UACM
Universidad Autónoma
de la Ciudad de México
NADA HUMANO ME ES AJENO

Redes Neuronales

Sabino Miranda

Redes Multicapa

3. Backpropagation-I (1)

- Objetivo: minimizar el error de predicción de la red neuronal ajustando los pesos y sesgos en función del gradiente del error.
- Se logra calculando el gradiente de la función de pérdida con respecto a los pesos y sesgos mediante la regla de la cadena para que cada peso y sesgo se actualice y mejorar la precisión de la red durante el entrenamiento.
- Se usa el **Descenso del Gradiente** para reducir el error de la red completa y mejorar las predicciones.

3. Backpropagation-I (2)

- Descenso del gradiente para actualización de los parámetros:

$$w_{t+1} = w_t - \eta \frac{\partial E}{\partial w_t}$$

  Donde:

  - w_t representa el peso actual
  - w_{t+1} representa el peso nuevo
  - E la función de error
  - η (eta) la tasa de aprendizaje

- Para determinar el error respecto a los parámetros (pesos W y sesgos B) se puede realizar por medio de la regla de la cadena del cálculo.

3. Backpropagation-I (3)

- Supongamos una función $f(g(x))$. La regla de la cadena nos dice que la derivada de f con respecto a x es:

$$\frac{d}{dx}f(g(x)) = \frac{df}{dg} \cdot \frac{dg}{dx}$$

- Por medio de la regla de la cadena, se puede calcular las derivadas de cada capa en el orden inverso de la propagación hacia adelante.
- En el ejemplo, la predicción $\hat{y}$ depende $w_{11}^{(2)}$ de la capa (2).
- **E** depende de $W^{(2)}$ a través del producto $z_1 = w_{11}^{(2)} a_1^{(1)} + w_{12}^{(2)} a_2^{(1)} + b_1^{(2)}$

3.1 Derivada (1)

- En términos simples, una derivada es la tasa de cambio de una función con respecto a los cambios en sus argumentos.
- Las derivadas pueden decirnos qué tan rápido **aumentaría** o **disminuiría** una función de pérdida si aumentáramos o disminuyéramos cada parámetro en una cantidad infinitesimalmente pequeña.
- La derivada de f en el punto x se define como

$$f'(x) = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}$$

3.1 Derivada (2)

- El límite nos dice a qué valor converge la relación entre una perturbación h y el cambio en el valor de la función $f(x+h) - f(x)$ a medida que reducimos su tamaño a cero.

3.2. Descenso del Gradiente

- En el aprendizaje profundo, se usa la optimización de la función de error o función de pérdida o función de costo. Esta se puede hacer de manera eficiente si se hace uso de la información del gradiente.
- Esto se logra, si se evalúan las derivadas de la función de error con respecto a los parámetros de la red (pesos y bias).
- Una preocupación es que la función representada por la red neuronal sea diferenciable por diseño. Asimismo, la función de error en sí también debe ser diferenciable.

Procesamiento y Visualización de datos

Consultar el NoteBook: 06_descenso_gradiente.ipynb

3.3 Regla de la cadena (1)

- La **regla de la cadena** es una técnica en cálculo para encontrar la derivada de una función compuesta, una función que está formada por otras funciones.
- Si una variable y depende de una variable u, la cual depende de una variable x. La tasa de cambio de y respecto a x, se calcula como el producto de la tasa de cambio de y respecto a u y de la tasa de cambio de u respecto a x.

$$\frac{dy}{dx} = \frac{dy}{du} \cdot \frac{du}{dx}$$

3.3 Regla de la cadena (2)

- Forma de la Regla de la Cadena:
  Supongamos que se tiene una función $y = f(g(x))$, donde:
  - $y = f(u)$ es una función que depende de $u = g(x)$.
  - $g(x)$ es una función que depende de x.

  La regla de la cadena nos dice que la derivada de y con respecto a x es:

$$\frac{d}{dx} f(g(x)) = \frac{dy}{du} \cdot \frac{du}{dx}$$

$$\frac{dy}{dx} = \frac{dy}{du} \cdot \frac{du}{dx}$$

3.3 Regla de la cadena (3)

Ejemplo

- Derivar $y = f(x) = (3x^2 + 1)^2$ aplicando la regla de la cadena:
  - Función interna: $u = g(x) = 3x^2 + 1$
  - Función externa: $y = f(u) = u^2$
- Aplicando la regla de la cadena:
  - La derivada de $y = u^2$ con respecto a u es $\frac{dy}{du} = 2u$.
  - La derivada de $u = 3x^2 + 1$ con respecto a x es $\frac{du}{dx} = 6x$.
- Se multiplica ambas derivadas sustituyendo u:

$$\frac{dy}{dx} = \frac{dy}{du} \cdot \frac{du}{dx} = 2u \cdot 6x = 2(3x^2 + 1) \cdot 6x = 12x(3x^2 + 1)$$

3.3 Regla de la cadena (4)

- En redes neuronales, la regla de la cadena se aplica en el algoritmo de retropropagación para calcular los gradientes de la función de pérdida respecto a los pesos y sesgos.
- Al derivar una función de pérdida que depende de múltiples capas de funciones de activación (como la sigmoide o ReLU), la regla de la cadena permite propagar los gradientes hacia atrás a través de la red.

3.3 Regla de la cadena: Ejercicios

- Derivar la siguiente función usando la regla de la cadena:

$$y = f(x) = \text{sen}(2x^2 + 3)$$

  - La función interna es $u = g(x) = ?$
  - La función externa es $y = f(u) = ?$
  - Regla de la cadena:

$$\frac{dy}{dx} = \frac{dy}{du} \cdot \frac{du}{dx}$$

$$\frac{dy}{dx} = ?$$

3.3 Regla de la cadena: Ejercicios

- Derivar la siguiente función usando la regla de la cadena:

$$y = f(x) = \ln(\cos(x^2 + 1))$$

  - La función interna es $u = g(x) = ?$
  - La función externa es $y = f(u) = ?$

$$\frac{dy}{dx} = ?$$

3.3 Regla de la cadena: Ejercicios

- Derivar la siguiente función usando la regla de la cadena:

$$y = f(x) = e^{2x^2+3x}$$

  - La función interna es $u = g(x) = ?$
  - La función externa es $y = f(u) = ?$

$$\frac{dy}{dx} = ?$$

3. Backpropagation-II (1)

- Para propagar el error hacia atrás en la red, se calcula el gradiente respecto a los parámetros de la red (pesos y sesgos)
- El objetivo es calcular la razón de cambio de la función de error respecto a los pesos y bias.
- En el ejemplo, se puede calcular el error de predicción con respecto a $w_{11}^{(2)}$, usando la regla de la cadena:

$$E = \frac{1}{2}(\hat{y}-y)^2 = \frac{1}{2}(\sigma(z_1)-y)^2 = \frac{1}{2}(\sigma(w_{11}^{(2)}a_1^{(1)}+w_{12}^{(2)}a_2^{(1)}+b_1^{(2)})-y)^2$$

3. Backpropagation-II (2)

$$\frac{\partial E}{\partial w_{11}^{(2)}} = \frac{\partial \frac{1}{2}(\sigma(w_{11}^{(2)} a_1^{(1)} + w_{12}^{(2)} a_2^{(1)} + b_1^{(2)}) - y)^2}{\partial w_{11}^{(2)}}$$

- La regla de la cadena permite calcular el gradiente del error (la tasa de cambio) para cada uno de los parámetros.

$$\frac{\partial E}{\partial w_{11}^{(2)}} = \frac{\partial E}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial z_1^{(2)}} \cdot \frac{\partial z_1^{(2)}}{\partial w_{11}^{(2)}}$$

3. Backpropagation-II (3)

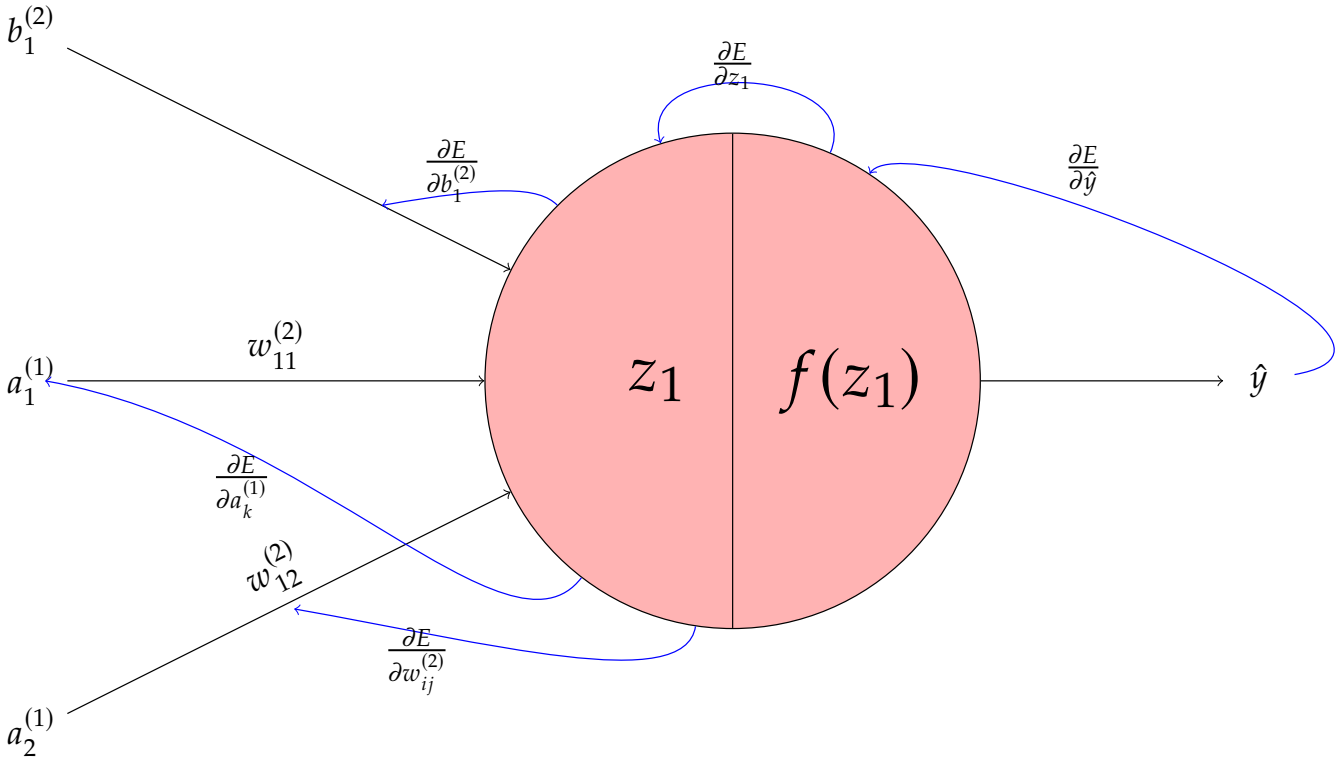

Figura 1: Backpropagation para la neurona de salida.

3. Backpropagation-II (4)

- Calcular el gradiente de la capa de salida con respecto a la predicción.

$$\nabla_{\hat{y}} E(\hat{y}, y) = \frac{\partial E}{\partial \hat{y}} = (\hat{y} - y)$$

- Calcular el gradiente de la capa de salida con respecto al lote (batch) N de predicciones.

$$\nabla_{\hat{y}_i} E(\hat{y}_i, y_i) = \frac{\partial E}{\partial \hat{y}_i} = \frac{1}{N}(\hat{y}_i - y_i)$$

Derivada parcial del Error (1)

- Para N ejemplos, el error cuadrático medio se calcula como:

$$Error(E) = \text{MSE} = \frac{1}{2N}\sum_{i=1}^{N}(\hat{y}_i - y_i)^2$$

- Dada la ecuación para un ejemplo i particular :

$$E(\hat{y}, y) = \frac{1}{2}(\hat{y} - y)^2$$

- Paso 1: Derivada de E con respecto a $\hat{y}$

$$\frac{\partial E}{\partial \hat{y}} = \frac{\partial}{\partial \hat{y}}\left(\frac{1}{2}(\hat{y} - y)^2\right)$$

Derivada parcial del Error (2)

- Paso 2: La derivada de $(\hat{y} - y)^2$ con respecto a $\hat{y}$ es:

$$\frac{\partial}{\partial \hat{y}} \left((\hat{y} - y)^2 \right) = 2(\hat{y} - y) \cdot \frac{\partial}{\partial \hat{y}} \left(\hat{y} - y \right)$$

- Paso 3: Multiplicamos por el factor externo $\frac{1}{2}$:

$$\frac{\partial E}{\partial \hat{y}} = \frac{1}{2} \cdot 2(\hat{y} - y) \cdot (1)$$

- Paso 4: Al simplificar, obtenemos:

$$\frac{\partial E}{\partial \hat{y}} = (\hat{y} - y)$$

Derivada parcial del Error (3)

- La derivada final de la función de error con respecto a $\hat{y}$ es:

$$\frac{\partial E}{\partial \hat{y}} = \hat{y} - y$$

- Para N ejemplos, el error cuadrático medio se calcula como:

$$\frac{\partial E}{\partial \hat{y}} = \frac{\hat{y} - y}{N}$$

3. Backpropagation-III (1)

- 1. **(Recordando) Propagación hacia adelante (Forward pass)**
- Nomenclatura:
  - z: Potencial de activación, suma ponderada de entradas y pesos + bias.
  - a: Activación, salida después de aplicar la función activación: p.j., $a = \sigma(z)$.
  - δ: Error de cada neurona en términos de la derivada del error con respecto a z.
  - W: Pesos.
  - b: Sesgos.

3. Backpropagation-III (2)

- 1. **(Recordando) Propagación hacia adelante (Forward pass)**
  - Capa Oculta (2 Neuronas)

$$z_1^{(1)} = w_{11}^{(1)} x_1 + w_{12}^{(1)} x_2 + b_1$$

$$z_2^{(1)} = w_{21}^{(1)} x_1 + w_{22}^{(1)} x_2 + b_2$$

Aplicando la función sigmoide $\sigma(z) = \frac{1}{1+e^{-z}}$ para obtener las activaciones:

$$a_1^{(1)} = \sigma(z_1) = \frac{1}{1 + e^{-z_1}}$$

$$a_2^{(1)} = \sigma(z_2) = \frac{1}{1 + e^{-z_2}}$$

3. Backpropagation-III (3)

- Capa de salida (1 neurona):

$$z_1^{(2)} = w_{11}^{(2)} a_1^{(1)} + w_{12}^{(2)} a_2^{(1)} + b_1^{(2)}$$

La activación en la capa de salida se calcula aplicando la función sigmoide:

$$\hat{y} = a_1^{(2)} = \sigma(z_1^{(2)}) = \frac{1}{1 + e^{-z_1^{(2)}}}$$

3. Backpropagation-III (4)

- 2. **(Recordando) Cálculo del error**
  - El error se calcula usando la función de pérdida del error cuadrático medio (MSE):

$$E = \frac{1}{2}(\hat{y} - y)^2$$

donde $\hat{y} = a_1^{(2)}$ es la salida de la red y y es el valor esperado.

3. Backpropagation-III (5)

- 3. **Cálculo del gradiente**
  - Cálculo del error en la capa de salida, representado por $\delta^{(2)}$:

$$\delta_j^{(2)} = \frac{\partial E}{\partial z_j^{(2)}}$$

$$\delta^{(2)} = \frac{\partial E}{\partial z_1^{(2)}} = \frac{\partial E}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial z_1^{(2)}}$$

$$\delta^{(2)} = \frac{\partial E}{\partial z_1^{(2)}} = (\hat{y} - y) \cdot \hat{y}(1 - \hat{y})$$

3. Backpropagation-III (6)

- Este término se obtiene de:

$$\frac{\partial E}{\partial \hat{y}} = (\hat{y} - y)$$

$$\frac{\partial \hat{y}}{\partial z_1^{(2)}} = \frac{\partial \sigma(z_1^{(2)})}{\partial z_1^{(2)}} = \sigma(z_1^{(2)})(1 - \sigma(z_1^{(2)})) = \hat{y}(1 - \hat{y})$$

Derivada de $\sigma(z)$ (1)

- La derivada de la función sigmoide $\sigma(z)$

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

- Se puede reescribir $\sigma(z)$ como:

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

- Para derivar $\sigma(z)$, usamos la regla de la cadena.
- La función externa es $g(u) = u^{-1}$ y la función interna es $u = 1 + e^{-z}$.

Derivada de $\sigma(z)$ (2)

- Se deriva ambas partes:

  1. Derivada de la función externa $g(u) = u^{-1}$:

$$\frac{d}{du}\left(u^{-1}\right) = -u^{-2}.$$

  2. Derivada de la función interna $u = 1 + e^{-z}$:

$$\frac{d}{dz}\left(1 + e^{-z}\right) = -e^{-z}.$$

- Se aplica la regla del producto

$$\frac{d}{dz}\sigma(z) = -(1 + e^{-z})^{-2} \cdot (-e^{-z}).$$

Derivada de $\sigma(z)$ (3)

- Se simplifica la expresión

$$\frac{d}{dz}\sigma(z) = \frac{e^{-z}}{(1+e^{-z})^2}.$$

Derivada de $\sigma(z)$ (4)

- Para simplificar la derivada de la función sigmoide $\frac{d}{dz}\sigma(z)$ en términos de $\sigma(z)$:
  Partimos de la expresión que ya tenemos:

$$\frac{d}{dz}\sigma(z) = \frac{e^{-z}}{(1+e^{-z})^2}$$

  Sabemos que la función sigmoide se define como:

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

Derivada de $\sigma(z)$ (5)

- Paso 1: Se reescribe e^{-z} en función de $\sigma(z)$

  De la definición de $\sigma(z)$:

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

  Por lo tanto:

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

- Paso 2: Sustituir e^{-z} en la derivada

$$\frac{d}{dz}\sigma(z) = \frac{e^{-z}}{(1+e^{-z})^2} = \frac{\frac{1-\sigma(z)}{\sigma(z)}}{\left(1+\frac{1-\sigma(z)}{\sigma(z)}\right)^2} = \frac{\frac{1-\sigma(z)}{\sigma(z)}}{\left(\frac{1}{\sigma(z)}\right)^2}$$

Derivada de $\sigma(z)$ (6)

- Paso 3: Simplificar la fracción
  Simplificamos el numerador y el denominador:

$$\frac{d}{dz}\sigma(z) = \frac{\frac{1-\sigma(z)}{\sigma(z)}}{\frac{1}{\sigma(z)^2}} = \frac{\sigma(z)^2(1-\sigma(z))}{\sigma(z)} = \sigma(z)\cdot(1-\sigma(z))$$

- Resultado final:

$$\frac{d}{dz}\sigma(z) = \sigma(z)\cdot(1-\sigma(z))$$

3. Backpropagation-IV:Cálculo del gradiente (1)

- **Gradientes en la capa de salida:**
  - Gradiente de los pesos de la capa de salida $w_{1j}^{(2)}$, el superíndice $^{(2)}$ representa el número de la capa, $\delta_1^{(2)}$ representa el error en la capa de salida:

$$\delta_1^{(2)} = \frac{\partial E}{\partial z_1^{(2)}} = \frac{\partial E}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial z_1^{(2)}}$$

$$\delta_1^{(2)} = \frac{\partial E}{\partial z_1^{(2)}} = (\hat{y} - y) \cdot \hat{y}(1 - \hat{y})$$

$$\delta_1^{(2)} = \frac{\partial E}{\partial z_1^{(2)}} = (\hat{y} - y) \odot \hat{y}(1 - \hat{y})$$

3. Backpropagation-IV:Cálculo del gradiente (2)

$$\frac{\partial E}{\partial w_{1j}^{(2)}} = \frac{\partial E}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial z_1^{(2)}} \cdot \frac{\partial z_1^{(2)}}{\partial w_{1j}^{(2)}} = \frac{\partial E}{\partial z_1^{(2)}} \cdot \frac{\partial z_1^{(2)}}{\partial w_{1j}^{(2)}} = \delta_1^{(2)} \cdot \frac{\partial z_1^{(2)}}{\partial w_{1j}^{(2)}}$$

Recordando: $z_1^{(2)} = w_{11}^{(2)} a_1^{(1)} + w_{12}^{(2)} a_2^{(1)} + b_1^{(2)}$

$$\frac{\partial z_1^{(2)}}{\partial w_{11}^{(2)}} = a_1^{(1)}$$

Para los pesos de la neurona de salida:

$$\frac{\partial z_1^{(2)}}{\partial w_{1k}^{(2)}} = a_k^{(1)}$$

3. Backpropagation-IV:Cálculo del gradiente (3)

En general:

$$\frac{\partial z_j^{(2)}}{\partial w_{jk}^{(2)}} = a_k^{(1)}$$

Para cada peso:

$$\frac{\partial E}{\partial w_{11}^{(2)}} = \delta_1^{(2)} \cdot a_1^{(1)}$$

$$\frac{\partial E}{\partial w_{12}^{(2)}} = \delta_1^{(2)} \cdot a_2^{(1)}$$

3. Backpropagation-IV:Cálculo del gradiente (4)

Para N ejemplos (batch/mini-batch): suma (o promedio en batch) de todos los $\delta \cdot a$

$$\frac{\partial E}{\partial w_{1k}^{(2)}} = \sum_{i}^{N} \delta_{ik}^{(2)} \cdot a_{ik}^{(1)}$$

- donde:
  - $\delta^{(2)}$ es el *delta* de la neurona de salida para ese ejemplo i.
  - $a_k^{(1)}$ es la activación de la neurona k en la capa oculta para ese mismo ejemplo i.

3. Backpropagation-IV:Cálculo del gradiente (5)

- En general:

$$\frac{\partial E}{\partial W^{(2)}} = (\delta^{(2)})^T \cdot a^{(1)}$$

$$\nabla W^{(2)} = (\Delta^{(2)})^T \cdot A^{(1)}$$

- donde:

$$\underbrace{\nabla W^{(2)}}_{\text{gradiente de los pesos de la capa 2}} = \underbrace{(\Delta^{(2)})^T}_{\substack{\text{delta de la neurona de salida} \\ \text{para cada ejemplo, } \mathbb{R}^{1\times N}}} \cdot \underbrace{A^{(1)}}_{\substack{\text{activaciones de la capa oculta} \\ \text{para cada ejemplo, } \mathbb{R}^{N\times H}}} \in \mathbb{R}^{1\times H(\text{neuronas de capa oculta})}$$

3. Backpropagation-IV:Cálculo del gradiente (6)

- Para el ejemplo de la XOR. En forma matricial. N=4 , H=2 , O=1; 4 ejemplos de entrenamiento (N), 2 neuronas ocultas (H) y 1 de salida (O)

$$\nabla W^{(2)} = (\Delta^{(2)})^T \cdot A^{(1)} \in \mathbb{R}^{1\times 2}$$

$$\Delta^{(2)} = \begin{bmatrix} \delta_{11}^{(2)} \\ \delta_{21}^{(2)} \\ \delta_{31}^{(2)} \\ \delta_{41}^{(2)} \end{bmatrix} \in \mathbb{R}^{4\times 1} \quad \text{(delta de la neurona de salida para cada ejemplo)}$$

$$\mathrm{A}^{(1)} = \begin{bmatrix} a_{11}^{(1)} & a_{12}^{(1)} \\ a_{21}^{(1)} & a_{22}^{(1)} \\ a_{31}^{(1)} & a_{32}^{(1)} \\ a_{41}^{(1)} & a_{42}^{(1)} \end{bmatrix} \in \mathbb{R}^{4\times 2} \quad \text{(activaciones de las 2 neuronas ocultas para cada ejemplo)}$$

3. Backpropagation-IV:Cálculo del gradiente (7)

$$\nabla W^{(2)} = (\Delta^{(2)})^T \cdot A^{(1)} \in \mathbb{R}^{1\times 2}$$

- donde:

$$\underbrace{\nabla W^{(2)}}_{\text{gradiente de los pesos de la capa 2}} = \underbrace{(\Delta^{(2)})^T}_{\substack{\text{delta de la neurona de salida} \\ \text{para cada ejemplo, } \mathbb{R}^{1\times N}}} \cdot \underbrace{A^{(1)}}_{\substack{\text{activaciones de la capa oculta} \\ \text{para cada ejemplo, } \mathbb{R}^{N\times H}}} \in \mathbb{R}^{1\times H}$$

$$(\Delta^{(2)})^T \cdot A^{(1)} = \begin{bmatrix} \delta_{11}^{(2)} & \delta_{21}^{(2)} & \delta_{31}^{(2)} & \delta_{41}^{(2)} \end{bmatrix} \begin{bmatrix} a_{11}^{(1)} & a_{12}^{(1)} \\ a_{21}^{(1)} & a_{22}^{(1)} \\ a_{31}^{(1)} & a_{32}^{(1)} \\ a_{41}^{(1)} & a_{42}^{(1)} \end{bmatrix} = \begin{bmatrix} \sum_{n=1}^{4} \delta_{n1}^{(2)} a_{n1}^{(1)} & \sum_{n=1}^{4} \delta_{n1}^{(2)} a_{n2}^{(1)} \end{bmatrix}$$

$$\boxed{\nabla W^{(2)} = \begin{bmatrix} \delta_{11}^{(2)} a_{11}^{(1)} + \delta_{21}^{(2)} a_{21}^{(1)} + \delta_{31}^{(2)} a_{31}^{(1)} + \delta_{41}^{(2)} a_{41}^{(1)} & \delta_{11}^{(2)} a_{12}^{(1)} + \delta_{21}^{(2)} a_{22}^{(1)} + \delta_{31}^{(2)} a_{32}^{(1)} + \delta_{41}^{(2)} a_{42}^{(1)} \end{bmatrix}}$$

Opcional: gradiente medio: $\nabla W_{\text{media}}^{(2)} = \frac{1}{4}(\Delta^{(2)})^T A^{(1)}$

3. Backpropagation-IV:Cálculo del gradiente (8)

- **Gradiente del sesgo de la capa de salida**:
  La derivada del error respecto al sesgo en la capa de salida $b_1^{(2)}$ es:

$$\frac{\partial E}{\partial b_1^{(2)}} = \frac{\partial E}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial z_1^{(2)}} \cdot \frac{\partial z_1^{(2)}}{\partial b_1^{(2)}} = \frac{\partial E}{\partial z_1^{(2)}} \cdot \frac{\partial z_1^{(2)}}{\partial b_1^{(2)}} = \delta_1^{(2)} \cdot \frac{\partial z_1^{(2)}}{\partial b_1^{(2)}}$$

Recordando: $z_1^{(2)} = w_{11}^{(2)} a_1^{(1)} + w_{12}^{(2)} a_2^{(1)} + b_1^{(2)}$

$$\frac{\partial z_1^{(2)}}{\partial b_1^{(2)}} = \frac{\partial b_1^{(2)}}{\partial b_1^{(2)}} = 1$$

$$\frac{\partial E}{\partial b_1^{(2)}} = \delta_1^{(2)}$$

3. Backpropagation-IV:Cálculo del gradiente (9)

Para N ejemplos (batch/mini-batch): suma (o promedio en batch) de todos los δ

$$\frac{\partial E}{\partial b_{1k}^{(2)}} = \sum_{i}^{N} \delta_{ik}^{(2)}$$

- donde:
  - $\delta^{(2)}$ es el *delta* de la neurona de salida k para ese ejemplo i.

3. Backpropagation-IV:Cálculo del gradiente (10)

- **Gradientes en la capa oculta:**
  - Gradientes de los pesos de la capa oculta:

$$\underbrace{\frac{\partial E}{\partial w_{ji}^{(1)}}}_{\text{gradiente del peso } ji \text{ de la capa 1}} = \underbrace{\underbrace{\frac{\partial E}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial z_1^{(2)}}}_{\delta_1^{(2)}} \cdot \frac{\partial z_1^{(2)}}{\partial a_j^{(1)}} \cdot \frac{\partial a_j^{(1)}}{\partial z_j^{(1)}}}_{\delta_j^{(1)}} \cdot \frac{\partial z_j^{(1)}}{\partial w_{ji}^{(1)}}$$

$$\frac{\partial E}{\partial w_{ji}^{(1)}} = \delta_j^{(1)} \cdot \frac{\partial z_j}{\partial w_{ji}^{(1)}}$$

  - Para cada neurona de la capa oculta, calculamos $\delta_j^{(1)}$, el error de esa capa.

3. Backpropagation-IV:Cálculo del gradiente (11)

- El error en la capa oculta $\delta_j^{(1)}$ se propaga desde el error de la capa de salida:

$$\delta_1^{(2)} = \frac{\partial E}{\partial z_1^{(2)}} = \frac{\partial E}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial z_1^{(2)}}$$

$$\delta_j^{(1)} = \frac{\partial E}{\partial z_j^{(1)}} = \frac{\partial E}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial z_1^{(2)}} \cdot \frac{\partial z_1^{(2)}}{\partial a_j^{(1)}} \cdot \frac{\partial a_j^{(1)}}{\partial z_j^{(1)}}$$

$$\delta_j^{(1)} = \frac{\partial E}{\partial z_j^{(1)}} = \delta_1^{(2)} \cdot \frac{\partial z_1^{(2)}}{\partial a_j^{(1)}} \cdot \frac{\partial a_j^{(1)}}{\partial z_j^{(1)}}$$

Recordando: $a_j^{(1)} = \sigma(z_j^{(1)})$

3. Backpropagation-IV:Cálculo del gradiente (12)

$$\frac{\partial a_j^{(1)}}{\partial z_j^{(1)}} = \frac{\partial \sigma(z_j^{(1)})}{\partial z_j^{(1)}} = \sigma(z_j^{(1)})(1 - \sigma(z_j^{(1)})) = a_j^{(1)}(1 - a_j^{(1)})$$

Recordando: $z_1^{(2)} = w_{11}^{(2)} a_1^{(1)} + w_{12}^{(2)} a_2^{(1)} + b_1^{(2)}$

$$\frac{\partial z_1^{(2)}}{\partial a_j^{(1)}} = w_{1j}^{(2)}$$

3. Backpropagation-IV:Cálculo del gradiente (13)

Por lo tanto:

$$\delta_j^{(1)} = \frac{\partial E}{\partial z_j^{(1)}} = \delta_1^{(2)} \cdot w_{1j}^{(2)} \cdot a_j^{(1)}(1 - a_j^{(1)})$$

Donde:

$$\delta_j^{(1)} = \frac{\partial E}{\partial z_j^{(1)}} = \underbrace{\delta^{(2)}}_{\text{error de la neurona de salida}} \cdot \underbrace{w_{1j}^{(2)}}_{\text{peso que conecta la neurona oculta j con la salida}} \cdot \underbrace{a_j^{(1)}(1 - a_j^{(1)})}_{\text{derivada de la activación sigmoide (das)}}$$

$$das^{(1)} = a^{(1)}(1 - a^{(1)})$$

$$\boldsymbol{\delta^{(1)}} = \boldsymbol{\delta^{(2)}} \cdot W^{(2)} \odot das^{(1)}$$

3. Backpropagation-IV:Cálculo del gradiente (14)

donde $w_{1j}^{(2)}$ son los pesos que conectan las neuronas de la capa oculta con la neurona de salida. $\odot$ producto elemento a elemento. sda matriz de las derivadas de las activaciones

- Para cada neurona en la capa oculta:
  - Para la primera neurona ($a_1^{(1)}$):

$$\delta_1^{(1)} = \delta_1^{(2)} \cdot w_{11}^{(2)} \cdot a_1^{(1)}(1 - a_1^{(1)})$$

  - Para la segunda neurona ($a_2^{(1)}$):

$$\delta_2^{(1)} = \delta_1^{(2)} \cdot w_{12}^{(2)} \cdot a_2^{(1)}(1 - a_2^{(1)})$$

3. Backpropagation-IV:Cálculo del gradiente (15)

- Para N ejemplos (XOR):

$$\Delta^{(2)} = \begin{bmatrix} \delta_{11}^{(2)} \\ \delta_{21}^{(2)} \\ \delta_{31}^{(2)} \\ \delta_{41}^{(2)} \end{bmatrix} \in \mathbb{R}^{4\times 1}$$

$$W^{(2)} = \begin{bmatrix} w_{11}^{(2)} & w_{12}^{(2)} \end{bmatrix} \in \mathbb{R}^{1\times 2}$$

$$A^{(1)} = \begin{bmatrix} a_{11}^{(1)} & a_{12}^{(1)} \\ a_{21}^{(1)} & a_{22}^{(1)} \\ a_{31}^{(1)} & a_{32}^{(1)} \\ a_{41}^{(1)} & a_{42}^{(1)} \end{bmatrix} \in \mathbb{R}^{4\times 2}$$

3. Backpropagation-IV:Cálculo del gradiente (16)

$$\sigma'(Z^{(1)}) = A^{(1)} \odot (1 - A^{(1)}) \in \mathbb{R}^{4\times 2}$$

$$\Delta^{(1)} = (\Delta^{(2)} \cdot W^{(2)}) \odot \sigma'(Z^{(1)}) \in \mathbb{R}^{4\times 2}$$

$$\Delta^{(1)} = \begin{bmatrix} \delta_{11}^{(2)} w_{11}^{(2)} a_{11}^{(1)}(1 - a_{11}^{(1)}) & \delta_{11}^{(2)} w_{12}^{(2)} a_{12}^{(1)}(1 - a_{12}^{(1)}) \\ \delta_{21}^{(2)} w_{11}^{(2)} a_{21}^{(1)}(1 - a_{21}^{(1)}) & \delta_{21}^{(2)} w_{12}^{(2)} a_{22}^{(1)}(1 - a_{22}^{(1)}) \\ \delta_{31}^{(2)} w_{11}^{(2)} a_{31}^{(1)}(1 - a_{31}^{(1)}) & \delta_{31}^{(2)} w_{12}^{(2)} a_{32}^{(1)}(1 - a_{32}^{(1)}) \\ \delta_{41}^{(2)} w_{11}^{(2)} a_{41}^{(1)}(1 - a_{41}^{(1)}) & \delta_{41}^{(2)} w_{12}^{(2)} a_{42}^{(1)}(1 - a_{42}^{(1)}) \end{bmatrix}$$

$$\Delta^{(1)} = \begin{bmatrix} \delta_{11}^{(1)} & \delta_{12}^{(1)} \\ \delta_{21}^{(1)} & \delta_{22}^{(1)} \\ \delta_{31}^{(1)} & \delta_{32}^{(1)} \\ \delta_{41}^{(1)} & \delta_{42}^{(1)} \end{bmatrix} \in \mathbb{R}^{4\times 2}$$

3. Backpropagation-IV:Cálculo del gradiente (17)

$$\delta_{n,j}^{(1)} = \delta_{n,1}^{(2)} \cdot w_{1j}^{(2)} \cdot a_{n,j}^{(1)}\left(1 - a_{n,j}^{(1)}\right), \quad n = 1, \ldots, 4, \; j = 1, 2$$

3. Backpropagation-IV:Cálculo del gradiente (18)

- Gradientes de los pesos de la capa oculta:

$$\frac{\partial E}{\partial w_{ji}^{(1)}} = \delta_j^{(1)} \cdot \frac{\partial z_j}{\partial w_{ji}^{(1)}}$$

Recordando: $z_j^{(1)} = w_{ji}^{(1)} x_i + b_j^{(1)}$

$$\frac{\partial z_j^{(1)}}{\partial w_{ji}^{(1)}} = x_i$$

3. Backpropagation-IV:Cálculo del gradiente (19)

- Primera neurona de la capa oculta:

$$\frac{\partial E}{\partial w_{11}^{(1)}} = \delta_1^{(1)} \cdot x_1$$

$$\frac{\partial E}{\partial w_{12}^{(1)}} = \delta_1^{(1)} \cdot x_2$$

- Segunda neurona de la capa oculta:

$$\frac{\partial E}{\partial w_{21}^{(1)}} = \delta_2^{(1)} \cdot x_1$$

$$\frac{\partial E}{\partial w_{22}^{(1)}} = \delta_2^{(1)} \cdot x_2$$

3. Backpropagation-IV:Cálculo del gradiente (20)

- Para N ejemplos (XOR):

$$\frac{\partial E}{\partial w_{1i}^{(1)}} = \sum_{n=1}^{N} \delta_{n1}^{(1)} x_{ni}, \quad i = 1, 2, \ldots$$

$$\nabla W^{(1)} = (\Delta^{(1)})^T \cdot X$$

- **Gradientes del bias (sesgo) de la capa oculta**:

$$\frac{\partial E}{\partial b_j^{(1)}} = \delta_j^{(1)} \cdot \frac{\partial z_j^{(1)}}{\partial b_j^{(1)}}$$

Recordando: $z_j^{(1)} = w_{ji}^{(1)} x_i + b_j^{(1)}$

3. Backpropagation-IV:Cálculo del gradiente (21)

$$\frac{\partial z_j^{(1)}}{\partial b_j^{(1)}} = 1$$

$$\frac{\partial E}{\partial b_j^{(1)}} = \delta_j^{(1)}$$

Para cada neurona en la capa oculta:

$$\frac{\partial E}{\partial b_1^{(1)}} = \delta_1^{(1)}$$

$$\frac{\partial E}{\partial b_2^{(1)}} = \delta_2^{(1)}$$

3. Backpropagation-IV:Cálculo del gradiente (22)

- Para N ejemplos (batch/mini-batch): suma (o promedio en batch) de todos los δ

$$\frac{\partial E}{\partial b_{1k}^{(2)}} = \sum_{i}^{N} \delta_{ik}^{(2)}$$

- donde:
  - $\delta^{(2)}$ es el *delta* de la neurona de salida k para ese ejemplo i.

4. Actualización de los pesos (1)

- Para completar el proceso de backpropagation, después de calcular los gradientes de los pesos y bias, se realiza la actualización de los pesos utilizando un método de optimización, como el gradiente descendente.
- En este caso, se usará la actualización con una tasa de aprendizaje η.
- Fórmula de actualización de pesos. Aplicamos la siguiente regla de actualización:

$$w_{ij}^{(l)} \leftarrow w_{ij}^{(l)} - \eta \frac{\partial E}{\partial w_{ij}^{(l)}}$$

$$b_{j}^{(l)} \leftarrow b_{j}^{(l)} - \eta \frac{\partial E}{\partial b_{j}^{(l)}}$$

4. Actualización de los pesos (2)

- Donde:
  - η es la tasa de aprendizaje (un valor positivo que controla el tamaño del paso de ajuste de los pesos $[0, 1]$).
  - $\frac{\partial E}{\partial w_{ij}^{(l)}}$ es el gradiente del error respecto al peso.
  - $\frac{\partial E}{\partial b_{j}^{(l)}}$ es el gradiente del error respecto al sesgo.
- 1. Actualización de los pesos en la capa de salida
  - Para los pesos que conectan las neuronas de la capa oculta con la salida ($w_{1j}^{(2)}$):

$$w_{1j}^{(2)} \leftarrow w_{1j}^{(2)} - \eta \frac{\partial E}{\partial w_{1j}^{(2)}}$$

4. Actualización de los pesos (3)

Sustituyendo el gradiente:

$$w_{1j}^{(2)} \leftarrow w_{1j}^{(2)} - \eta \left(\delta^{(2)} \cdot a_j^{(1)} \right)$$

Esto se aplica para cada peso en la capa de salida:
- Primer peso de la capa de salida:

$$w_{11}^{(2)} \leftarrow w_{11}^{(2)} - \eta \left(\delta^{(2)} \cdot a_1^{(1)} \right)$$

- Segundo peso de la capa de salida:

$$w_{12}^{(2)} \leftarrow w_{12}^{(2)} - \eta \left(\delta^{(2)} \cdot a_2^{(1)} \right)$$

4. Actualización de los pesos (4)

- Para el sesgo en la capa de salida:

$$b^{(2)} \leftarrow b^{(2)} - \eta \delta^{(2)}$$

- 2. Actualización de los pesos en la capa oculta
  - Para los pesos que conectan las entradas x_1, x_2 con las neuronas de la capa oculta ($w_{ij}^{(1)}$):

$$w_{ij}^{(1)} \leftarrow w_{ij}^{(1)} - \eta \frac{\partial E}{\partial w_{ij}^{(1)}}$$

Sustituyendo el gradiente:

$$w_{ij}^{(1)} \leftarrow w_{ij}^{(1)} - \eta \left(\delta_i^{(1)} \cdot x_j \right)$$

4. Actualización de los pesos (5)

Esto se aplica a cada peso en la capa oculta:
- Primer peso que conecta la entrada x_1 con la primera neurona en la capa oculta:

$$w_{11}^{(1)} \leftarrow w_{11}^{(1)} - \eta \left(\delta_1^{(1)} \cdot x_1 \right)$$

- Segundo peso que conecta la entrada x_2 con la primera neurona en la capa oculta:

$$w_{12}^{(1)} \leftarrow w_{12}^{(1)} - \eta \left(\delta_1^{(1)} \cdot x_2 \right)$$

- Primer peso que conecta la entrada x_1 con la segunda neurona en la capa oculta:

$$w_{21}^{(1)} \leftarrow w_{21}^{(1)} - \eta \left(\delta_2^{(1)} \cdot x_1 \right)$$

4. Actualización de los pesos (6)

- Segundo peso que conecta la entrada x_2 con la segunda neurona en la capa oculta:

$$w_{22}^{(1)} \leftarrow w_{22}^{(1)} - \eta \left(\delta_2^{(1)} \cdot x_2 \right)$$

- Para los sesgos en la capa oculta:

$$b_j^{(1)} \leftarrow b_j^{(1)} - \eta \delta_j^{(1)}$$

Para cada neurona en la capa oculta:
- Para la primera neurona en la capa oculta:

$$b_1^{(1)} \leftarrow b_1^{(1)} - \eta \delta_1^{(1)}$$

4. Actualización de los pesos (7)

- Para la segunda neurona en la capa oculta:

$$b_2^{(1)} \leftarrow b_2^{(1)} - \eta \delta_2^{(1)}$$

Actualización de los pesos (1)

- Las actualizaciones de los pesos pueden ocurrir de diferentes maneras con respecto al conjunto de ejemplos de entrenamiento:
  - Modo en línea (***Online mode***).
    - La actualización de pesos se produce después de que cada ejemplo recorre la red.
    - El algoritmo trata los ejemplos de aprendizaje como un flujo del que aprende en tiempo real.
    - Este modo usado cuando el conjunto de entrenamiento no cabe en la memoria RAM.
    - Sin embargo, este método es sensible a los valores atípicos, por lo que se debe mantener baja la tasa de aprendizaje. (En consecuencia, el algoritmo es lento para converger a una solución).
  - Modo por lotes (***Batch mode***).

Actualización de los pesos (2)

- La actualización de pesos se produce después de procesar todos los ejemplos en el conjunto de entrenamiento.
- Esta técnica hace que la optimización sea más rápida y menos propensa a que aparezcan variaciones en el flujo de ejemplos.
- En el modo por lotes, la retropropagación considera los gradientes sumados de todos los ejemplos.

- Modo de minilotes (o estocástico) (***mini-batch (or stochastic) mode***).
  - La actualización de pesos se produce después de que la red haya procesado una submuestra de ejemplos del conjunto de entrenamiento seleccionados aleatoriamente.
  - Este enfoque combina las ventajas del modo en línea (bajo uso de memoria) y el modo de lotes (una convergencia rápida)

Actualización de los pesos (3)

- Introduce un elemento aleatorio (el submuestreo) para evitar que el descenso del gradiente se quede atrapado en un mínimo local (una caída en el valor que no es el mínimo real).

Referencias (1)

1. Deep Learning. Ian Goodfellow, Yoshua Bengio, and Aaron Courville. MIT Press, 2016. http://www.deeplearningbook.org
2. Dive into Deep Learning. Aston Zhang, Zachary C. Lipton, Mu li, and Alexander J. Smola. Cambridge University Press, 2023. https://d2l.ai
3. Neural Networks and Deep Learning A Textbook (2nd Edition). Charu C. Aggarwal. Springer, 2023. https://doi.org/10.1007/978-3-031-29642-0
4. Deep Learning: Foundations and Concepts. Christopher M. Bishop and Hugh Bishop. Springer, 2024. https://doi.org/10.1007/978-3-031-45468-4

Referencias (2)

5. PyTorch documentation.
   `https://pytorch.org`
6. Numpy documentation.
   `https://numpy.org`
7. Python documentation.
   `https://www.python.org`