Grafos computacionales

Redes Neuronales

Gamaliel Moreno Chávez

gamalielmch@uaz.edu.mx

Enero-julio 2021

Contenido

Grafos computacionales

- Cálculo de $J(\theta)$ y $\nabla_\theta J(\theta)$ es complejo: Concatenación de productos matriz-vector, extensiones para el sesgo y funciones de activación no linenales constribuyen a complejidad.
- En redes modernas, capas varían (función de activación, tipo, tamaño), lo que aumenta complejidad.
- Redes clásicas: $J(\theta)$ y $\nabla_\theta J(\theta)$ calculados de previo y plasmados en código fijo para el caso concreto de J.
- Marcos de trabajo (TensorFlow, Torch, ...) son flexibles: tipos de capas y funciones de activación se cambian fácilmente.
- ¿cómo se pueden hacer los cálculos eficientemente?
- ¿cómo puedo agregar mis propios tipos de capas?

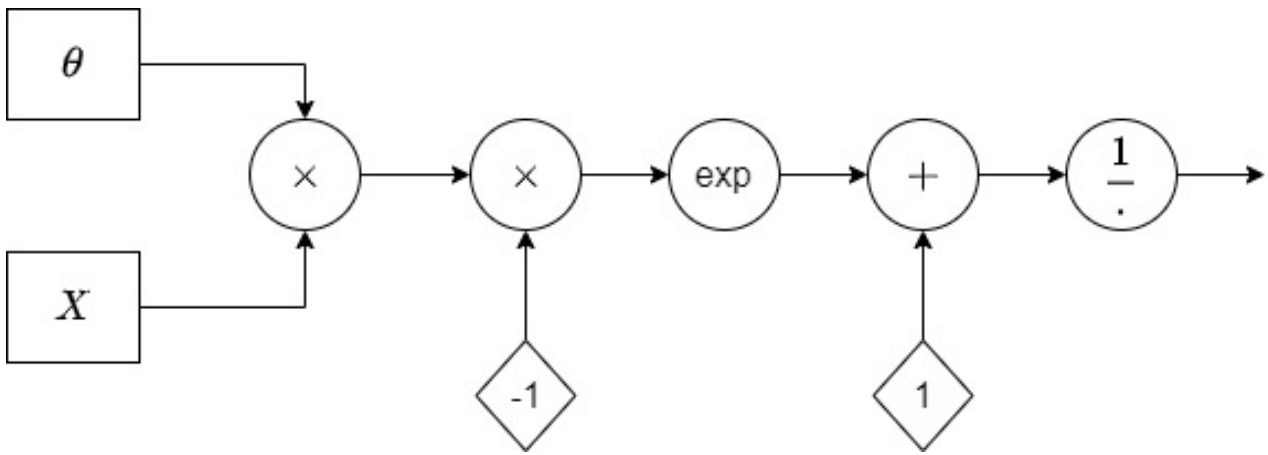
Grafos computacionales

- Cálculo analítico y código correspondiente es complejo, tarda tiempo, y es poco reutilizable.
- Cálculo numérico del gradiente:
  - es excesivamente caro
  - diferencias divididas requieren la evaluiación de J 1 o 2 veces por cada parámetro escalar θ
  - es muy fácil y rápido de implementar
  - se usa para verificar cálculo analítico
- Alternativa: grafos computacionales:
  - Similar a la diferenciación automática
  - Aplican regla de la cadena para descomponer cálculo complejo en paso sencillos
  - Compromiso entre los métodos numéricos y los métodos analíticos
  - Altamente flexibles

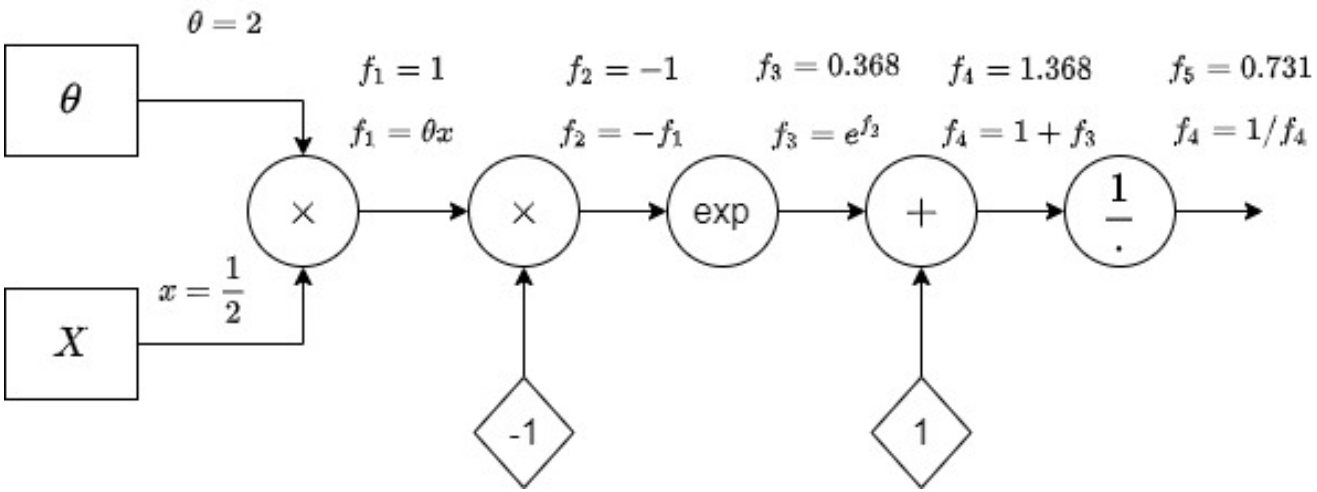
Grafos computacionales

- idea:
  1. Partir evaluación de una función en pasos elementales
  2. Cada paso es un nodo del grafo
  3. Relación entre pasos se expresa con aristas del grafo
- Explicaremos conceptos con ejemplo:

$$f(x,\theta) = \frac{1}{1+e^{-\theta x}}$$



Grafos computacionales



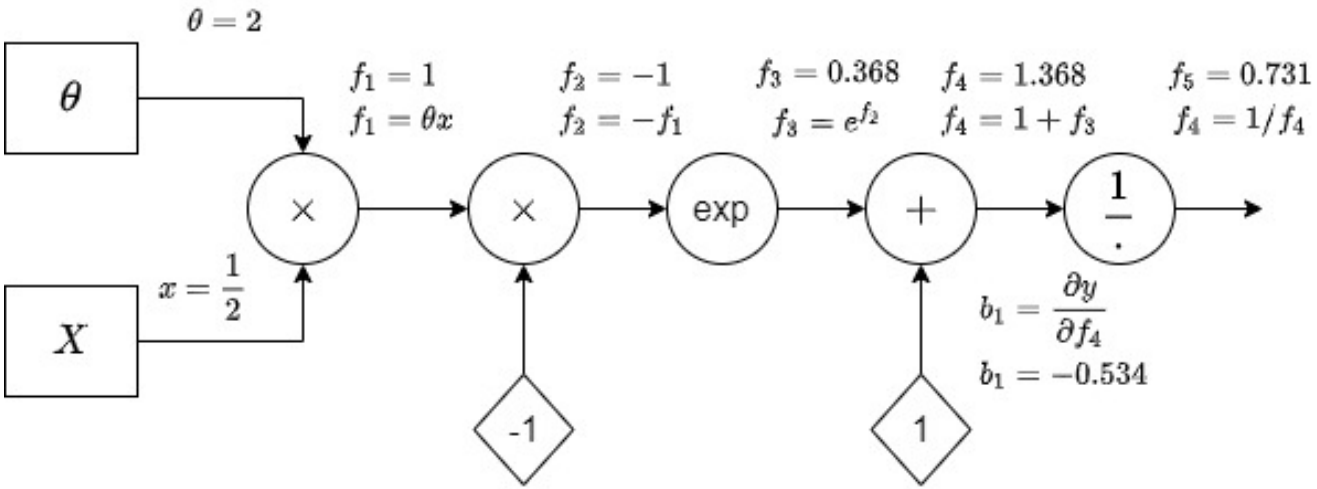
Propagación hacia atrás

- Buscamos ahora $dy/d\theta$ y dy/dx
- Esto se interpreta como el cambio en la salida debido a un cambio en la entrada
- La regla de la cadena subyace el cálculo nodo por nodo:

$$\frac{\partial y}{\partial \theta} = \underbrace{\underbrace{\underbrace{\frac{\partial y}{\partial f_4}}_{b1} \cdot \frac{\partial f_4}{\partial f_3}}_{b2} \cdot \frac{\partial f_3}{\partial f_2}}_{b3} \cdot \frac{\partial f_2}{\partial f_1} \cdot \frac{\partial f_1}{\partial \theta}}_{b4}$$

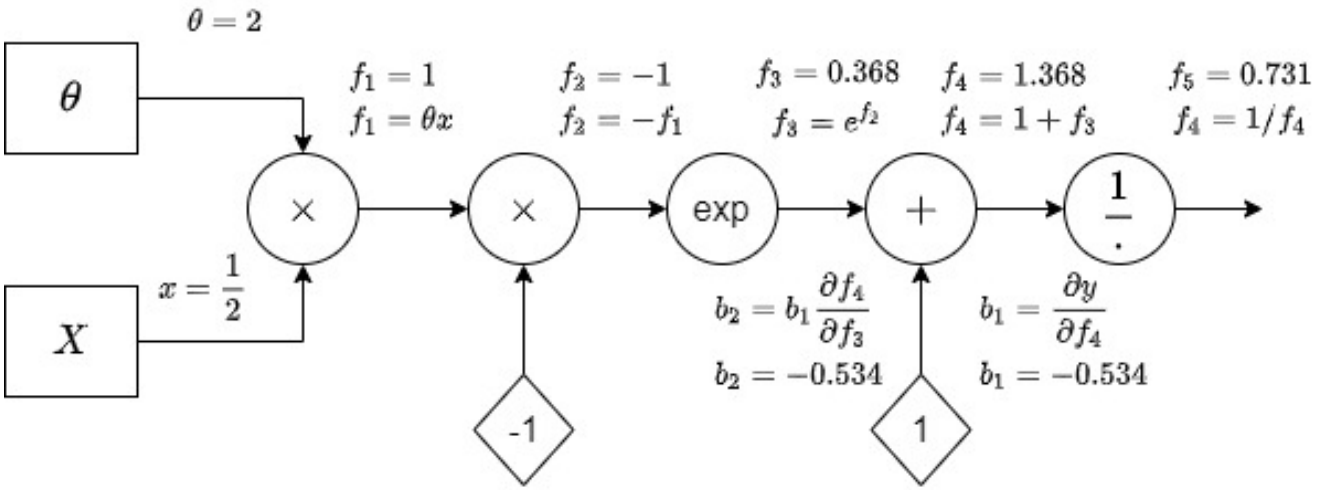
- Lo calculamos de atrás hacia adelante

Propagación hacia atrás



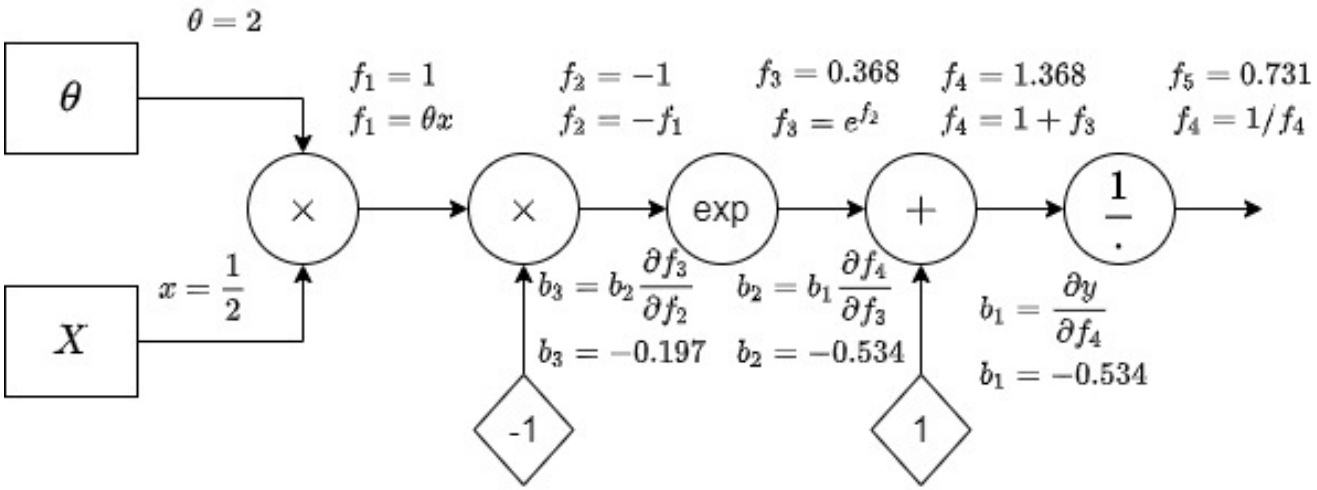
$$\frac{dy}{df_4} = \frac{d}{df_4}\left(\frac{1}{f_4}\right) = -\frac{1}{f_4^2} = -\frac{1}{1.368^2} = -0.534$$

Propagación hacia atrás



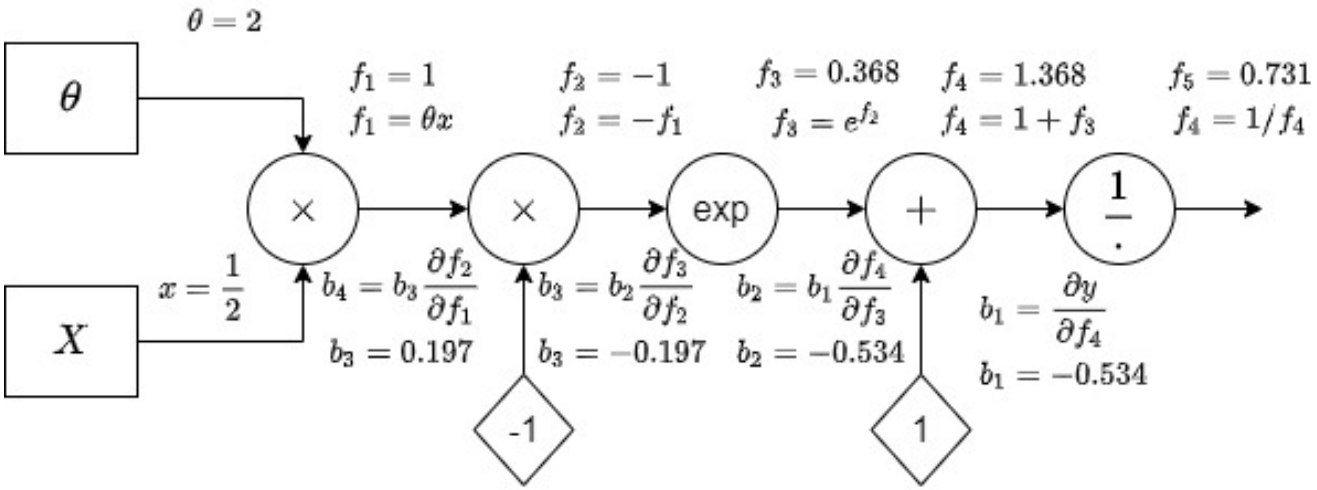
$$\frac{df_4}{df_3} = \frac{d}{df_3}(1 + f_3) = 1$$

Propagación hacia atrás



$$\frac{df_3}{df_2} = \frac{d}{df_2}(e^{f_2}) = e^{f_2} = e^{-1} = 0.368$$

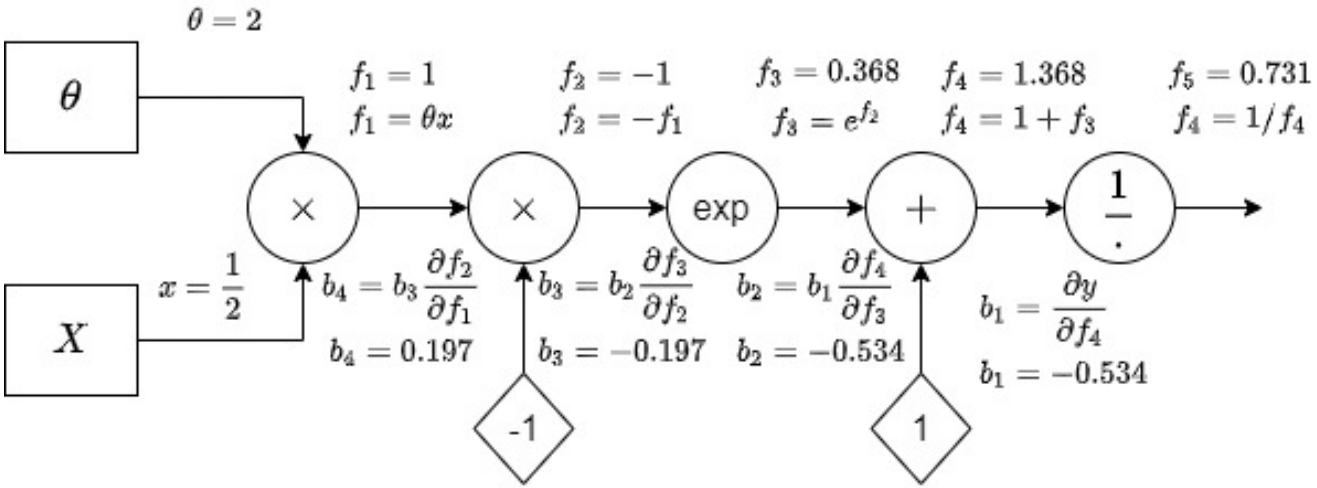
Propagación hacia atrás



$$\frac{df_2}{df_1} = \frac{d}{df_1}(-f_1) = -1$$

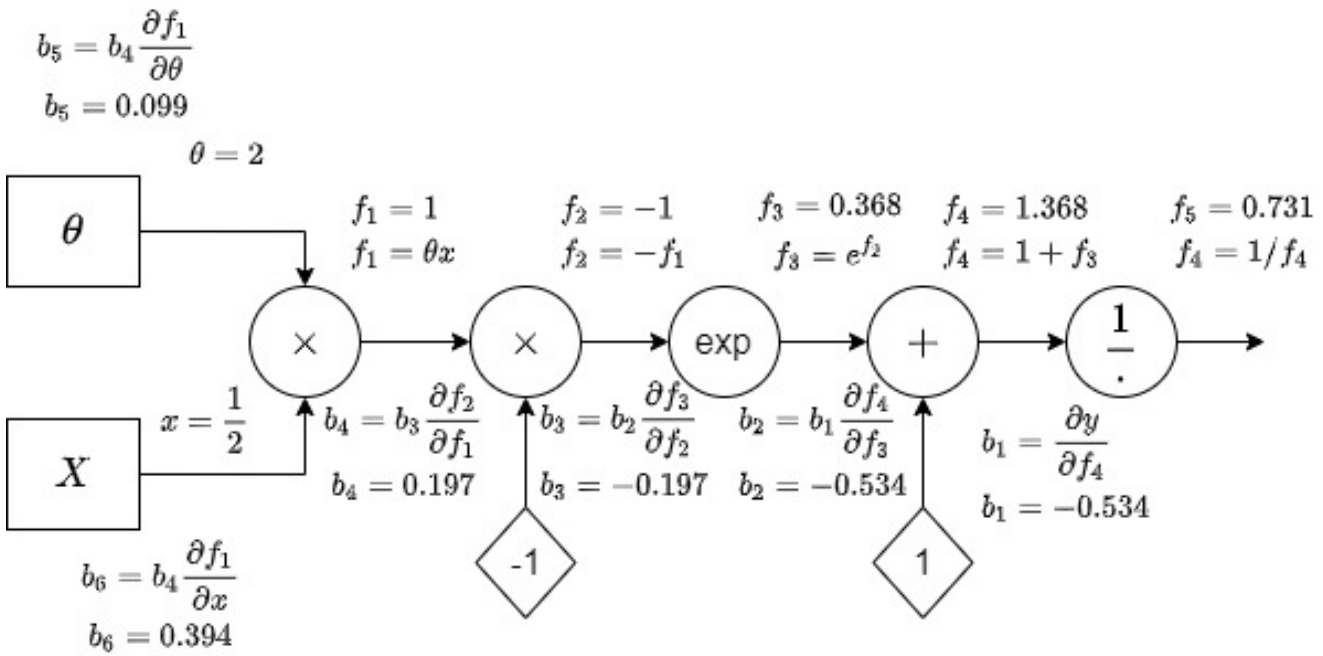
Propagación hacia atrás

$$b_5 = b_4 \frac{\partial f_1}{\partial \theta}$$
$$b_5 = 0.099$$

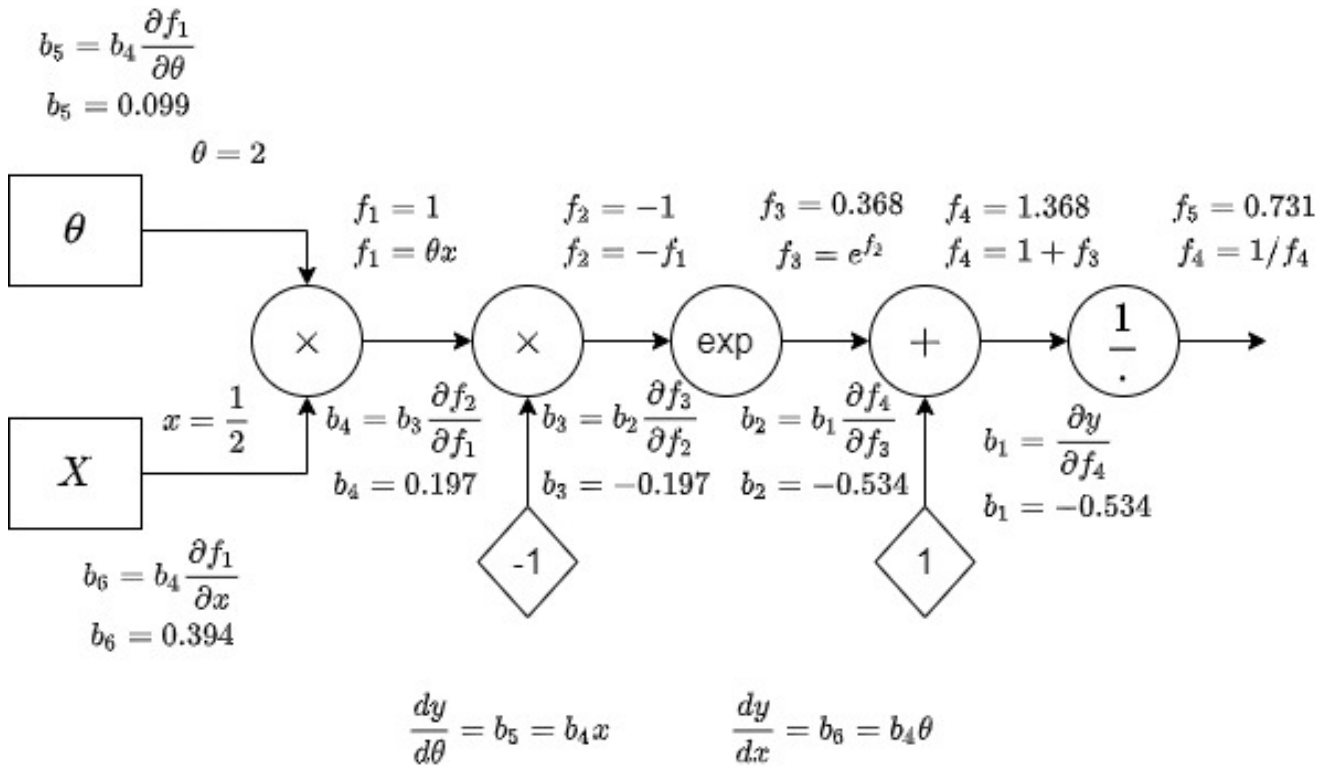


$$\frac{df_1}{d\theta} = \frac{d}{d\theta}\theta x = x$$

Propagación hacia atrás



Propagación hacia atrás



Propagación hacia atrás

- Obsérvese que claculamos $dy/d\theta$ y dy/dx
- con eso, podemos adaptar las entradas en un proceso de descenso de gradiente:

$$\theta' \leftarrow \theta - \lambda \frac{\partial y}{\partial \theta}$$

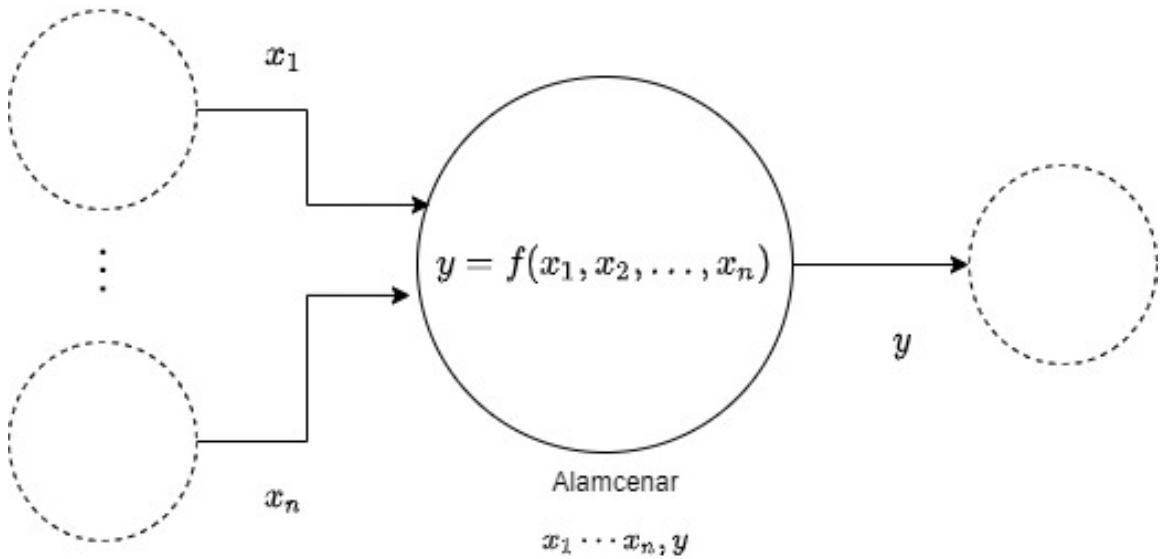
$$x' \leftarrow x - \lambda \frac{\partial y}{\partial x}$$

- Estos cambios en θ y x asegurarían que $y = f(x', \theta') \leq f(x\theta)$
- La iteración de este proceso llevará hasta el mínimo (si λ está bien elegido)

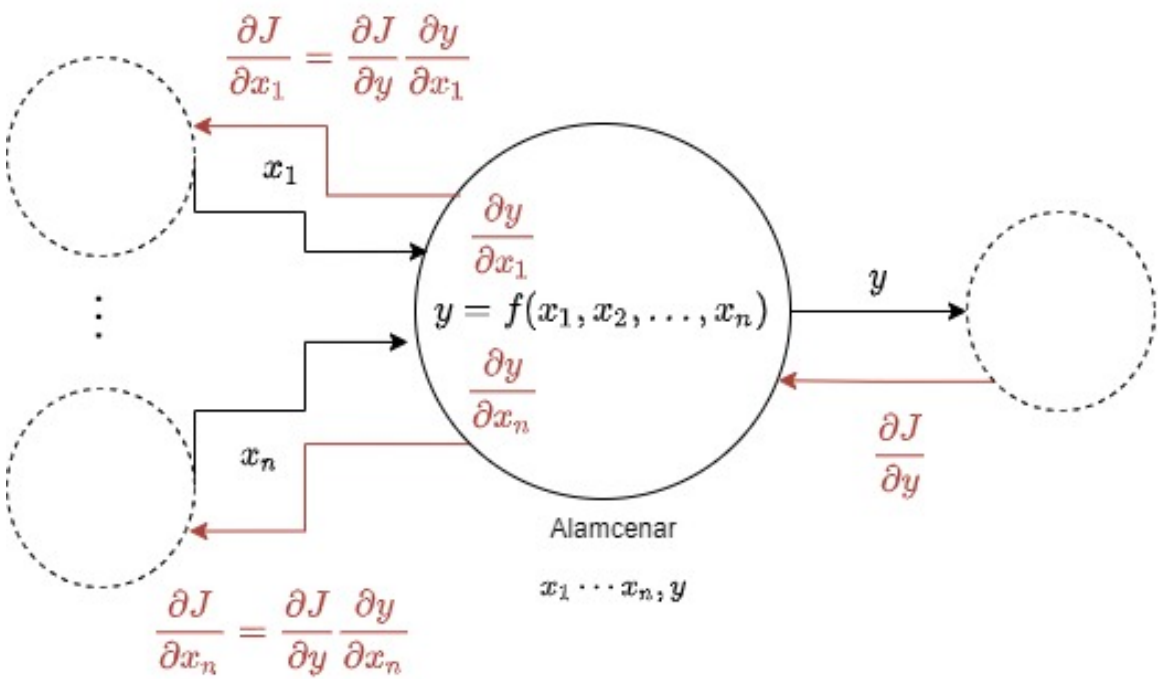
Operación de un nodo

- Comportamiento de nodos en escalares
  - Múltiples entradas y una salida
  - Una entrada y múltiples salidas
  - Múltiples entradas y múltiples salidas
- Generalización a vectores y matrices

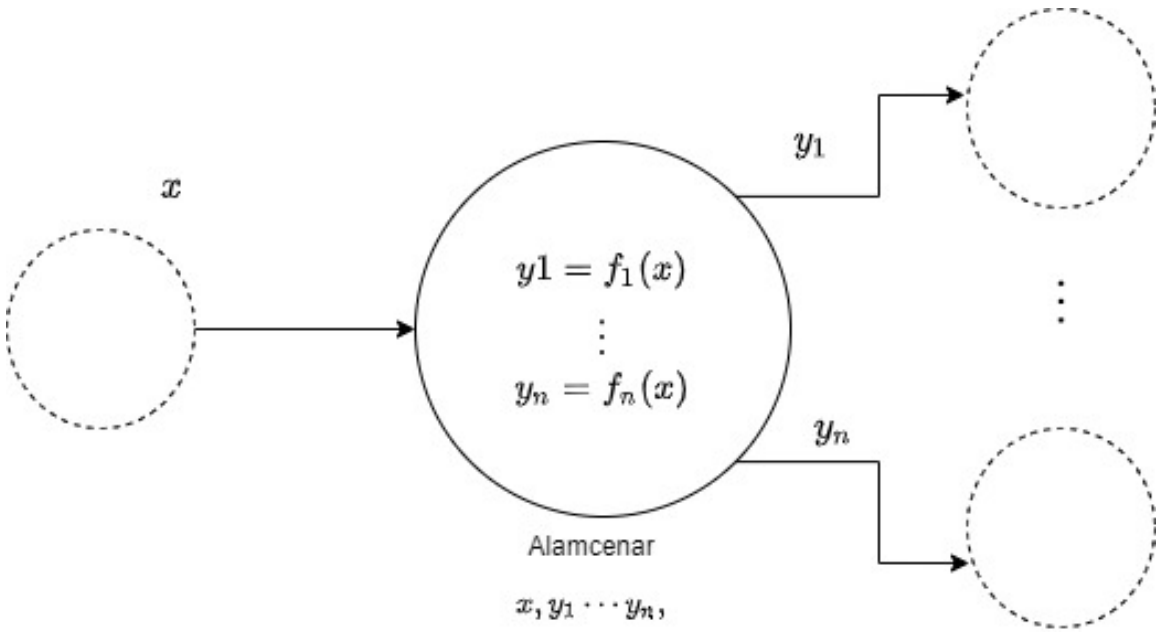
Operación de un nodo



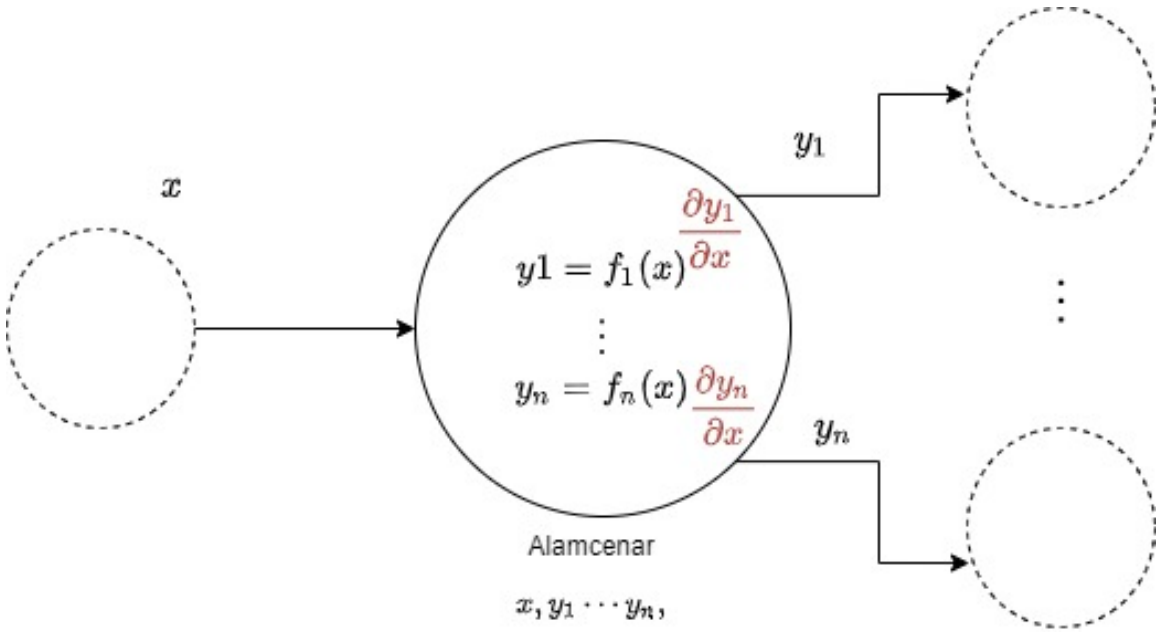
Operación de un nodo



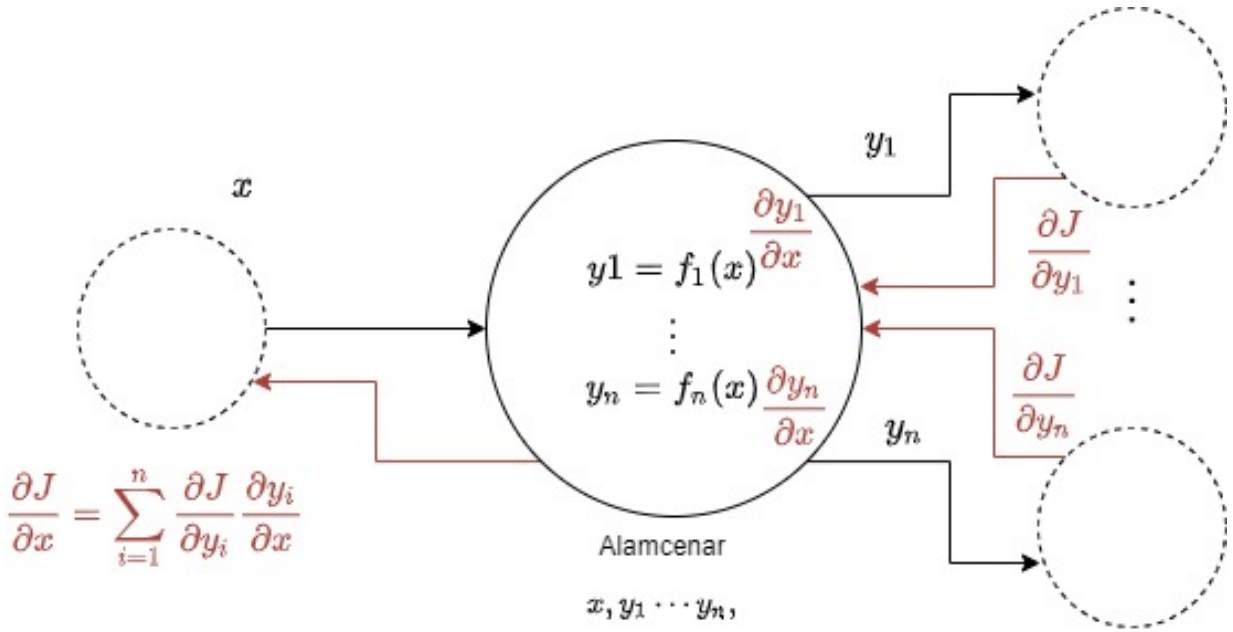
Operación de un nodo



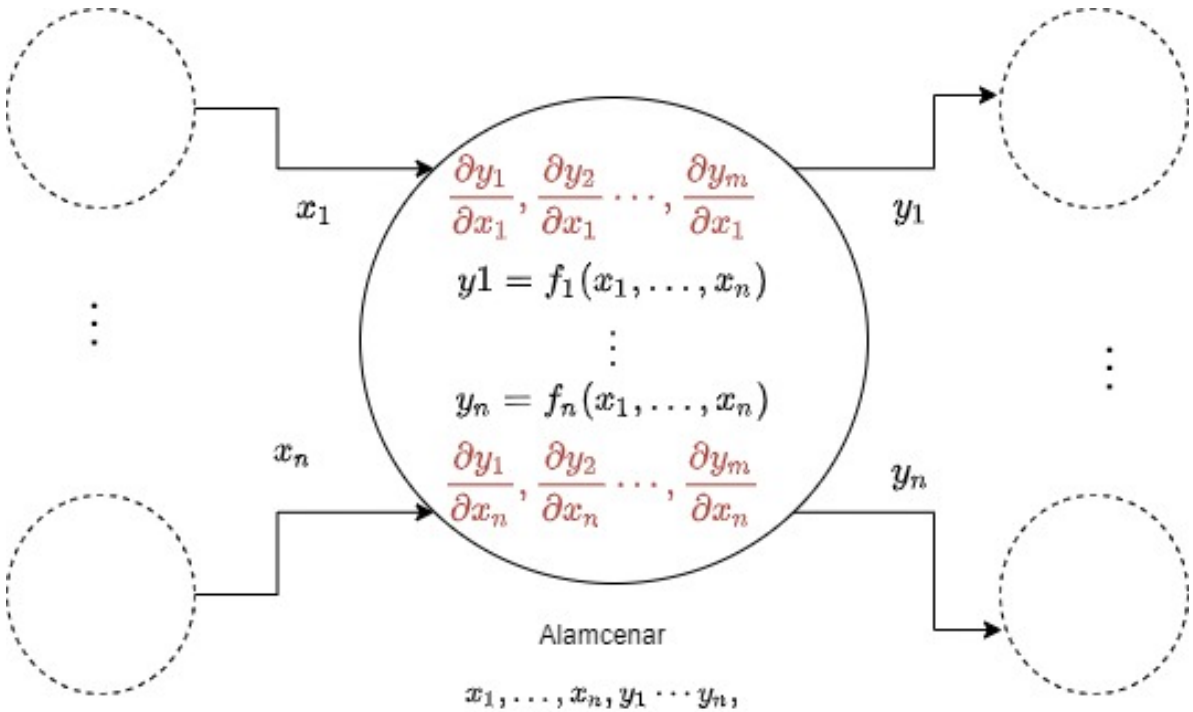
Operación de un nodo



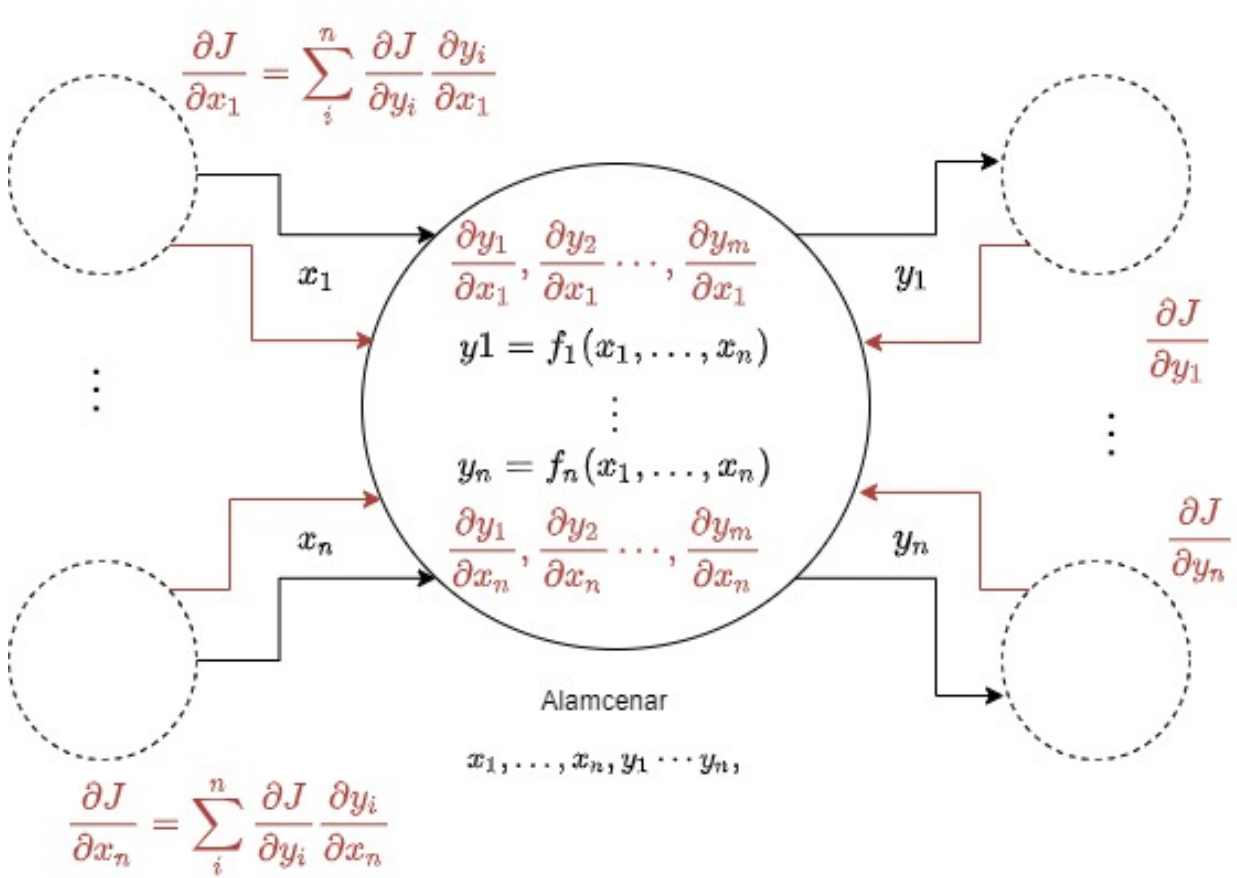
Operación de un nodo



Operación de un nodo



Operación de un nodo



Usando notación vectorial

- El nodo anterior se simplifica usando notación vectorial

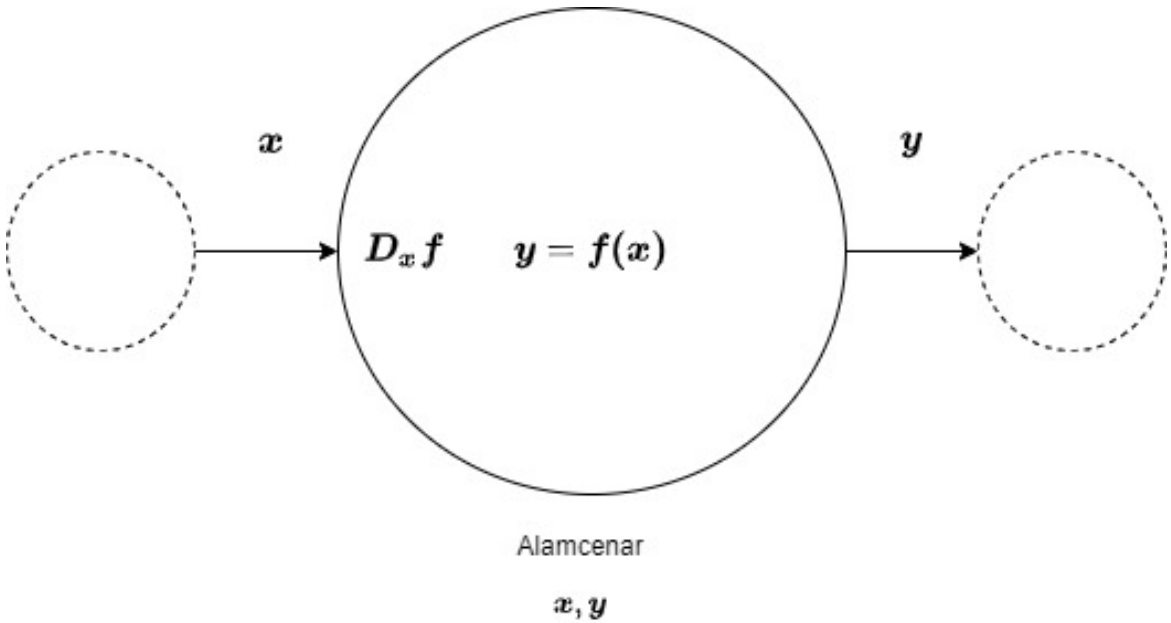
$$\boldsymbol{x} = [x_1 \quad x_2 \cdots x_n]^T$$
$$\boldsymbol{y} = [y_1 \quad y_2 \cdots y_m]^T = [f_1(\boldsymbol{x}) \quad f_2(\boldsymbol{x}) \cdots f_m(\boldsymbol{x})]$$

$$\nabla_{\boldsymbol{x}} J = \left[\frac{\partial J}{\partial x_1} \frac{\partial J}{\partial x_2} \cdots \frac{\partial J}{\partial x_n}\right]^T \quad \nabla_{\boldsymbol{y}} J = \left[\frac{\partial J}{\partial y_1} \frac{\partial J}{\partial y_2} \cdots \frac{\partial J}{\partial y_m}\right]^T$$

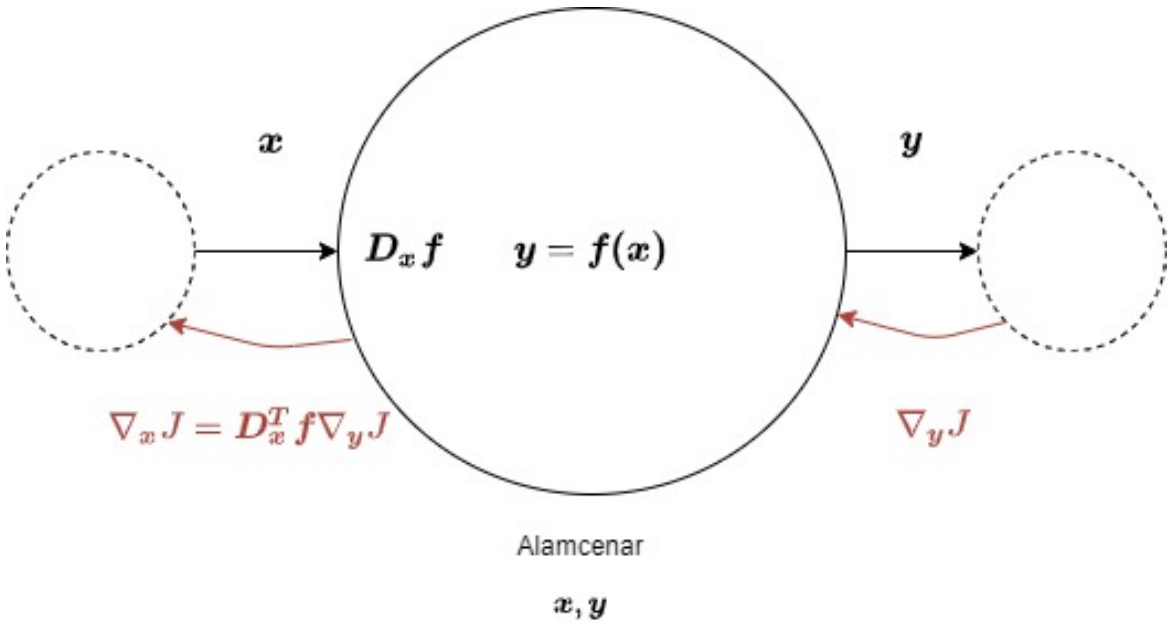
$$\boldsymbol{D}_{\boldsymbol{x}} \boldsymbol{f} = \begin{bmatrix} \nabla_{\boldsymbol{x}}^T f_1(\boldsymbol{x}) \\ \nabla_{\boldsymbol{x}}^T f_2(\boldsymbol{x}) \\ \vdots \\ \nabla_{\boldsymbol{x}}^T f_m(\boldsymbol{x}) \end{bmatrix} = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} & \cdots & \frac{\partial f_2}{\partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial f_m}{\partial x_1} & \frac{\partial f_m}{\partial x_2} & . & \frac{\partial f_m}{\partial x_n} \end{bmatrix}$$

$$\nabla_{\boldsymbol{x}} J = \boldsymbol{D}_{\boldsymbol{x}}^T \boldsymbol{f} \nabla_y J$$

Operación de un nodo, vectorial



Operación de un nodo, vectorial



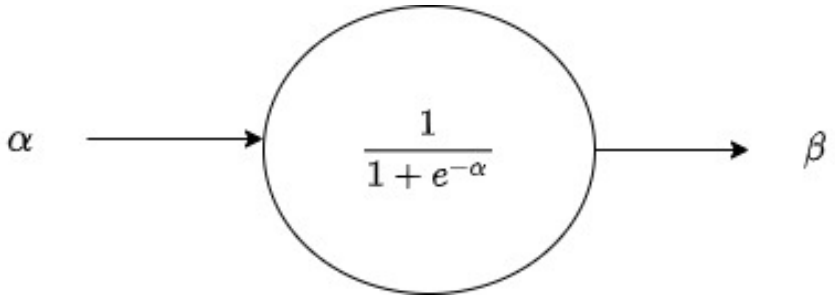
Consideraciones prácticas

- $\boldsymbol{D}_{\boldsymbol{x}}^{\boldsymbol{T}}\boldsymbol{f}$ es el jacobiano de $\boldsymbol{f}$ respecto $\boldsymbol{x}$
- La ecuación $\nabla_{\boldsymbol{x}} J = \boldsymbol{D}_{\boldsymbol{x}}^{\boldsymbol{T}}\boldsymbol{f} \nabla_{\boldsymbol{y}} J$ integra todas las posibles influencias de cada salida y cada entrada, y aplica la regla de la cadena para ello.
- con frecuencia sólo existen algunas dependencias y el Jacobiano $\boldsymbol{D}_{\boldsymbol{x}}^{\boldsymbol{T}}\boldsymbol{f}$ es disperso (tiene muchos ceros).
  - Si y_i solo depende de x_i entonces el Jacobiano es diagonal.
- En la práctica se calcula $\nabla_{\boldsymbol{x}} J$, que es lo que se retropropaga a los nodos precedentes.
- Por eso el nombre: retro-propagación (backpropagation)

Generalización

- Observe que $\boldsymbol{f(x)}$ puede ser cualquier función
- $\boldsymbol{f(x)}$ agrupa varios cálculos en un nodo
- Granularidad según complejidad deseada para cada nodo
- Resumiendo: cada nodo tiene tres tareas:
  - propagar entrada a salida
  - almacenar valores temporales para calcular gradientes
  - calcula gradiente de J respecto a entradas del nodo, combinando información del gradiente J respecto a las salidas y derivadas/gradientes de salidas respecto a entradas
- Esto último es conceptual. En la práctica se resume el cálculo
- Si $\boldsymbol{x}$ o $\boldsymbol{y}$ fuese matrices, se analiza cada elemento de la matriz por separado, $\nabla_{\boldsymbol{x}} J$ o $\nabla_{\boldsymbol{y}} J$ serían matrices.

Ejemplo: Sigmoide

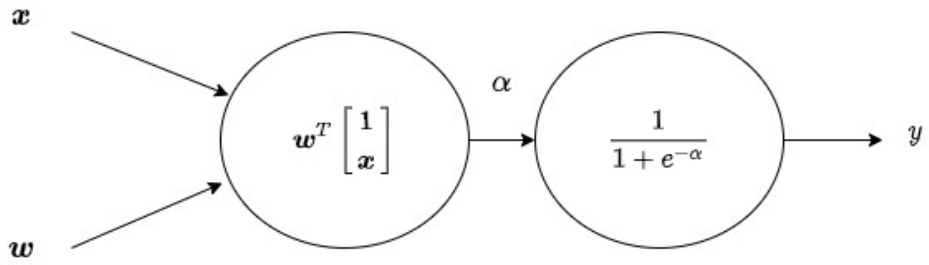


- Entrada: α
- Salida: β
- Cálculo en propagación hacia adelante: $\frac{1}{1+e^{-\alpha}}$
- Cálculo en propagación hacia atrás:

$$\frac{\partial \beta}{\partial \alpha} = \frac{e^{-\alpha}}{(1+e-\alpha)^2} = \frac{e^{-\alpha}}{(1+e-\alpha)} + \frac{1}{(1+e-\alpha)}$$
$$= \frac{1+e^{-\alpha}-1}{(1+e-\alpha)}\beta = \left(\frac{1+e^{-\alpha}}{1+e^{-\alpha}} - \frac{1}{1+e^{-\alpha}}\right) = (1-\beta)\beta$$

- Basta con almacenar β para calcular $d\beta/d\alpha$

Ejemplo: Neurona totalmente conectada con sesgo

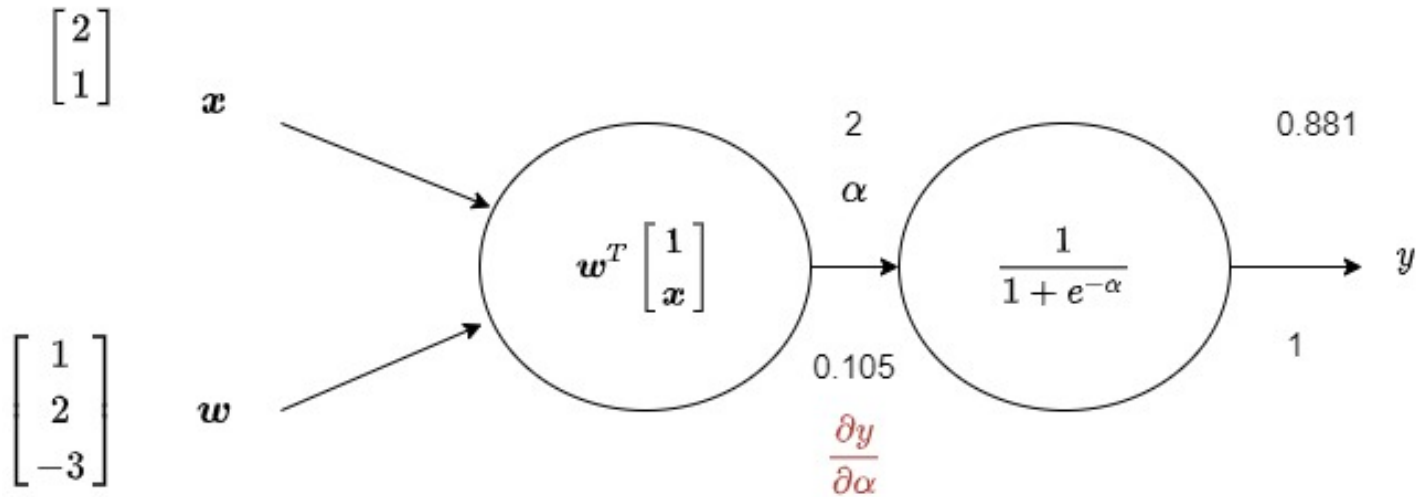


- primer nodo extiende vector $\boldsymbol{x}$ con 1 para poder aplicar el sesgo w_0 (primer componente de $\boldsymbol{w}$)

$$\alpha = \begin{pmatrix} w_0 \\ +w_1 x_1 \\ +w_2 x_2 \\ \vdots \\ +w_n x_n \end{pmatrix} \Rightarrow \boldsymbol{D}_{\boldsymbol{w}}^{\boldsymbol{T}} \alpha = \nabla_{\boldsymbol{w}} \alpha = \begin{bmatrix} 1 \\ \boldsymbol{x} \end{bmatrix}$$

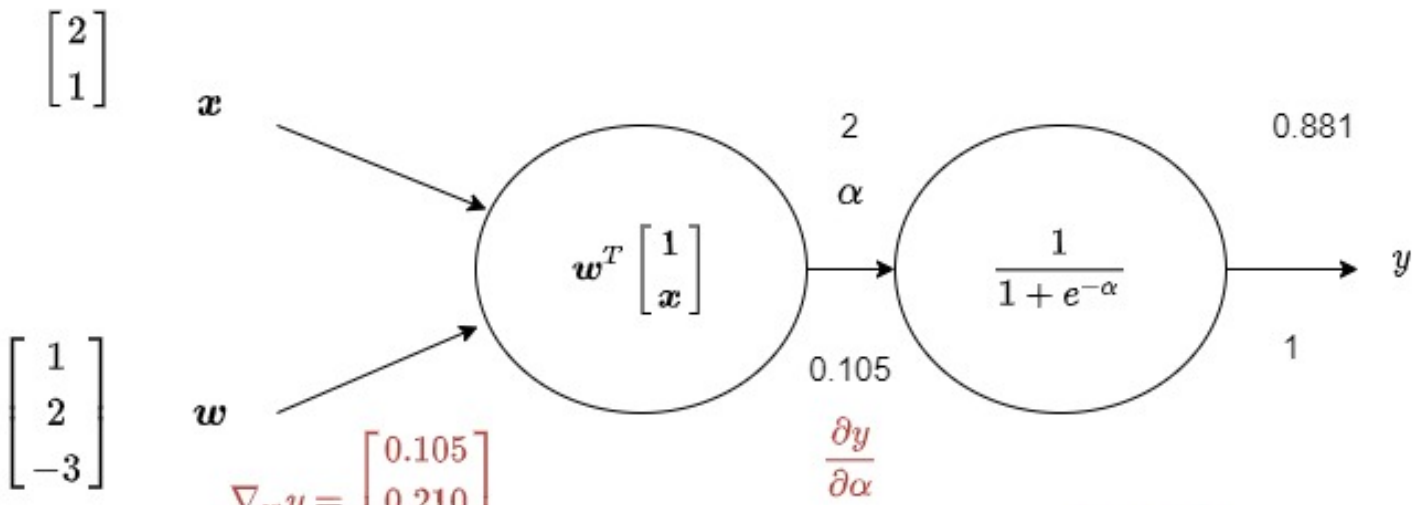
$$\boldsymbol{D}_{\boldsymbol{x}}^{\boldsymbol{T}} \alpha = \nabla_{\boldsymbol{x}} \alpha = \boldsymbol{w} = \begin{pmatrix} w_1 \\ w_2 \\ \vdots \\ w_n \end{pmatrix}$$

Ejemplo: Neurona totalmente conectada con sesgo

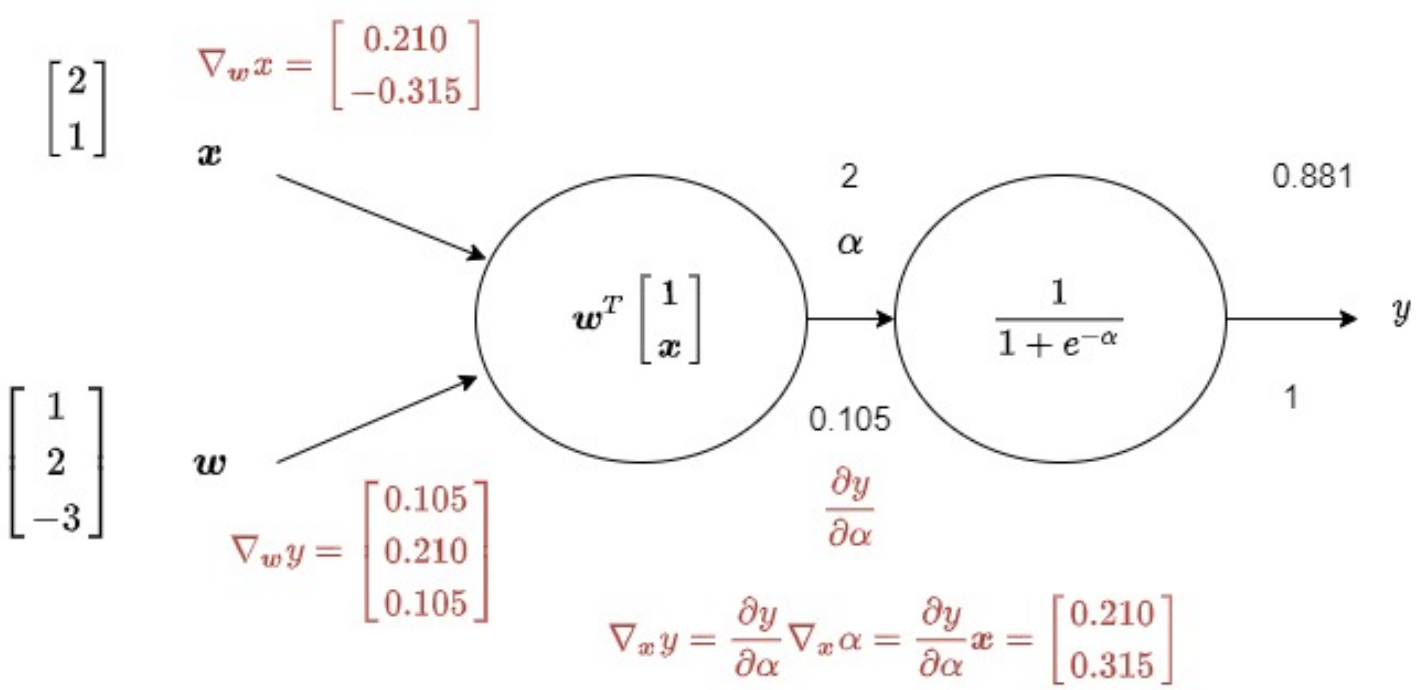


$$\frac{\partial y}{\partial \alpha} = (1-y)y = 0.105$$

Ejemplo: Neurona totalmente conectada con sesgo



Ejemplo: Neurona totalmente conectada con sesgo



Compuertas

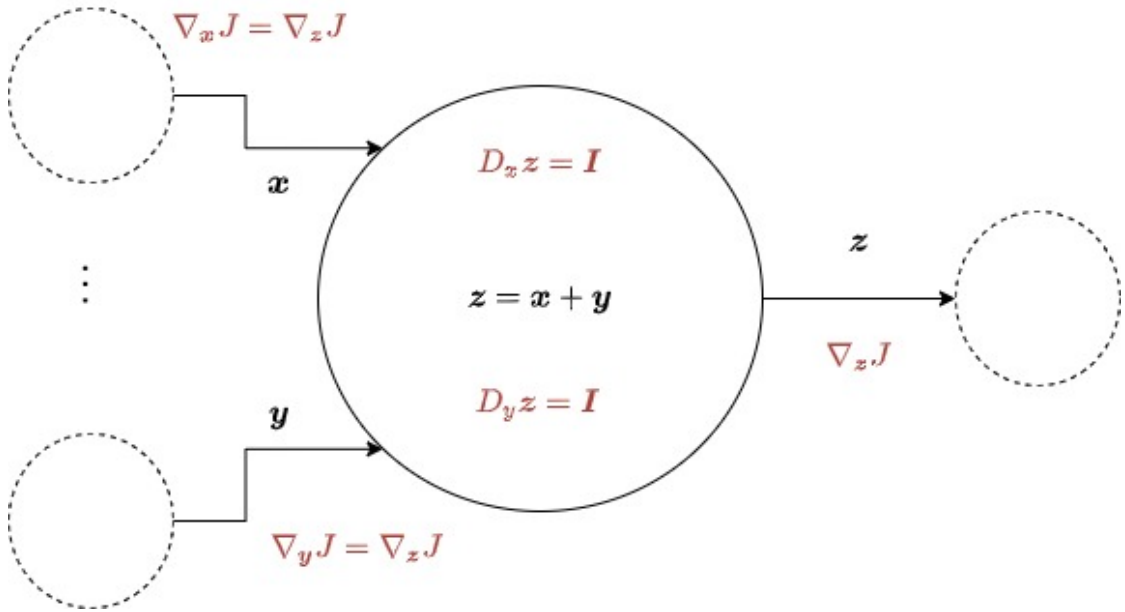
- Los nodos de un grafo computacional se comportan como compuertas
- A continuación se analizarán algunos casos comunes

Compuertas

- Los nodos de un grafo computacional se comportan como compuertas
- A continuación se analizarán algunos casos comunes

Suma

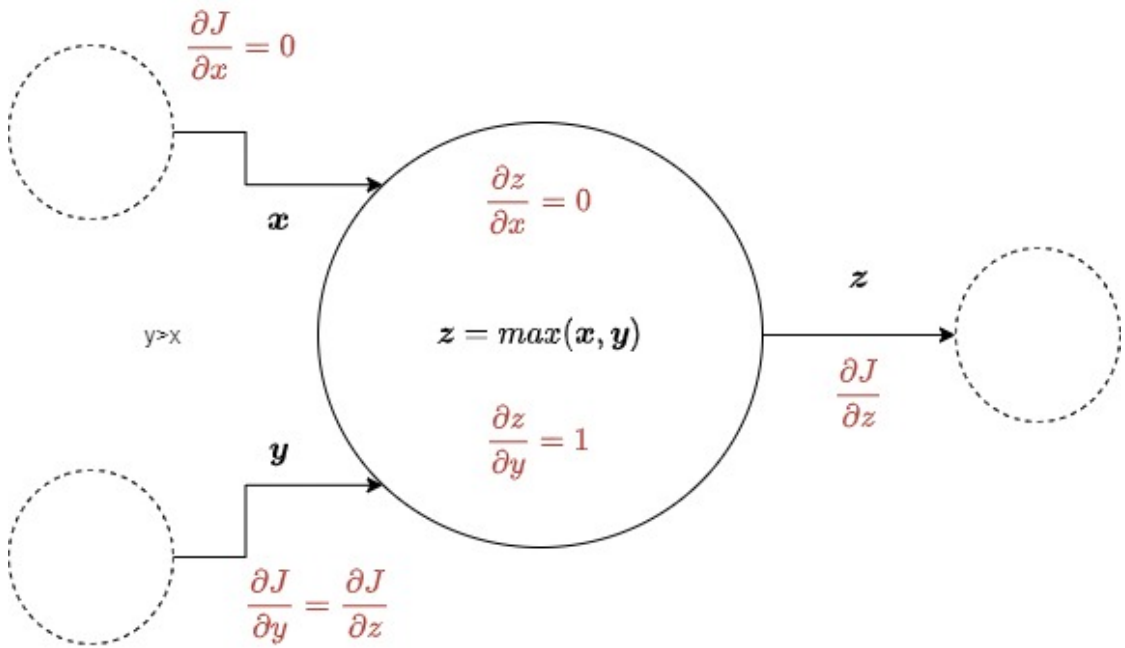
- La suma distribuye el gradiente a todas las entradas



Máximo

- El nodo $z = max(x, y)$ produce el máximo de variables de entrada
- la pregunta es ¿qué hace la derivada?
- la interpretación es la siguiente
  - si $x > y$, entonces $z = x$ y por tanto $\frac{\partial z}{\partial x} = 1$ y $\frac{\partial z}{\partial y} = 0$
  - si $x < y$, entonces $z = y$ y por tanto $\frac{\partial z}{\partial x} = 0$ y $\frac{\partial z}{\partial y} = 1$
- La compuerta *max* enruta entonces el gradiente por aquella entrada con el máximo valor
- Esto se generaliza fácilmente a vectores componente por componente.

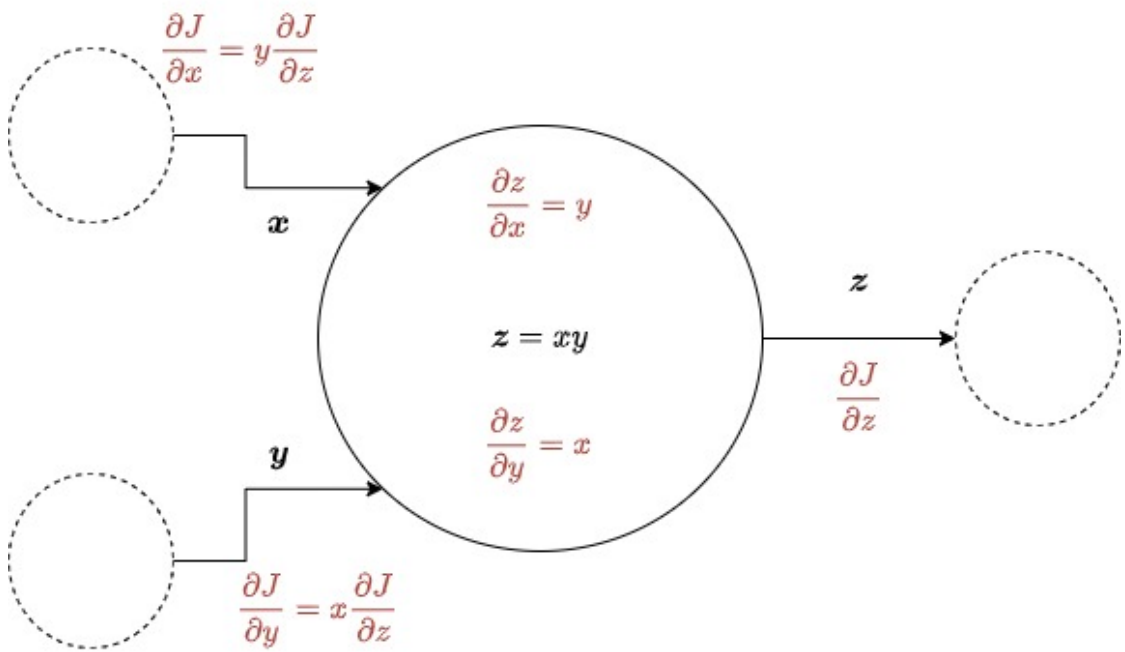
Máximo



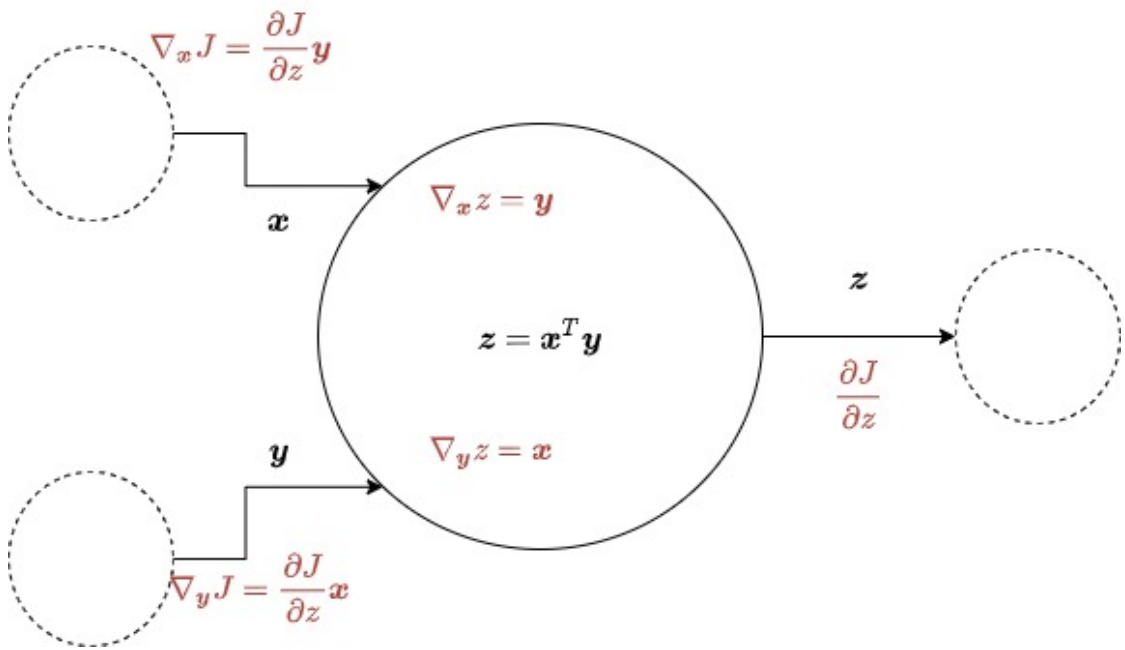
Conmutadores

- Los distintos tipos de producto intercambian de algún modo u otro las entradas en la propagación hacia atrás.
- Solo en algunos casos es posible calcular el jacobiano
- A continuación analizaremos cuatro casos:
  - Producto de dos escalares $z = xy$
  - Producto interno de dos vectores $z = \boldsymbol{x}^T \boldsymbol{y}$
  - Producto externo de dos vectores $\boldsymbol{Z} = \boldsymbol{x}\boldsymbol{y}^T$
  - Producto de dos matrices $\boldsymbol{Z} = \boldsymbol{X}\boldsymbol{Y}$

Producto escalar



Producto interno de dos vectores



Producto interno de dos vectores

- El producto externo produce una matriz a la salida
- Supongamos $\boldsymbol{x} \in \mathbb{R}^n$ e $\boldsymbol{y} \in \mathbb{R}^m$

$$\boldsymbol{Z} = \begin{bmatrix} z_{11} & z_{12} & \cdots & z_{1m} \\ z_{21} & z_{22} & \cdots & z_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ z_{n1} & z_{n2} & \cdots & z_{nm} \end{bmatrix} = \boldsymbol{x}\boldsymbol{y}^T = \begin{bmatrix} x_1 y_1 & x_1 y_2 & \cdots & x_1 y_m \\ x_2 y_1 & x_2 y_2 & \cdots & x_2 y_m \\ \vdots & \vdots & \ddots & \vdots \\ x_n y_1 & x_n y_2 & \cdots & x_n y_m \end{bmatrix}$$

- siguendo el principio de múltiples salidas (elementos de $\boldsymbol{Z}$)

$$\frac{\partial J}{\partial x_k} = \sum_i \sum_j \frac{\partial J}{\partial z_{ij}} \frac{\partial z_{ij}}{\partial x_k} = \sum_i \sum_j \frac{\partial J}{\partial z_{ij}} \frac{\partial (x_i y_j)}{\partial x_k}$$

donde solo los términos con $i = k$ sobreviven

$$\frac{\partial J}{\partial x_k} = \sum_j \frac{\partial J}{\partial z_{kj}} y_j \Rightarrow \nabla_{\boldsymbol{x}} J = \nabla_{\boldsymbol{Z}} J \boldsymbol{y}$$

Gracias

Gamaliel Moreno Chávez