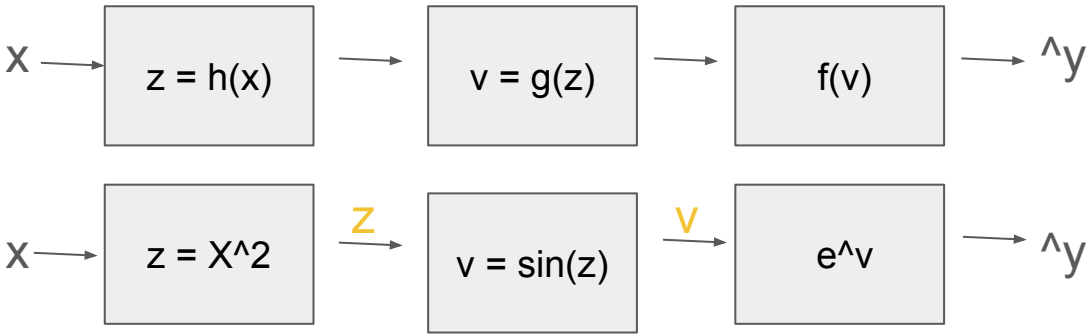
Entrenamiento de Redes Neuronales Artificiales Profundas

U-TAD

Grafo computacional

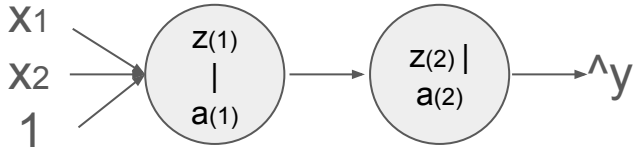
- Son grafos dirigidos (DAG) cuyos nodos se corresponden con una operación o una variable y los arcos son valores de entrada/salida

f(g(h(x))) → $e^{\sin(x^2)}$ función compuesta: f(x) = e^x, g(x) = sin(x), h(x) = x^2

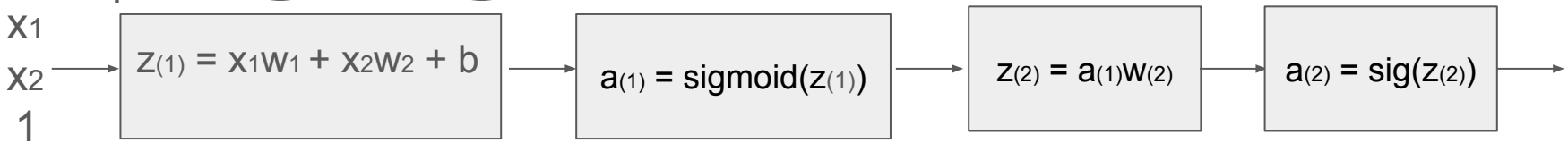


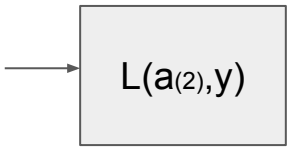
Grafo computacional

- Aplicación del concepto de grafo computacional a una RNA sencilla



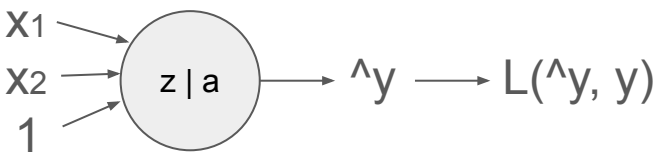
$\hat{y} = \text{sigmoid}(x_1 w_1 + x_2 w_2 + b)$ a(z)





dL / dw (a continuación veremos cómo hacer esta derivada para la función de error L)

Derivadas (parciales ∂) con el Grafo Computacional



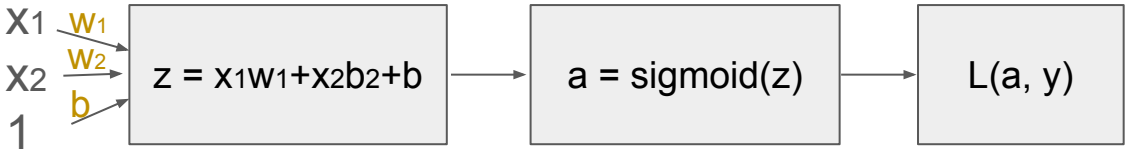
$\hat{y} = \text{sigmoid}(x_1w_1 + w_2w_2 + b)$

[Gradient Descent]

$w_1 = w_1 - n\, dL/dw_1$

$w_2 = w_2 - n\, dL/dw_2$

$b = b - n\, dL/db$



dL/dw_1, dL/dw_2, dL/w_3

Calculamos la derivada de la función de error *L* respecto a los parámetros *w,* además de lo que ya vimos respecto de la pendiente, es que cuando calculamos la derivada de esta función L es calcular cómo varía el resultado de esta función de error cuando variamos el valor de estos parámetros. Por ejemplo, para dL/dw_1, lo que hacemos de manera intuitiva es ver cómo varía la función de error cuando variamos el parámetro w_1. De manera que lo podamos modificar cuando esa modificación varía la función de error minimizándola. Es decir, es un manera de saber cómo varía la función de error.

Regla de la cadena

Si queremos calcular, concretamente dL/dw_1, lo que vamos a hacer es ir recorriendo todos los nodos en nuestro grafo computacional de manera inversa.

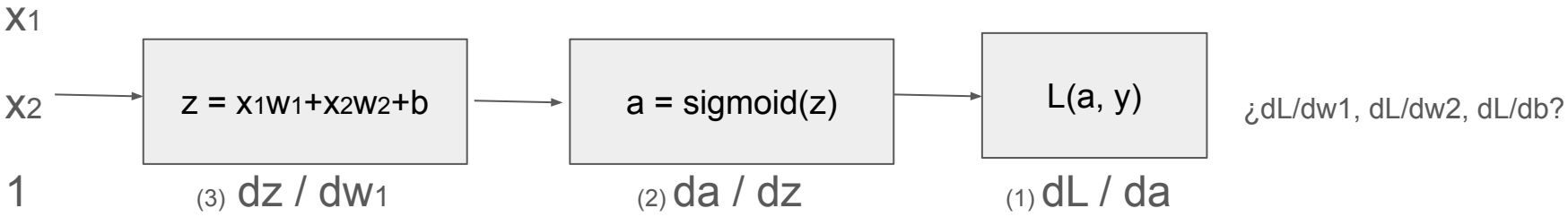
Es decir, primero tendremos que calcular la derivada de la función de error respecto al nodo anterior (dL/da)

Lo siguiente es calcular la derivada del resultado de la función de agregación que tenemos en el nodo anterior (da/dz)

Y, también, voy a tener que saber cómo varía esta función de agregación cuando varío w1 (dz/dw_1)

Por lo que iré computando la derivada de un nodo respecto al nodo anterior de manera secuencial, lo que se denomina regla de la cadena

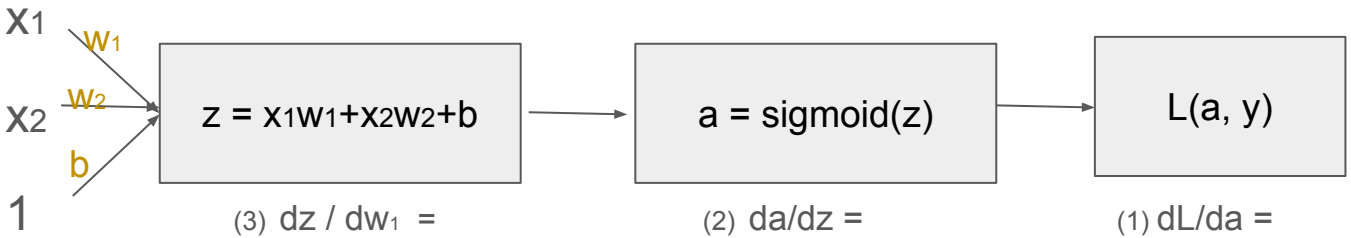
Regla de la cadena



Regla de la cadena -> calcula la derivada de la composición de dos o más funciones

$dL/dw_1 = dL/da * da/dz * dz/dw_1$

Regla de la cadena



$= d[w_1x_1+w_2x_2+b]/dw_1 =$

$= x_1$

$= d[1/(1+e^{-z})]/dz =$

$= a(1-a)$

$= d[-y\log(a)-(1-y)\log(1-a)]/da =$

$= (1-y)/(1-a) - y/a$

$dL/dw_1 = dL/da * da/dz * dz/dw_1 = ((1-y)/(1-a) - y/a) * (a(1-a))x_1$

$= ((1-y)a - y(1-a) / (1-a)a) * a(1-a)x_1 =$

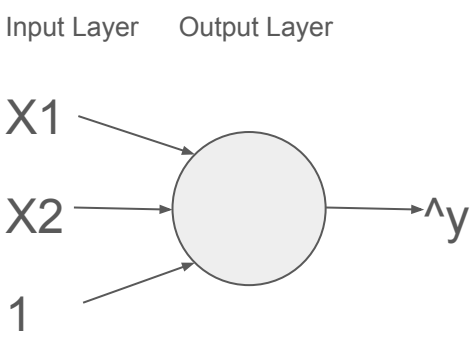
$= ((1-y)a - y(1-a))x1 = (a - ya - y + ya)x_1 =$

$= \mathbf{(a-y)x1}$

$w_1 = w_1 - n(dL/dw_1) = w_1 - n(a-y)x_1$

$a = \hat{y}$

Backward Propagation



(Gradient Descent) $w_1 = w_1 - n(dL/dw_1) = w_1 - nX_1(a - y)$

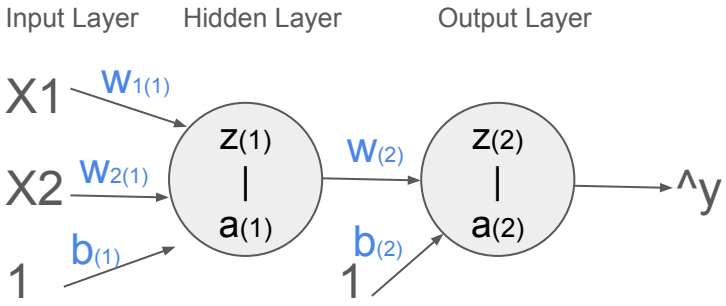
$w_2 = w_2 - n(dL/dw_2) = w_2 - nX_2(a - y)$

$b = b - n(dL/db) = b - n(a - y)$

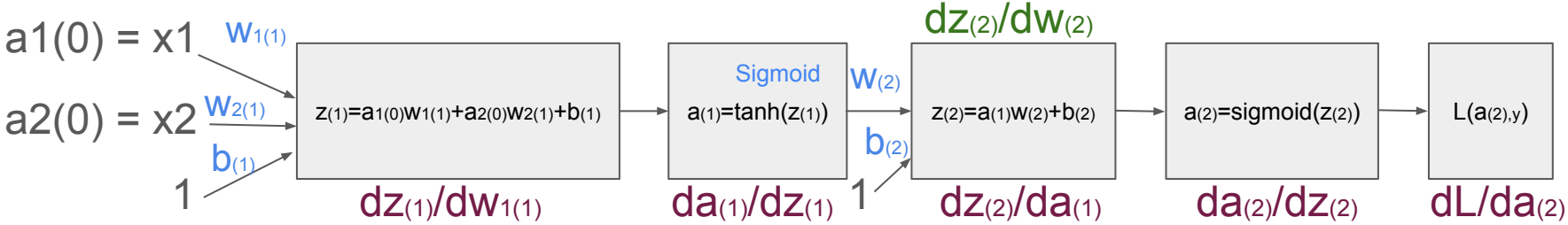
Vamos a ir al final del grafo computacional, al nodo donde se calcula el resultado de esa función de error respecto a la predicción de nuestra red neuronal y a la etiqueta de nuestro conjunto de datos para ese ejemplo. Y propagamos hacia atrás todas las derivadas de manera que lleguemos a una conclusión sobre cómo modificar ese parámetro del modelo (w1), en función de cómo varía la función de error cuando variamos un poco el valor de ese parámetro.

Todo este proceso de cálculo de derivadas hacia atrás es “Backward Propagation”.

Backward Propagation



$dL/dw_{1(1)}, dL/dw_{(2)}$



$$dL/dw_{1(1)} = dL/da_{(2)} * da_{(2)}/dz_{(2)} * dz_{(2)}/da_{(1)} * da_{(1)}/dz_{(1)} * dz_{(1)}/dw_{1(1)}$$

$$dL/dw_{(2)} = dL/da_{(2)} * da_{(2)}/dz_{(2)} * dz_{(2)}/dw_{(2)}$$

Backward Propagation

Backward Propagation

$dw^{(2)} = a^{(1)} * (a^{(2)} - y)$

$db^{(2)} = a^{(2)} - y$

$dw1^{(1)} = (a^{(2)} - y) * w^{(2)} * (1 - a^{(1)}\text{^}2) * x1$

$dw2^{(1)} = (a^{(2)} - y) * w^{(2)} * (1 - a^{(1)}\text{^}2) * x2$

$db^{(1)} = (a^{(2)} - y) * w^{(2)} * (1 - a^{(1)}\text{^}2)$

Parameter update (Gradient Descent)

$w^{(2)} = w^{(2)} - n * dw^{(2)}$

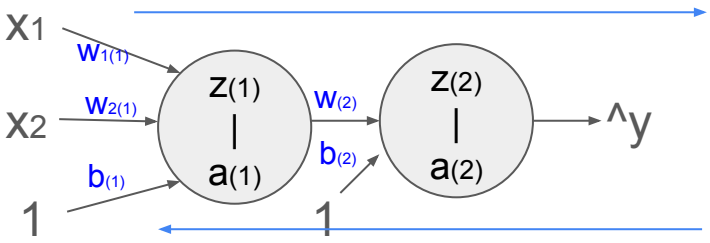
$b^{(2)} = b^{(2)} - n * db^{(2)}$

$w1^{(1)} = w1^{(1)} - n * dw1^{(1)}$

$w2^{(1)} = w2^{(1)} - n * dw2^{(1)}$

$b^{(1)} = b^{(1)} - n * db^{(1)}$

Entrenando una RNA simple



Realiza predicción

Comprueba con L(^y, y)

Modifica w_1, w_2, … (hasta que no varíen)

Forward Propagation

$z_{(1)} = w_{1(1)}x_1 + w_{2(1)}x_2 + b_{(1)}$

$a_{(1)} = \tanh(z_{(1)})$

$z_{(2)} = w_{(2)}a_{(1)} + b_{(2)}$

$a_{(2)} = \text{sigmoid}(z_{(2)})$

^y

Backward Propagation

$dw_{(2)} = a_{(1)}(a_{(2)} - y)$

$db_{(2)} = a_{(2)} - y$

$dw_{(1)} = (a_{(2)} - y)w_{(2)}(1 - a_{(1)}\text{^}2)x_1$

$dw_{(2)} = (a_{(2)} - y)w_{(2)}(1 - a_{(1)}\text{^}2)x_2$

$db_{(1)} = (a_{(2)} - y)w_{(2)}(1 - a_{(1)}\text{^}2)$

Parameter update

$w_{(2)} = w_{(2)} - n*dw_{(2)}$

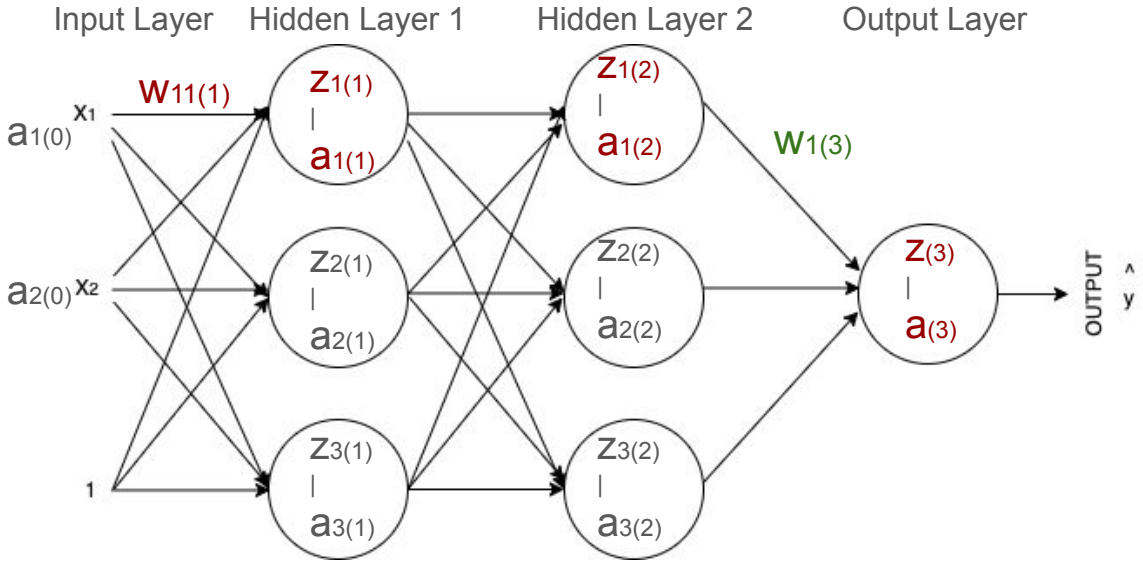
$b_{(2)} = b_{(2)} - n*db_{(2)}$

$w_{1(1)} = w_{1(1)} - n*dw_{1(1)}$

$w_{2(1)} = w_{2(1)} - n*dw_{2(1)}$

$b_{(1)} = b_{(1)} - n*db_{(1)}$

Entrenando una RNA profunda



$dL/dw_{1(3)} = dL/da_{(3)} * da_{(3)}/dz_{(3)} * dz_{(3)}/dw_{1(3)}$

$dL/dw_{11(1)} = dL/da_{(3)} * da_{(3)}/dz_{(3)} * dz_{(3)}/da_{1(2)} * da_{(2)}/dz_{1(2)} * dz_{1(2)}/da_{1(1)} * da_{1(1)}/dz_{1(1)} * dz_{1(1)}/w_{11(1)}$

Caso Práctico: Clasificación de audio

Clasificación de audio con el Perceptrón Multicapa I.ypnb *(malos resultados)*

Clasificación de audio con el Perceptrón Multicapa II.ypnb