UNIVERSIDAD NACIONAL DE INGENIERÍA | Facultad de Ingeniería Industrial y de Sistemas

MAESTRÍA EN INTELIGENCIA ARTIFICIAL

Tarea 1

Curso

MIA-207 Redes Neuronales y Aprendizaje Profundo

Integrantes

- Paul Cusi H
- Cesar Vicuña H
- Adolfo Ramon P
- Johan Callomamani B
- Jairo Pinedo T

1) Hacer las operaciones de forward y backward propagation de forma manual para un MLP que tome 3 entradas (3x1), tenga 2 hidden layers de tamaño 4, y una salida de 1x1. El cálculo lo deben hacer para la siguiente data:
La data de entrada $\mathbf{X}$ y el vector deseado es el siguiente $\mathbf{y}$ son:

$$\mathbf{X_s} = \begin{bmatrix} 2.5 & 3.5 & -0.5 \\ 4.0 & -1.0 & 0.5 \\ 0.5 & 1.5 & 1.0 \\ 3.0 & 2.0 & -1.5 \end{bmatrix}$$

$$\mathbf{y_s} = \begin{bmatrix} 1.0 \\ -1.0 \\ -1.0 \\ 1.0 \end{bmatrix}$$

Desarrollo:

Conceptos fundamentales para entender el desarrollo del trabajo:

Las operaciones de forward propagation y backward propagation son fundamentales en el entrenamiento de redes neuronales.

- ✓ **Forward Propagation**: Este proceso consiste en pasar la entrada a través de la red, capa por capa, para calcular la salida. En cada capa, se aplican transformaciones matemáticas, como la multiplicación por pesos y la aplicación de funciones de activación. El resultado final se compara con la salida esperada para calcular la pérdida o error.
- ✓ **Backward Propagation**: Luego de calcular el error, la propagación hacia atrás se utiliza para actualizar los pesos de la red. Se aplica el algoritmo de retropropagación, que utiliza la regla de la cadena para calcular el gradiente de la función de pérdida con respecto a cada peso. Estos gradientes indican cómo ajustar los pesos para reducir el error en la siguiente iteración.

Ambas operaciones son cruciales para el aprendizaje de la red, permitiendo que esta mejore su rendimiento con el tiempo a través de la minimización de la pérdida.

El problema nos solicita realizar las operaciones de **forward propagation** y **backward propagation** de manera manual para una red MLP (Multi-Layer Perceptron) con:

- 3 entradas (3x1).
- 2 capas ocultas de tamaño 4.

- Una salida de 1x1.

Dado el conjunto de datos de entrada X_s y el vector deseado y_s, debemos resolverlo paso a paso.

Definición de la arquitectura:

1. **Entradas (Input layer):**
   - Número de entradas: 3
   - X_s es una matriz de 4 filas y 3 columnas, representando 4 ejemplos con 3 características cada uno.
2. **Primera capa oculta (Hidden Layer 1):**
   - Tiene 4 neuronas.
   - Necesitamos una matriz de pesos de tamaño 4×3 para conectar las entradas con las 4 neuronas.
3. **Segunda capa oculta (Hidden Layer 2):**
   - También tiene 4 neuronas.
   - La matriz de pesos será de tamaño 4×4.
4. **Capa de salida (Output Layer):**
   - Tiene una única neurona de salida (1x1), por lo que la matriz de pesos será de tamaño 1×4.

Paso 1: Inicialización de pesos y bias

Para realizar el forward y backward propagation, debemos asignar valores iniciales a los pesos y bias. Asumimos que se usan los siguientes:

- W_1 (Pesos de la primera capa): Una matriz de 4×3 (puede ser inicializada aleatoriamente o con valores pequeños).
- b_1 (Bias de la primera capa): Un vector de tamaño 4.
- W_2 (Pesos de la segunda capa): Una matriz de 4×4.
- b_2 (Bias de la segunda capa): Un vector de tamaño 4.
- W_3 (Pesos de la capa de salida): Una matriz de 1×4.
- b_3 (Bias de la capa de salida): Un escalar.

Paso 2: Forward propagation

1. **Cálculo de la activación de la primera capa oculta:**

$$Z_1 = W_1 \cdot X_s^T + b_1$$

Aplicamos la función de activación (usualmente la función sigmoide o ReLU):

$$A_1 = \text{ReLU}(Z_1)$$

2. **Cálculo de la activación de la segunda capa oculta:**

$$Z_2 = W_2 \cdot A_1 + b_2$$

Aplicamos nuevamente la función de activación:

$$A_2 = \text{ReLU}(Z_2)$$

3. **Cálculo de la salida:**

$$Z_3 = W_3 \cdot A_2 + b_3$$

Aplicamos la función de activación final (puede ser sigmoide si es clasificación):

$$A_3 = \text{Sigmoide}(Z_3)$$

Paso 3: Cálculo del error (Loss)

La función de pérdida más común es el error cuadrático medio para este tipo de problemas:

$$L = \frac{1}{m} \sum_{i=1}^{m} (y_i - A_3)^2$$

Donde m es el número de ejemplos de entrenamiento, y_i es el valor real y A_3 es la salida estimada.

Paso 4: Backward propagation

1. **Derivada respecto a Z_3 (última capa):**

$$dZ_3 = A_3 - y_s$$

2. **Derivada de los pesos y bias de la última capa:**

$$dW_3 = \frac{1}{m} dZ_3 \cdot A_2^T$$

$$db_3 = \frac{1}{m} \sum dZ_3$$

3. **Retropropagación al segundo hidden layer:**

$$dA_2 = W_3^T \cdot dZ_3$$

Derivada respecto a Z_2:

$$dZ_2 = dA_2 \cdot \mathrm{ReLU}'(Z_2)$$

Derivadas de los pesos y bias de la segunda capa:

$$dW_2 = \frac{1}{m} dZ_2 \cdot A_1^T$$

$$db_2 = \frac{1}{m} \sum dZ_2$$

4. **Retropropagación al primer hidden layer:**

$$dA_1 = W_2^T \cdot dZ_2$$

Derivada respecto a Z_1:

$$dZ_1 = dA_1 \cdot \mathrm{ReLU}'(Z_1)$$

Derivadas de los pesos y bias de la primera capa:

$$dW_1 = \frac{1}{m} dZ_1 \cdot X_s$$

$$db_1 = \frac{1}{m} \sum dZ_1$$

Paso 5: Actualización de los pesos

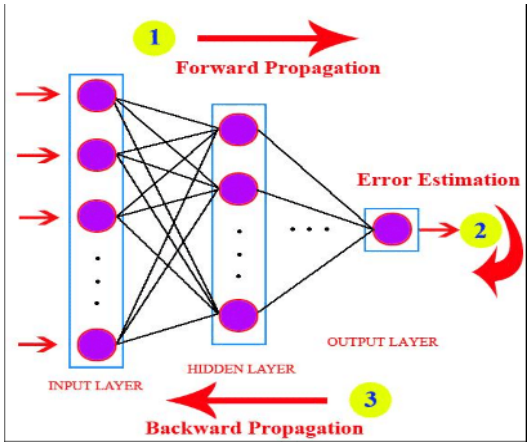
Actualizamos los pesos y bias con el gradiente descendente usando una tasa de aprendizaje α:

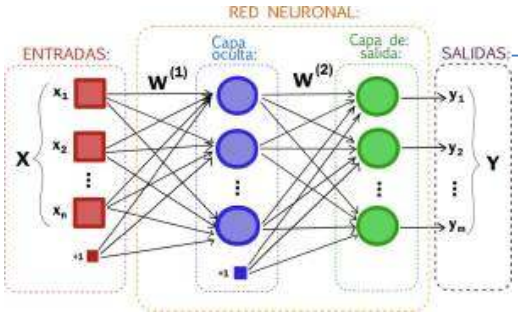
$$W = W - \alpha dW$$

$$b = b - \alpha db$$

- **03 INPUT**
- **02 CAPAS OCULTAS**

- **01 OUTPUT**
- **Iteraciones realiizadas:**





micrograd

```
Predicciones finales:
Entrada: [2.5, 3.5, -0.5], Predicción: 0.9837, Objetivo: 1.0
Entrada: [4.0, -1.0, 0.5], Predicción: -1.0210, Objetivo: -1.0
Entrada: [0.5, 1.5, 1.0], Predicción: -0.9209, Objetivo: -1.0
Entrada: [3.0, 2.0, -1.5], Predicción: 1.0031, Objetivo: 1.0
```

```
Gradientes después de backward propagation:
Gradiente del peso 1: -0.0046
Gradiente del peso 2: 0.0011
Gradiente del peso 3: -0.0006
Gradiente del peso 4: -0.0011
Gradiente del peso 5: -0.0054
Gradiente del peso 6: -0.0113
Gradiente del peso 7: -0.0008
Gradiente del peso 8: -0.0028
Gradiente del peso 9: 0.0044
Gradiente del peso 10: 0.0001
Gradiente del peso 11: 0.0036
Gradiente del peso 12: 0.0012
Gradiente del peso 13: 0.0000
Gradiente del peso 14: 0.0000
Gradiente del peso 15: 0.0000
Gradiente del peso 16: 0.0000
Gradiente del peso 17: 0.0000
Gradiente del peso 18: -0.0043
Gradiente del peso 19: -0.0069
Gradiente del peso 20: 0.0000
Gradiente del peso 21: -0.0030
Gradiente del peso 22: 0.0056
Gradiente del peso 23: -0.0040
Gradiente del peso 24: 0.0124
Gradiente del peso 25: 0.0000
Gradiente del peso 26: -0.0068
Gradiente del peso 27: 0.0000
Gradiente del peso 28: -0.0115
Gradiente del peso 29: -0.0000
Gradiente del peso 30: 0.0000
Gradiente del peso 31: -0.0045
Gradiente del peso 32: 0.0000
Gradiente del peso 33: 0.0000
Gradiente del peso 34: 0.0000
Gradiente del peso 35: 0.0000
Gradiente del peso 36: 0.0000
```

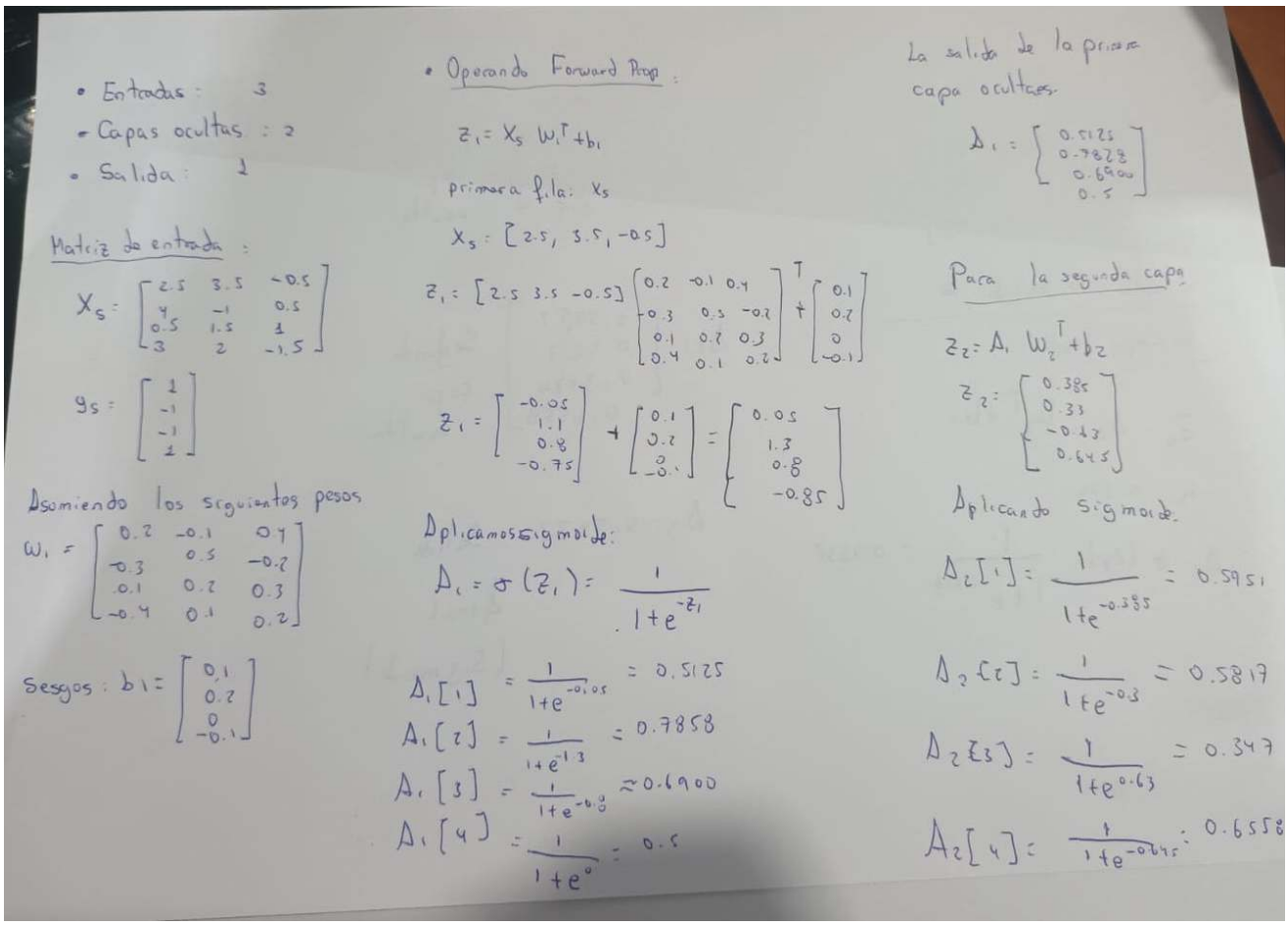
```
Pesos en la forward
Peso 1: 0.5862
Peso 2: -1.1550
Peso 3: -0.3241
Peso 4: 0.0652
Peso 5: -0.1636
Peso 6: 0.9073
Peso 7: -0.3656
Peso 8: -0.4145
Peso 9: 1.0388
Peso 10: -0.8407
Peso 11: -0.5563
Peso 12: 0.0681
Peso 13: -0.9404
Peso 14: -0.5627
Peso 15: 0.0107
Peso 16: 0.0000
Peso 17: -1.2822
Peso 18: -0.5240
Peso 19: 0.8567
Peso 20: 0.0899
Peso 21: 0.2560
Peso 22: -0.4538
Peso 23: 0.0426
Peso 24: 0.6874
Peso 25: -0.9870
Peso 26: 0.0758
Peso 27: 0.5643
Peso 28: 0.8755
Peso 29: -0.3881
Peso 30: -0.6890
Peso 31: -0.4369
Peso 32: 0.9144
Peso 33: -0.3268
Peso 34: -0.8145
Peso 35: -0.8066
Peso 36: 0.0000
```

Pytorch

```
Predicciones finales:
Entrada: [ 2.5  3.5 -0.5], Predicción: 0.9828799366950989, Objetivo: 1.0
Entrada: [ 4.  -1.   0.5], Predicción: -1.0049536228179932, Objetivo: -1.0
Entrada: [0.5 1.5 1. ], Predicción: -0.9466397504806519, Objetivo: -1.0
Entrada: [ 3.   2.  -1.5], Predicción: 0.9797263741493225, Objetivo: 1.0
```

ANEXOS:

Resolución de la primera iteración a mano



La salida de la segunda capa oculta es

$$A_2 = \begin{bmatrix} 0.5951 \\ 0.5817 \\ 0.3470 \\ 0.6558 \end{bmatrix}$$

Calculo de salida de la capa final

$$Z_3 = A_2 \cdot W_3^T + b_3$$

$$Z_3 = 0.336$$

$$A_3 = \sigma(Z_3) = \frac{1}{1 + e^{-0.336}} = 0.5832$$

Conclusión:

$$A_1 = \begin{bmatrix} 0.5125 \\ 0.7858 \\ 0.69 \\ 0.5 \end{bmatrix}$$ Primera capa oculta

$$A_2 = \begin{bmatrix} 0.5951 \\ 0.5817 \\ 0.3470 \\ 0.6558 \end{bmatrix}$$ Segunda capa oculta

$A_3 = 0.5832$: Salida final (Sigmoide)