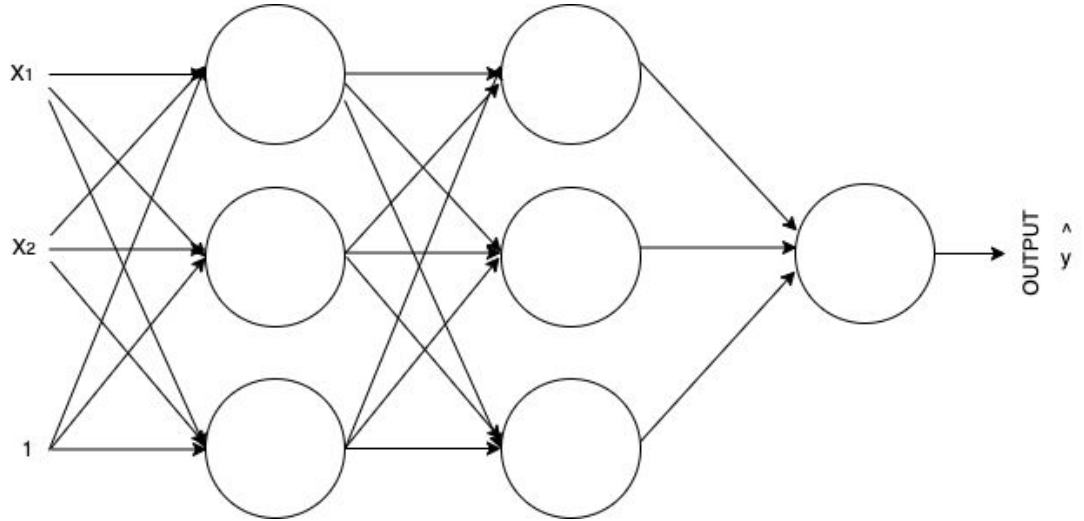
Machine Learning II
Perceptrón Multicapa

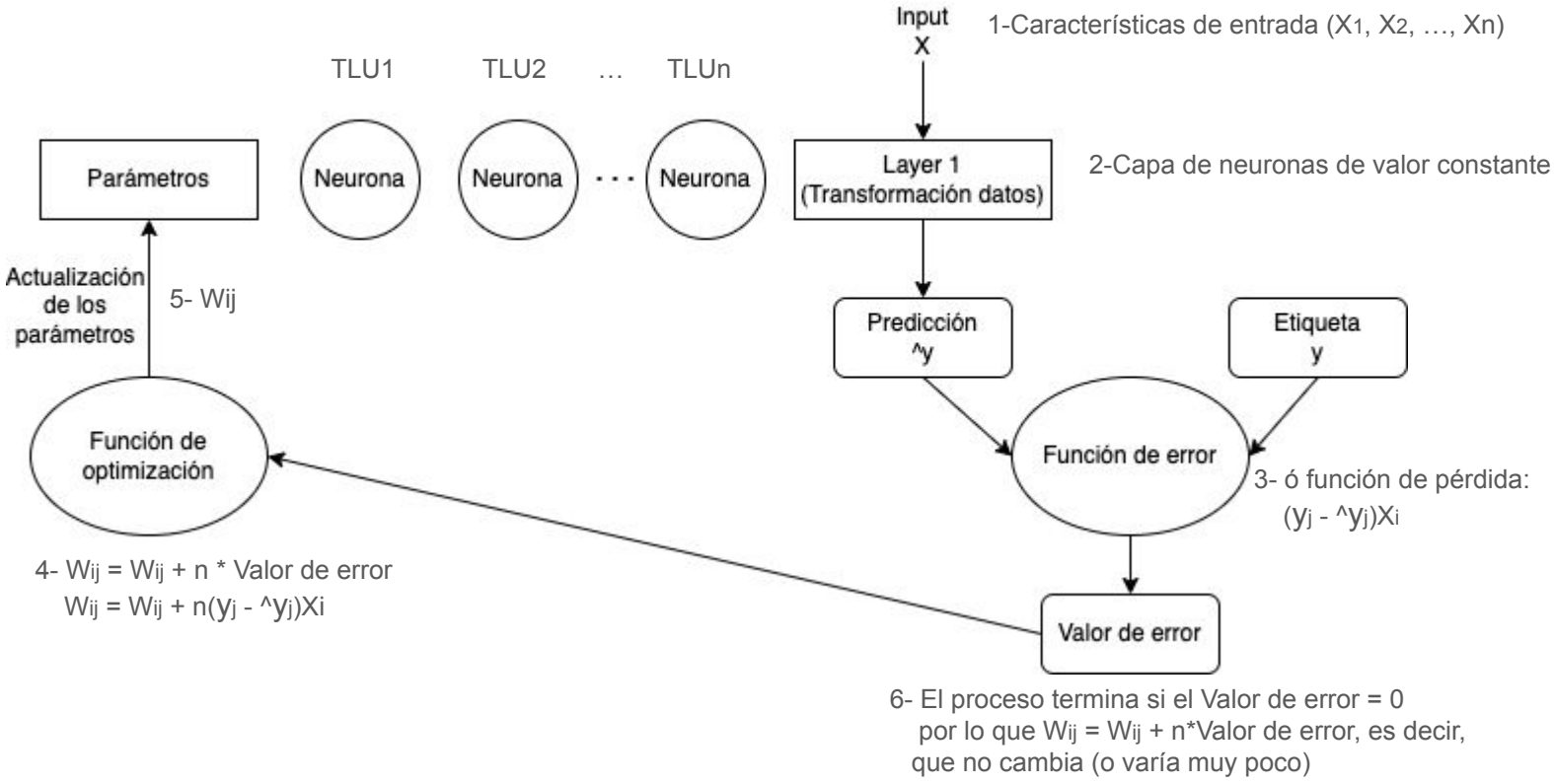
U-TAD

Introducción al Perceptrón Multicapa

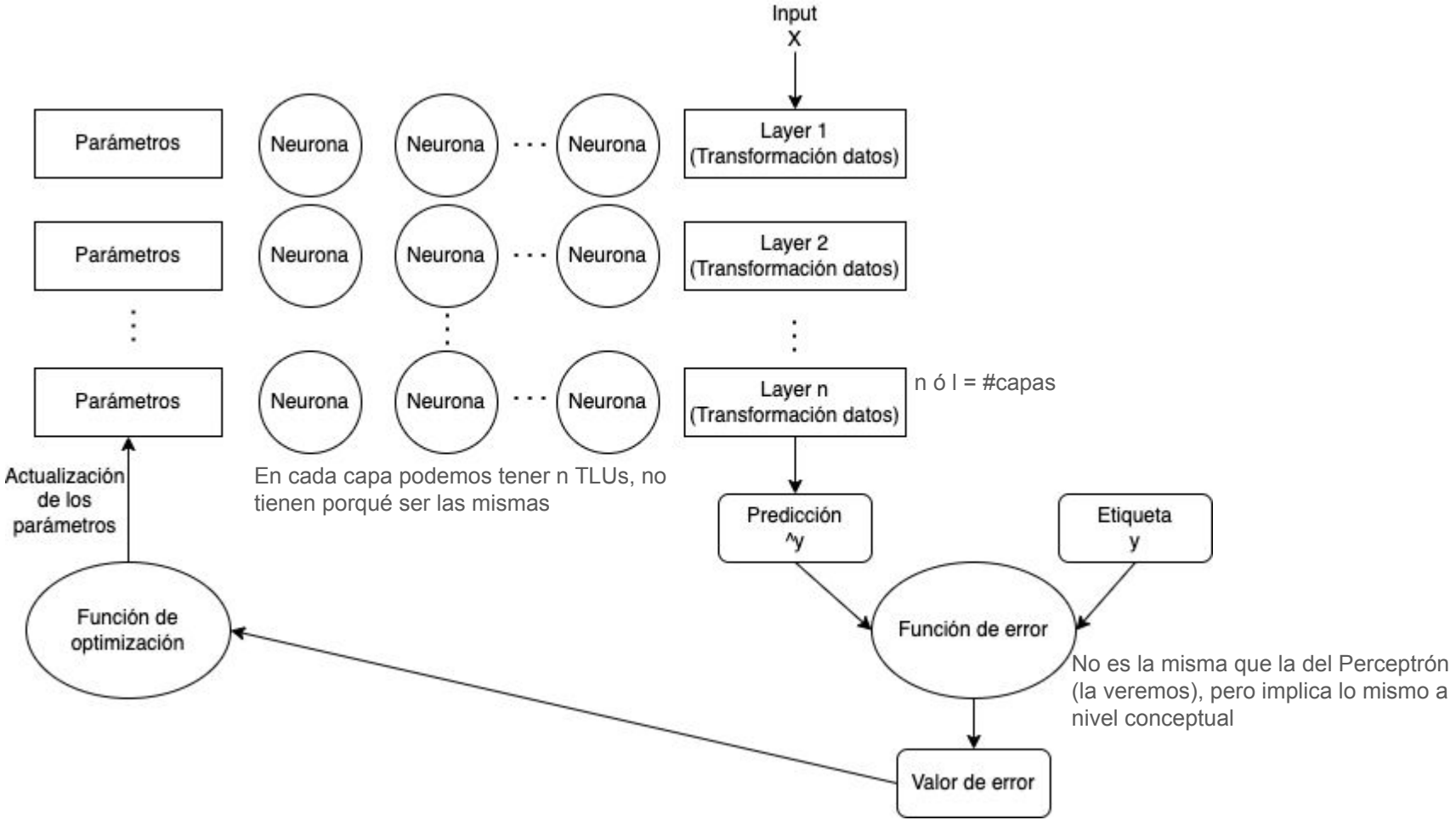
- El Perceptrón multicapa o Deep feedforward network es el modelo más popular dentro del Deep Learning (y base para resolver problemas concretos)
- Se denomina feedforward porque la información fluye desde las entradas hasta la salida sin conexiones de feedback (la información siempre fluye desde el Input Layer hasta el Output Layer en la misma dirección).
- Se caracterizan por la composición de numerosas funciones



Arquitectura del Perceptrón Multicapa



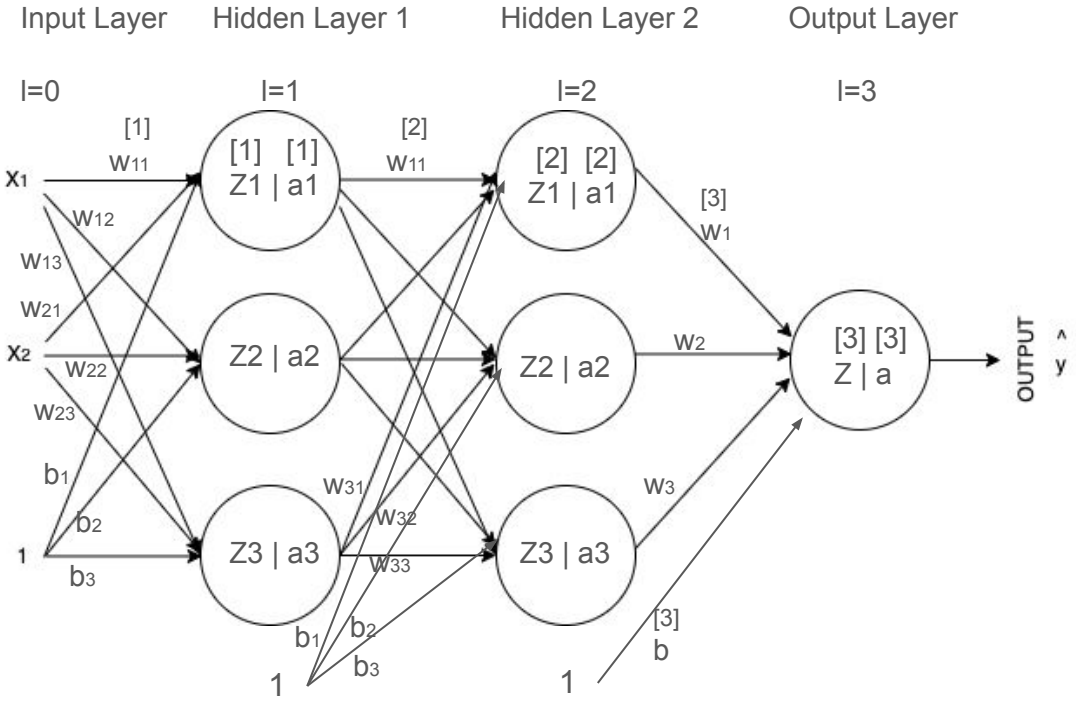
Componentes de una RNA profunda



Notación y funcionamiento del Perceptrón multicapa

- El perceptrón Multicapa se compone de una **input layer** y **una o más capas de TLUs**
- Cada capa de TLUs intermedia se denomina **hidden layer**
- La capa final de TLUs es la **output layer**
- Todas las capas excepto la output layer incluyen la **bias neuron**
- Todas las capas se encuentran totalmente conectadas (**fully connected**) con las siguientes capas
- Cuando una ANN tiene dos o más hidden layers, se denomina Deep Neural Network (DNN)

Perceptrón Multicapa



l = #capas RNA

X1 y X2 las podemos considerar una neurona con un valor constante tal que, a veces, lo podemos ver como:

$a_1^{[0]} = X_1$ $a_2^{[0]} = X_2$

Fórmulas:

$$Z_1^{[1]} = W_{11}^{[1]} * a_1^{[0]} + W_{21}^{[1]} * a_2^{[0]} + b_1^{[1]}$$

$$a_1^{[1]} = \text{activation}(Z_1^{[1]})$$

(que será el input de la siguiente neurona multiplicado por el peso asociado w)

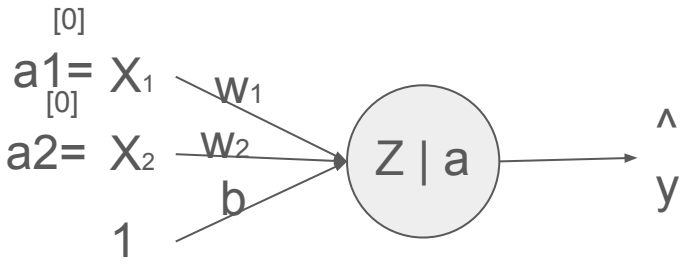
$$Z_1^{[2]} = W_{11}^{[2]} * a_1^{[1]} + W_{21}^{[2]} * a_2^{[1]} + W_{31}^{[2]} * a_3^{[1]} + b_1^{[2]}$$

$$a_1^{[2]} = \text{activation}(Z_1^{[2]})$$

Componentes principales del Perceptrón Multicapa

- Durante mucho tiempo una de las mayores limitaciones del Perceptrón Multicapa era que no existía una forma adecuada de entrenarlo
- En 1986, D.E. Rumelhart presenta un algoritmo que revoluciona la manera de entrenar el Perceptrón Multicapa, el algoritmo **backpropagation**
- El cambio clave del algoritmo backpropagation respecto a los utilizados anteriormente fue reemplazar la función de activación *Heaviside step function* por la **sigmoide**
- En la actualidad, existen otras funciones de activación populares, como la **tanh(z)** o la **ReLU(z)**

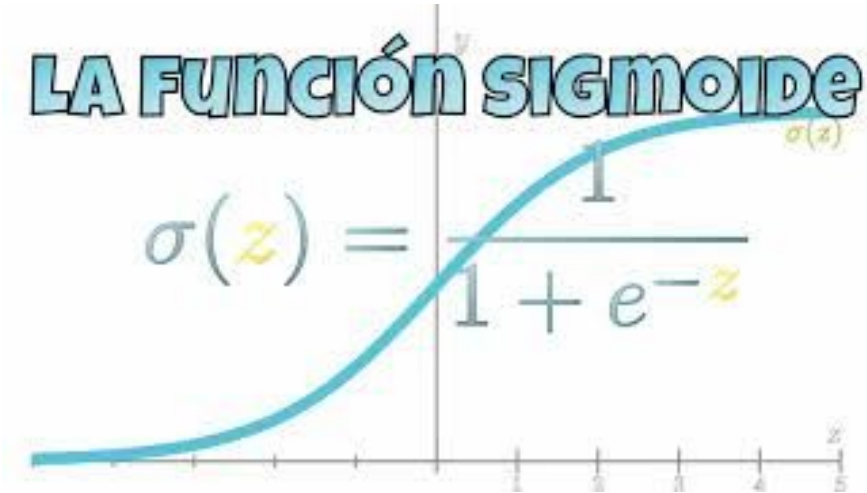
Función de activación (sigmoide)



$Z = X_1W_1 + X_2W_2 + b = a_1W_1 + a_2W_2 + b$

^ (solo cambia la función de activación)
y = activation(Z) = sigmoid(Z)

A diferencia de *heaviside*, que se basaba en un threshold estático, y no devuelve valores intermedios, solo 0 o 1 en función del threshold. Ahora tendremos valores continuos entre 0 y 1:
0 <= sigmoid(Z) <= 1
^
y = a(Z(X)) = sigmoid(x1w1 + x2w2 +b) =
$\frac{1}{1 + e^{-(x1w1 + x2w2 + b)}}$ -> probabilidad de que pertenezca a la clase positiva (y = 1)



Caso práctico: Límite de decisión del Perceptrón Multicapa

Visualizando el límite de decisión del perceptrón multicapa.ypynb

(Veremos más adelante cómo hacerlo con Keras, más optimizado que con sklearn)

Obtenemos límites de decisión más flexibles y complejas (funciones matemáticas que son líneas curvas).

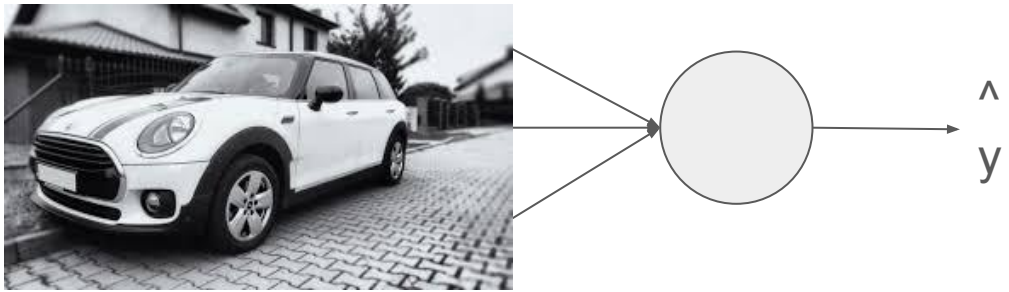
Introducción a Forward Propagation

Se basa en que la información fluye desde las entradas hasta las salidas, sin que haya ningún tipo de conexión de feedback. Es decir, el output de una neurona siempre se va a corresponder con el input de una neurona posterior, y nunca se va a corresponder con el input de una neurona anterior.

Este concepto forma uno de los procesos fundamentales y más importantes, no solamente del perceptrón multicapa, sino de la mayoría de las redes neuronales, que en este caso se corresponden con Feed Forward Networks.

Input Layer

Output Layer



Problema:
Quiero reconocer si aparece o no un coche en la imagen

Introducción a Forward Propagation

Para resolver el problema, se requiere de un montón de imágenes en las que aparecen coches y en las que no, y etiquetarlas (1 hay coche, 0 no hay coche)

Se toman las imágenes sin la etiqueta y se le proporciona a la red neuronal, poniendo un valor aleatorio para los pesos (parámetros del modelo).

La red neuronal recibirá una imagen, realizará una predicción determinada y, después, utilizará una función de error de mi red neuronal respecto a la etiqueta que tenía en el conjunto de datos.

Ya estamos usando el Forward Propagation, dado que para poder llegar a calcular ese error, primero tengo que haber calculado la predicción. Además, lo utilizaremos una vez entrenada la red neuronal, una vez encontrado ese valor óptimo para los parámetros, y queremos realizar una predicción.

Introducción a Forward Propagation

Cada uno de los píxeles de la imagen en B/N, lo tomamos como una característica de entrada, como un input feature individual, cuyo valor va a ser la intensidad de ese píxel. Por ejemplo, de 300px de alto por 400 px de ancho.

n = #input features = 300*400 = 120000

Tendremos 120k características de entrada que comenzaremos a propagar hacia delante, hacia la TLU: (Z | a (sigmoid))

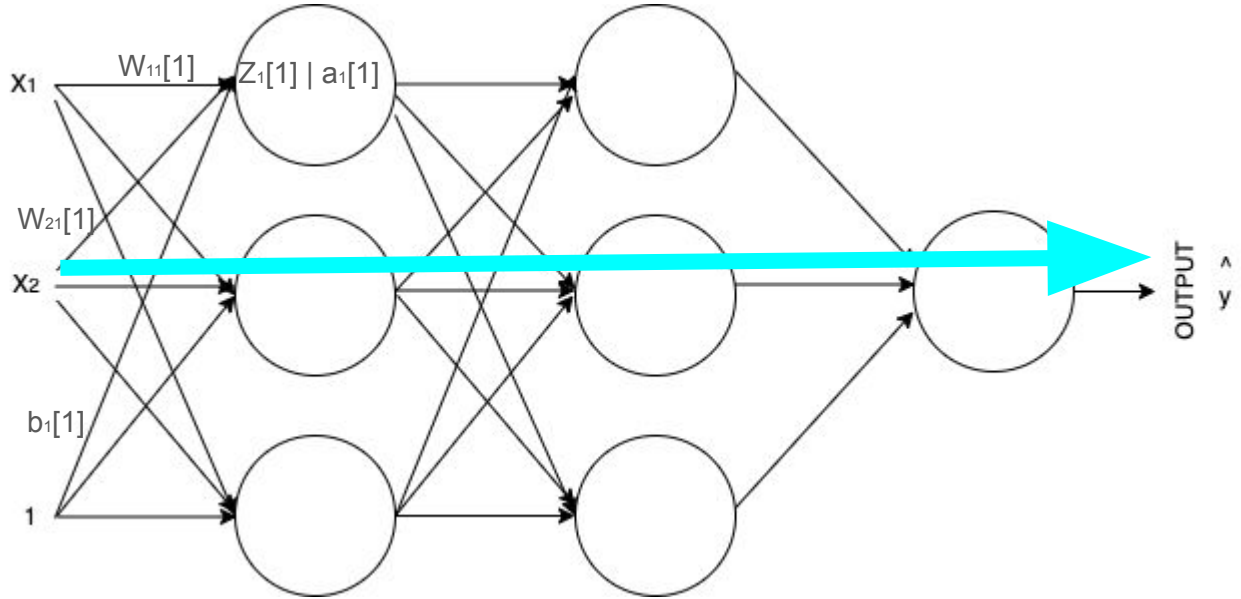
$(X_1, X_2, X_{120000}, 1)$

$Z = X_1 * W_1 + X_2 * W_2 + \ldots + X_{120k} * W_{120k} + b$

^y = a = sigmoid(Z) = 1 / (1+e^(-z))

Forward Propagation con el Perceptrón Multicapa

Input Layer Hidden Layer 1 Hidden Layer 2 Output Layer



*Recordamos que en cada una de las capas intermedias se añade la neurona **bias** (1)

Forward Propagation con el Perceptrón Multicapa

Vamos a ir propagando todas las entradas $a_1[0] = X_1$, $a_2[0] = X_2$, …, por todas las TLUs ($Z_1[1]$, $a_1[1]$, …, $Z[3]$, $a[3]$). Ahora, calculamos las ecuaciones para la primera neurona ($Z_1[1]$, $a_1[1]$):

$Z_1[1] = a_1[0]*W_{11}[1] + a_2[0]*W_{21}[1] + b_1[1]$

$a_1[1] = \text{sigmoid}(Z_1[1]) = 1 / (1+e^{(-z)})$ *Este output será el input de las siguientes neuronas

Entendido esto, las ecuaciones finales serían:

$Z_1[1] = a_1[0]*W_{11}[1] + a_2[0]*W_{21}[1] + b_1[1]$ $\quad$ $a_1[1] = \text{sigmoid}(Z_1[1])$

$Z_2[1] = a_1[0]*W_{12}[1] + a_2[0]*W_{22}[1] + b_2[1]$ $\quad$ $a_2[1] = \text{sigmoid}(Z_2[1])$

$Z_3[1] = a_1[0]*W_{13}[1] + a_2[0]*W_{23}[1] + b_3[1]$ $\quad$ $a_3[1] = \text{sigmoid}(Z_3[1])$

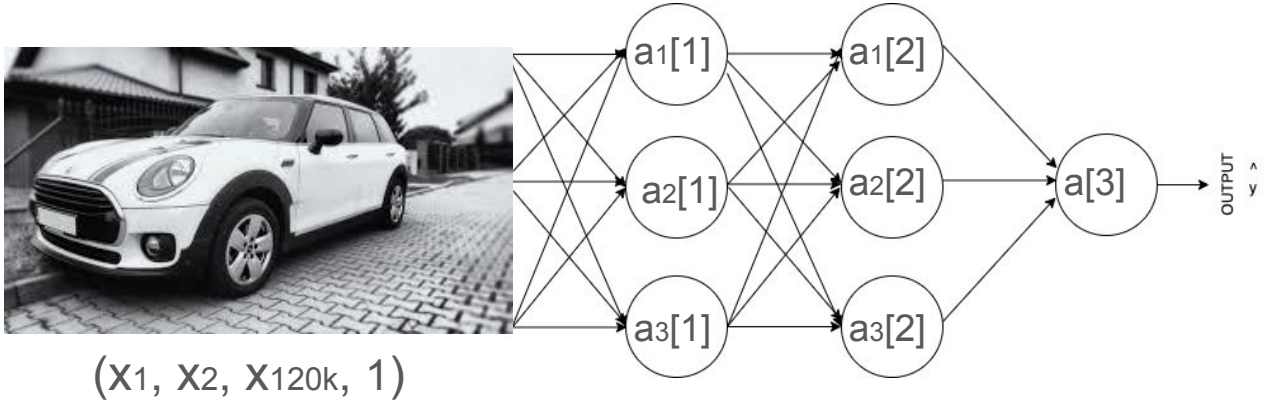
$Z_1[2] = a_1[1]*W_{11}[2] + a_2[1]*W_{21}[2] + a_3[1]*W_{31}[2] + b_1[2]$ $\quad$ $a_1[2] = \text{sigmoid}(Z_1[2])$

$Z_2[2] = a_1[1]*W_{12}[2] + a_2[1]*W_{22}[2] + a_3[1]*W_{32}[2] + b_2[2]$ $\quad$ $a_2[2] = \text{sigmoid}(Z_2[2])$

$Z_3[2] = a_1[1]*W_{13}[2] + a_2[1]*W_{23}[2] + a_3[1]*W_{33}[2] + b_3[2]$ $\quad$ $a_3[2] = \text{sigmoid}(Z_3[2])$

$Z[3] = a_1[2]*W_1[3] + a_2[2]*W_2[3] + a_3[2]*W_3[3] + b[3]$ $\quad$ **$\hat{y} = a[3] = \text{sigmoid}(Z[3])$** $(0 <= \hat{y} <= 1)$

Forward Propagation para múltiples entradas



$(x_1, x_2, x_{120k}, 1)$

$$a_1[1] = \text{sigmoid}(a_1[0]\, W_{11}[1] + a_2[0]\, W_{21}[1] + \ldots + a_{120k}[0]\, W_{\mathbf{120k1}}[1] + b_1[1])$$

Tenemos 120k+1 parámetros por cada neurona de la capa 1, más 4 parámetros por cada neurona de la capa 2, y 4 de la capa de salida, es decir, **360019** parámetros W a calcular.