En tanto que la métrica de precisión se usa para medir la capacidad de generalizar y acertar la clase correcta descrita en la base de datos MNIST sobre el conjunto de datos de prueba. Así que, cuando la clase que proporciona la red neuronal coincide con la clase correcta de los datos de entrada, la precisión es uno, y cero en caso contrario. La máxima precisión de

prueba es 100%, que se calcula dividiendo entre $10{,}000$ el número de casos correctamente clasificados.

En el experimento 3, se usa la función $PolyTanh$ y, de igual forma, se comparan los resultados de clasificación con respecto a los de la función $Tanh$. Es importante mencionar que no se pretende maximizar primordialmente la precisión en la clasificación de los datos de MNIST, sino que el objetivo es comparar los resultados de las funciones $Tanh$ y $PolyTanh$, a fin de obtener esencialmente los mismos resultados de clasificación. Esto se hará calculando el factor de correlación entre las respectivas predicciones al ejecutar 20 épocas de entrenamiento. En otras palabras, se busca comparar qué tanto se parecen las precisiones alcanzadas al reemplazar $Tanh$ con $PolyTanh$.

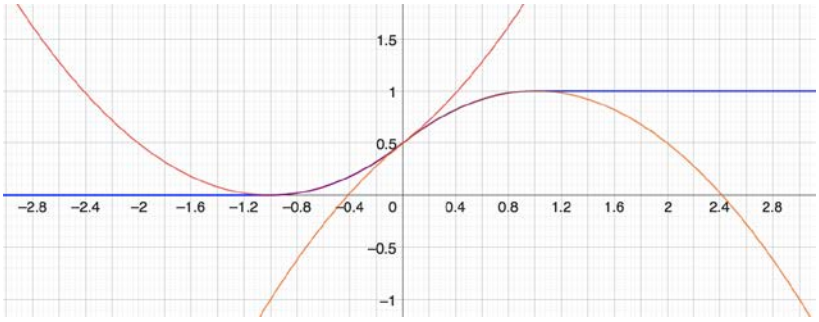
Experimento 1

La función $PolySigmoid$ está dada por [Herrera, 2024], Ecuación 8.

$$PolySigmoid(x) = \begin{cases} 0, & x \leq -1 \\ \dfrac{(x+1)^2}{2}, & -1 < x \leq 0 \\ \dfrac{(-x^2+2x+1)}{2}, & 0 < x \leq 1 \\ 1, & 1 < x \end{cases} \tag{8}$$

La cual se obtiene como caso especial de una función de activación llamada Morph [Herrera, 2024]. La Ecuación 8 consta de cuatro trozos: para $x < -1$ el valor es 0, y para $x > 1$ el valor es 1. En la parte no lineal, para $-1 < x \leq 0$ se tiene una parábola que evaluada en los extremos -1 y 0 tiene los valores de 0 y 0.5 respectivamente. Por otro lado, para $0 < x \leq 1$ se tiene otra parábola que evaluada en los extremos 0 y 1 obtiene el valor de 0.5 y 1 respectivamente, así que la función es continua, y no presenta el problema de ”salto”, como sucedía al usar la Ecuación 7. La Figura 7 ilustra estas cuatro componentes de la gráfica. En cuanto a la primera derivada, esta se denota como $PolySigmoid^{(1)}(x)$, y está dada por la Ecuación 9.

$$PolySigmoid^{(1)}(x) = \begin{cases} 0, & x \leq -1 \; y \; 1 < x \\ x+1, & -1 < x \leq 0 \\ -x+1, & 0 < x \leq 1 \end{cases} \tag{9}$$

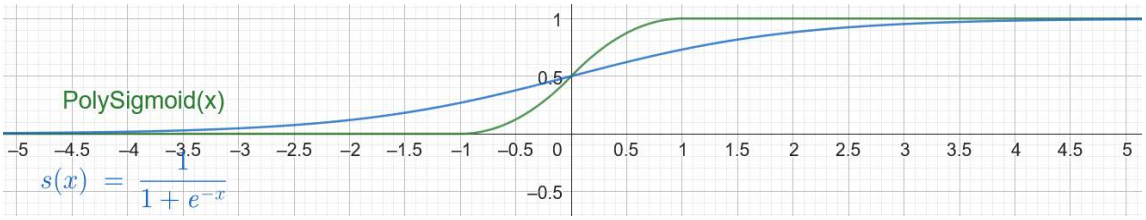


Fuente: elaboración propia

Figura 7 Parábolas $\frac{(x+1)^2}{2}$ y $\frac{(-x^2+2x+1)}{2}$ que aproximan la función Sigmoid.

Así que, en $x = -1$ está definida toda vez que por la izquierda es 0, y por la derecha también es 0. De la misma forma, en $x = 1$ por la izquierda es 0 y por la derecha también es 0.

En la Figura 8 se muestra la gráfica de la función $PolySigmoid(x)$, y de la función $Sigmoid\ s(x)$ a manera de comparación.

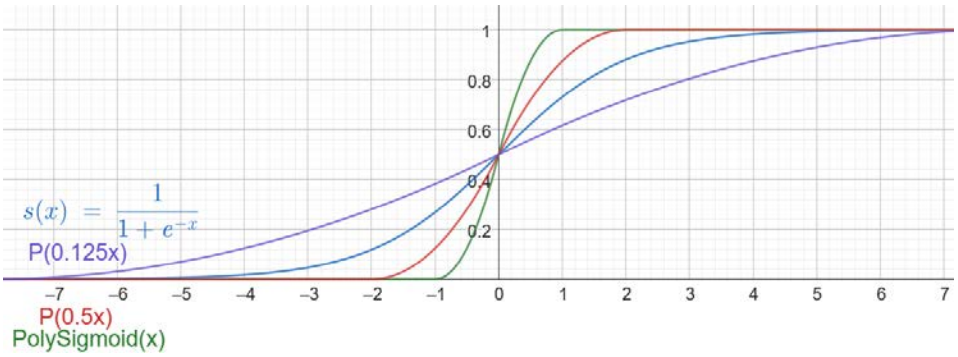


Fuente: elaboración propia

Figura 8 Gráfica de la función PolySigmoid y de la función Sigmoid.

Esencialmente, ambas funciones comparten la misma forma. Sin embargo, la función PolySigmoid alcanza el valor de 1 para $x > 1$, en tanto que la función Sigmoid es igual a 0.9933 para $x = 5$ (y es asíntota a 1). Si se introduce un factor de escalamiento α es posible "aplanarla o desaplanarla", es decir se puede usar PolySigmoid(αx) con $0 < \alpha \leq 1$ para hacer que se asemejen más. En esto radica fundamentalmente la optimización del factor α. En la Figura 9, se muestra la gráfica de la función PolySigmoid(αx) para $\alpha = 1$, 0.5, y 0.125. Dada la simetría de la función, es posible enfocarse en el plano negativo de x en la Figura 9. Obsérvese que para $\alpha = 1$ (en verde) y $\alpha = 0.5$ (en rojo) la función PolySigmoid(αx) queda por

debajo de la función Sigmoid $s(x)$, en tanto que para para $\alpha = 0.125$ (en morado), la función se "aplana" demasiado y queda arriba de $s(x)$. En [Herrera, 2024] se reporta que $\alpha = 0.25$ es un buen valor para la aproximación. En este trabajo se explorará cuál es el valor óptimo de α y para ello se usará un algoritmo genético, que determine el valor de α que provea el mejor ajuste hacia la función $s(x)$.



Fuente: elaboración propia

Figura 9 Gráfica de PolySigmoid(αx) para $\alpha = 1, 0.5\ y\ 0.125$ vs. función Sigmoid $s(x)$.

La función PolySigmoid(αx) alcanza el valor de 0 en $x = -\frac{1}{\alpha}$ y 1 en $x = \frac{1}{\alpha}$. Entonces, como se mencionó anteriormente y dada la simetría, es posible enfocarse en el intervalo $-1 < x \leq 0$. A continuación se presenta el algoritmo para determinar el valor óptimo de α.

Algoritmo para optimizar α:

- ENTRADAS: Rango de búsqueda (min, max) para los valores de α:
  1. Generar la población de P individuos de un algoritmo genético en donde cada individuo representa un valor de α inicializado en el intervalo (min, max).
  2. Evaluar cada individuo de la población P, considerando el valor α que se representa como número de punto flotante:
     - Generar una partición del intervalo $(-\frac{1}{\alpha}, 0)$ con K valores de muestreo de $s(x)$ en $x_k = -\frac{k}{K\alpha}$, donde k=1,2, ..., K.

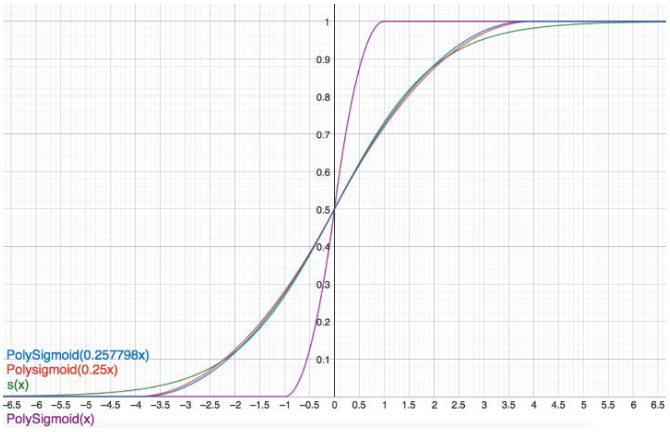
- Generar el vector de muestras $\mathrm{sk}(x_k)$ al evaluar $Sigmoid\ (x_k)$.
- Calcular el error absoluto de aproximación E, sobre las K muestras comparando el valor de PolySigmoid(αx_k) con $s(x_k)$. Es importante aclarar que, dada la simetría de la función, se limitará la aproximación al intervalo $(-\frac{1}{\alpha}, 0)$, y entonces la función Sigmoid $s(x)$ se debe aproximar con la función $\alpha\frac{(x+1)^2}{2}$. En caso de usar el intervalo $(0, \frac{1}{\alpha})$ se usaría la función $\alpha\frac{(-x^2+2x+1)}{2}$.

3. Las referencias deben ser colocadas entre paréntesis rectangulares.
4. Ordenar ascendentemente los individuos en base al error E calculado con el valor de α
5. Seleccionar P/2 individuos con menor error E. El criterio para identificar al individuo más apto es aquel con índice 0, porque tiene el menor error de aproximación E, que es la función a optimizar en base al valor de α
6. Cruzar los individuos en pares aleatorios tomados de la población con rango de individuos $(0, P/2-1)$ y crear la población de tamaño P para la siguiente generación tomando el promedio de los dos individuos padres, donde un individuo hijo es un valor real dado por $hijo = r \cdot padre1 + (1-r) \cdot padre2$, para un valor aleatorio $0 \leq r \leq 1$.
7. Mutar cada individuo con probabilidad Pmut, que significa darle otro valor aleatorio de α en el rango válido de búsqueda.
8. Repetir el paso 2 a 5 un número máximo de iteraciones MAXITER. Preservar la población de la última iteración (y no generar una nueva como al inicio)
9. Imprimir el valor óptimo de α codificado en el individuo 0.

- SALIDA: El valor optimizado de α.

El algoritmo fue implementado en Python, con rango de búsqueda $\alpha \in [0.125, 0.5]$. La población tiene $P = 20$ individuos, probabilidad de mutación $Pmut = 0.1$, $MAXITER = 1000$ y $K = 3000$ puntos de muestreo. Como resultado se obtuvo $\alpha = 0.257798$, el cual es ciertamente cercano (pero no igual) a 0.25 [Herrera, 2024].

En la Figura 10, se muestran las gráficas de las funciones $Sigmoid$, $PolySigmoid(0.25\,x)$ y $PolySigmoid(0.257798\,x)$ (Figura 10). Estas últimas aproximan suficientemente a la función $Sigmoid$. Lo que sigue es evaluar su desempeño como funciones de activación en un problema de clasificación de datos con una red neuronal.



Fuente: elaboración propia

Figura 10 $PolySigmoid(\alpha x)$ para $\alpha = 1, 0.25$ y $\alpha = 0.257798$ vs $Sigmoid\ s(x)$.

Experimento 2

Consiste en entrenar una red neuronal con la siguiente arquitectura:

- Capa 1: Lineal(input_size, hidden_size)
- Capa 2: Lineal(hidden_size, num_classes)
- Función de activation: {Sigmoid, PolySigmoid}

Donde, $input\_size$ es 784 que corresponde a los datos de MNIST, clasificados en 10 clases, es decir, $num\_classes = 10$. El número de neuronas ocultas es $hidden\_size = 128$. La implementación se hizo en Python ejecutándose sobre una GPU NVIDIA GeForce RTX 3070. El optimizador es Adam [Kingma, 2015], con métrica Precisión (Accuracy), tasa de aprendizaje $lr = 0.001$, y 20 épocas de entrenamiento. Los resultados de precisión sobre el conjunto de datos de prueba, Tabla 1, para la función $Sigmoid\ s(x)$, $PolySigmoid(x)$, $PolySigmoid(0.25x)$ y $PolySigmoid(0.257798\,x)$.

Tabla 1 Precisión con MNIST para funciones de activación Sigmoid y PolySigmoid

Época	Precisión de $s(x)$	Precisión de $Polysigmoid(\alpha x)$		
		$\alpha = 1$	$\alpha = 0.25$	$\alpha = 0.257798$
1	*92.63*	*94.46*	*92.61*	*92.47*
2	*94.23*	*95.79*	*94.29*	*94.31*
3	*95.22*	*96.68*	*95.31*	*95.29*
4	*95.93*	*96.86*	*95.96*	*95.93*
5	*96.42*	*97.37*	*96.5*	*96.54*
6	*96.66*	*97.47*	*96.67*	*96.93*
7	*96.92*	*97.36*	*97.06*	*97.04*
8	*97.18*	*97.46*	*97.22*	*97.33*
9	*97.17*	*97.41*	*97.32*	*97.44*
10	*97.35*	*97.43*	*97.48*	*97.5*
11	*97.5*	*97.54*	*97.46*	*97.72*
12	*97.59*	*97.55*	*97.58*	*97.75*
13	*97.73*	*97.5*	*97.84*	*97.84*
14	*97.79*	*97.4*	*97.77*	*97.76*
15	*97.72*	*97.73*	*97.76*	*97.89*
16	*97.71*	*97.38*	*97.67*	*97.85*
17	*97.76*	*97.52*	*97.81*	*97.86*
18	*97.84*	*97.58*	*97.79*	*97.82*
19	*97.89*	*97.43*	*97.91*	*97.97*
20	*97.83*	*97.4*	*97.77*	*97.8*

Fuente: elaboración propia

La correlación entre los valores de precisión de las columnas de la Tabla 1 se muestran en la Tabla 2.

Tabla 2 Correlación de precisiones de las funciones de activación Sigmoid y PolySigmoid

	$s(x)$	$PolySigmoid(x)$	$Polysigmoid(0.25x)$	$Polysigmoid(0.257798x)$
$s(x)$				
$Polysigmoid(x)$	*0.9476*			
$Polysigmoid(0.2x)$	*0.9977*	*0.9593*		
$Polysigmoid(0.257798x)$	***0.9988***	*0.9535*	*0.9981*	

Fuente: elaboración propia

En la tabla 2 es posible verificar que efectivamente se logra una mayor correlación (con valor de 0.9988 marcado en negritas) entre las precisiones de la red neuronal usando la función Sigmoid $s(x)$ y la función $PolySigmoid(0.257798\,x)$. La correlación entre $s(x)$ y $PolySigmoid(0.25x)$ es 0.9977, que es menor a 0.9988, y con esto se valida el resultado del experimento 1. En otras palabras, al mejorar la aproximación de la función $Sigmoid$ optimizando el parámetro $\alpha = 0.257798$ con el algoritmo genético del experimento 1, también se mejoran los resultados de clasificación, en el sentido de que son más parecidos a los obtenidos con la función

$Sigmoid$. La ventaja de usar la aproximación de $PolySigmoid(\alpha x)$ con $\alpha = 0.257798$ es que la función Sigmoid requiere la evaluación de la función exponencial, en tanto que la función $PolySigmoid(\alpha x)$ es una función a trozos aproximada con dos parábolas $\frac{(x+1)^2}{2}$ y $\frac{(-x^2+2x+1)}{2}$ en los intervalos $-1 < x \leq 0$ y $0 < x \leq 1$ respectivamente, y a pesar de ser un polinomio de $grado\ 2$, se logran los mismos resultados.

Experimento 3

En este experimento se entrena la misma red neuronal del experimento 2 pero con las funciones $Tanh$ y $PolyTanh$. La función $Tanh$ es aproximada en la Ecuación 10 con la función $PolySigmoid(\alpha x)$ con $\alpha = 0.257798$.

$$PolyTanh(x) = 2PolySigmoid(2\ \alpha\ x) - 1 \qquad (10)$$

Los valores de precisión con 20 épocas de entrenamiento, calculados sobre los datos de prueba, se muestran en la Tabla 3. Considerando los datos de la Tabla 3, el coeficiente de correlación entre las precisiones de $PolyTanh$ y $Tanh$ es de 0.9867, que ciertamente es menor que la de $PolySigmoid$ y $Sigmoid$. Sin embargo, también es una aproximación aceptable.

Tabla 3 Precisión de las funciones de activación $Tanh$ y $PolyTanh$ con $\alpha = 0.257798$.

Época	Precisión de $PolyTanh(x)$	Precisión de $Tanh(x)$
1	*92.54*	*92.63*
2	*94.30*	*94.23*
3	*95.30*	*95.22*
4	*95.87*	*95.93*
5	*96.49*	*96.42*
6	*96.86*	*96.66*
7	*97.97*	*96.92*
8	*97.34*	*97.18*
9	*97.43*	*97.17*
10	*97.46*	*97.35*
11	*97.63*	*97.50*
12	*97.67*	*97.59*
13	*97.77*	*97.73*
14	*97.74*	*97.79*
15	*97.81*	*97.72*
16	*97.81*	*97.71*
17	*97.84*	*97.76*
18	*97.94*	*97.84*
19	*97.89*	*97.89*
20	*97.80*	*97.83*

Fuente: elaboración propia

4. Discusión

De los experimentos realizados, se pueden hacer los siguientes comentarios. Ha sido posible optimizar un parámetro de escalamiento α para aproximar de la mejor manera la función $Sigmoid\ s(x)$ con una función polinomial a trozos $PolySigmoid$. La optimización de α se hizo exitosamente a través de un algoritmo genético.

La función $PolySigmoid$ usa dos parábolas (polinomios de $grado\ 2$) para la parte no lineal, y toma valores exactos de 0 para $x < 1$, y 1 para $x > 1$. Además, es continua (no presenta saltos) y existe la derivada en los puntos de conexión entre los trozos. A pesar de que $PolySigmoid$ usa polinomios de grado finito (cuadráticos) el desempeño como función de activación es equiparable a la original $Sigmoid$, la cual involucra un número infinito de términos de aproximación. Esto puede tener múltiples usos, por ejemplo, en la implementación donde se tiene hardware limitado, o inclusive para reducir el costo computacional al ejecutar múltiples evaluaciones de la función de activación en redes neuronales profundas y con muchas neuronas.

A partir de la función $PolySigmoid$, es posible obtener otras expresiones de funciones de activación con polinomios de grado finito, como es el caso de la función tangente hiperbólica. Por tanto, se abre la posibilidad de obtener otras aproximaciones para más funciones de activación. El factor de correlación cercano a 1.0 en los resultados de clasificación, soporta la idea de que es factible usar la función $PolySigmoid$ como reemplazo eficiente de la función $Sigmoid$.

5. Conclusiones

En base a los experimentos realizados se puede concluir que:

- Es posible encontrar aproximaciones de funciones de activación mediante polinomios definidos a trozos y de bajo orden. Ejemplo son las aproximaciones de las funciones de activación $Sigmoid$ y $Tanh$, usando polinomios de $grado\ 2$.
- Los resultados de clasificación al usar funciones polinomiales a trozos resultaron aceptables en los experimentos realizados, específicamente al compararlos con las funciones $Sigmoid$ y $Tanh$.

- En trabajos futuros se espera estudiar y proponer otras funciones de activación relacionadas con la función $PolySigmoid$. Por ejemplo, $Lish(x) = x\,Tanh(x)$, en donde como ya se estudió en este trabajo, $Tanh$ puede aproximarse exitosamente usando la función $PolySigmoid$.

6. Bibliografía y Referencias

[1] Agostinelli, F. Learning activation functions to improve deep neural networks. arXiv preprint arXiv:1412.6830, 2014.

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

[3] Catalbas, B., Morgül, Ö. Deep learning with ExtendeD Exponential Linear Unit (DELU). Neural Computing and Applications, No.35(30), 22705-22724, 2023.

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

[5] Davies, B. Integral Transforms and their Applications. Springer, No. 28, 2002.

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

[7] Hendrycks, D., Gimpel, K. Gaussian error linear units (GELUs). Proceedings of the Advances in Neural Information Processing Systems, Barcelona, España, 5-10, 2016.

[8] Herrera-Alcántara, O., Arellano-Balderas, S. Adaptive Morphing Activation Function for Neural Networks. Fractal Fract., No. 8(8), 444, 2024.

[9] Kingma, D., Ba, J. Adam: A Method for Stochastic Optimization. 3rd International Conference for Learning Representations, San Diego, CA, 2015.

[10] Kumar, R. APTx: better activation function than MISH, SWISH, and ReLU's variants used in deep learning. arXiv preprint arXiv:2209.06119, 2022.

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

[12] Misra, D. Mish: A Self Regularized Non-Monotonic Neural Activation Function. arXiv:1908.08681, 2019.

[13] Narayan, S. The generalized sigmoid activation function: Competitive supervised learning. Information Sciences, No. 99(1–2), 69-82, 1997.

[14] Qiumei, Z., Dan, T., Fenghua, W. Improved Convolutional Neural Network Based on Fast Exponentially Linear Unit Activation Function. IEEE Access No. 7, 151359-151367, 2019.

[15] Ramachandran, P., Zoph, B., Le, Q.V. Searching for Activation Functions. arXiv:1710.05941v2, 2017.

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

[17] Zhou, Y., Li, D., Huo, S., Kung, S. Shape autotuning activation function. Expert Syst. Appl. No. 171, 114534, 2020.