Intro a las Redes Neuronales

Modelo de Hopfield (No Supervizado)

1. Implemente el modelo de Hopfield, para reconocer una imagen de 10x10 pixeles, en blanco y negro. Plotee cuando considere necesario hacerlo, para ir viendo los resultados de su red. Implemente con libertad todas aquellas funciones auxiliares que crea convenientes para obtener un código claro, elegante en lo posible y útil.

   a) Teniendo una sola memoria almacenada, que será generada al azar, como si fuera ruido. ¿Ha notado qué pasa si, como estado inicial, le ingresa una imagen un poco parecida *(cambiando el valor de 10 pixeles)*, muy parecida *(cambiando el valor de solo 3 pixeles)* o completamente diferente *(generando todos los pixeles al azar o cambiando al menos la mitad)*? ¿Qué produce de respuesta?

   b) Teniendo almacenadas 2 memorias, generadas al azar cada una. ¿Qué salidas produce su red ahora?

   c) Extienda a p memorias almacenadas y observe los resultados. ¿La red empeoró o mejoró su capacidad de reconocimiento? Para $p = 4, p = 8, p = 20$. Pruebe cambiando solo un pixel en todos los casos de p y observe si reconoce o no.

   d) Calcular y graficar la evolución temporal de la función energía H para los casos del punto c), y para varias condiciones iniciales generadas aleatoriamente.

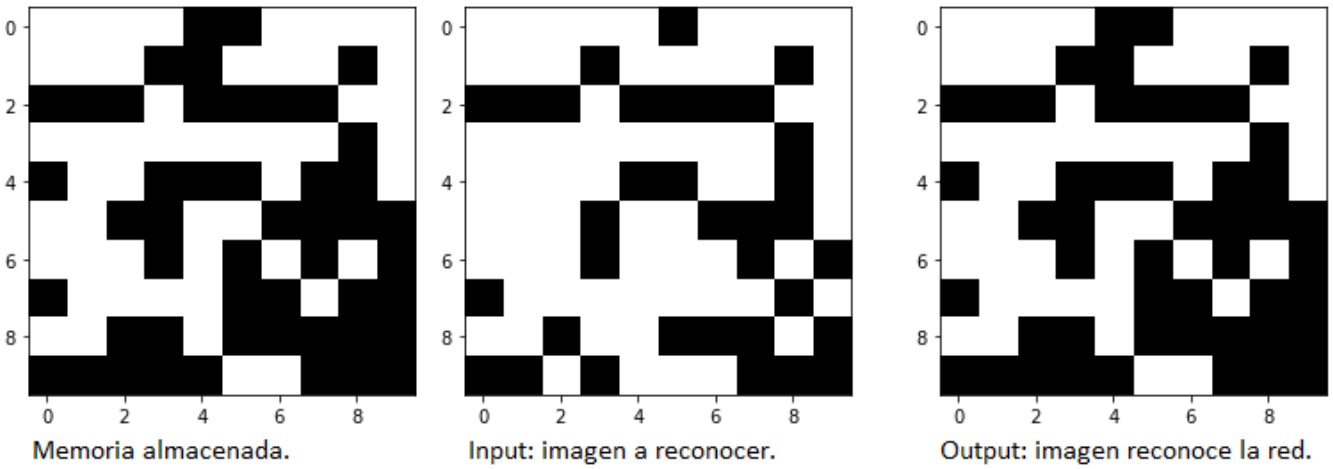

Figura 1: Hopfield de 1 memoria - Imagen en blanco y negro de 10x10 pixeles.

2. Implemente el modelo de Hopfield, para 3 memorias, de 20x20, y luego 40x40 pixeles. Genere un estado espúreo. Iniciando la dinámica de la red desde un estado cercano *(cambiando un 1 % de los pixeles)* a ese estado espúreo, calcule la función energía H y grafíquela. ¿Qué puede observar en el comportamiento de H? ¿El sistema evoluciona a alguna de las memorias? *(Responda utilizando la distancia de Hamming)*

Extensión a Hopfield: Dinámica de Glauber

3. Implemente el modelo de Hopfield, para reconocer una imagen de 20x20 pixeles, en blanco y negro. Realizar cada uno de los siguientes puntos para valores de $\beta = 10, 1, 0.3$

   a) Teniendo una sola memoria almacenada, que será generada al azar, como si fuera ruido.

   b) Teniendo almacenadas 2 memorias, generadas al azar cada una.

   c) Extienda a p memorias almacenadas y observe los resultados. Pruebe con $p = 4, 6, 8$ y 10.

   d) Graficar la función energía H en función del tiempo, y comparar.

   e) En todos los casos, grafique la distancia de Hamming entre el estado del sistema en cada instante t y cada una de las memorias. *(Tendrá tantas curvas, como memorias almacenadas).*

Perceptrones (Aprendizaje Supervisado - Clasificadores)

4. Graficar las siguientes funciones, que suelen ser usadas como funciones de activación continuas. Con $\beta = 0.3, 1, 3$.

   a) Tangente hiperbólica $g(x) = tanh(\beta x)$

   b) La función Rectificador *(RELU)* definida por:

$$g(x) = \begin{cases} x & \text{si} \quad x > 0 \\ 0 & \text{si} \quad x \leq 0 \end{cases}$$

   c) La función $f(x)$ definida por:

$$f(x) = \frac{e^{\beta x}}{1 + e^{\beta x}}$$

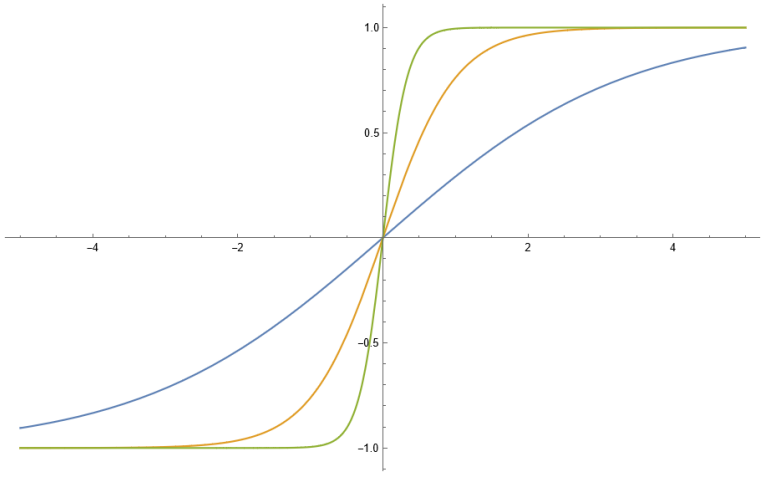

Figura 2: Función tangente hiperbólica para diferentes valores de β.

5. Desarrolle analíticamente *(en papel)* usando un Perceptrón Simple para resolver la función lógica **and** que devuelve verdadero solo y solo si ambos parámetros son verdaderos a la vez, y falso en los otros 3 casos posibles, como se lista en la tabla a continuación, dados dos parámetros p y q que tengan valores lógicos. Reemplace el True y False por 1 y -1 respectivamente. ¿Cuántas neuronas tendrá su red?

p	**q**	**and(p, q)**
$True$	$True$	$True$
$True$	$False$	$False$
$False$	$True$	$False$
$False$	$False$	$False$

6. Idem al ejercicio anterior, desarrolle analíticamente usando un Perceptrón Simple para resolver la función lógica **or** que devuelve verdadero si alguno o ambos parámetros son verdaderos, y falso solo si ambos son falsos a la vez, como se lista en la tabla a continuación, dados dos parámetros p y q que tengan valores lógicos.

p	**q**	**or(p, q)**
$True$	$True$	$True$
$True$	$False$	$True$
$False$	$True$	$True$
$False$	$False$	$False$

7. Implemente usando un Perceptrón Simple para resolver una función llamada *mayoria* que tome una cadena de 0's y 1's, por ejemplo 10001010 de 9 bits *(cualquier longitud impar de dígitos binarios es válida)*, lo cual dará un total de 512 cadenas posibles de 0s y 1s. Elegir $P = 8, 32, 64$, y determinar si tiene mayoría de 1's o no. Determine para cada P, el error de Generalización.

8. Implemente una versión equivalente al ejercicio anterior para mayoría de 0's en vez de unos.

9. Implementar un Perceptrón Simple para reconocer un dígito numérico puntual. Por ejemplo, el dígito 6. *(Clasificador Binario)* A continuación las líneas de código que deberá incluir en su programa para descargar el dataset de imagenes de números escritos a mano y digitalizados. Deberá preparar los datos descargados a fines de utilizarlos en el perceptrón. A modo información:

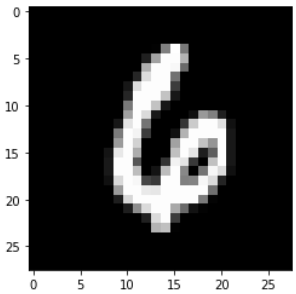

Figura 3: Ejemplo de un dígito numérico de Mnist.

```
""" MNIST
El conjunto de entrenamiento contiene 60k imagenes y el conjunto
de testeo contiene 10k imagenes.
La dimension del vector input de entrenamiento es (60000 x 28 x 28).
La dimension del vector de salida es de (60000 x 1).
Cada vector de entrada es de 28 x 28.
Cada vector de salida es de dimension 1. """

from keras.datasets import mnist
from matplotlib import pyplot

(train_X, train_y), (test_X, test_y) = mnist.load_data()

# Mostrarlo
# Por ejemplo, trae un 6 en la posición 36 del conjunto de entrenamiento.
pyplot.imshow(train_X[39], cmap=pyplot.get_cmap('gray'))
pyplot.show()
```

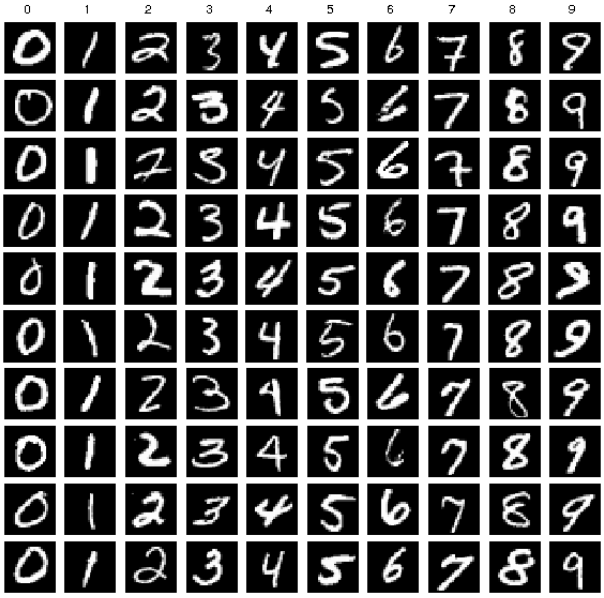

Figura 4: Ejemplo del dataset Mnist.

Implementelo utilizando:

a) El algoritmo de Aprendizaje de un Perceptrón, para dos funciones de activación diferentes. *(RELU y exponencial por ejemplo)*

b) Descenso por el Gradiente para la función de activación $g(h) = tanh(h)$

c) ¿Qué conclusiones puede obtener? ¿Considera que un Perceptrón Simple funciona adecuadamente para este problema? ¿Qué pasaría si entrena su red para el número 1, y le ingresa un 7? ¿Serviría para clasificar los 10 dígitos numéricos 0 al 9?

Referencias

[1] Rectifier *(Rectificador o función RELU)*. Wikipedia. *(Artículo)*.

[2] Lakshmi Panneerselvam, *Activation functions and their derivatives, a quick guide.* (2021) *(Artículo)*.

[3] Wikipedia, *The MNIST database (Modified National Institute of Standards and Technology database)* (1994) *(Artículo)*.