Información General

Curso : Aprendizaje profundo
Semestre : 2026 - 1
Profesor : Gibrán Fuentes Pineda
Entrega : Septiembre 25 de 2025
Alumno : Pablo Uriel Benítez Ramírez, 418003561

Tarea: Redes densas

1. Red de unidades de umbral lineal

Programa y evalúa una red de neuronas con funciones de activación escalón unitario que aproxime la operación XNOR ($\odot$) dada por Para ello debes asignarle los pesos y sesgos adecuados a cada neurona

x_1	x_2	y
0	0	1
0	1	0
1	0	0
1	1	1

Tabla 1: Tabla de verdad XNOR

manualmente. Explica la elección de la red y los valores de los pesos y sesgos.

Procedimiento 1.1. La lógica de implementación de XNOR se sigue del siguiente diagrama: definiendo así

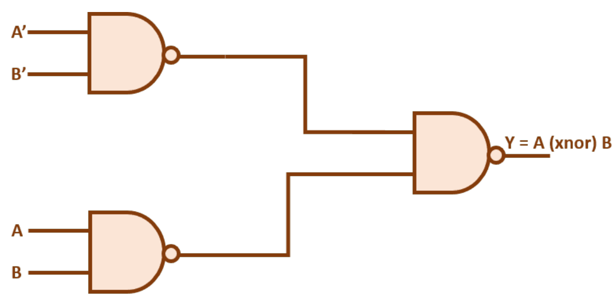

Figura 1: Diagrama XNOR

la siguiente ecuación

$$XNOR(x_1, x_2) = (x_1 \cap x_2) \;\cup\; (\neg x_1 \cap \neg x_2) \tag{1}$$

teniendo así la siguiente configuración de activación

$$\text{escalon}(z) = 1 \iff z \geq 0, \tag{2}$$

y es 0 si no. Se consideran dos entradas, dos neuronas ocultas y una salida, siguiendo la Figura (1) la capa oculta calula $(x_1 \cap x_2)$ y $(\neg x_1 \cap \neg x_2)$, luego en la salida se hace la $\cup$. Para ello es necesario que la ecuación

1. Neurona $(x_1 \cap x_2)$

$$\begin{aligned} z_1 &= x_1 + x_2 - 1.5 \\ h_1 &= \text{escalon}(z_1) \end{aligned}$$

para cualquier valor $x_1, x_2 \in \{0, 1\}$, z_1 es positivo unicamente cuando $x_1 = x_2 = 1$.

$$\implies \text{pesos: } [1, 1]$$
$$\text{sesgo:-1.5}$$

2. Neurona $(\neg x_1 \cap \neg x_2)$

$$
\begin{aligned}
z_2 &= -(x_1 + x_2) + 0.5 \\
h_2 &= \text{escalon}(z_2)
\end{aligned}
$$

para cualquier valor $x_1, x_2 \in \{0, 1\}$, z_2 es positivo unicamente cuando $x_1 = x_2 = 0$.

$$
\begin{aligned}
\implies &\text{pesos: } [-1, -1] \\
&\text{sesgo: } 0.5
\end{aligned}
$$

3. Capa de salida $\cup$

$$
\begin{aligned}
z &= h_1 + h_2 - 0.5 \\
y &= \text{escalon}(z)
\end{aligned}
$$

para cualquier valor $h_1, h_2 \in \{0, 1\}$, z es positivo unicamente cuando h_1 o h_2 es 1.

$$
\begin{aligned}
\implies &\text{pesos: } [1, 1] \\
&\text{sesgo:-0.5}
\end{aligned}
$$

Demostración 1.2.

$$
\begin{aligned}
x = (0,0) &: h_1 = 0,\ h_2 = 1 \to y = \text{escalon}(1 - 0.5) = 1 \\
x = (0,1) &: h_1 = 0,\ h_2 = 0 \to y = \text{escalon}(0 - 0.5) = 0 \\
x = (1,0) &: h_1 = 0,\ h_2 = 0 \to y = \text{escalon}(0 - 0.5) = 0 \\
x = (1,1) &: h_1 = 1,\ h_2 = 0 \to y = \text{escalon}(1 - 0.5) = 1
\end{aligned}
$$

Procedimiento 1.3. Otra forma de realizarlo es mediante matrices, del mismo modo que el anterior. La red tendría dos neuronas conectadas a las entradas que realizan las operaciones $(x_1 \cap x_2)$ con $w_{11}^{\{1\}} = 1$, $w_{12}^{\{1\}} = 1$ y $b_1^{\{1\}} = -1.5$; y $(\neg x_1 \cap \neg x_2)$ con $w_{21}^{\{1\}} = -1$, $w_{22}^{\{1\}} = -1$ y $b_1^{\{1\}} = 0.5$, teniendo así

$$
W_1 = \begin{pmatrix} 1 & -1 \\ 1 & -1 \end{pmatrix} \qquad\qquad b_1 = \begin{pmatrix} -1.5 \\ 0.5 \end{pmatrix} \tag{3}
$$

La salida de estas neuronas se conecta a una tercera neurona que realiza la operación $\cup$ con $w_{11}^{\{2\}} = 1$, $w_{12}^{\{2\}} = 1$ y $b_1^{\{2\}} = -0.5$, teniendo así

$$
W_1 = \begin{pmatrix} 1 \\ 1 \end{pmatrix} \qquad\qquad b_1 = \begin{pmatrix} -0.5 \end{pmatrix}. \tag{4}
$$

Obteniendo así

```
W_1 = [[ 1 -1][ 1 -1]], b_1 = [-1.5  0.5]
W_2 = [[1][1]], b_2 = [-0.5]
-----------------------------
x_1  x_2  y y_hat
-----------------------------
0.0 0.0 1.0 1.0
0.0 1.0 0.0 0.0
1.0 0.0 0.0 0.0
1.0 1.0 1.0 1.0
```

Código 1.4. *Nombre de archivo:* `Red de unidades de umbral lineal.ipynb`

2. Retropropagación en red densa

Programa el algoritmo de retropopagación usando `numpy` para una tarea de clasificación binaria presuponiendo una red densa con tres capas ocultas y una conexión residual que va de la salida de la primera capa oculta a la salida de la tercera capa oculta. Las neuronas de las capas ocultas cuentan con una función de activación $ReLU(z) = \text{máx}(0, z)$. Por su parte, la capa de salida está compuesta por una sola neurona sigmoide $\sigma(z) = \frac{1}{1+e^{-z}}$. Para el entrenamiento minimiza el promedio de la función de pérdida de entropía cruzada binaria:

$$ECB(\mathbf{y},\hat{\mathbf{y}}) = -\frac{1}{n}\sum_{i=1}^{n}\left[y^{(i)}\log\left(\hat{y}^{(i)}\right) + \left(1 - y^{(i)}\right)\log\left(1 - y^{(i)}\right)\right] \tag{5}$$

- Entrena y evalúa la red mediante descenso por gradiente y el algoritmo de retropropagación de errores en algún problema de clasificación no lineal.

Procedimiento 2.1. La estructura de la red densa es requerida con tres capas ocultas, con la función de activación $ReLU$. Para facilidad usé modifique el notebook disponible: `1b_retropropagacion.ipynb`.

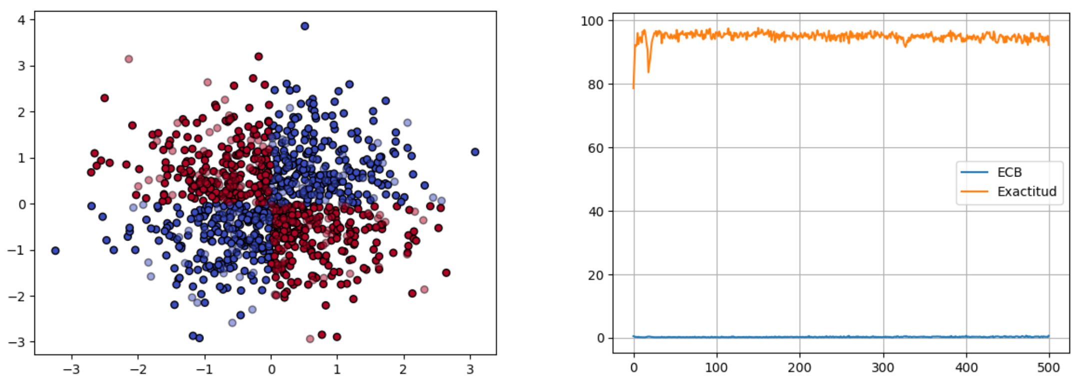

Figura 2: Visualización de los datos utilizados (izq). Entrenamiento: 94.5 % de desempeño (der)

- Describe las fórmulas y reglas de actualización de los pesos y sesgos de cada capa.

Procedimiento 2.2. Las fórmulas y reglas de actualización se incluyen en el código adjunto.

Se define la función que propaga hacia adelante una entrada $\mathbf{X} \in \mathbb{R}^{m\times d}$.

Como la red está compuesta de 4 capas densas (3 oculta y 1 de salida), tenemos 4 matrices de pesos con sus correspondientes vectores de sesgos

$$\{\mathbf{W}^{\{1\}} \in \mathbb{R}^{d\times l}, \mathbf{b}^{\{1\}} \in \mathbb{R}^{l\times 1}\}$$
$$\{\mathbf{W}^{\{2\}} \in \mathbb{R}^{l\times k}, \mathbf{b}^{\{2\}} \in \mathbb{R}^{k\times 1}\}$$
$$\{\mathbf{W}^{\{3\}} \in \mathbb{R}^{l\times k}, \mathbf{b}^{\{3\}} \in \mathbb{R}^{k\times 1}\}$$
$$\{\mathbf{W}^{\{4\}} \in \mathbb{R}^{l\times k}, \mathbf{b}^{\{4\}} \in \mathbb{R}^{k\times 1}\}$$

de las capas ocultas y la capa de salida respectivamente. Así, podemos llevar a cabo la propagación hacia adelante en esta red de la siguiente manera:

$$
\begin{aligned}
\mathbf{A}^{\{1\}} &= \mathbf{X} \\
\mathbf{Z}^{\{2\}} &= \mathbf{A}^{\{1\}} \cdot \mathbf{W}^{\{1\}} + \mathbf{b}^{\{1\}} \\
\mathbf{A}^{\{2\}} &= ReLU(\mathbf{Z}^{\{2\}}) \\
\mathbf{Z}^{\{3\}} &= \mathbf{A}^{\{2\}} \cdot \mathbf{W}^{\{2\}} + \mathbf{b}^{\{2\}} \\
\mathbf{A}^{\{3\}} &= ReLU(\mathbf{Z}^{\{3\}}) \\
\mathbf{Z}^{\{4\}} &= \mathbf{A}^{\{3\}} \cdot \mathbf{W}^{\{3\}} + \mathbf{b}^{\{3\}} + \mathbf{A}^{\{2\}} \\
\mathbf{A}^{\{4\}} &= ReLU(\mathbf{Z}^{\{4\}}) \\
\mathbf{Z}^{\{5\}} &= \mathbf{A}^{\{4\}} \cdot \mathbf{W}^{\{4\}} + \mathbf{b}^{\{4\}} \\
\mathbf{A}^{\{5\}} &= \sigma(\mathbf{Z}^{\{5\}}) \\
\hat{\mathbf{y}} &= \mathbf{A}^{\{5\}}
\end{aligned}
$$

Se define la función para entrenar nuestra red neuronal usando descenso por gradiente. Para calcular el gradiente de la función de pérdida respecto a los pesos y sesgos en cada capa empleamos el algoritmo de retropropagación. Para este caso, serían las siguientes expresiones.

Dado

$$
\begin{aligned}
Z_2 &= X \cdot W_1 + b_1, & A_2 &= ReLU(Z_2), \\
Z_3 &= A_2 \cdot W_2 + b_2, & A_3 &= ReLU(Z_3), \\
Z_4 &= A_3 \cdot W_3 + b_3 + A_2, & A_4 &= ReLU(Z_4), \\
Z_5 &= A_4 \cdot W_4 + b_4, & \hat{y} &= \sigma(Z_5),
\end{aligned}
$$

Sea

$$\boldsymbol{\delta}^{\{5\}} = \hat{\mathbf{y}}^{(i)} - \mathbf{y}^{(i)}$$

Capa de salida (W_4, b_4):

$$
\begin{aligned}
\frac{\partial ECB(\mathbf{y}, \hat{\mathbf{y}})}{\partial \mathbf{W}^{\{4\}}} &= \mathbf{A}^{\{4\}\top} \cdot \boldsymbol{\delta}^{\{5\}} \\
\frac{\partial ECB(\mathbf{y}, \hat{\mathbf{y}})}{\partial \mathbf{b}^{\{4\}}} &= \sum_{j=1}^{k} \boldsymbol{\delta}^{\{5\}}_{j,:} \\
\boldsymbol{\delta}^{\{4\}} &= (\boldsymbol{\delta}^{\{5\}} \cdot \mathbf{W}^{\{4\}\top}) \odot \frac{\partial \mathbf{A}^{\{4\}}}{\partial \mathbf{Z}^{\{4\}}}
\end{aligned}
$$

Tercera capa oculta (W_3, b_3):

$$
\begin{aligned}
\frac{\partial ECB(\mathbf{y}, \hat{\mathbf{y}})}{\partial \mathbf{W}^{\{3\}}} &= \mathbf{A}^{\{3\}\top} \cdot \boldsymbol{\delta}^{\{4\}} \\
\frac{\partial ECB(\mathbf{y}, \hat{\mathbf{y}})}{\partial \mathbf{b}^{\{3\}}} &= \sum_{j=1}^{k} \boldsymbol{\delta}^{\{4\}}_{j,:} \\
\boldsymbol{\delta}^{\{3\}} &= (\boldsymbol{\delta}^{\{4\}} \cdot \mathbf{W}^{\{3\}\top}) \odot \frac{\partial \mathbf{A}^{\{3\}}}{\partial \mathbf{Z}^{\{3\}}}
\end{aligned}
$$

segunda capa oculta (W_2, b_2):

$$
\begin{aligned}
\frac{\partial ECB(\mathbf{y}, \hat{\mathbf{y}})}{\partial \mathbf{W}^{\{2\}}} &= \mathbf{A}^{\{2\}\top} \cdot \boldsymbol{\delta}^{\{3\}} \\
\frac{\partial ECB(\mathbf{y}, \hat{\mathbf{y}})}{\partial \mathbf{b}^{\{2\}}} &= \sum_{j=1}^{k} \boldsymbol{\delta}^{\{3\}}_{j,:}
\end{aligned}
$$

$$
\boldsymbol{\delta}^{\{2\}} = (\boldsymbol{\delta}^{\{3\}} \cdot \mathbf{W}^{\{2\}\top} + \delta_4) \odot \frac{\partial \mathbf{A}^{\{2\}}}{\partial \mathbf{Z}^{\{2\}}}
$$

primera capa oculta (W_1, b_1):

$$
\begin{aligned}
\frac{\partial ECB(\mathbf{y}, \hat{\mathbf{y}})}{\partial \mathbf{W}^{\{1\}}} &= \mathbf{X}^{\{1\}\top} \cdot \boldsymbol{\delta}^{\{2\}} \\
\frac{\partial ECB(\mathbf{y}, \hat{\mathbf{y}})}{\partial \mathbf{b}^{\{1\}}} &= \sum_{j=1}^{k} \boldsymbol{\delta}^{\{2\}}_{j,:}
\end{aligned}
$$

- Compara las fórmulas y el comportamiento en el entrenamiento (incluyendo las normas de los gradientes por capa) de dicha red con otra igual pero sin la conexión residual y usando una función de activación sigmoide en las capas ocultas.

Procedimiento 2.3. Las fórmulas y reglas de actualización se incluyen en el código adjunto.

Para dicho cambio se quita la conexión residual de $\mathbf{Z}^{\{4\}}$ y cambiar $ReLU$ por $\sigma(z)$. Se define la función que propaga hacia adelante una entrada $\mathbf{X} \in \mathbb{R}^{m \times d}$. La propagación hacia adelante en esta red queda de la siguiente manera:

$$
\begin{aligned}
\mathbf{A}^{\{1\}} &= \mathbf{X} \\
\mathbf{Z}^{\{2\}} &= \mathbf{A}^{\{1\}} \cdot \mathbf{W}^{\{1\}} + \mathbf{b}^{\{1\}} \\
\mathbf{A}^{\{2\}} &= \sigma(\mathbf{Z}^{\{2\}}) \\
\mathbf{Z}^{\{3\}} &= \mathbf{A}^{\{2\}} \cdot \mathbf{W}^{\{2\}} + \mathbf{b}^{\{2\}} \\
\mathbf{A}^{\{3\}} &= \sigma(\mathbf{Z}^{\{3\}}) \\
\mathbf{Z}^{\{4\}} &= \mathbf{A}^{\{3\}} \cdot \mathbf{W}^{\{3\}} + \mathbf{b}^{\{3\}} \\
\mathbf{A}^{\{4\}} &= \sigma(\mathbf{Z}^{\{4\}}) \\
\mathbf{Z}^{\{5\}} &= \mathbf{A}^{\{4\}} \cdot \mathbf{W}^{\{4\}} + \mathbf{b}^{\{4\}} \\
\mathbf{A}^{\{5\}} &= \sigma(\mathbf{Z}^{\{5\}}) \\
\hat{\mathbf{y}} &= \mathbf{A}^{\{5\}}
\end{aligned}
$$

Se define la función para entrenar nuestra red neuronal usando descenso por gradiente. Para calcular el gradiente de la función de pérdida respecto a los pesos y sesgos en cada capa empleamos el algoritmo de retropropagación. Para este caso, serían las siguientes expresiones.

Dado

$$
\begin{aligned}
Z_2 &= X \cdot W_1 + b_1, & A_2 &= \sigma(Z_2), \\
Z_3 &= A_2 \cdot W_2 + b_2, & A_3 &= \sigma(Z_3), \\
Z_4 &= A_3 \cdot W_3 + b_3, & A_4 &= \sigma(Z_4), \\
Z_5 &= A_4 \cdot W_4 + b_4, & \hat{y} &= \sigma(Z_5),
\end{aligned}
$$

Sea

$$\boldsymbol{\delta}^{\{5\}} = \hat{\mathbf{y}}^{(i)} - \mathbf{y}^{(i)}$$

Capa de salida (W_4, b_4):

$$\frac{\partial ECB(\mathbf{y}, \hat{\mathbf{y}})}{\partial \mathbf{W}^{\{4\}}} = \mathbf{A}^{\{4\}\top} \cdot \boldsymbol{\delta}^{\{5\}}$$
$$\frac{\partial ECB(\mathbf{y}, \hat{\mathbf{y}})}{\partial \mathbf{b}^{\{4\}}} = \sum_{j=1}^{k} \boldsymbol{\delta}_{j,:}^{\{5\}}$$
$$\boldsymbol{\delta}^{\{4\}} = (\boldsymbol{\delta}^{\{5\}} \cdot \mathbf{W}^{\{4\}\top}) \odot \frac{\partial \mathbf{A}^{\{4\}}}{\partial \mathbf{Z}^{\{4\}}}$$

Tercera capa oculta (W_3, b_3):

$$\frac{\partial ECB(\mathbf{y}, \hat{\mathbf{y}})}{\partial \mathbf{W}^{\{3\}}} = \mathbf{A}^{\{3\}\top} \cdot \boldsymbol{\delta}^{\{4\}}$$
$$\frac{\partial ECB(\mathbf{y}, \hat{\mathbf{y}})}{\partial \mathbf{b}^{\{3\}}} = \sum_{j=1}^{k} \boldsymbol{\delta}_{j,:}^{\{4\}}$$
$$\boldsymbol{\delta}^{\{3\}} = (\boldsymbol{\delta}^{\{4\}} \cdot \mathbf{W}^{\{3\}\top}) \odot \frac{\partial \mathbf{A}^{\{3\}}}{\partial \mathbf{Z}^{\{3\}}}$$

segunda capa oculta (W_2, b_2):

$$\frac{\partial ECB(\mathbf{y}, \hat{\mathbf{y}})}{\partial \mathbf{W}^{\{2\}}} = \mathbf{A}^{\{2\}\top} \cdot \boldsymbol{\delta}^{\{3\}}$$
$$\frac{\partial ECB(\mathbf{y}, \hat{\mathbf{y}})}{\partial \mathbf{b}^{\{2\}}} = \sum_{j=1}^{k} \boldsymbol{\delta}_{j,:}^{\{3\}}$$
$$\boldsymbol{\delta}^{\{2\}} = (\boldsymbol{\delta}^{\{3\}} \cdot \mathbf{W}^{\{2\}\top}) \odot \frac{\partial \mathbf{A}^{\{2\}}}{\partial \mathbf{Z}^{\{2\}}}$$

primera capa oculta (W_1, b_1):

$$\frac{\partial ECB(\mathbf{y}, \hat{\mathbf{y}})}{\partial \mathbf{W}^{\{1\}}} = \mathbf{X}^{\{1\}\top} \cdot \boldsymbol{\delta}^{\{2\}}$$
$$\frac{\partial ECB(\mathbf{y}, \hat{\mathbf{y}})}{\partial \mathbf{b}^{\{1\}}} = \sum_{j=1}^{k} \boldsymbol{\delta}_{j,:}^{\{2\}}$$

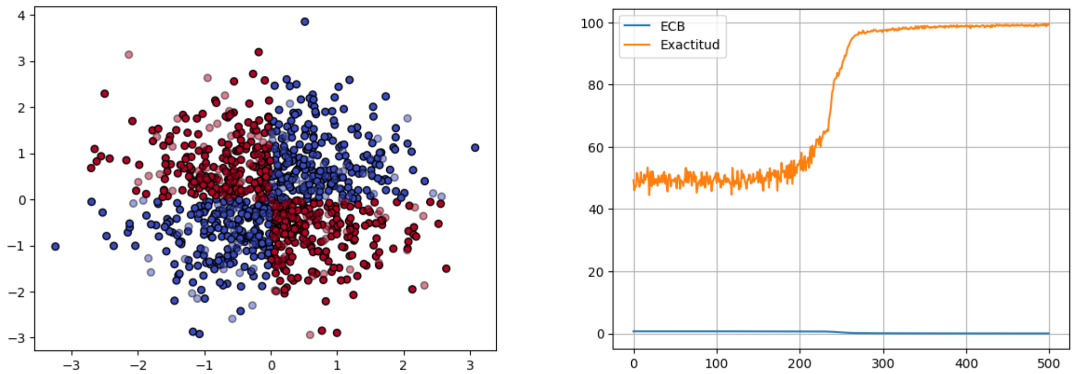

Figura 3: Visualización de los datos utilizados (izq). Entrenamiento: 97.5 % de desempeño (der)

- Estudia el efcto en el entrenamiento de no usar el promedio en la función (5) esto es

$$ECB(\mathbf{y},\hat{\mathbf{y}}) = -\sum_{i=1}^{n}\left[y^{(i)}\log\left(\hat{y}^{(i)}\right) + \left(1-y^{(i)}\right)\log\left(1-y^{(i)}\right)\right]$$

  Procedimiento 2.4. Solo cambié la definición del ECB y lo corrí en ambos casos con $ReLu$ y con sigmoide. Se hizo con los mismo parámetros de α que en los anteriores, los resultados son los siguientes

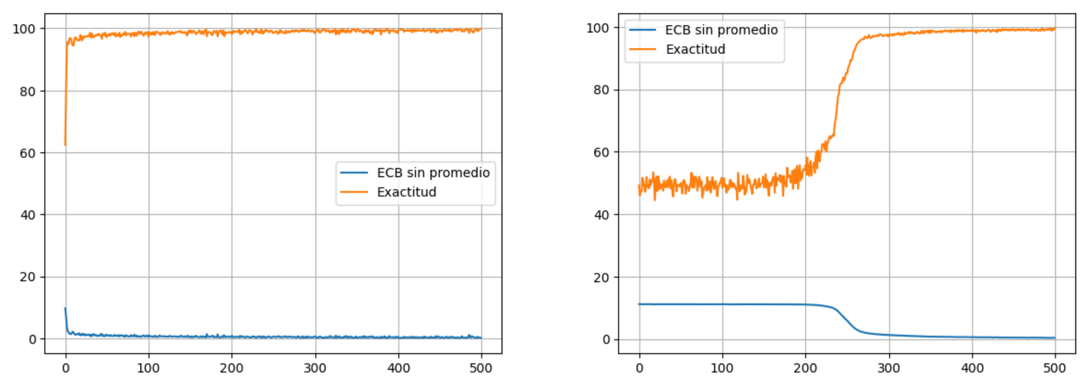

Figura 4: Visualización $ReLU$ sin promedio (izq) . Visualización σ sin promedio(der)

Código 2.5. *Nombre de archivo:* `Retropropagación en red densa.ipynb`

3. Diferenciación automática

Considera la siguiente función:

$$f(x_1, x_2, x_3, w_1, w_2, w_3, w_4) = x_3 \cdot tanh\{w_4 \cdot \sigma(w_3 \cdot [ReLU(w_1 \cdot x_1) + \sigma(w_2 \cdot x_2)])\}$$

- Calcula el paso hacia adelante y el paso hacia atrás, presentando los valores intermedios para $x_1 = -1.5,\ x_2 = 3.1,\ x_3 = 7.3,\ w_1 = 4.5,\ w_2 = -2.7,\ w_3 = 0.2$ y $w_4 = -5$. (α)
- Dibuja el gráfico de cómputo de la diferenciación automática. (β)
- Describe las expresiones correspondientes al paso hacia atrás. (δ)

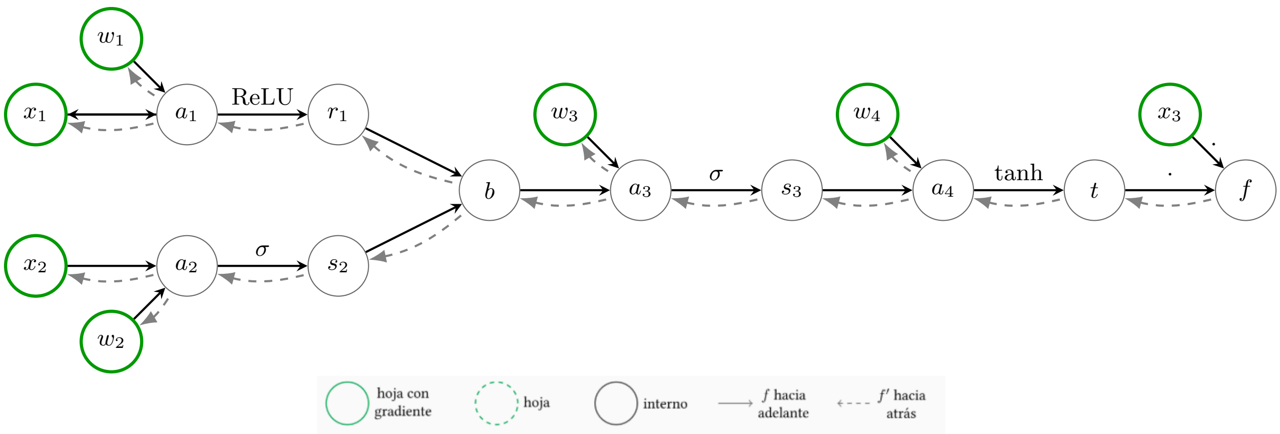

Figura 5: Gráfico de cómputo de diferenciación automática (β)

Procedimiento 3.1. Se definen las hojas del diagrama que será útil para el paso hacia adelante y el paso hacia atrás dado lo siguiente

$$f = \underbrace{x_3 \cdot \underbrace{tanh\{\underbrace{w_4 \cdot \underbrace{\sigma(\underbrace{w_3 \cdot [\underbrace{\underbrace{ReLU(\underbrace{w_1 \cdot x_1}_{a_1})}_{r_1} + \underbrace{\sigma(\underbrace{w_2 \cdot x_2}_{a_2})}_{s_2}}_{b}]}_{a_3})}_{s_3}}_{a_4}\}}_{t}}_{f} \tag{6}$$

Definición 3.1. Dada la función es necesario definir las activaciones

- $$ReLU(z) = \text{máx}(0, z)$$

  que omite los valores negativos, su derivada es

  $$ReLU'(z) = \begin{cases} 1 & \text{si} \quad z > 0 \\ 0 & \text{si} \quad z \leq 0 \end{cases}$$

- $$\sigma(z) = \frac{1}{1 + e^{-z}}$$

  que regresa un valor $v_i \in [0, 1]$, su derivada es

  $$\sigma'(z) = \sigma(z) \cdot (1 - \sigma(z)),$$

- $$\tanh(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}} = \frac{\sinh(z)}{\cosh(z)}$$

  que regresa un valor $v_j \in [-1, 1]$, su derivada es

  $$\tanh'(z) = 1 - \tanh^2(z).$$

Los datos que se tienen son los siguientes: $x_1 = -1.5,\ x_2 = 3.1,\ x_3 = 7.3,\ w_1 = 4.5,\ w_2 = -2.7,\ w_3 = 0.2,\ w_4 = -5.$

(α) El paso hacia adelante (siguiendo los nodos definidos en la Ecuación 6) se calcula de adentro hacia afuera, guardando así cada resultado intermedio

1. $a_1 = w_1 \cdot x_1 = 4.5 \cdot (-1.5) = -6.75,$
2. $r_1 = ReLU(a_1) = ReLU(-6.75) = 0,$
3. $a_2 = w_2 \cdot x_2 = -2,7 \cdot 3.1 = -8.37,$
4. $s_2 = \sigma(a_2) = \sigma(-8.37) \approx 0.00023,$
5. $b = r_1 + s_2 = 0 + 0.00023 = 0.00023,$
6. $a_3 = w_3 \cdot b = 0.2 \cdot 0.00023 = 0.000046,$
7. $s_3 = \sigma(a_3) = \sigma(0.000046) \approx 0.5000115,$
8. $a_4 = w_4 \cdot s_3 = -5 \cdot 0.5000115 = -2.5000579,$
9. $t = \tanh(a_4) = \tanh(-2.5000579) \approx -0.9866,$
10. $f = x_3 \cdot t = 7.3 \cdot -0.9866 \approx -7.202$

(δ) El paso hacia atrás (siguiendo los nodos definidos en la Ecuación 6 y las derivadas definidas en la Definición 3.1) se hace del final al inicio, partiendo de que $f' = \frac{\partial f}{\partial f} = 1$

1. $f = x_3 \cdot t$
   - $t' = f' \cdot \frac{\partial f}{\partial t} = x_3 = 7.3$
   - $x_3' = \frac{\partial f}{\partial x_3} = t \approx -0.9866$
2. $t = \tanh(a_4)$
   - $a_4' = t' \cdot \tanh'(a_4) = t' \cdot (1 - t^2) = 7.3 \cdot (1 - (-0.9866)^2) \approx 7.3 \cdot 0.0266 \approx 0.1943$
3. $a_4 = w_4 \cdot s_3$
   - $w_4' = a_4' \cdot s_3 \approx 0.1943 \cdot 0.5000115 \approx 0.0971$
   - $s_3' = a_4' \cdot w_4 \approx 0.1943 \cdot (-5) \approx -0.9715$
4. $s_3 = \sigma(a_3)$
   - $a_3' = s_3' \cdot \sigma'(a_3) = s_3' \cdot \sigma(a_3) \cdot (1 - \sigma(a_3)) = -0.9715 \cdot (0.5000115) \cdot (0.4999885) \approx -0.2428$
5. $a_3 = w_3 \cdot b$
   - $w_3' = a_3' \cdot b \approx -0.2428 \cdot 0.00023 \approx -0.0000558$
   - $b' = a_3' \cdot w_3 \approx -0.2428 \cdot 0.2 \approx -0.04856$
6. $b = r_1 + s_2$
   - $r_1' = b' \approx -0.04856$
   - $s_2' = b' \approx -0.04856$
7. $s_2 = \sigma(a_2)$
   - $a_2' = s_2' \cdot \sigma'(a_2) = s_2' \cdot \sigma(a_2) \cdot (1 - \sigma(a_2)) = -0.04856 \cdot (0.00023) \cdot (0.99977) \approx -0.0000111$
8. $r_1 = ReLU(a_1)$
   - $a_1' = r_1' \cdot ReLU'(-6.75) = -0.04856 \cdot 0 = 0$
9. $a_2 = w_2 \cdot x_2$
   - $w_2' = a_2' \cdot x_2 \approx -0.0000111 \cdot 3.1 \approx -0.00003441$
   - $x_2' = a_2' \cdot w_2 \approx -0.0000111 \cdot (-2.7) \approx 0.00002997$
10. $a_1 = w_1 \cdot x_1$
    - $w_1' = a_1' \cdot x_1 = 0$
    - $x_1' = a_1' \cdot w_1 = 0$

Resumen:

- $\frac{\partial f}{\partial x_1} \approx 0$
- $\frac{\partial f}{\partial x_2} \approx 0.00002997$
- $\frac{\partial f}{\partial x_3} \approx -0.9866$
- $\frac{\partial f}{\partial w_1} \approx 0$
- $\frac{\partial f}{\partial w_2} \approx -0.00003441$
- $\frac{\partial f}{\partial w_3} \approx -0.0000558$
- $\frac{\partial f}{\partial w_4} \approx 0.0971$

4. Red densa con PyTorch

A partir de la libreta https://github.com/gibranfp/CursoAprendizajeProfundo/blob/2026-1/notebooks/1c_redes_densas.ipynb, evalúa tres distintas configuraciones de redes densas que incluyan lo siguiente:

- Más capas ocultas.
- Distintas funciones de activación de las capas ocultas.
- Técnicas de regularización.
- Capas de normalización.

Procedimiento 4.1. A continuación se presentan las especificaciones y resultados de cada red densa.

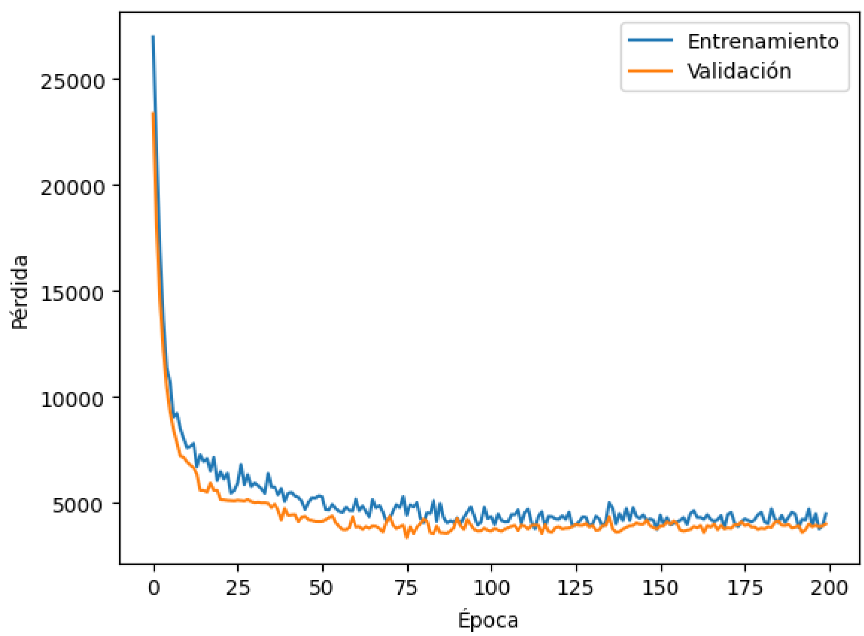

Figura 6: Configuración 1
Más capas ocultas: [32, 16, 8, 2]
Activación: ReLU y Sigmoide en ocultas
Regularización: Dropout 0.2
Normalización: BatchNorm después de cada capa oculta

Código 4.2. *Nombre de archivo:* `Red densa con PyTorch.ipynb`

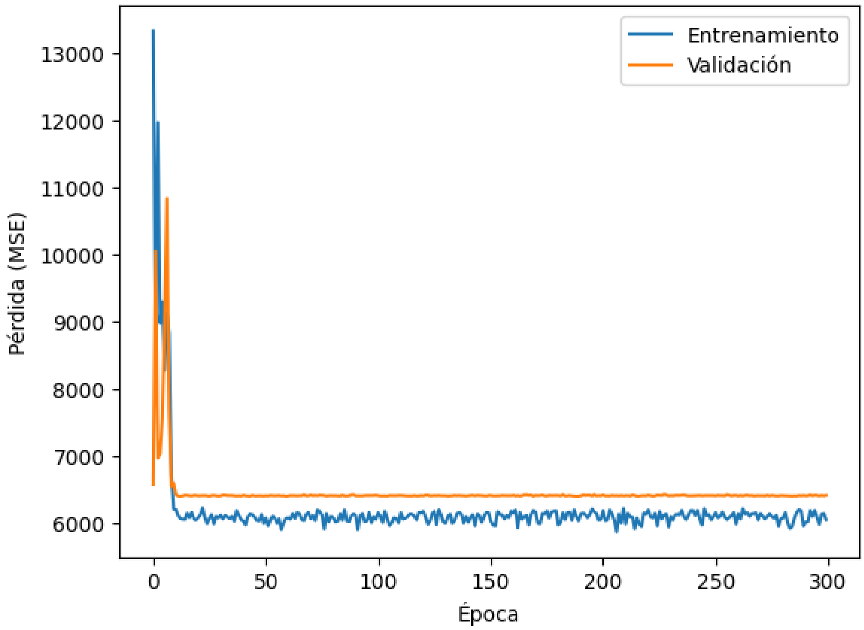

Figura 7: Configuración 2
Más capas ocultas: [64, 128, 64, 32, 16]
Activación: ELU, LeakyReLU, GELU, SiLU, ReLU
Regularización: Dropout 0.1, 0.2, 0.3, 0.3, 0.2
Normalización: GroupNorm después de cada capa oculta

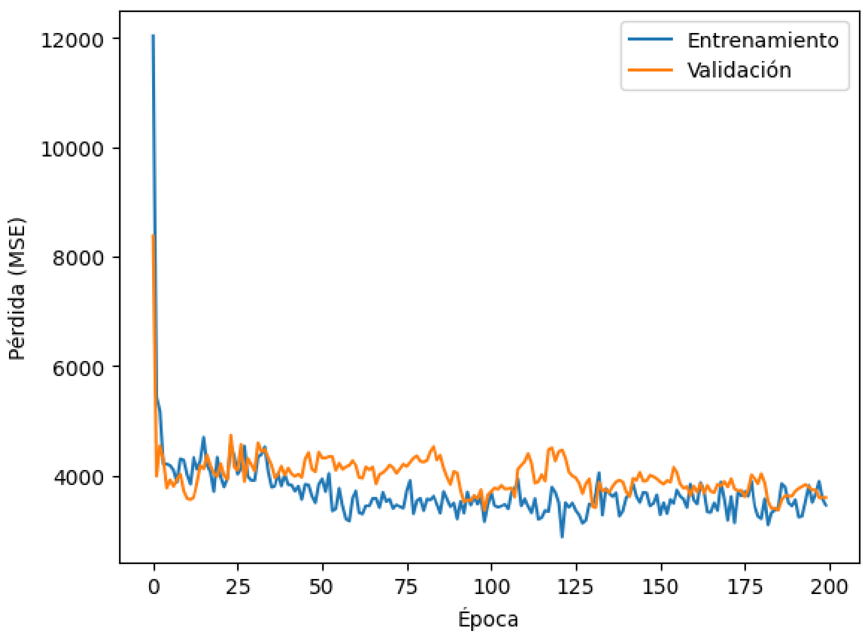

Figura 8: Configuración 3
Más capas ocultas: [d, 10, 20, 1]
Activación: Sigmoid
Regularización: Dropout 0.2 despues de cada act
Normalización: por penalizacion L2, el weight_decay=1e-3