Tema 2.4

Implementación Matricial

Deep Learning

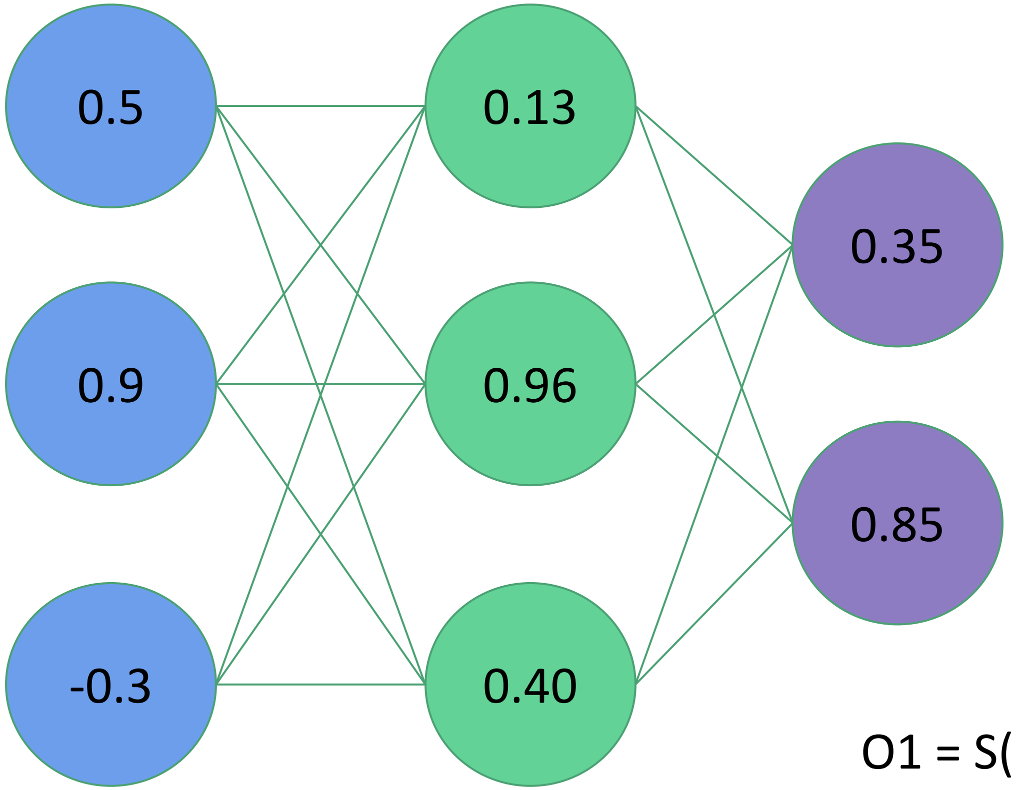
Máster Oficial en Ingeniería Informática

Universidad de Sevilla

Contenido

- Formulación matricial
- Propagación de gradiente con vectorización
- Ejemplo completo

Formulación matricial

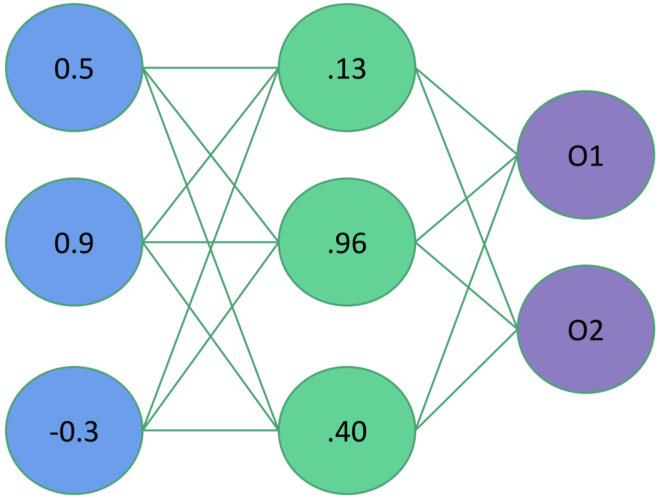


H1 Weights = (1.0, -2.0, 2.0)
H2 Weights = (2.0, 1.0, -4.0)
H3 Weights = (1.0, -1.0, 0.0)

O1 Weights = (-3.0, 1.0, -3.0)
O2 Weights = (0.0, 1.0, 2.0)

O1 = S(0.13 * -3.0 + 0.96 * 1.0 + 0.40 * -3.0) = S(-.63) = 0.35
O2 = S(0.13 * 0.0 + 0.96 * 1.0 + 0.40 * 2.0) = S(1.76) = 0.85

Formulación matricial



H1 Weights = (1.0, -2.0, 2.0)
H2 Weights = (2.0, 1.0, -4.0)
H3 Weights = (1.0, -1.0, 0.0)

Hidden Layer Weights, Inputs, Hidden Layer Outputs

$$S\left(\begin{bmatrix} 1.0 & -2.0 & 2.0 \\ 2.0 & 1.0 & -4.0 \\ 1.0 & -1.0 & 0.0 \end{bmatrix} * \begin{bmatrix} 0.5 \\ 0.9 \\ -0.3 \end{bmatrix}\right) = S\left(\begin{bmatrix} -1.9 & 3.1 & -0.4 \end{bmatrix}\right) = \begin{bmatrix} 0.13 & 0.96 & 0.4 \end{bmatrix}$$

Formulación matricial

- Una capa oculta (k_h neuronas)

$$o = f(X) = S\left(\sum_{i=0}^{n}\sum_{j=0}^{k_h} w_{ij}\, x_i\right) = S(W * X^T)$$

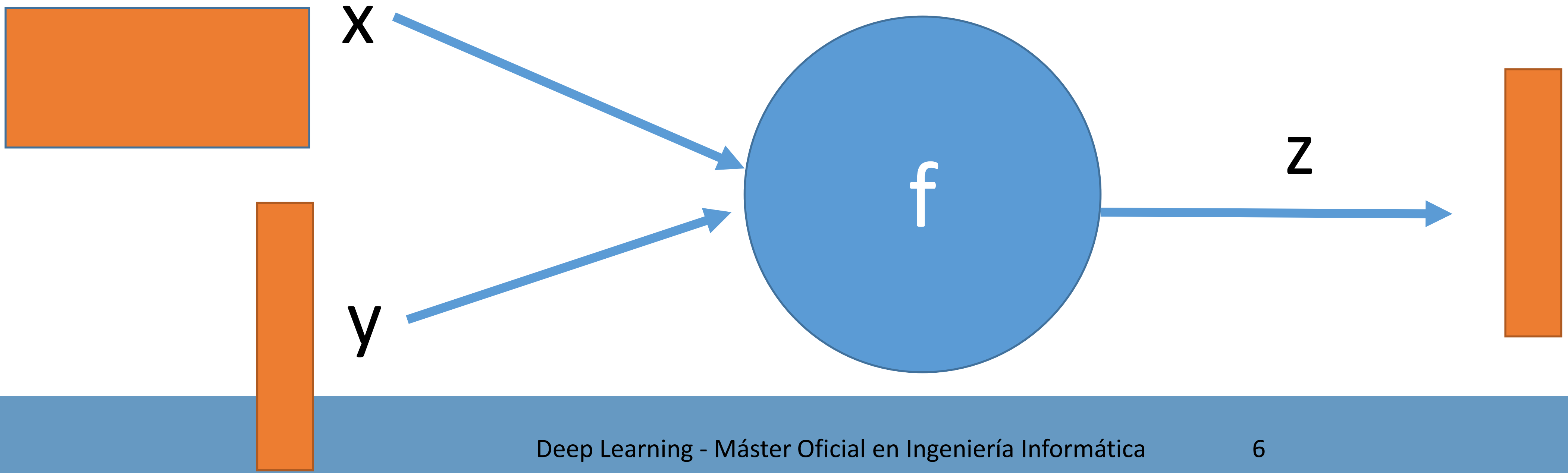
w_{10}	w_{11}	w_{12}	w_{13}	w_{14}	w_{15}	w_{16}
w_{20}	w_{21}	w_{22}	w_{23}	w_{24}	w_{25}	w_{26}
w_{30}	w_{31}	w_{32}	w_{33}	w_{34}	w_{35}	w_{36}
w_{40}	w_{41}	w_{42}	w_{43}	w_{44}	w_{45}	w_{46}

x_0	x_1	x_2	x_3	x_4	x_5	x_6

S(W * X) = o

Propagación de gradiente con vectorización

- Si usamos **formulación matricial**, las funciones reciben vectores y matrices, no escalares.
- Normalmente los valores de entrada de una red no son escalares, sino **tensores** (múltiples dimensiones, como imágenes, vídeo…)



Propagación de gradiente con vectorización

- Hay que tener en cuenta que el cálculo diferencial cambia
- Para f(x) = y, según el tipo de x e y:
  - Si **x** e **y** son escalares: **derivada** de f respecto de x
    - Si x cambia un poco, ¿cuánto cambia y?

$$x \in \mathbb{R}, y \in \mathbb{R}$$

$$\frac{\partial y}{\partial x} \in \mathbb{R}$$

  - Si **x** es un vector, **y** es un escalar: **gradiente** de f respecto de x
    - Para cada elemento de x, si cambia un poco, ¿cuánto cambia y?

$$x \in \mathbb{R}^N, y \in \mathbb{R}$$

$$\frac{\partial y}{\partial x} \in \mathbb{R}^N \quad \left(\frac{\partial y}{\partial x}\right)_n = \frac{\partial y}{\partial x_n}$$

  - Si **x** e **y** son vectores: **jacobiana** de f respecto de x
    - Para cada elemento de x, si cambia un poco, ¿cuánto cambia cada elemento de y?

$$x \in \mathbb{R}^N, y \in \mathbb{R}^M$$

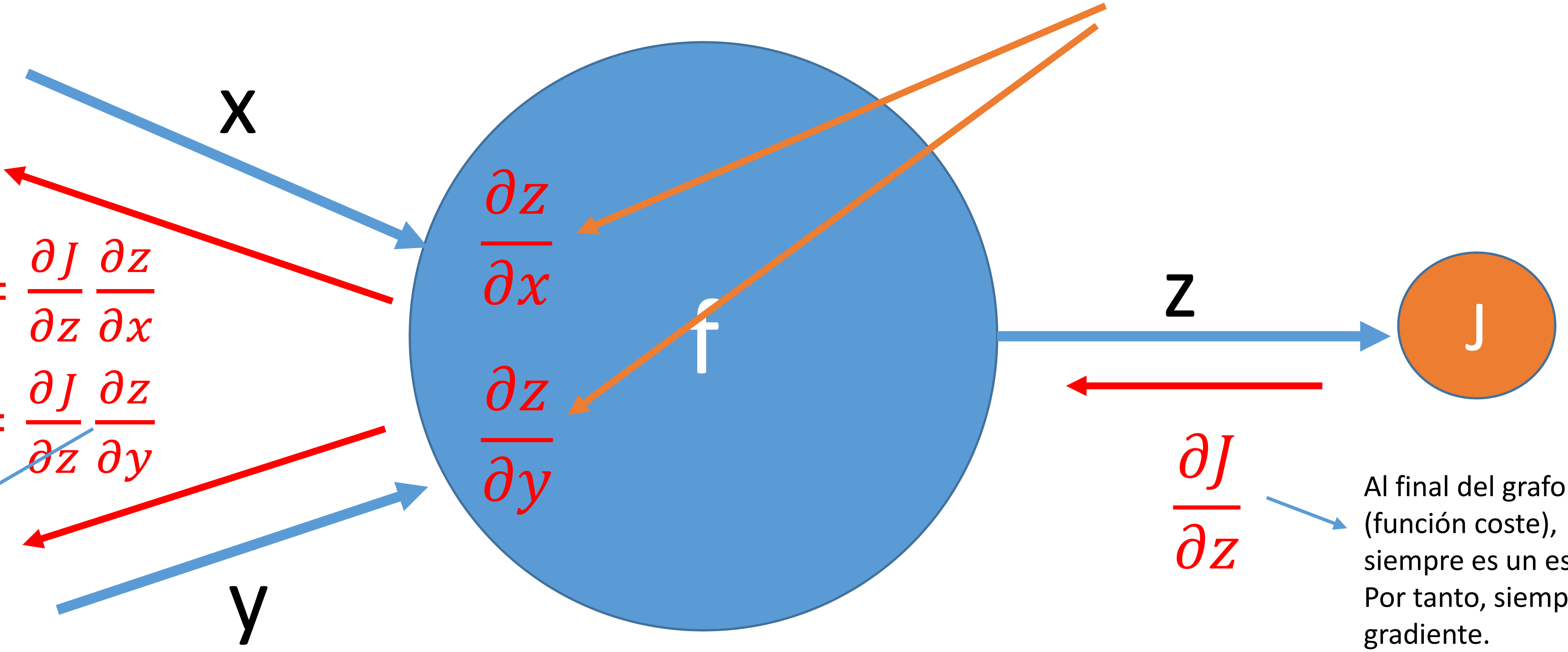
$$\frac{\partial y}{\partial x} \in \mathbb{R}^{N \times M} \quad \left(\frac{\partial y}{\partial x}\right)_{n,m} = \frac{\partial y_m}{\partial x_n}$$

Propagación de gradiente con vectorización

- ¿Qué ocurre cuando las datos son vectores? Matrices jacobianas.

$$\frac{\partial J}{\partial x} = \frac{\partial J}{\partial z}\frac{\partial z}{\partial x}$$

$$\frac{\partial J}{\partial y} = \frac{\partial J}{\partial z}\frac{\partial z}{\partial y}$$



Producto elemento a elemento (Hadamard)

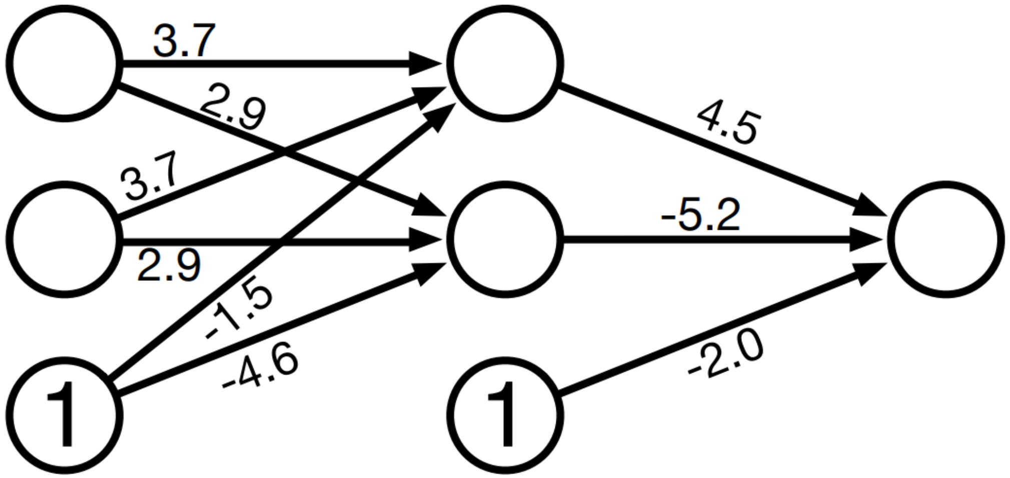
Al final del grafo (función coste), siempre es un escalar. Por tanto, siempre es gradiente.

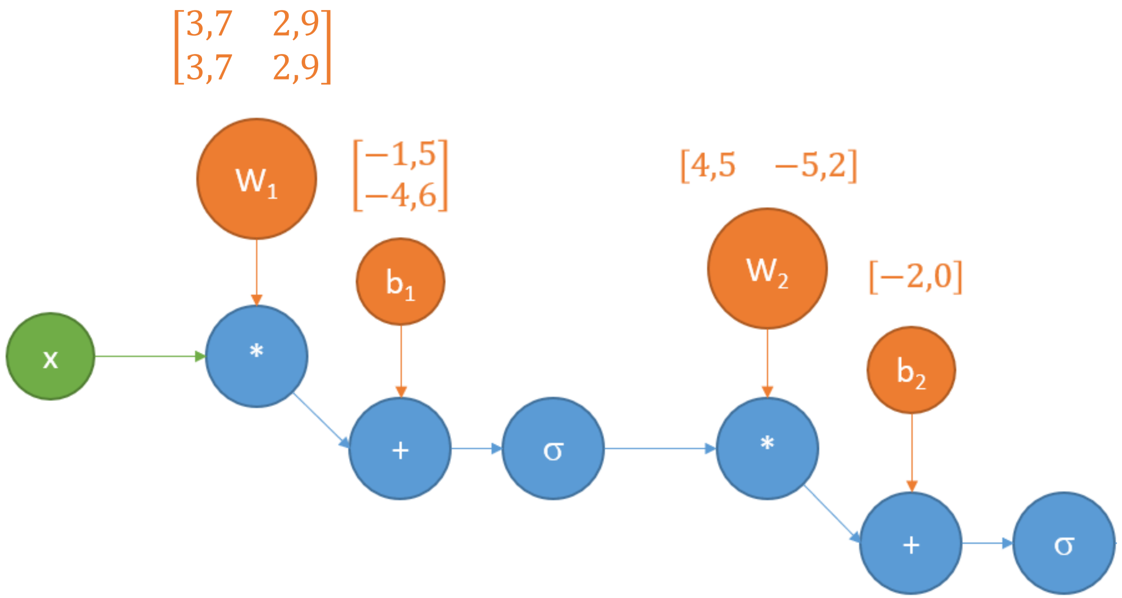
Propagación de gradiente con vectorización

- Dado que para la mayoría de funciones usadas en redes neuronales, las matrices jacobianas son muy **dispersas** (no todos los elementos de x afectan a todos los elementos de y).
  - Es decir, la mayoría de elementos son 0.
  - El cálculo se simplifica y no hace falta usar una matriz jacobiana de forma explícita.
- Por ejemplo, para **multiplicación** de matrices $y = f(X) = W * X$, sea G el gradiente que nos llega de la capa siguiente. La propagación es:
  - $\frac{\partial y}{\partial x} = W^T * G, \quad \frac{\partial y}{\partial W} = G * X^T$ (sigue siendo un intercambiador de gradientes)

Ejemplo completo

- Veamos un ejemplo de una red usando formulación matricial y grafo computacional:

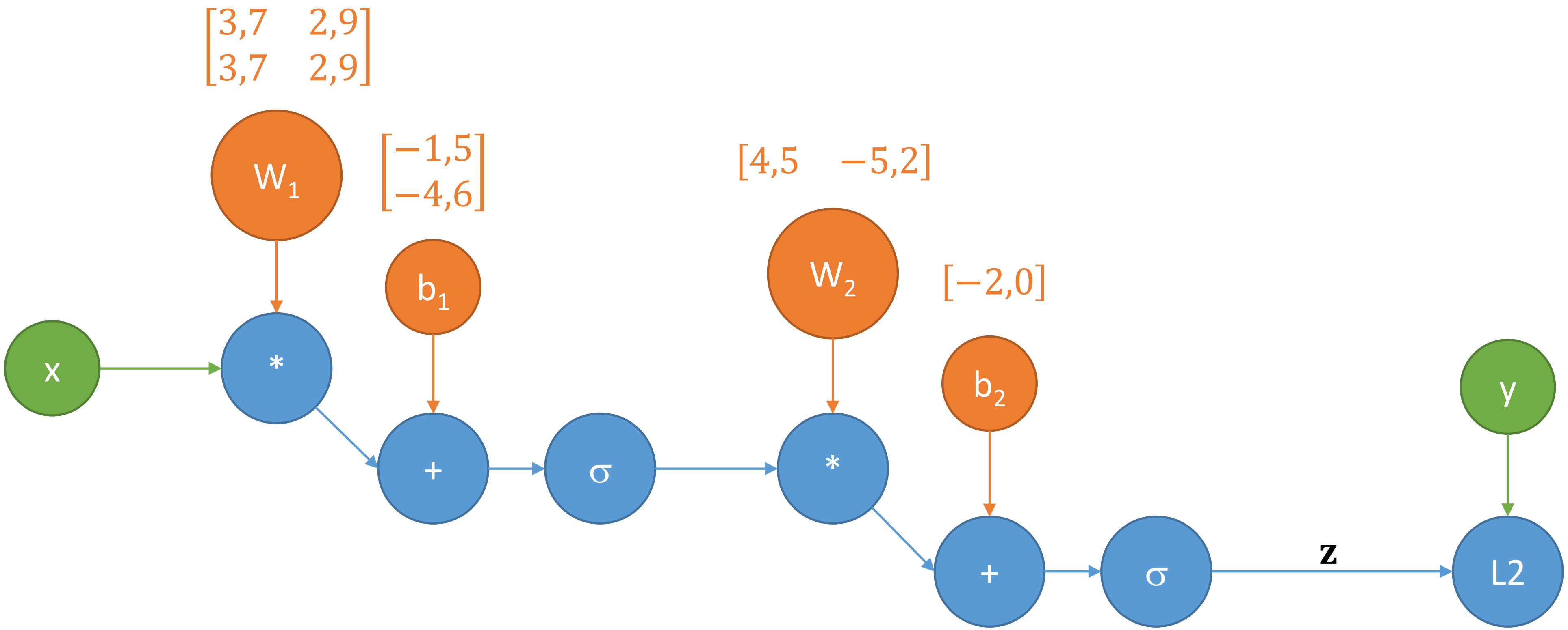




Ejemplo Philipp Koehn

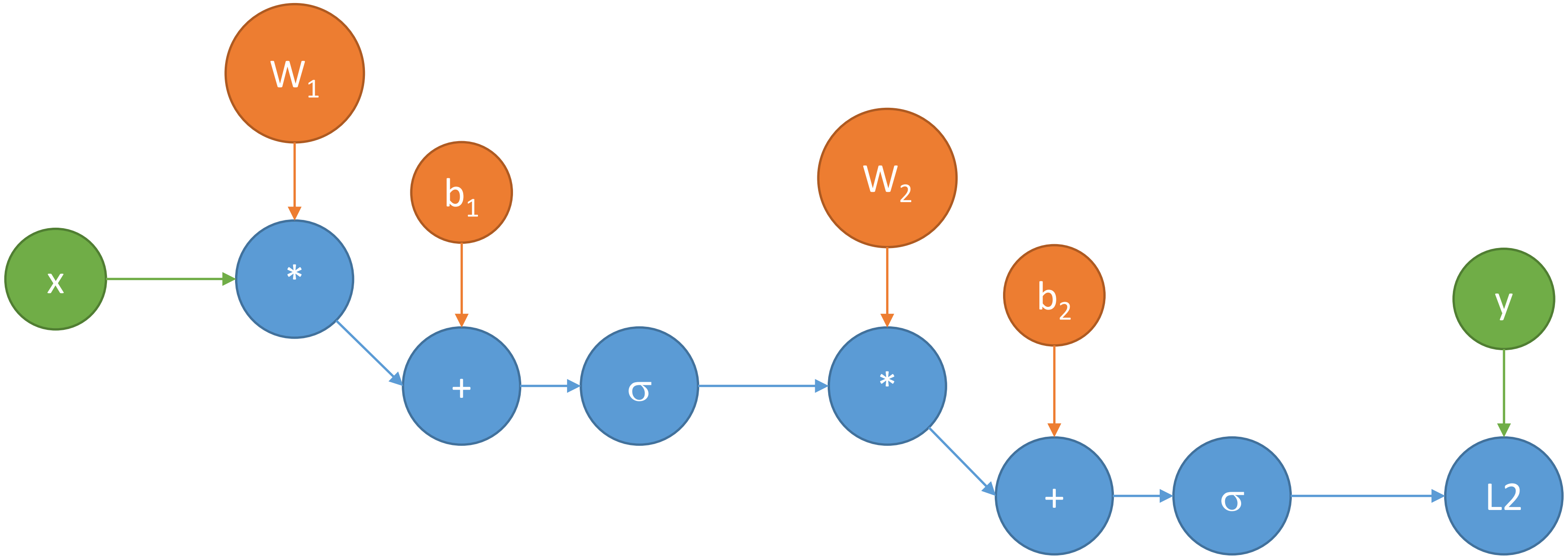
Ejemplo completo

- Función pérdida con error cuadrático medio: $\mathrm{L2}(\mathrm{y},\mathrm{z}) = \frac{1}{2}(z-y)^2$



Ejemplo completo

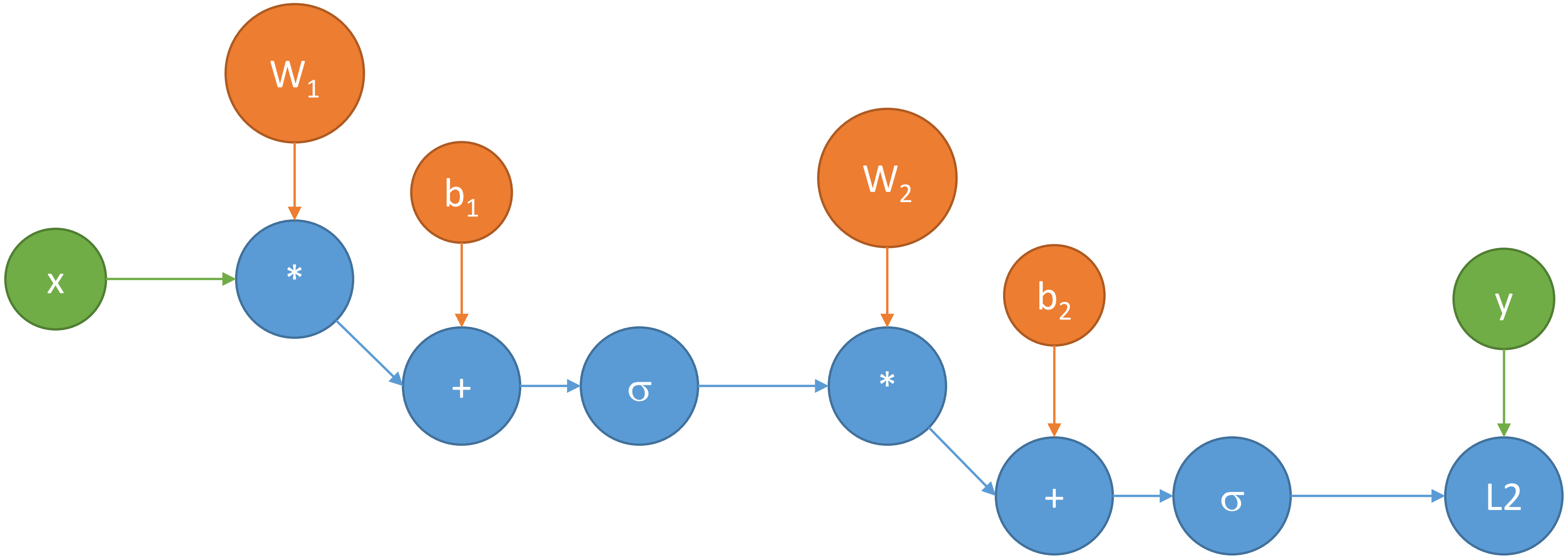
- Cálculo de los gradientes locales (se hace solo una vez al comienzo):



$$\frac{\partial L2}{\partial z} = \frac{\partial}{\partial z}\frac{1}{2}(z-y)^2 = z - y$$

Ejemplo completo

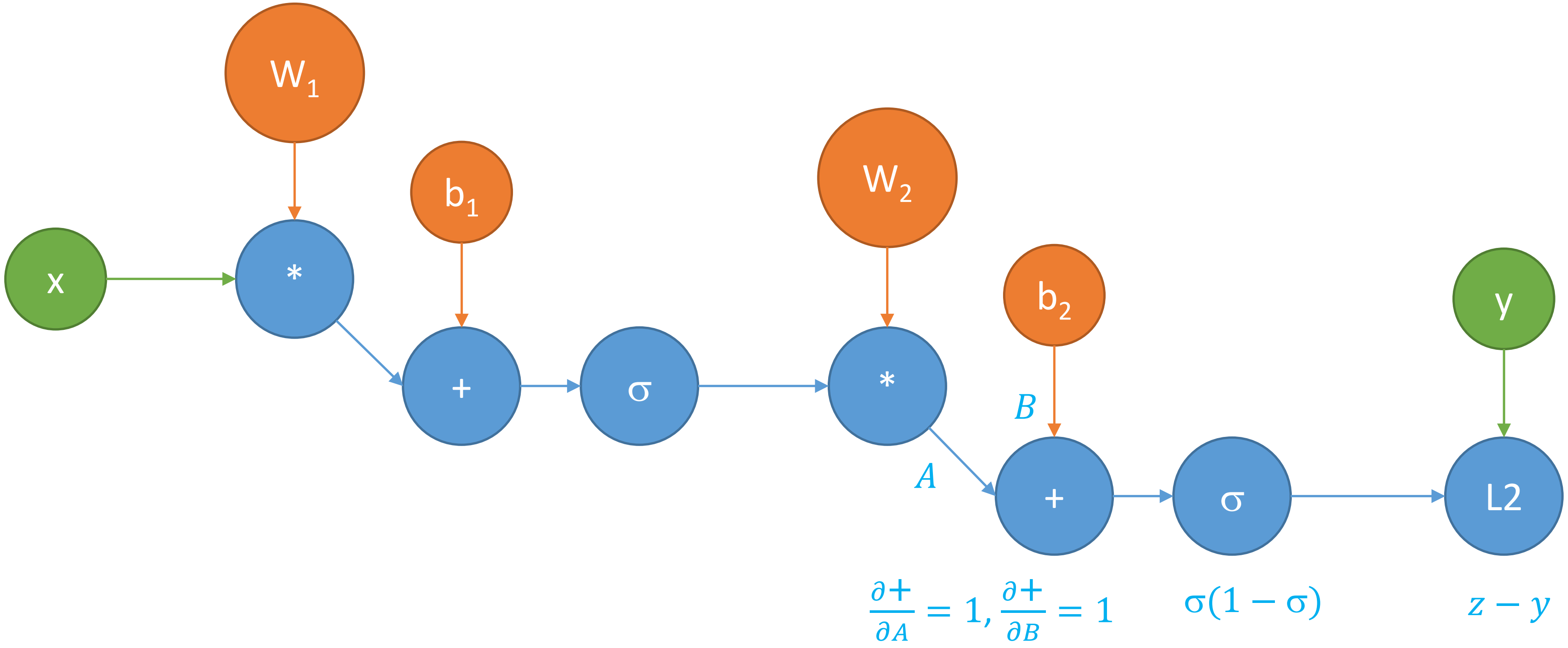
- Cálculo de los gradientes locales (se hace solo una vez al comienzo):



$$\frac{\partial \sigma}{\partial +} = \sigma(1-\sigma) \qquad z - y$$

Ejemplo completo

- Cálculo de los gradientes locales (se hace solo una vez al comienzo):

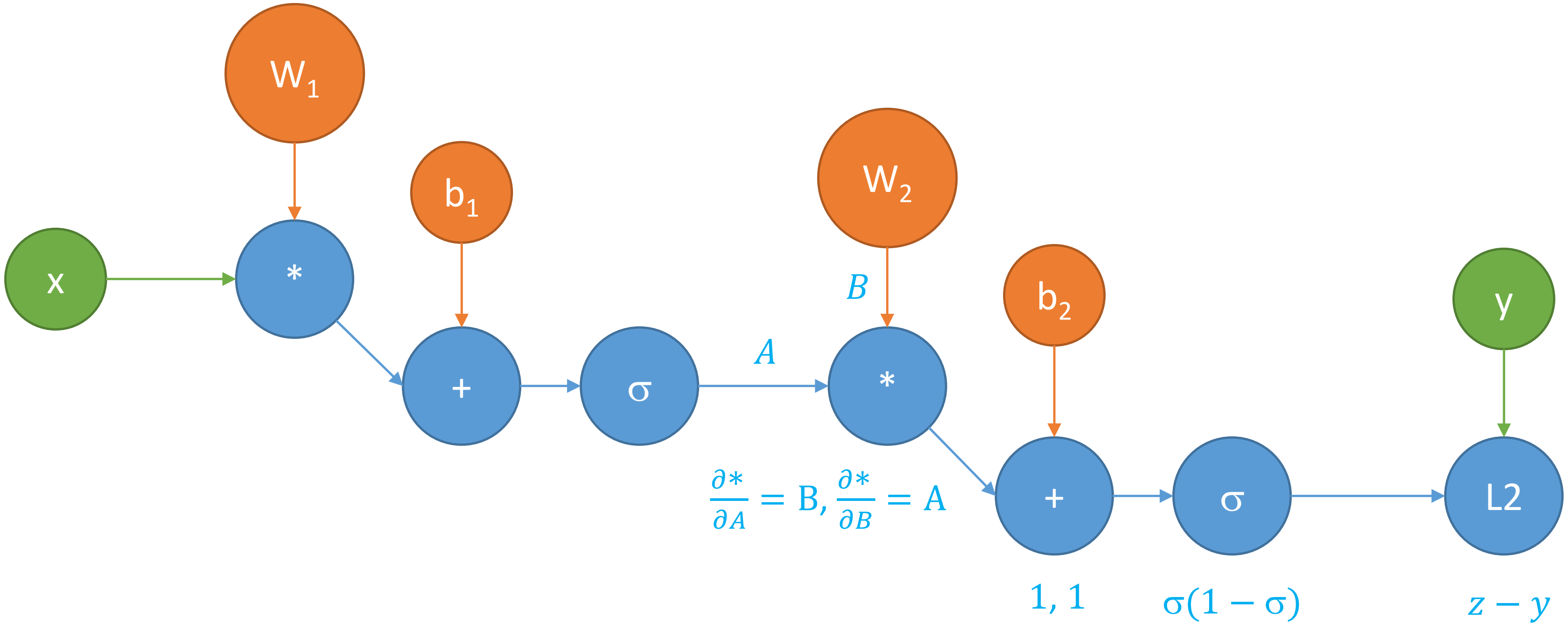


Escuela Técnica Superior de Ingeniería Informática

Máster Universitario en Ingeniería Informática

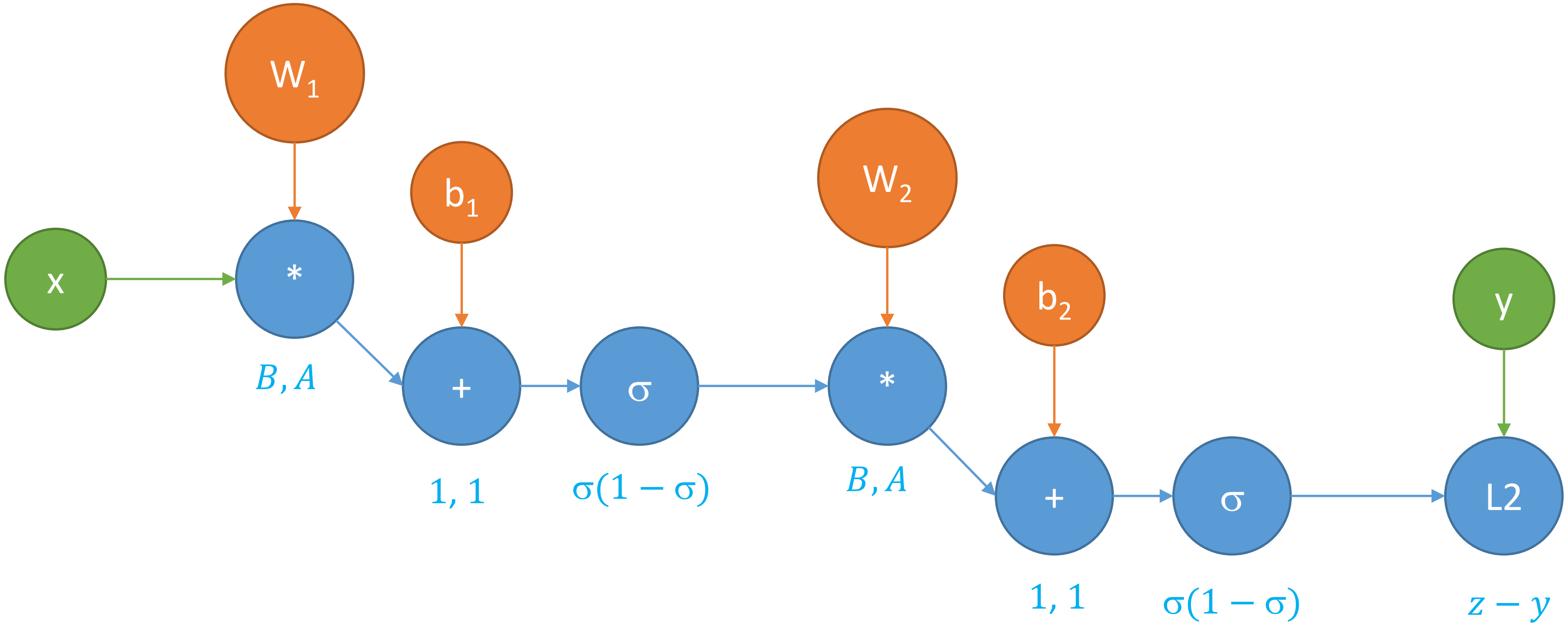
Ejemplo completo

- Cálculo de los gradientes locales (se hace solo una vez al comienzo):



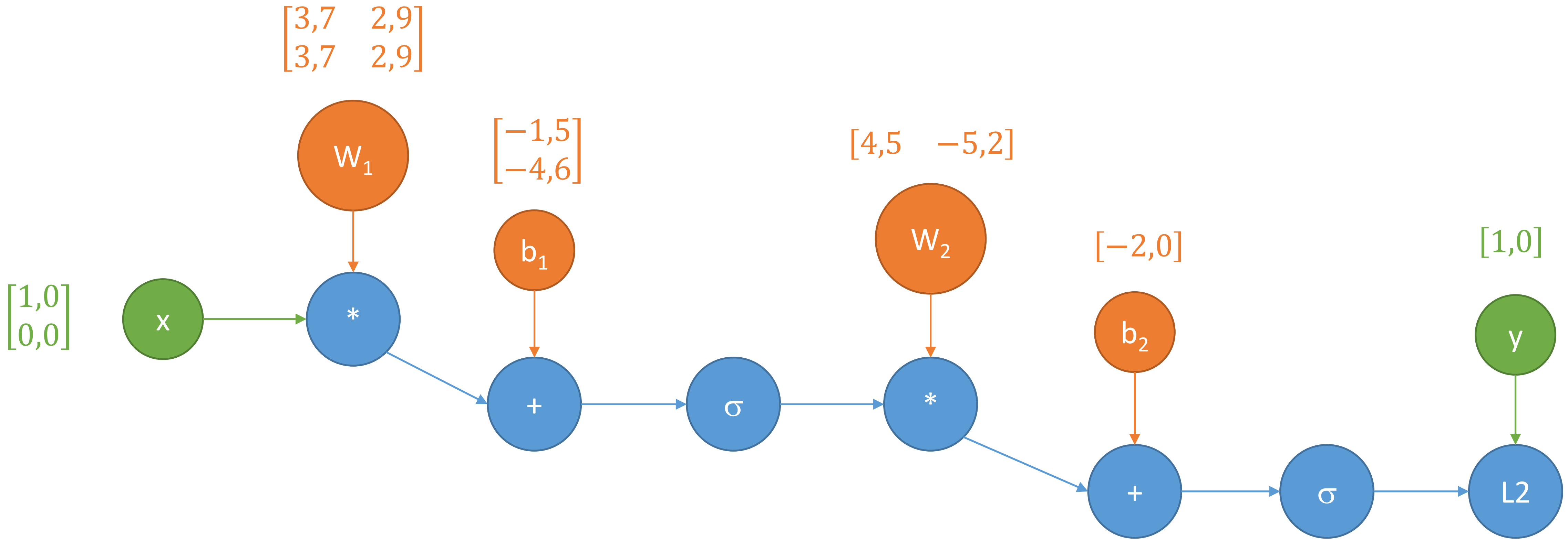
Ejemplo completo

- Cálculo de los gradientes locales (se hace solo una vez al comienzo):



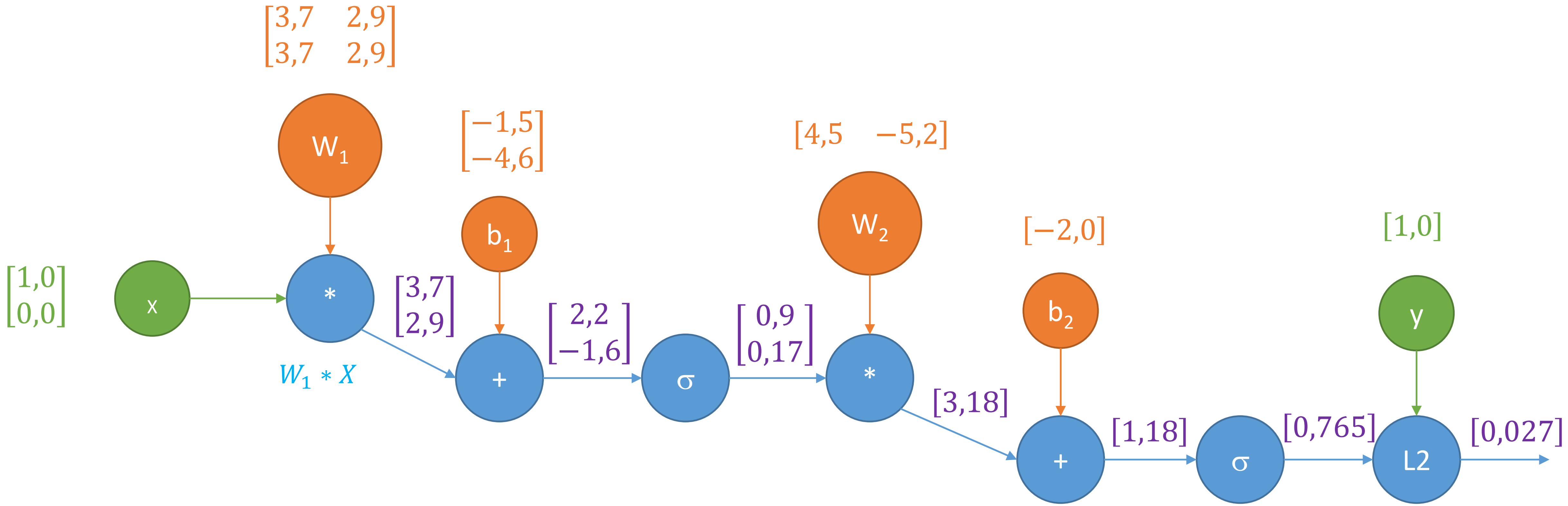
Ejemplo completo

- Propagación hacia adelante con x = [1,0] e y=1:



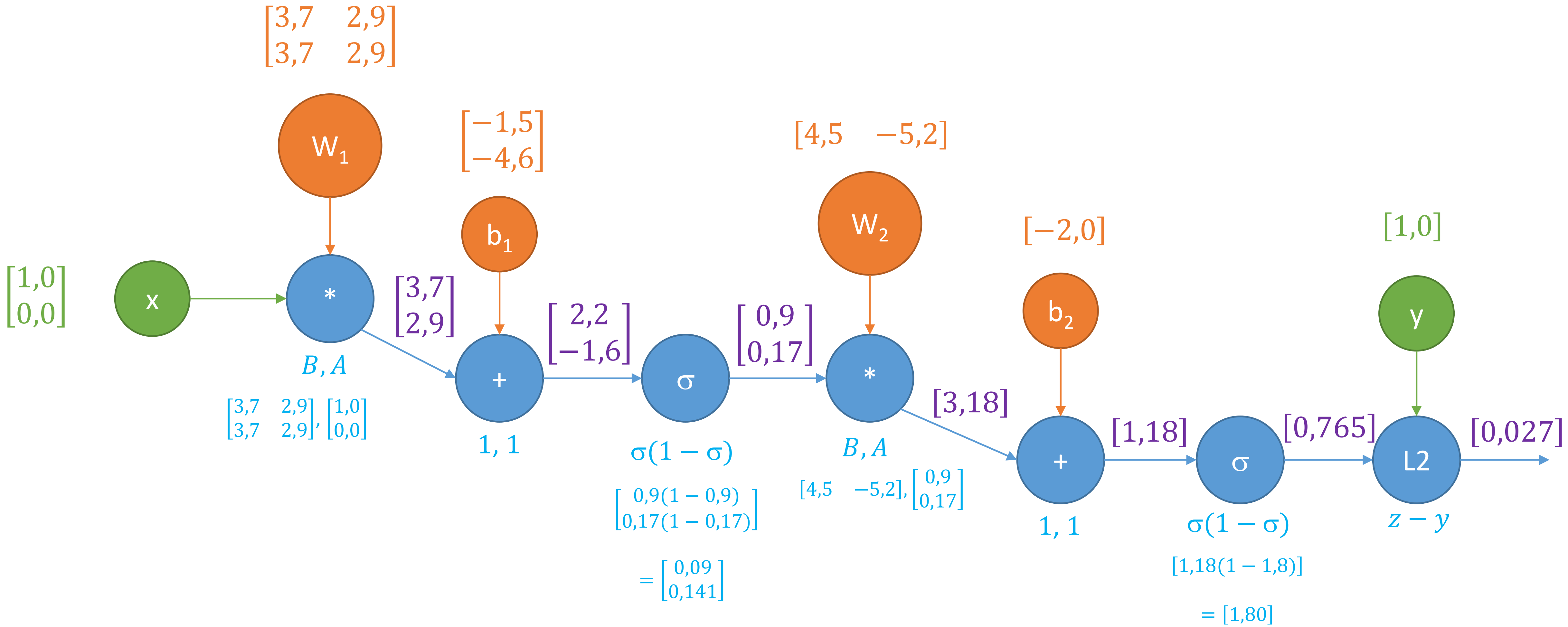
Ejemplo completo

- Propagación hacia adelante con x = [1,0] e y=1:



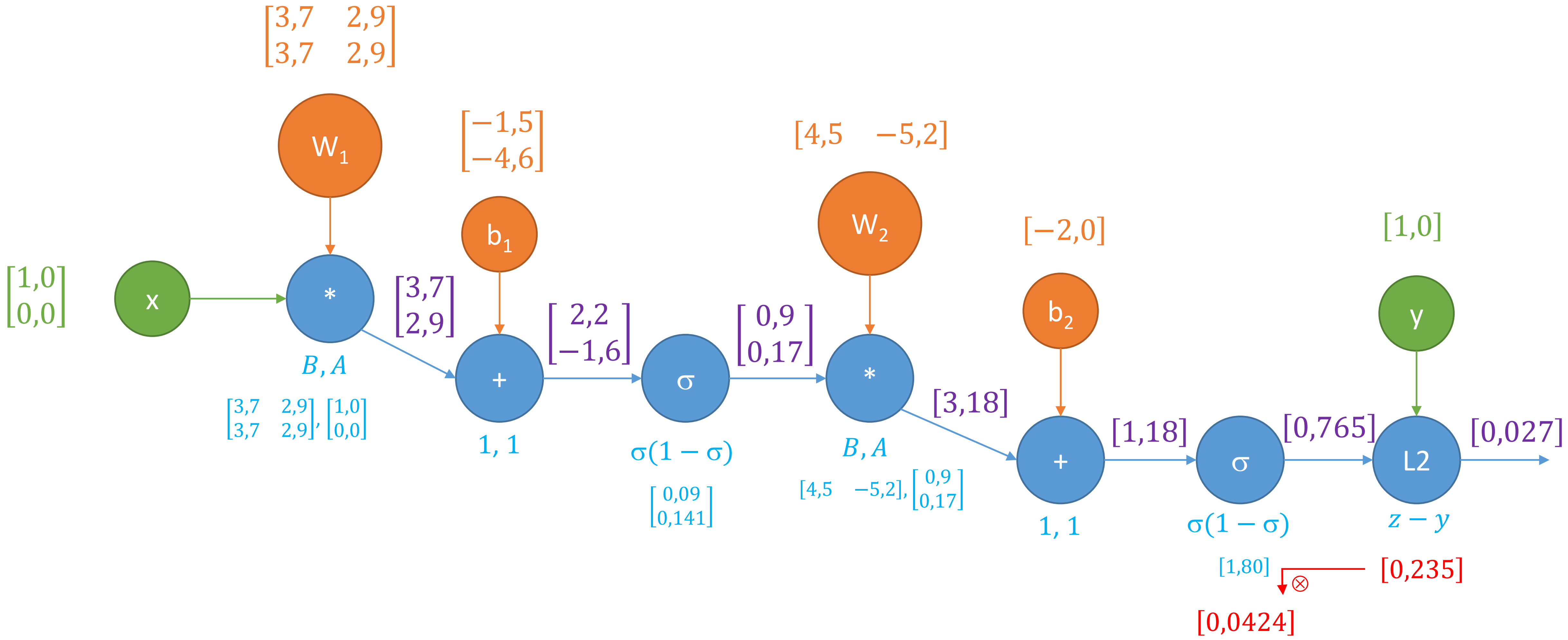
Ejemplo completo

- Mientras se hacía la propagación hacia adelante, se calculaban algunos gradientes locales:



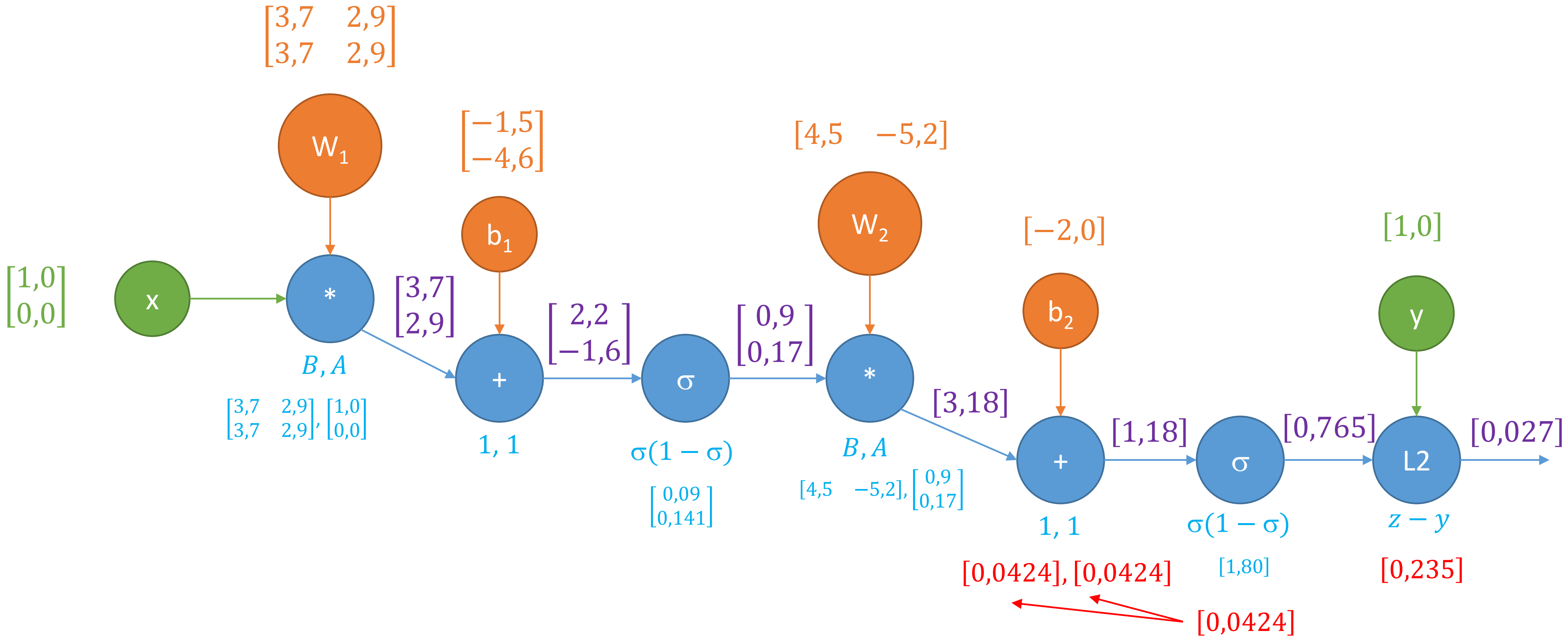
Ejemplo completo

- Propagación de gradientes:



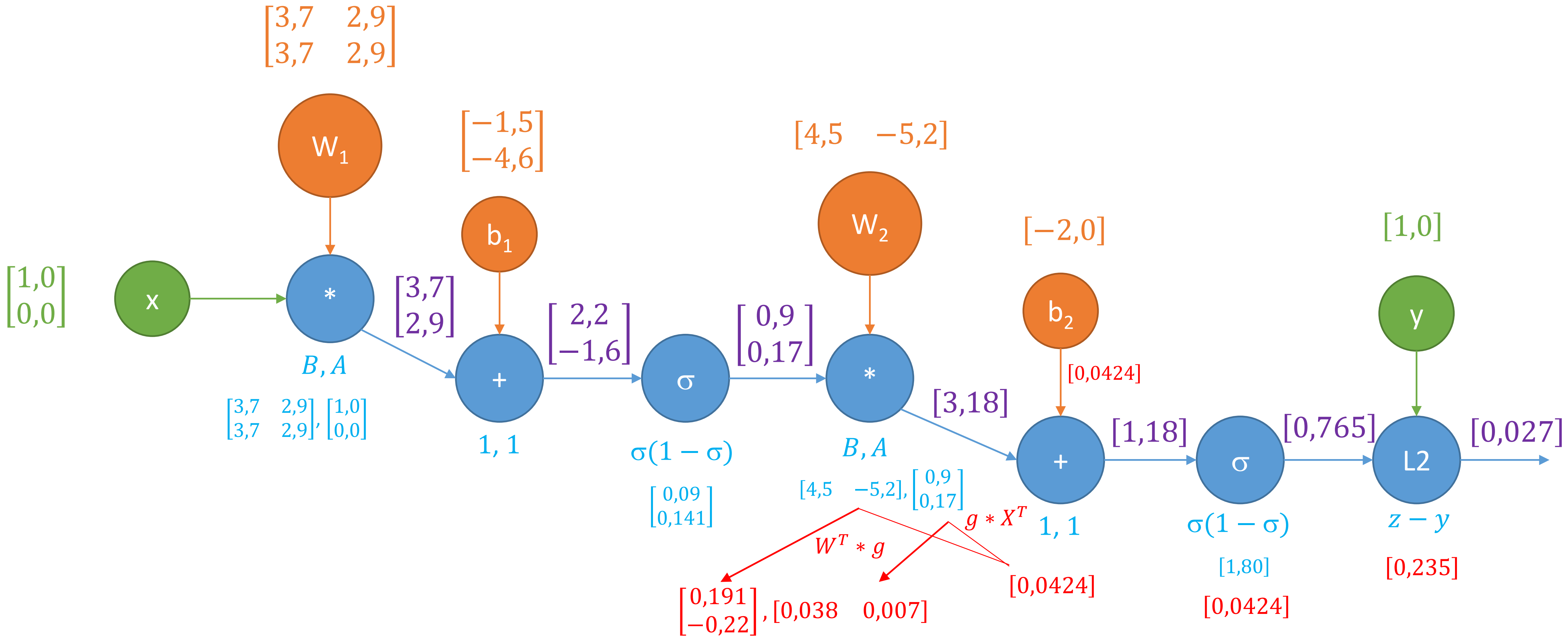
Ejemplo completo

- Propagación de gradientes:



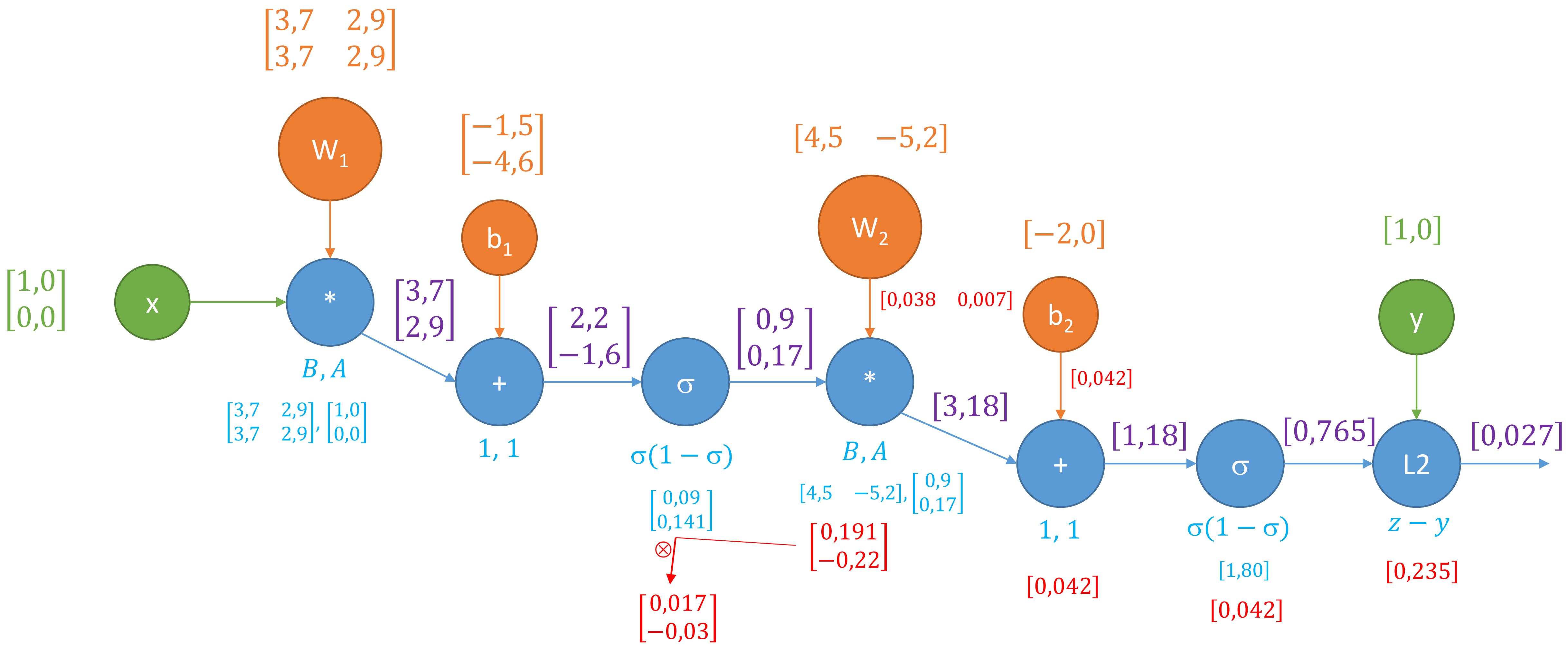
Ejemplo completo

- Propagación de gradientes:



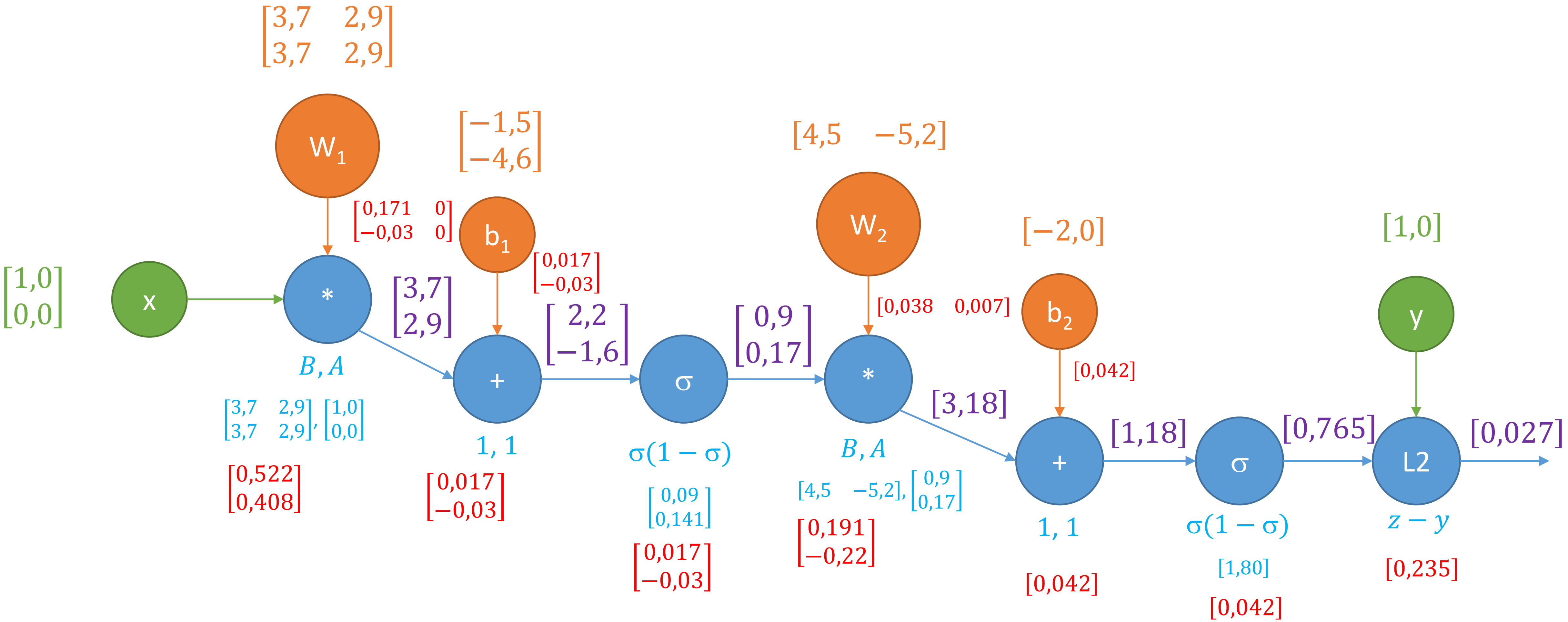
Ejemplo completo

- Propagación de gradientes:



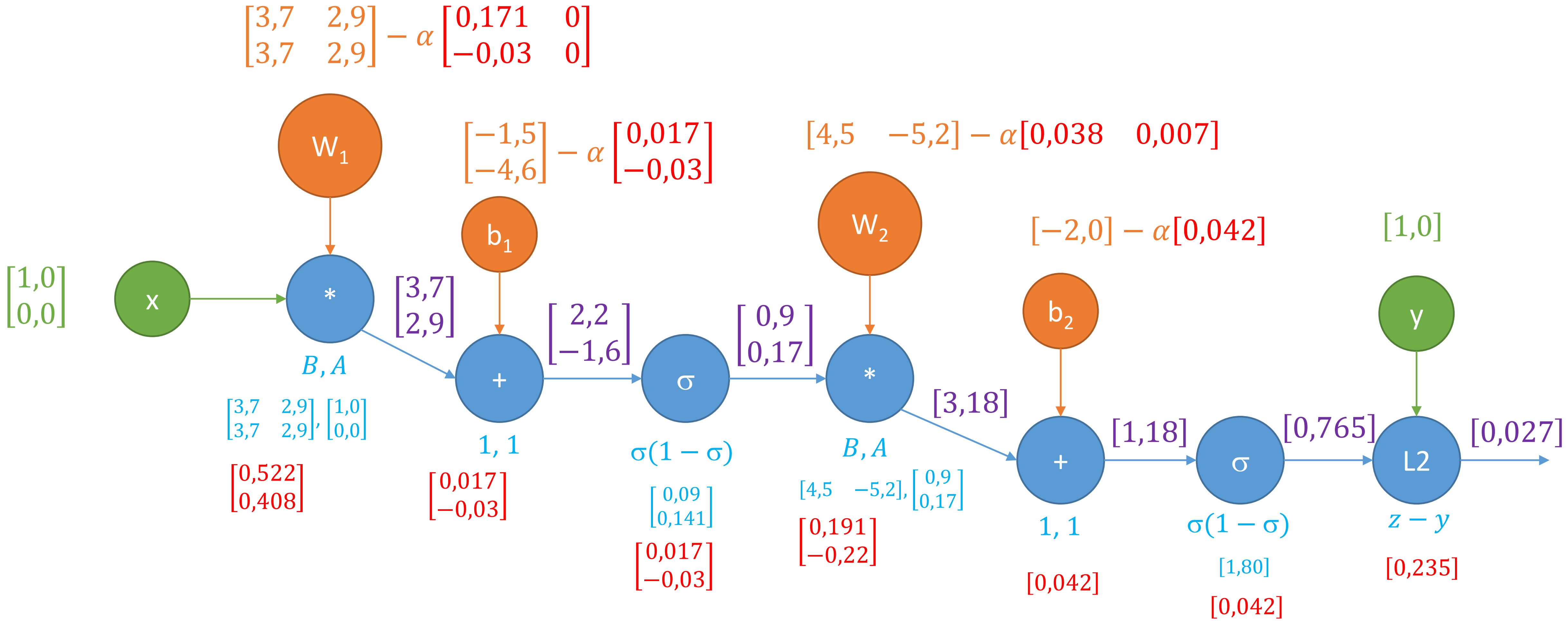
Ejemplo completo

- Propagación de gradientes:



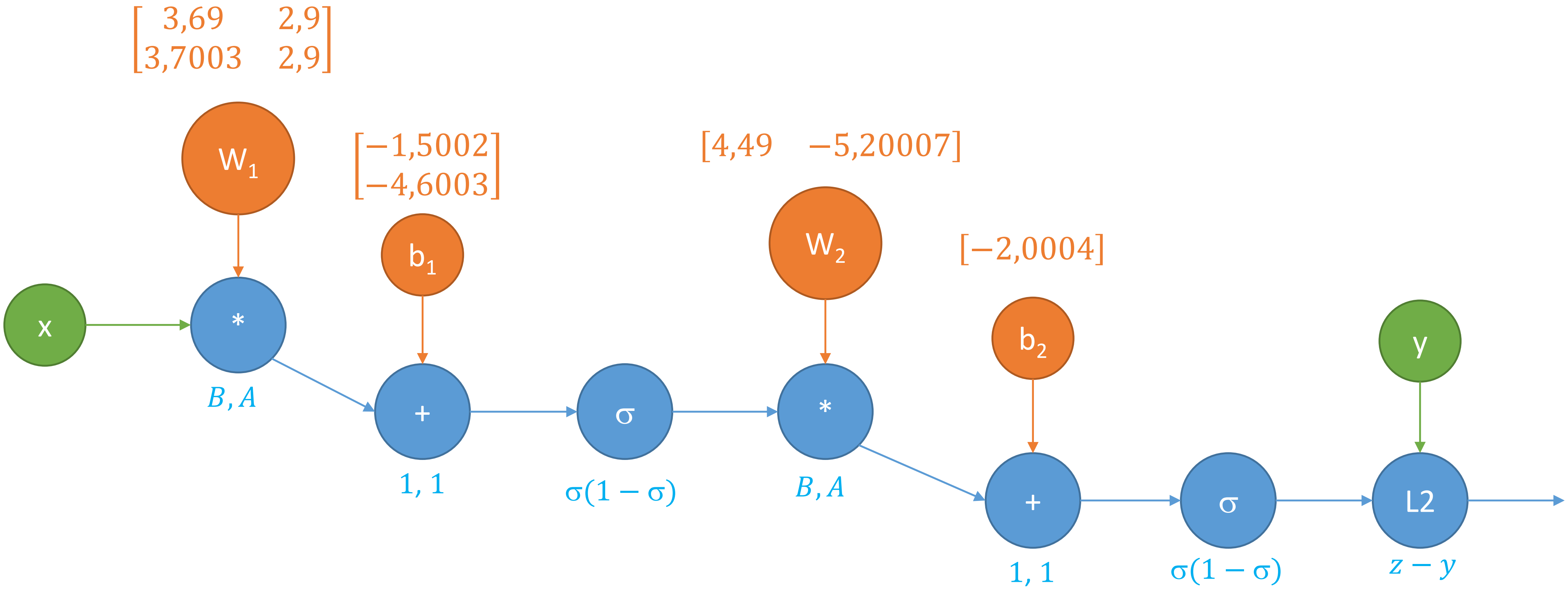
Ejemplo completo

- Ya solo queda actualizar los pesos usando un learning rate α=0,01



Ejemplo completo

- Ya tenemos la red preparada para el siguiente ejemplo



Recapitulación

- Cada capa de una red se puede representar mediante una matriz. Esto da lugar a una implementación eficiente basada en **matrices.**
- La propagación de gradientes de forma vectorial se basa en matrices **jacobianas**. El cálculo de las mismas se **simplifica** mucho y se reduce a las mismas reglas de propagación de gradientes.