Conceptos Generales

Redes Neuronales y Aprendizaje Automático

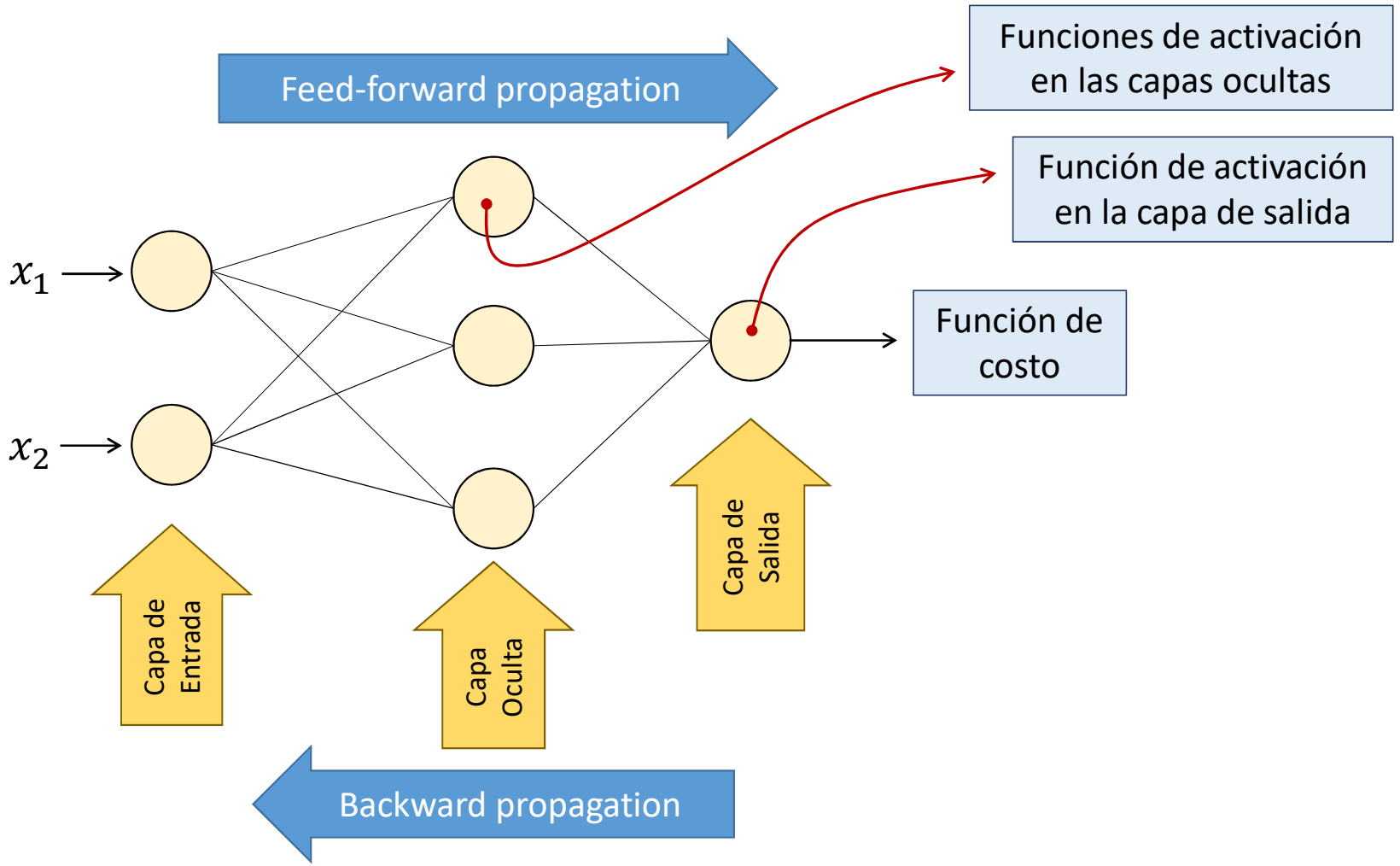
Parte 2

Dr. Luis Eduardo Falcón Morales

ITESM

Campus Guadalajara

¿Cómo aprende/encuentra sus pesos una red neuronal? Propagación hacia adelante y hacia atrás



Existen varias reglas empíricas para el número de neuronas en la capa oculta de un modelo de red neuronal Perceptrón Multicapa (MLP).

Podemos mencionar algunas de las más comunes:

- **Regla empírica-1**: la cantidad de neuronas en la capa oculta debe ser de al menos:

$$nh = \frac{2}{3}nx + ny,$$

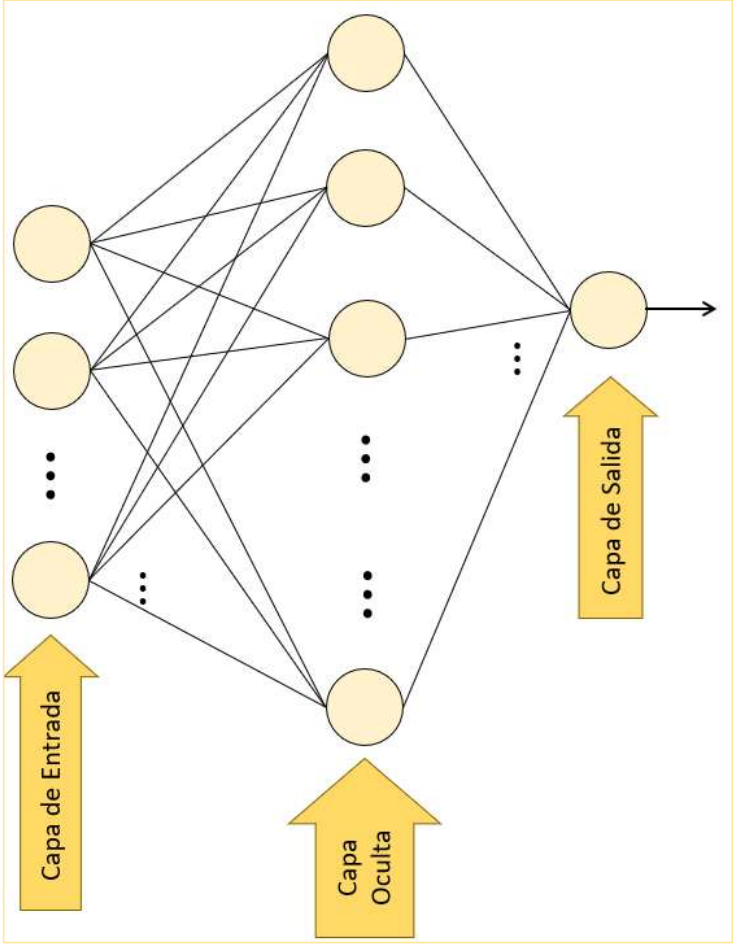
donde

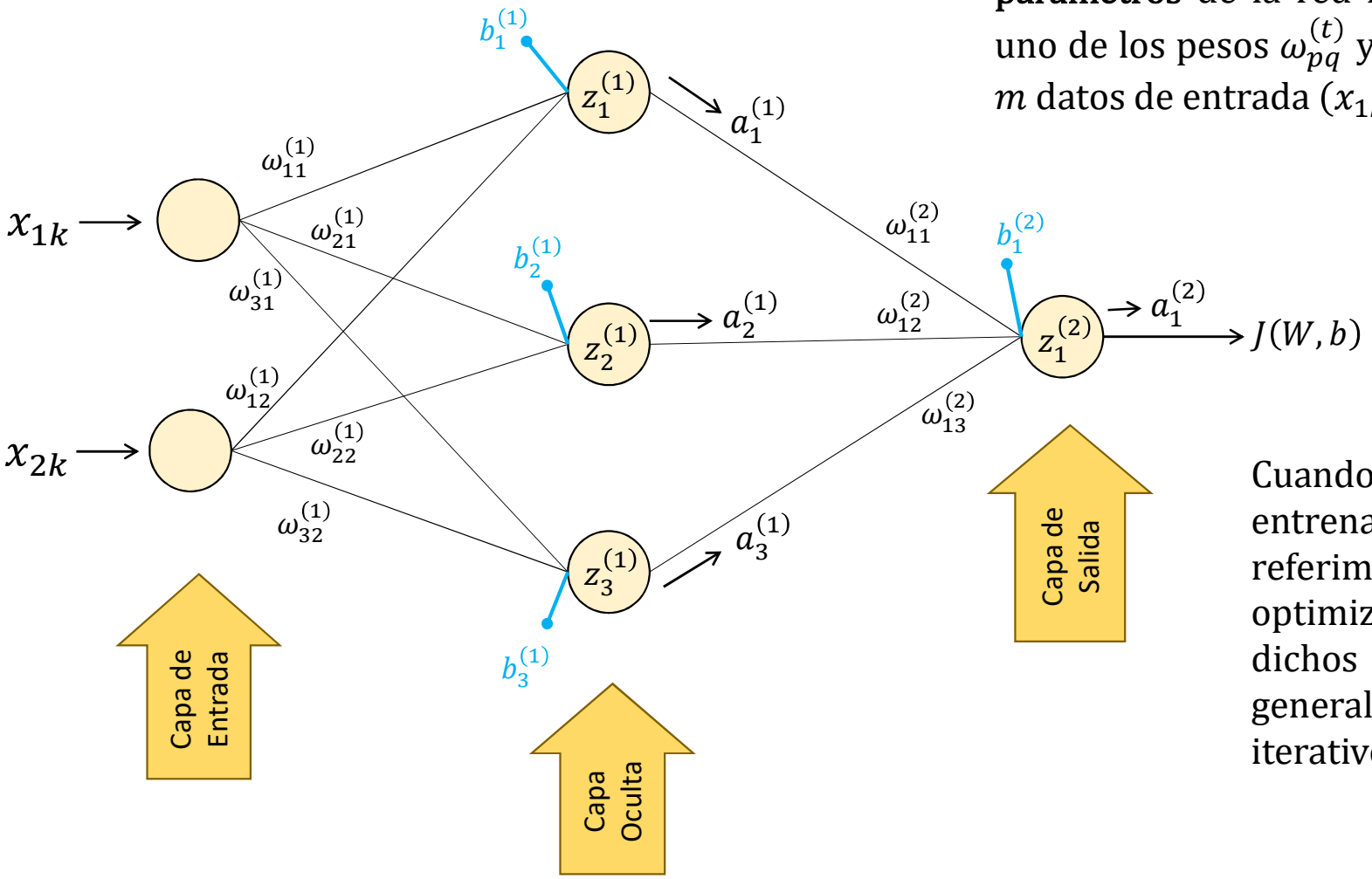
nx : número de neuronas en la capa de entrada
nh : número de neuronas en la capa oculta
ny : número de neuronas en la capa de salida

- **Regla empírica-2**: el máximo de neuronas en la capa oculta debiera ser el doble de neuronas del total de las de entrada y salida.

- **Regla empírica-3**: El mínimo de neuronas debe ser la mitad de total de las de entrada y salida.

¿Cuántas neuronas se requieren en la capa oculta?

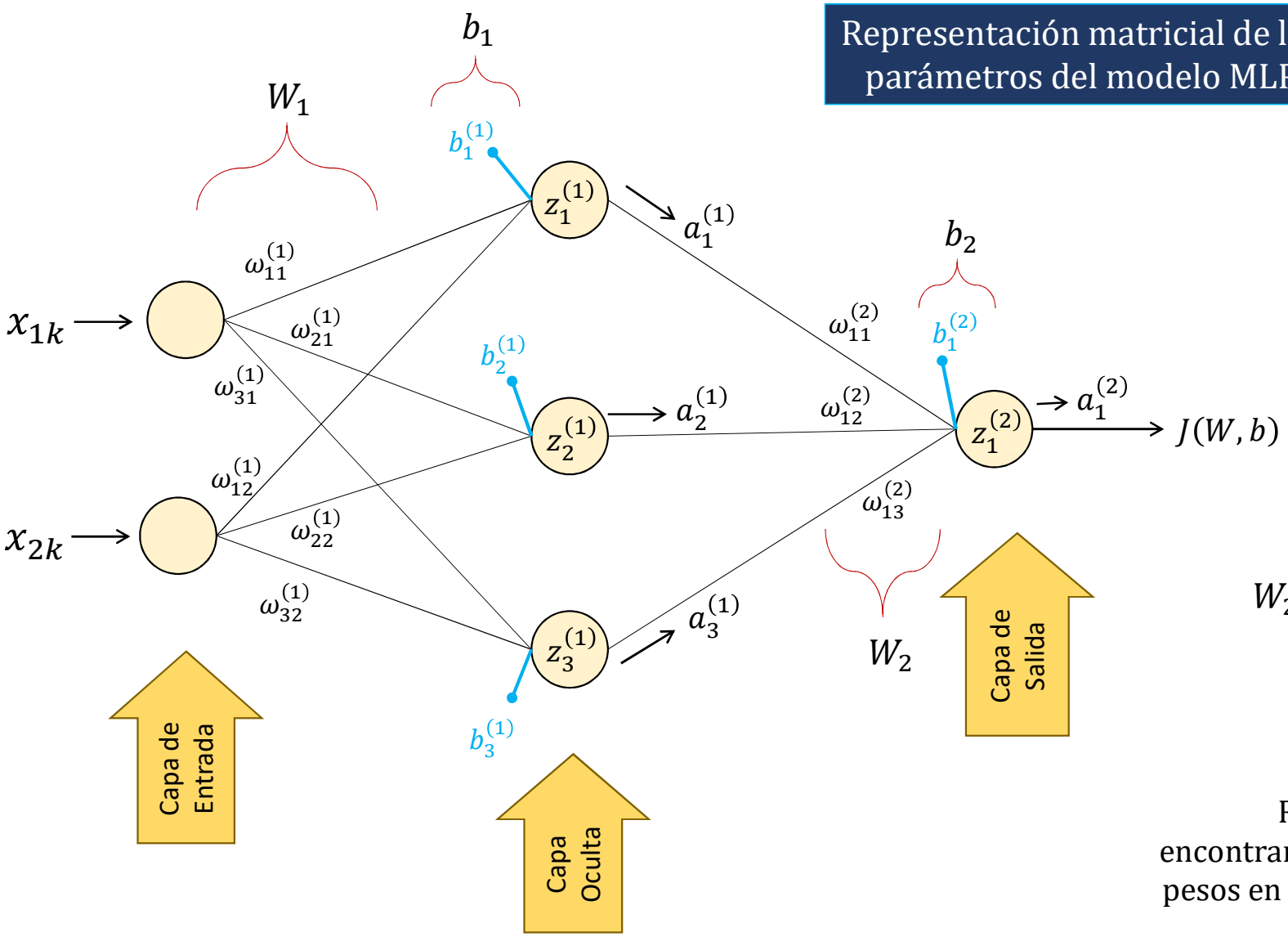




El objetivo es encontrar los llamados **parámetros** de la red neuronal, es decir, cada uno de los pesos $\omega_{pq}^{(t)}$ y los bias $b_p^{(t)}$ a partir de m datos de entrada $(x_{1k}, x_{2k}), k = 1,2, \dots, m$.

Cuando decimos que se está entrenando el modelo, nos referimos al proceso de optimización para encontrar dichos parámetros, que en general será un proceso iterativo numérico.

Representación matricial de los parámetros del modelo MLP



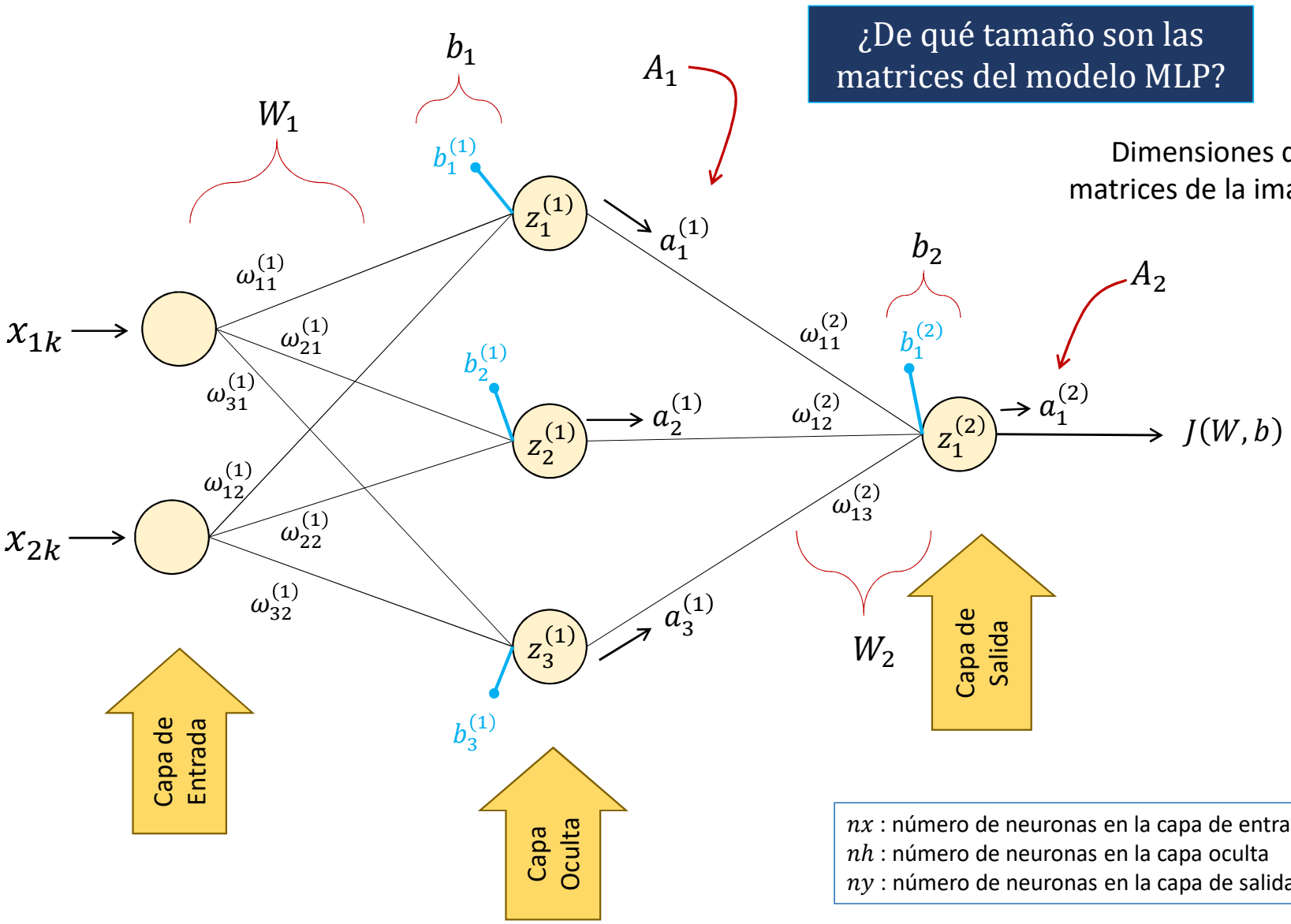
$$W_1 = \begin{bmatrix} \omega_{11}^{(1)} & \omega_{12}^{(1)} \\ \omega_{21}^{(1)} & \omega_{22}^{(1)} \\ \omega_{31}^{(1)} & \omega_{32}^{(1)} \end{bmatrix}$$

$$b_1 = \begin{bmatrix} b_1^{(1)} \\ b_2^{(1)} \\ b_3^{(1)} \end{bmatrix}$$

$$W_2 = \begin{bmatrix} \omega_{11}^{(2)} & \omega_{12}^{(2)} & \omega_{13}^{(2)} \end{bmatrix}$$

$$b_2 = \begin{bmatrix} b_1^{(2)} \end{bmatrix}$$

Recuerda que el objetivo es encontrar los valores de todos estos pesos en las matrices W_1, W_2, b_1, b_2.



¿De qué tamaño son las matrices del modelo MLP?

Dimensiones de las matrices de la imagen:

$W_1 : 3 \times 2$
$b_1 : 3 \times 1$

$W_2 : 1 \times 3$
$b_2 : 1 \times 1$

$A_1 : 3 \times m$
$A_2 : 1 \times m$

m : total datos de entrada

En general

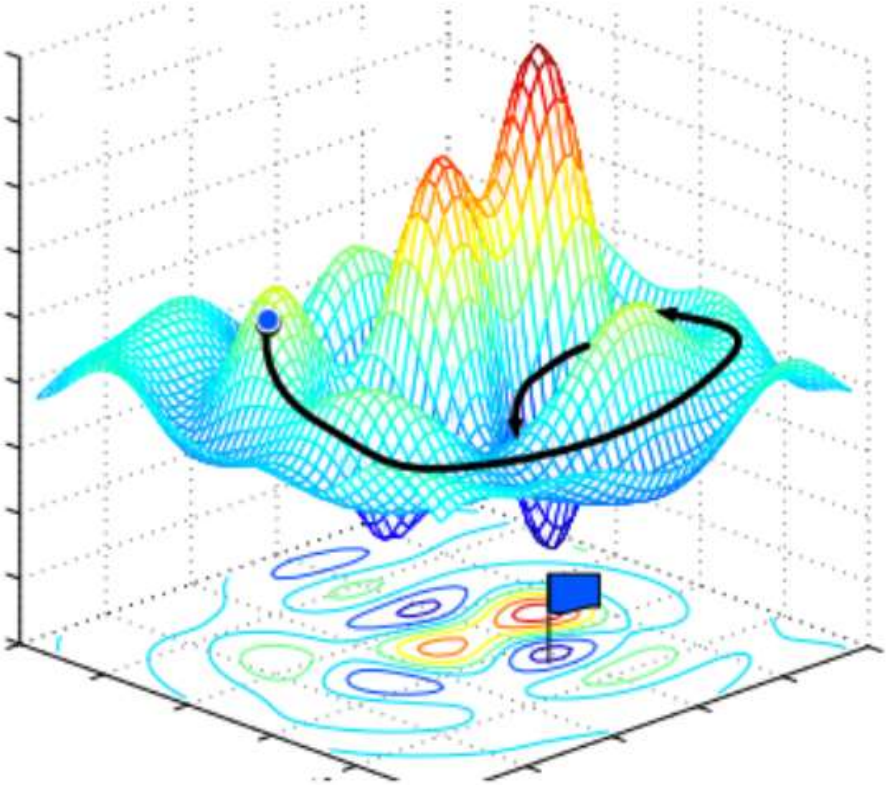
$W_1 : nh \times nx$
$b_1 : nh \times 1$

$W_2 : 1 \times nh$
$b_2 : 1 \times 1$

$A_1 : nh \times m$
$A_2 : 1 \times m$

nx : número de neuronas en la capa de entrada
nh : número de neuronas en la capa oculta
ny : número de neuronas en la capa de salida

Inicialización de los Pesos de la Red Neuronal



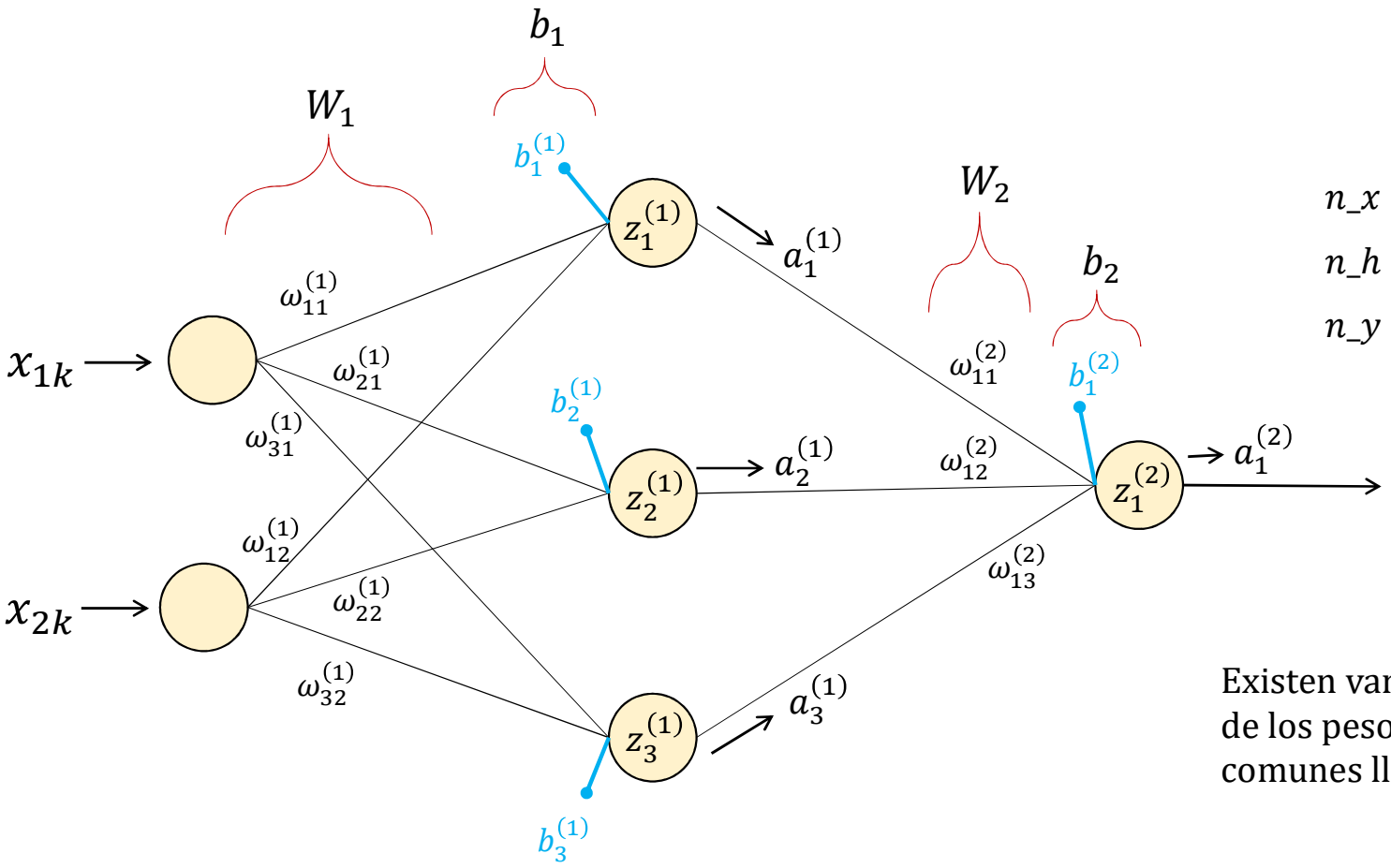
Espacio de búsqueda del mínimo en la función de costo.

En el proceso de minimizar una función de costo se suelen utilizar métodos no deterministas, estocásticos/aleatorios. En particular el gradiente descendente estocástico es de los más estándar.

Dichos métodos utilizan la aleatoriedad tanto en la inicialización, como durante el proceso de búsqueda del óptimo y evitar así quedar atrapado en un mínimo local.

Por esta razón, te darás cuenta al entrenar un modelo, que se obtienen soluciones diferentes cada vez que se lleva a cabo un entrenamiento diferente.

El método del gradiente descendente estocástico (SGD) y el uso de funciones de activación como la sigmoide y la tangente hiperbólica, nos llevan a inicializar aleatoriamente los pesos W_1 y W_2 con valores cercanos a cero y los vectores de sesgos b_1 y b_2 en cero.



$W_1 : n\_h \times n\_x$
$b_1 : n\_h \times 1$

$W_2 : 1 \times n\_h$
$b_2 : 1 \times 1$

n_x : neuronas en la capa de entrada.

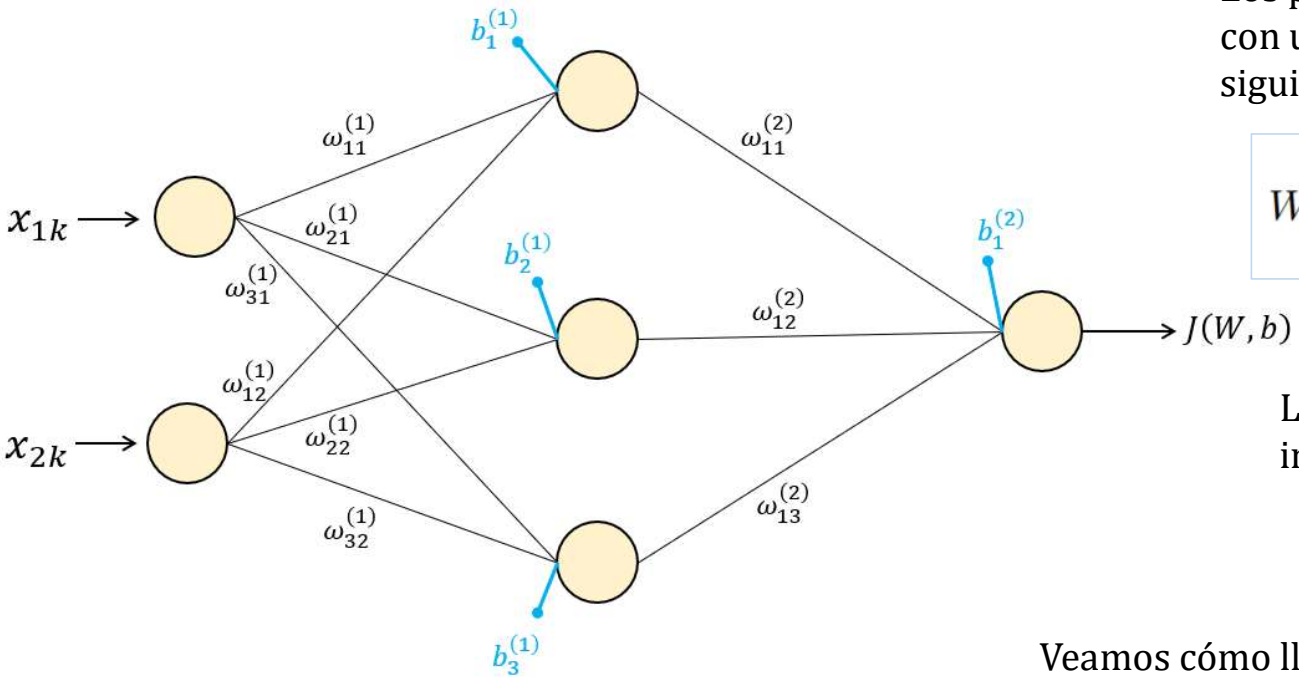
n_h : neuronas en la capa oculta.

n_y : neuronas en la capa de salida.

Existen varios métodos de inicialización de los pesos, veamos uno de los más comunes llamado de Glorot o Xavier.

Inicialización de Xavier o Inicialización de Glorot

¿Cómo encontrar los mejores valores de inicialización de los pesos $W = \omega_{ij}^{(m)}$ y *bias* $b = b_i^{(m)}$ al aplicar el método de optimización?



Los pesos W se inicializan aleatoriamente con una distribución uniforme U entre los siguientes valores:

$$W \sim U\left[-\frac{\sqrt{6}}{\sqrt{n_j + n_{j+1}}}, \frac{\sqrt{6}}{\sqrt{n_j + n_{j+1}}}\right]$$

Los valores de los bias se pueden inicializar siempre en cero:

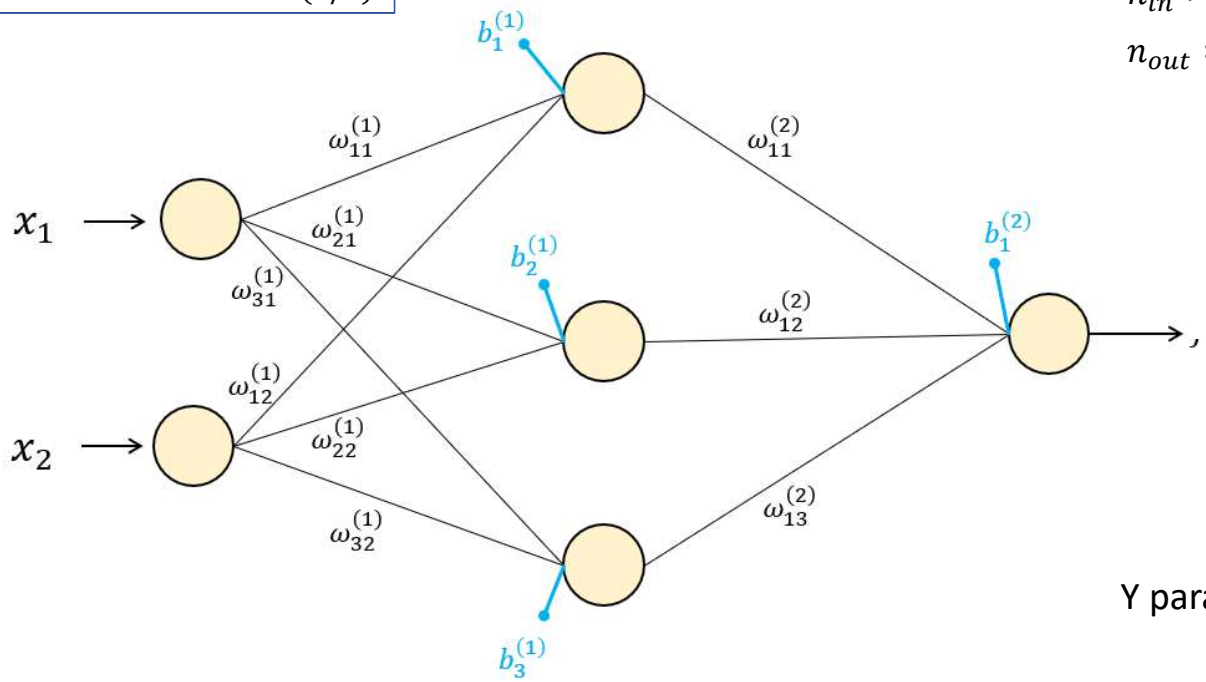
$$b = 0$$

Veamos cómo llegan los autores a estas expresiones…

Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks, 2010.

https://www.semanticscholar.org/paper/Understanding-the-difficulty-of-training-deep-Glorot-Bengio/b71ac1e9fb49420d13e084ac67254a0bbd40f83f

Inicialización de Glorot (1/2)



Por simplicidad veamos el análisis para los pesos de la capa de entrada a la capa oculta del diagrama. Supongamos que se tienen las siguientes distribuciones:

$$X \sim D(\mu_X = 0, \sigma_X^2)$$

$$W \sim D(\mu_W = 0, \sigma_W^2)$$

n_{in} : total neuronas en la capa de entrada

n_{out} : total neuronas en la capa de salida (oculta en este caso)

$$z_i = \sum_{j=1}^{n_{in}} \omega_{ij} x_j$$

Entonces, de Estadística el valor esperado de z_i:

$$E[z_i] = E\left[\sum_{j=1}^{n_{in}} \omega_{ij} x_j\right] = \sum_{j=1}^{n_{in}} E[\omega_{ij}] E[x_j] = 0$$

Y para la varianza:

$$Var[z_i] = E[z_i^2] - (E[z_i])^2 = \sum_{j=1}^{n_{in}} E[\omega_{ij}^2 x_j^2] - 0$$

$$= \sum_{j=1}^{n_{in}} E[\omega_{ij}^2] E[x_j^2] = n_{in} \sigma_W^2 \sigma_X^2$$

La cual finalmente podemos poner la restricción a los pesos W a partir de la capa oculta como:

$$n_{in} \sigma_W^2 = 1$$

Inicialización de Glorot (1/2)

Considerando ahora los pesos de la capa oculta a la capa de salida W, se puede obtener de manera análoga:

$$n_{out}\sigma_W^2 = 1$$

Observa que la varianza de los datos de entrada X no los podemos restringir, por ello aplicamos las restricciones solo a todos los pesos de la red, los cuales sí podemos controlar.

La manera de hacer que estas dos restricciones se cumplan será mediante su promedio, es decir:

$$\frac{1}{2}(n_{in} + n_{out})\sigma_W^2 = 1$$

Es decir, los pesos W los podemos inicializar a partir de una función de distribución W con media 0 y varianza:

$$\sigma_W^2 = \frac{2}{n_{in} + n_{out}}$$

o desviación estándar: $\sigma_W = \sqrt{\frac{2}{n_{in} + n_{out}}}$

Usualmente se considera la distribución normal (gaussiana) o bien la distribución uniforme, en cuyo caso coniserando esta última

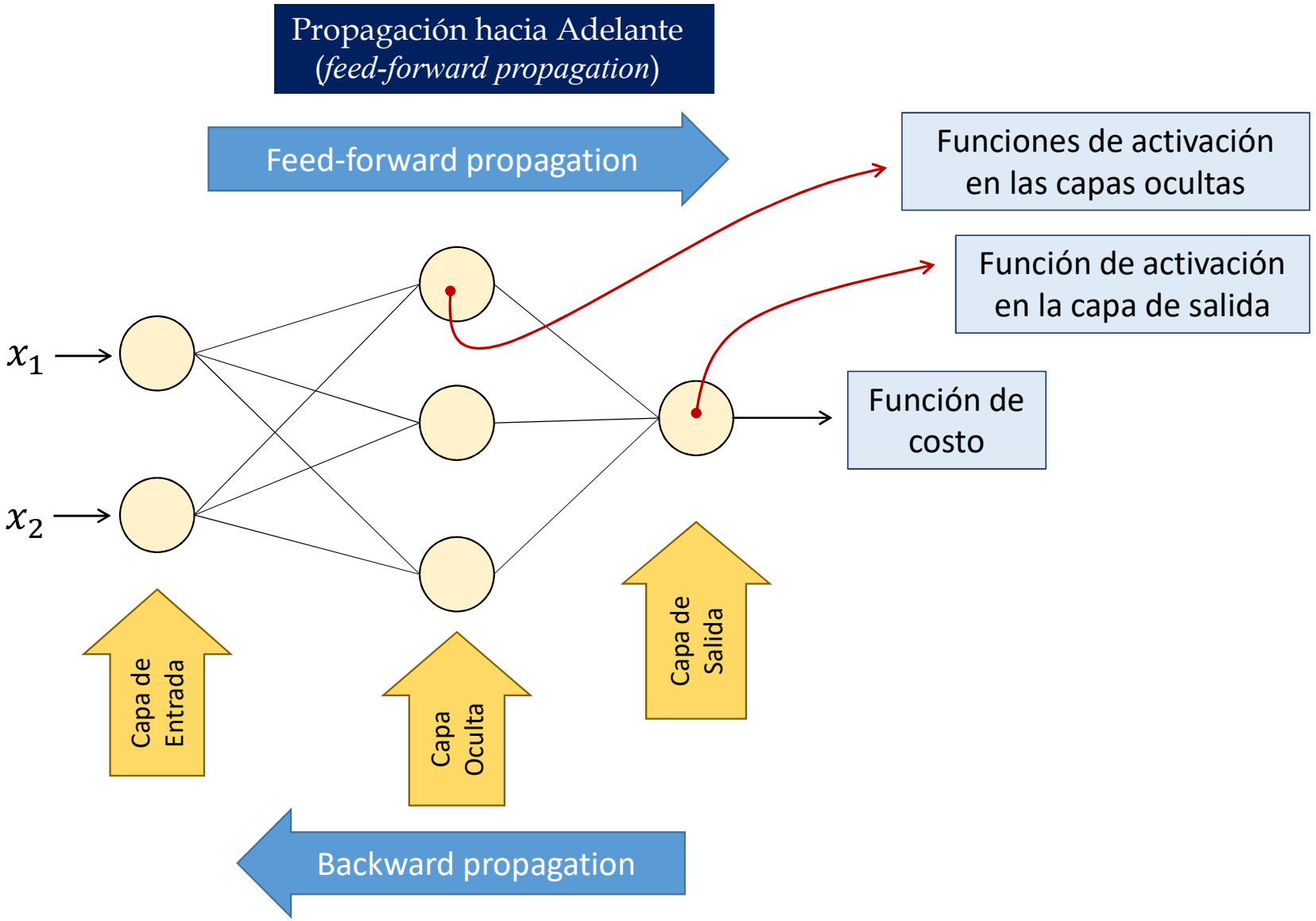
$$\sigma_W = \sqrt{\frac{6}{n_{in} + n_{out}}}$$

Ya que la varianza de una distribución uniforme $U[-a, a]$ está dada como $\frac{a^2}{3}$.

A estos parámetros de inicialización se les conoce como **inicialización de Xavier**, o **inicialización de Glorot**, por uno de sus autores.

Propagación hacia Adelante
(*feed-forward propagation*)

- problema biclase -


Propagación hacia Adelante
(feed-forward propagation)
Feed-forward propagation
Funciones de activación en las capas ocultas
Función de activación en la capa de salida
Función de costo
x_1
x_2
Capa de Entrada
Capa Oculta
Capa de Salida
Backward propagation

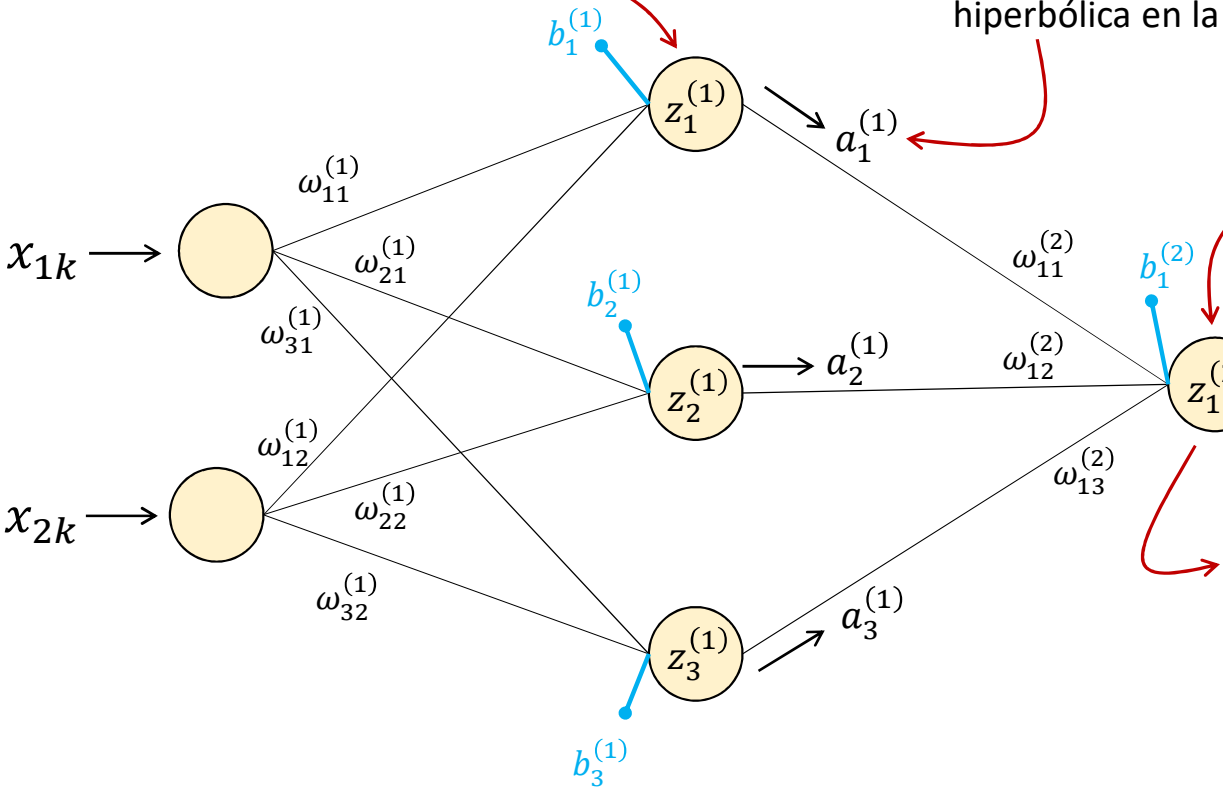
Ejemplo del método de propagación hacia adelante en una MLP.
En particular veamos el comportamiento de la primera neurona en la siguiente red:

Propagación hacia Adelante (*feed-forward propagation*)

1-Combinación lineal dentro de cada neurona de la capa oculta, en particular en $z_1^{(1)}$:

$$z_1^{(1)} = \omega^T x_k + b_1^{(1)} = \left(\omega_{11}^{(1)}, \omega_{12}^{(1)}\right)\begin{pmatrix} x_{1k} \\ x_{2k} \end{pmatrix} + b_1^{(1)} = \omega_{11}^{(1)} x_{1k} + \omega_{12}^{(1)} x_{2k} + b_1^{(1)}$$

2-Función de activación tangente hiperbólica en la capa oculta: $a_1^{(1)} = tanh\left(z_1^{(1)}\right)$



3-Combinación lineal dentro de la neurona/capa de salida:

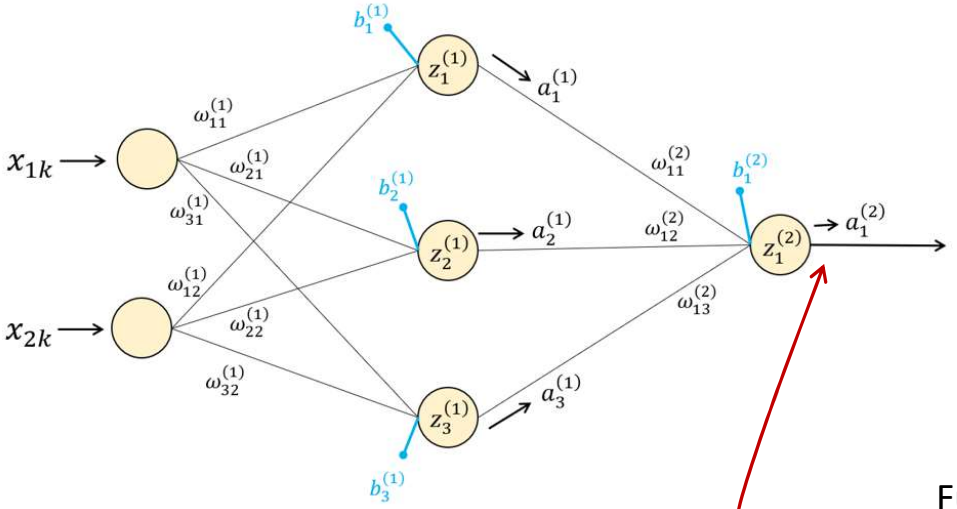
$$z_1^{(2)} = \left(\omega_{11}^{(2)}, \omega_{12}^{(2)}, \omega_{13}^{(2)}\right)\begin{pmatrix} a_1^{(1)} \\ a_2^{(1)} \\ a_3^{(1)} \end{pmatrix} + b_1^{(2)}$$

4-Función de activación sigmoide en la neurona/capa de salida:

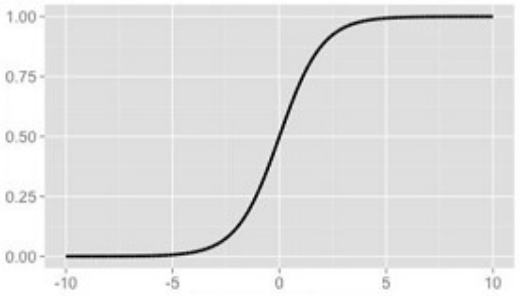
$$a_1^{(2)} = sigmoid\left(z_1^{(2)}\right)$$

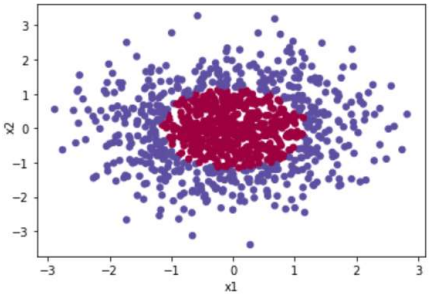
Propagación hacia Adelante (*feed-forward propagation*)

De una predicción de valor real a una predicción binaria:



Función Sigmoide: σ



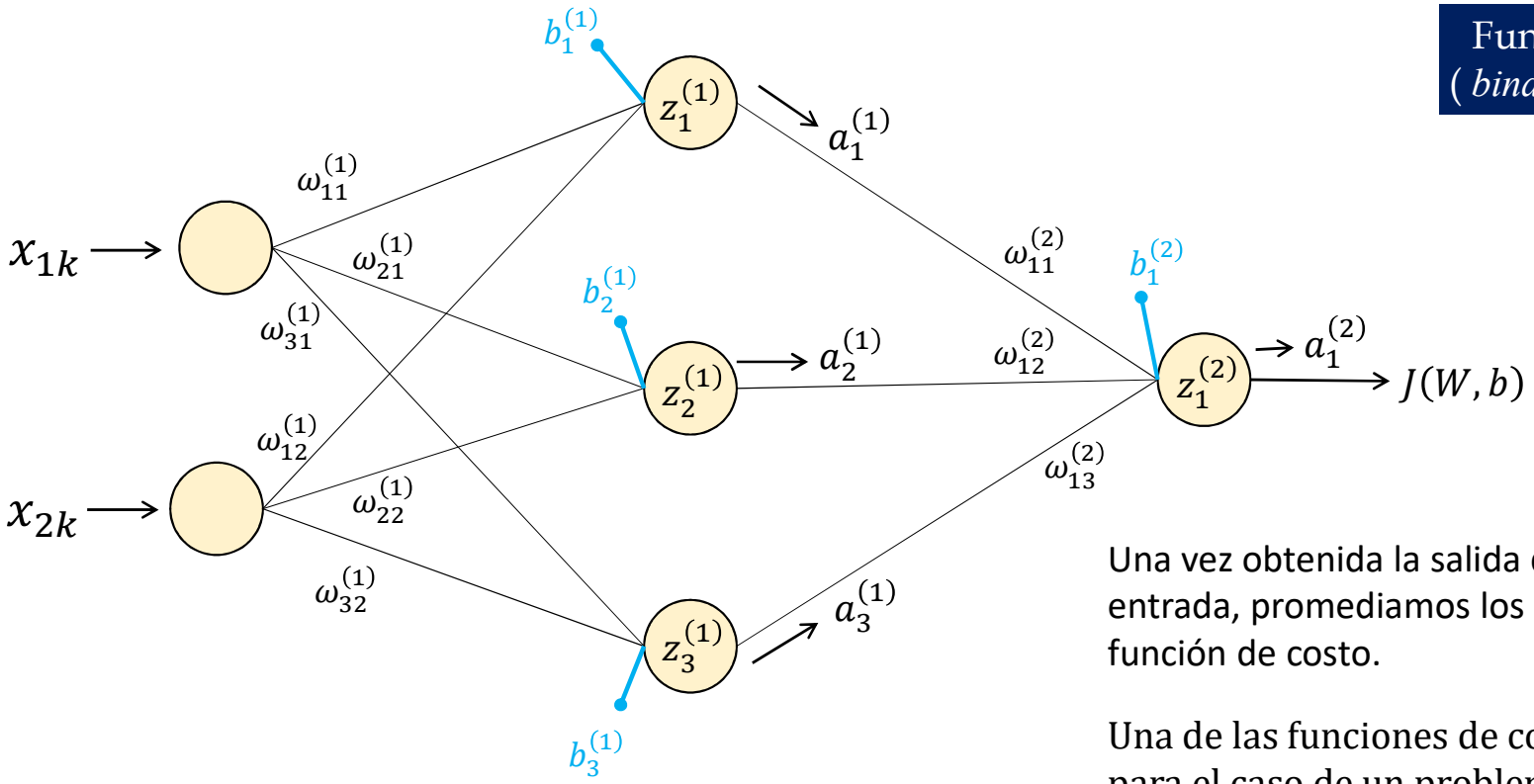


Función de activación sigmoide en la capa de salida: $a_1^{(2)} = sigmoid\left(z_1^{(2)}\right)$

El valor $a_1^{(2)}$ decimos que es **la predicción del modelo** para el dato de entrada (x_{1k}, x_{2k}).

Así, depués de llevar a cabo la propagación hacia adelante con el dato de entrada (x_{1k}, x_{2k}), la salida de la función sigmoide obtenida $a_1^{(2)} = sigmoid\left(z_1^{(2)}\right) \in (0,1)$ la podemos interpretar como **la probabilidad** de que dicho dato de entrada pertenezca a la clase del 0 si $a_1^{(2)} \leq 0$ (por ejemplo, el conjunto de los puntos rojos), o bien pertenezca a la clase del 1 si $a_1^{(2)} > 0$ (por ejemplo la clase de los puntos azules).

Función de Costo/Pérdida
(*binary cross entropy loss/cost*)



En relación a la notación del diagrama, los valores de salida o predicciones $\hat{y}_k$ estarían dados como $a_{1k}^{(2)}$.

Una vez obtenida la salida de todos los m datos de entrada, promediamos los errores con base a alguna función de costo.

Una de las funciones de costo utilizadas usualmente para el caso de un problema de clasificación biclase es la llamada log-loss, definida como:

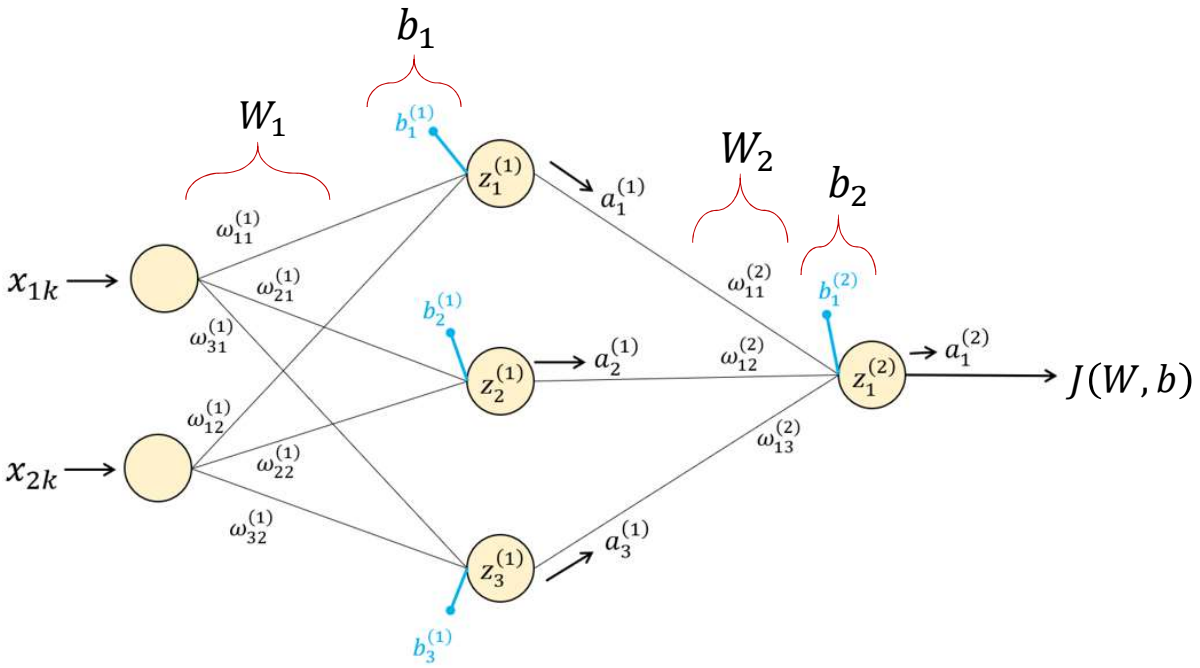
$$J = -\frac{1}{m}\sum_{k=1}^{m}[y_k \log_2(\hat{y}_k) + (1-y_k)\log_2(1-\hat{y}_k)]$$

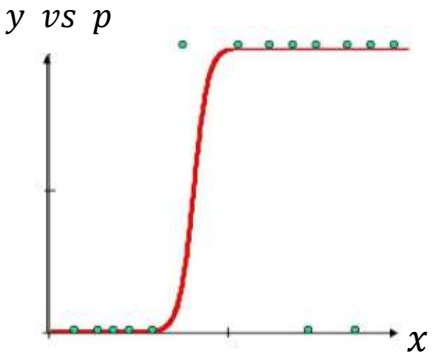
donde $\hat{y}_k$ es la predicción del dato de entrada k-ésimo y y_k su valor real o etiqueta..

Propagación hacia Atrás
(*Back-propagation*)

- problema biclase -

Para un problema de clasificación biclase, consideremos en este ejemplo una arquitectura de red neuronal con una capa oculta con funciones de activación Tangente Hiperbólica y una neurona de salida con función de activación Sigmoide.





La pregunta es ¿cómo seleccionamos y ajustamos los pesos $W_1,\ b_1, W_2, b_2$ para que la red neuronal aproxime los valores de salida 0 y 1, en cada dato de entrada?

El método para ir mejorando los pesos de la red se llama propagación hacia atrás (backpropagation) y está basado en las derivadas parciales.

La técnica de propagación hacia atrás (backpropagation) fue redescubierta por el británico Geoffrey Hinton y el estadounidense David Rumelhart en 1986, con el objetivo de mejorar los pesos de la red neuronal en cada iteración.

Obtengamos primero las derivadas de las funciones sigmoide y tangente hiperbólica, que las necesitaremos después.

Para la función logística, en particular sigmoide :

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

Derivamos usando la regla del cociente:

$$\frac{dz}{dx} = \frac{(1 + e^{-x})(0) - (1)(-e^{-x})}{(1 + e^{-x})^2}$$

$$= \frac{1}{(1 + e^{-x})^2} e^{-x}$$

y como $e^{-x} = \frac{1}{z} - 1$, sustituyendo:

$$= z^2 \left(\frac{1}{z} - 1\right)$$

$$= z(1 - z)$$

Es decir, en el caso de la función sigmoide:

$$\frac{dz}{dx} = z(1 - z)$$

Para la función tangente hiperbólica:

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

$$\frac{dz}{dx} = \frac{(e^x + e^{-x})\{e^x - e^{-x}(-1)\} - (e^x - e^{-x})\{e^x + e^{-x}(-1)\}}{(e^x + e^{-x})^2}$$

$$= \frac{(e^x + e^{-x})^2 - (e^x - e^{-x})^2}{(e^x + e^{-x})^2}$$

$$= 1 - \frac{(e^x - e^{-x})^2}{(e^x + e^{-x})^2}$$

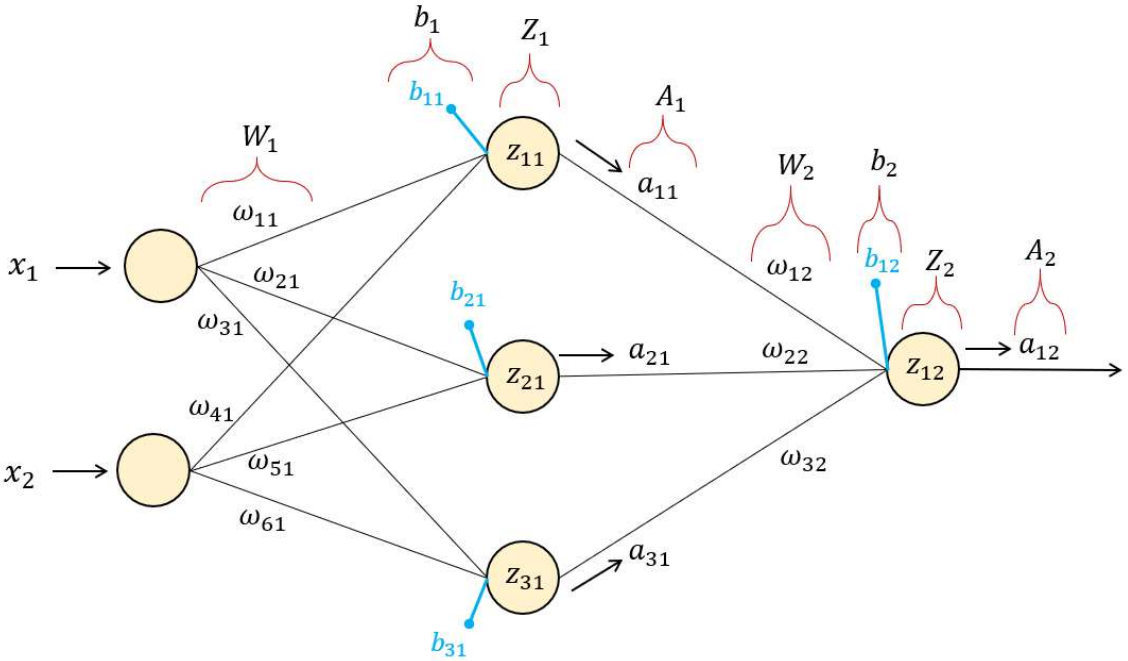
$$= 1 - z^2$$

Es decir, en el caso de la función tangente hiperbólica

$$\frac{dz}{dx} = 1 - z^2$$

Propagación hacia Atrás (*Back-propagation*)

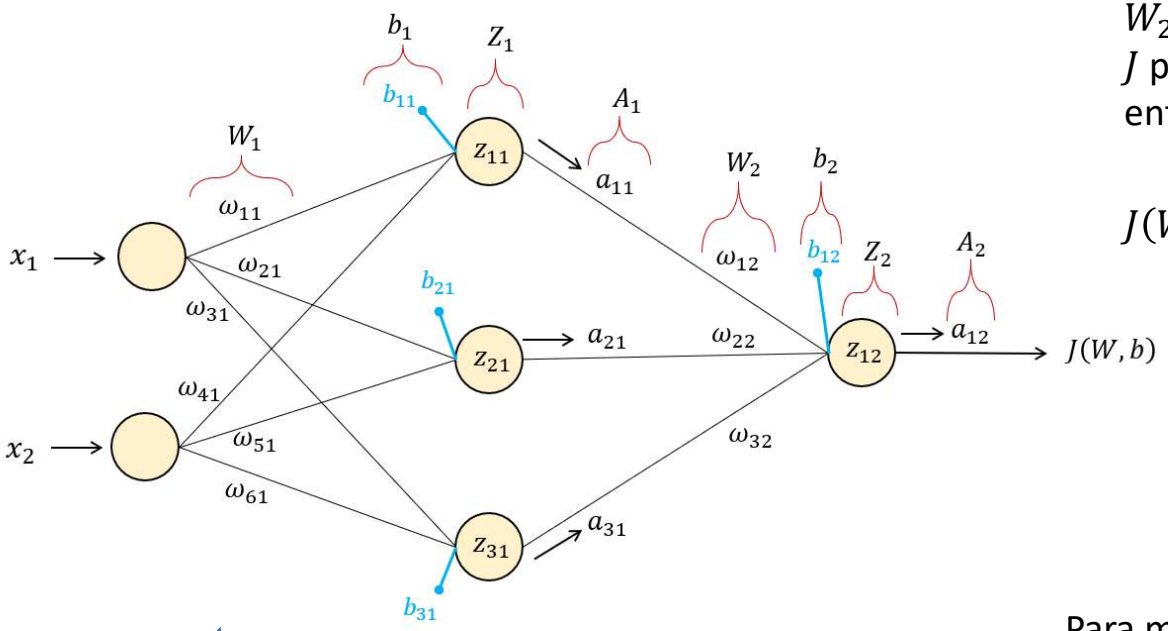
Para cuestiones del ejercicio simplifiquemos la notación como se muestra en la figura:



$$J(W,b) = -\frac{1}{m}\sum_{k=1}^{m}[y_k log(A_2) + (1-y_k) log(1-A_2)]$$

W_1 : Pesos de la capa inicial a la oculta.
b_1 : sesgo/umbral/bias de la capa inicial.
$Z_1 = W_1^T X + b_1$: Combinación lineal
$A_1 = Tanh(Z_1)$: Función de activación no-lineal.

W_2 : Pesos de la capa oculta a la de salida.
b_2 : sesgo/umbral/bias de la capa de salida.
$Z_2 = W_2^T A_1 + b_2$: Combinación lineal.
$A_2 = Sigmoid(Z_2)$: Función de activación no-lineal.



Backward propagation

Se desean encontrar los valores de los pesos W_1, b_1, W_2, b_2 que minimicen el valor de la función de costo J para los m datos de entrada del conjunto de entrenamiento:

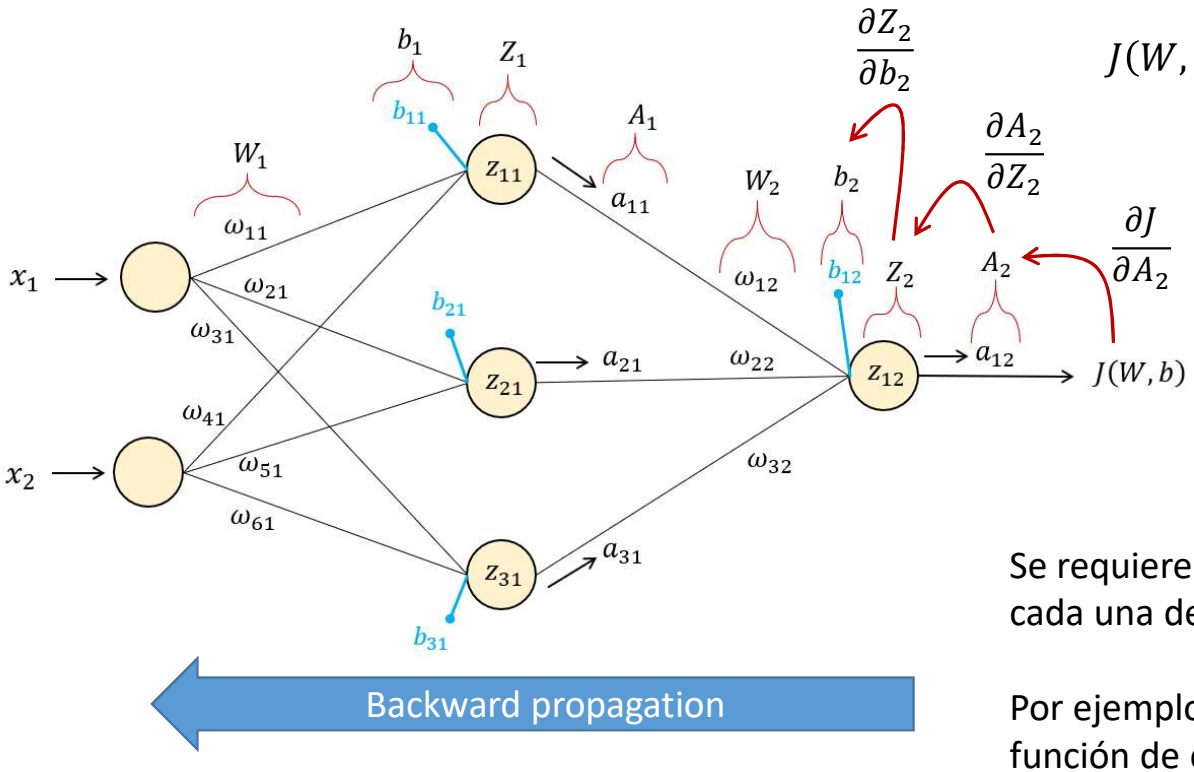
$$J(W,b) = -\frac{1}{m}\sum_{k=1}^{m}[y_k log(A_2) + (1-y_k) log(1-A_2)]$$

Para minimizar la función de costo J mediante el método de propagación hacia atrás, debemos obtener las derivadas parciales de J con respecto a $W_1,\ b_1, W_2, b_2$:

Se requiere hacer uso de regla de la cadena para calcular cada una de estas derivadas parciales:

$$\frac{\partial J}{\partial b_2} \qquad \frac{\partial J}{\partial W_2} \qquad \frac{\partial J}{\partial b_1} \qquad \frac{\partial J}{\partial W_1}$$

Propagación hacia Atrás (*Back-propagation*)



$$J(W,b) = -\frac{1}{m}\sum_{k=1}^{m}[y_k log(A_2) + (1-y_k) log(1-A_2)]$$

$$A_2 = sigmoid(Z_2)$$

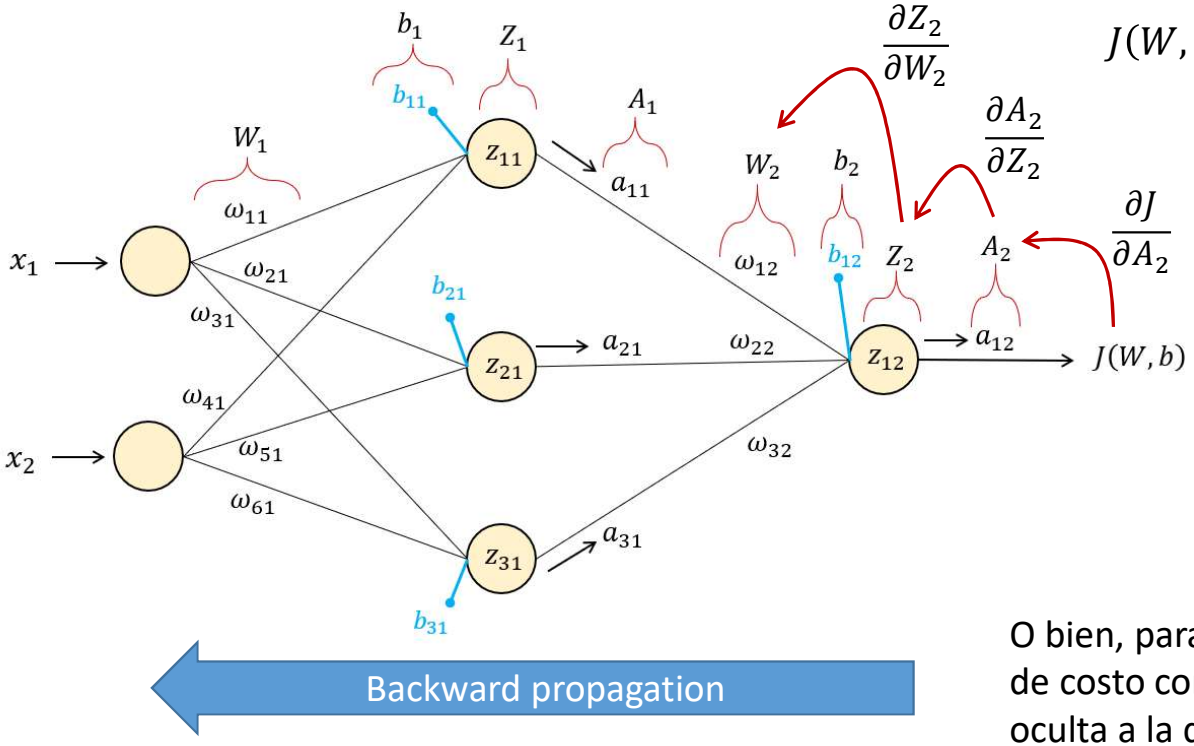
$$Z_2 = W_2^T A_1 + b_2$$

Se requiere hacer uso de regla de la cadena para calcular cada una de estas derivadas parciales.

Por ejemplo, para obtener la razón de cambio de la función de costo con respecto al bias b_2:

$$\frac{\partial J}{\partial b_2} = \frac{\partial J}{\partial A_2}\frac{\partial A_2}{\partial Z_2}\frac{\partial Z_2}{\partial b_2}$$

Propagación hacia Atrás (*Back-propagation*)



Backward propagation

$$J(W,b) = -\frac{1}{m}\sum_{k=1}^{m}[y_k log(A_2) + (1-y_k) log(1-A_2)]$$

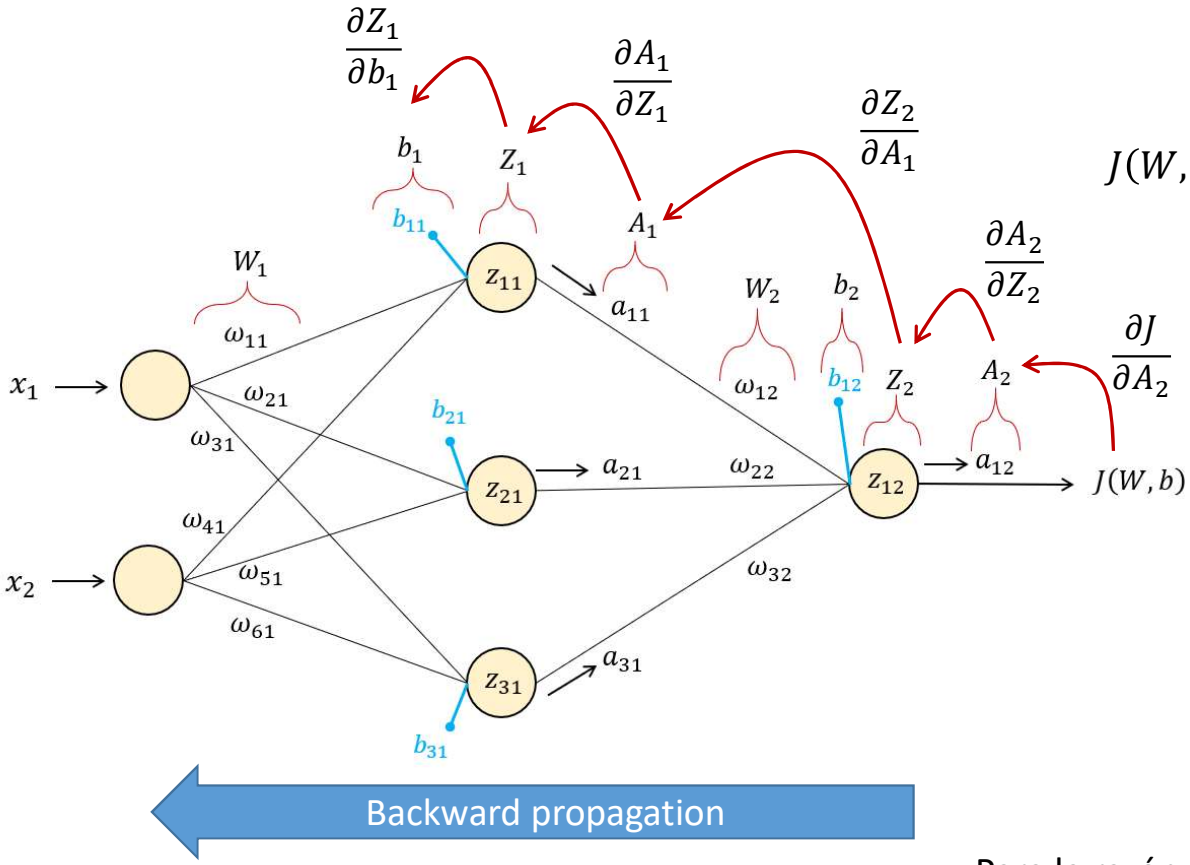
$$A_2 = sigmoid(Z_2)$$

$$Z_2 = W_2^T A_1 + b_2$$

O bien, para la razón de cambio de la función de costo con respecto al los pesos de la capa oculta a la de salida, W_2 sería:

$$\frac{\partial J}{\partial W_2} = \frac{\partial J}{\partial A_2}\frac{\partial A_2}{\partial Z_2}\frac{\partial Z_2}{\partial W_2}$$

Propagación hacia Atrás (*Back-propagation*)



$$J(W,b) = -\frac{1}{m}\sum_{k=1}^{m}[y_k log(A_2) + (1-y_k) log(1-A_2)]$$

$$A_2 = sigmoid(Z_2)$$

$$Z_2 = W_2^T A_1 + b_2$$

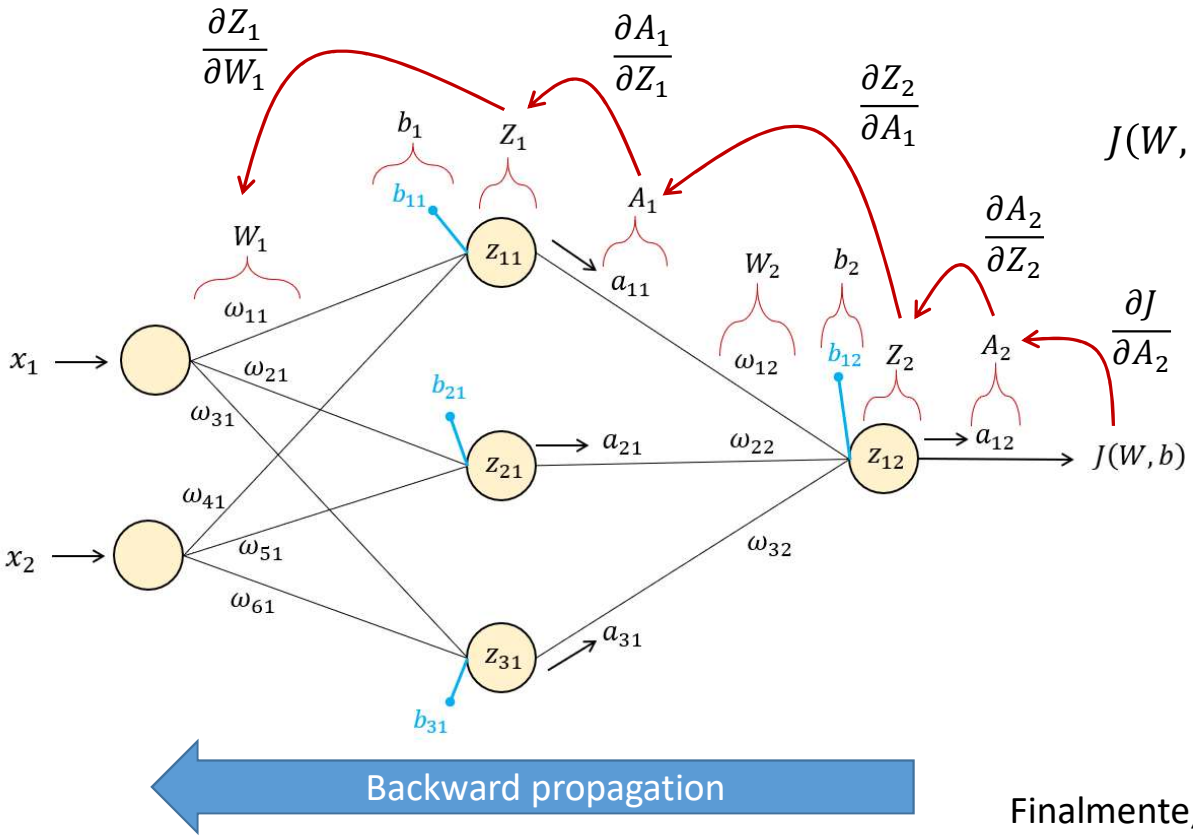
$$A_1 = tanh(Z_1)$$

$$Z_1 = W_1^T X + b_1$$

Para la razón de cambio con respecto a b_1:

$$\frac{\partial J}{\partial b_1} = \frac{\partial J}{\partial A_2}\frac{\partial A_2}{\partial Z_2}\frac{\partial Z_2}{\partial A_1}\frac{\partial A_1}{\partial Z_1}\frac{\partial Z_1}{\partial b_1}$$

Propagación hacia Atrás (*Back-propagation*)



$$J(W,b) = -\frac{1}{m}\sum_{k=1}^{m}[y_k log(A_2) + (1-y_k) log(1-A_2)]$$

$$A_2 = sigmoid(Z_2)$$

$$Z_2 = W_2^T A_1 + b_2$$

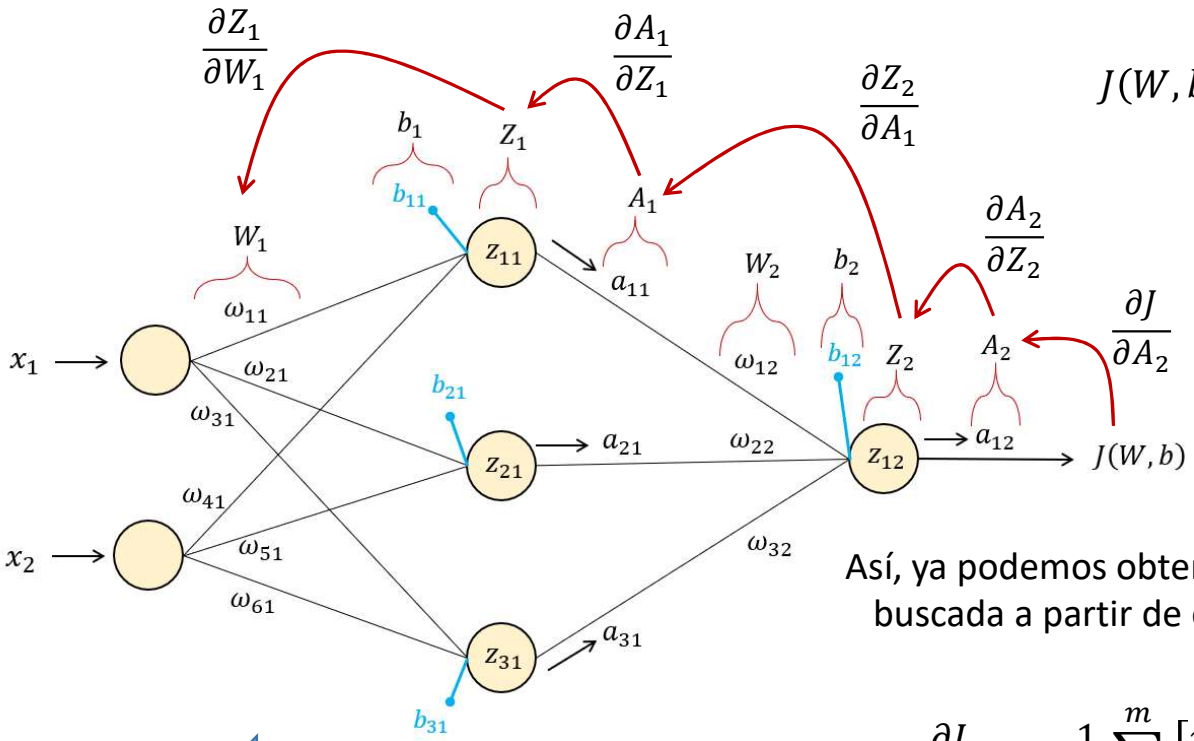
$$A_1 = tanh(Z_1)$$

$$Z_1 = W_1^T X + b_1$$

Finalmente, para las razones de cambio de la función de costo con respecto a los pesos y bias de la capa de entrada a la oculta nos apoyaremos en las parciales anteriores:

$$\frac{\partial J}{\partial W_1} = \frac{\partial J}{\partial A_2}\frac{\partial A_2}{\partial Z_2}\frac{\partial Z_2}{\partial A_1}\frac{\partial A_1}{\partial Z_1}\frac{\partial Z_1}{\partial W_1}$$

Propagación hacia Atrás (*Back-propagation*)



$$J(W,b) = -\frac{1}{m}\sum_{k=1}^{m}[y_k log(A_2) + (1-y_k) log(1-A_2)]$$

$$A_2 = sigmoid(Z_2)$$

$$Z_2 = W_2^T A_1 + b_2$$

$$A_1 = tanh(Z_1)$$

$$Z_1 = W_1^T X + b_1$$

Así, ya podemos obtener la derivada buscada a partir de cada factor de:

$$\frac{\partial J}{\partial W_1} = \frac{\partial J}{\partial A_2}\frac{\partial A_2}{\partial Z_2}\frac{\partial Z_2}{\partial A_1}\frac{\partial A_1}{\partial Z_1}\frac{\partial Z_1}{\partial W_1}$$

$$\frac{\partial J}{\partial A_2} = -\frac{1}{m}\sum_{k=1}^{m}\left[\frac{y_k}{A_2} - \frac{1-y_k}{1-A_2}\right]$$

Se puede omitir la constante $log_2(e)$ de la derivada del logaritmo base 2 sin pérdida de generalidad.

$$\frac{\partial A_2}{\partial Z_2} = A_2(1-A_2) \qquad \frac{\partial A_1}{\partial Z_1} = 1 - A_1^2$$

$$\frac{\partial Z_2}{\partial A_1} = W_2 \qquad \frac{\partial Z_1}{\partial W_1} = X$$

NOTA: Las derivadas matriciales son análogas a las que viste en tus cursos de Cálculo, pero queda fuera del alcance de este curso. Este ejemplo es a manera de ilustración.

Propagación hacia Atrás (*Back-propagation*)

Finalmente los pesos y bias se mejoran en cada iteración sumando un incremento delta mediante la fórmula:

$$\theta_{(new)} = \theta_{(old)} + \Delta\theta\Big|_{\theta_{(old)}}$$

Que para el caso del método del gradiente descendente, tendremos sustituyendo el incremento delta por el gradiente (parciales) de la función de costo en cada parámetro del modelo:

$$\theta_{(new)} = \theta_{(old)} - \mu\,\nabla J(\theta)\Big|_{\theta=\theta(old)}$$

Entonces, la mejora en cada iteración en los pesos W_1 de la primer capa del modelo MLP con las características indicadas previamente y la fórmula anterior, se actualizan incluyendo la expresión siguiente en lugar del gradiente:

$$\frac{\partial J}{\partial W_1} = \frac{\partial J}{\partial A_2}\frac{\partial A_2}{\partial Z_2}\frac{\partial Z_2}{\partial A_1}\frac{\partial A_1}{\partial Z_1}\frac{\partial Z_1}{\partial W_1} = \left\{-\frac{1}{m}\sum_{k=1}^{m}\left[\frac{y_k}{A_2} - \frac{1-y_k}{1-A_2}\right]\right\}\{A_2(1-A_2)\}\{W_2\}\{1-A_1^2\}\{X\}$$

W_1 : Pesos de la capa inicial a la oculta.
b_1 : sesgo/umbral/bias de la capa inicial.
$Z_1 = W_1^T X + b_1$: Combinación lineal
$A_1 = Tanh(Z_1)$: Función de activación no-lineal.

W_2 : Pesos de la capa oculta a la de salida.
b_2 : sesgo/umbral/bias de la capa de salida.
$Z_2 = W_2^T A_1 + b_2$: Combinación lineal.
$A_2 = Sigmoid(Z_2)$: Función de activación no-lineal.