



3. Fundamentos de redes neuronales

3.1 Introducción

Este capítulo está diseñado para guiarlos a través de los conceptos fundamentales y las estructuras que forman la base de las redes neuronales modernas. Al desentrañar estos elementos, adquirirán no solo una comprensión teórica, sino también práctica de cómo funcionan estas redes y cómo se pueden aplicar para resolver problemas complejos.

Comenzaremos nuestra exploración con el perceptrón, el bloque de construcción más básico de una red neuronal. Aquí, se sumergirán en la comprensión de los pesos y la combinación lineal, elementos esenciales que permiten a una red procesar y aprender de los datos de entrada. También abordaremos las funciones de activación, un componente crucial que añade la capacidad de modelar decisiones no lineales y complejas.

A continuación, avanzaremos hacia redes con varias salidas, explorando cómo se pueden construir para realizar tareas múltiples simultáneamente. Esta sección proporcionará una visión más profunda de cómo las redes neuronales pueden ser versátiles y potentes en el manejo de una variedad de desafíos en el procesamiento de datos.

Profundizaremos aún más en el tema al discutir las redes de varias capas. Estas estructuras más complejas permiten abordar problemas que una neurona simple no puede manejar, abriendo un mundo de posibilidades en términos de modelado y capacidad de predicción.

Finalmente, llegaremos al perceptrón multicapa, una forma de red neuronal que combina múltiples capas de perceptrones para formar una arquitectura capaz de aprender y modelar relaciones extremadamente complejas en los datos. Esta sección será crucial para entender cómo las redes neuronales se han convertido en una herramienta tan poderosa en el campo de la inteligencia artificial.

Cada tema está acompañado de ejercicios matemáticos detallados y ejemplos de programación en Python utilizando NumPy. Estos ejercicios están diseñados para solidificar su comprensión teórica y mejorar sus habilidades prácticas. Mediante la resolución de estos ejercicios, no solo fortalecerán su comprensión de los conceptos, sino que también desarrollarán una intuición práctica que es esencial para aplicar estas técnicas en el mundo real.

Al final de este capítulo, habrán adquirido no solo un conocimiento profundo de los fundamentos

de las redes neuronales, sino también la experiencia práctica necesaria para implementarlos en sus propios proyectos. Este conocimiento y habilidad son esenciales para cualquier aspirante a científico de datos, ingeniero de software o investigador en el campo de la inteligencia artificial.

3.2 Perceptrón

El perceptrón es la unidad de una red neuronal artificial que se basa en el modelo de unidad lógica de umbral (TLU) propuesto por Rosenblatt en 1962. Este modelo hipotético de sistema nervioso o máquina puede aprender a separar correctamente las clases contenidas en un conjunto de datos linealmente separable.

El perceptrón se asemeja a una neurona biológica (ver figura ??), que cuenta con dendritas para adquirir información, un núcleo para procesarla y un axón para enviarla. Sin embargo, en el perceptrón, la información se representa en forma de números, y el procesamiento se realiza mediante operaciones matemáticas.

3.2.1 El modelo del perceptrón

El perceptrón es una función matemática que mapea un conjunto de valores de entrada a otro conjunto de valores de salida. La figura 3.1a muestra un ejemplo de este mapeo. La figura 3.1b muestra los componentes del modelo del perceptrón: entradas (X), pesos (w), sesgo (b), función de activación (f) y salida (\hat{y} o y).

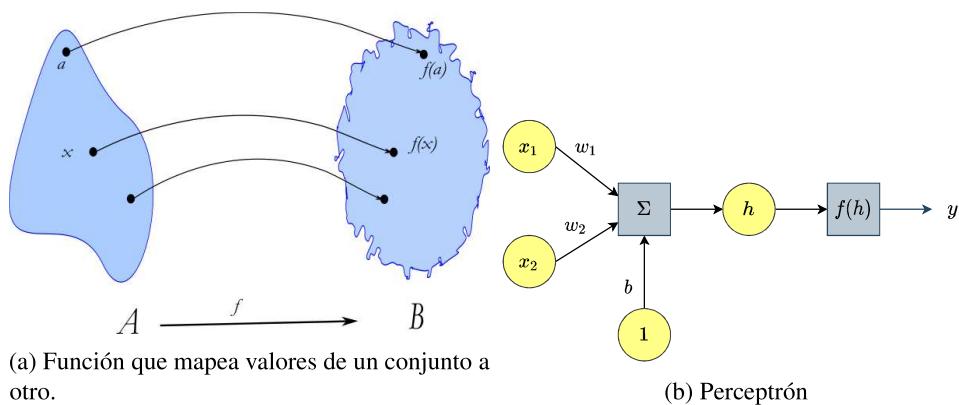


Figure 3.1: El modelo del perceptrón.

Entradas

Las entradas del perceptrón, x , son representaciones numéricas de información capturada de algún entorno. Esta información puede ser extraída con el uso de sensores, o bases de datos (*datasets*), esta información es suministrada a las entradas del perceptrón :

$$\mathbf{X} = [x_1, x_2, \dots, x_n]^T \quad (3.1)$$

donde x es un vector que representa el conjunto que contiene n valores de entradas que tiene el perceptrón.

Pesos

Cada entrada al perceptrón tiene un peso asociado, w , que representa la importancia que tiene esa entrada. Los pesos se multiplican por las entradas para calcular la salida del perceptrón:

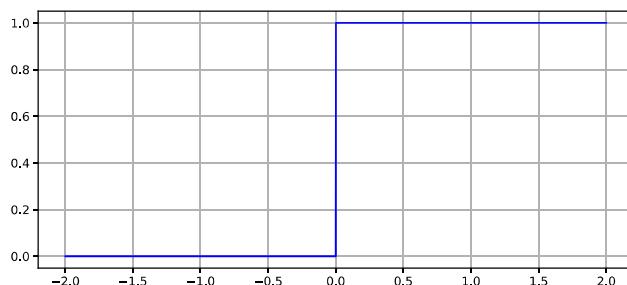


Figure 3.2: Gráfica de función Heaviside o escalón unitario.

$$w_i \cdot x_i \quad (3.2)$$

y considerando todas las entradas que puede presentar, se puede ver de la siguiente manera:

$$\begin{aligned} \mathbf{X} &= [x_1, x_2, \dots, x_n]^T \\ \mathbf{W} &= [w_1, w_2, \dots, w_n] \end{aligned} \quad (3.3)$$

Haciendo uso de estos valores se puede calcular el valor escalar h como:

$$h = \sum_{i=1}^n w_i \cdot x_i \quad (3.4)$$

h representa el nivel de estímulo que presenta el perceptrón resultado de la interacción con los valores recibidos en las entradas.

Funciones de activación

La función de activación es la encargada de determinar la salida que presentará el perceptrón. Rossenblat asumió que algunos puntos de estímulo (entradas) presentan un comportamiento "todo o nada", por lo que utilizó la función escalón unitario o también conocida como función de Heaviside (ver figura 3.2). Esta función se define de la siguiente forma:

$$f(t - a) = \begin{cases} 0 & \text{if } t < a \\ 1 & \text{if } t \geq a \end{cases} \quad (3.5)$$

Donde a es una constante que indica el valor de t en el que la función cambia de 0 a 1.

Para la aplicación se requiere que cualquier valor ≥ 0 de a la salida un valor igual a 1, lo que nos lleva a reescribir la función de la siguiente forma:

$$f(h) = \begin{cases} 0 & \text{if } \sum_{i=1}^n w_i \cdot x_i < 0 \\ 1 & \text{if } \sum_{i=1}^n w_i \cdot x_i \geq 0 \end{cases} \quad (3.6)$$

Sesgo

El valor de sesgo o *bias*, b , permite hacer desplazamientos a la función de activación, que de forma muy similar a la ecuación de una recta $y = mx + b$ que nos indica la pendiente que presenta la recta y el valor b indica cual será su desplazamiento; esto se verá aplicado de la misma forma, consecuencia de la combinación lineal realizada entre los pesos y las entradas se crea una linea recta o hiperplano, desplazando así las salidas de las funciones de activación empleadas en el perceptrón y esto se puede escribir de la siguiente forma:

$$\hat{y} = f\left(\sum_{i=1}^n w_i \cdot x_i + b\right) \quad (3.7)$$

Modelo completo

Con lo establecido anteriormente, es posible definir una formulación completa del comportamiento que presentara nuestro modelo de perceptrón, que puede ser escrita de la siguiente forma:

$$f(x_1, x_2, \dots, x_n) = \begin{cases} 0 & \text{if } \sum w_i \cdot x_i + b < 0 \\ 1 & \text{if } \sum w_i \cdot x_i + b \geq 0 \end{cases} \quad (3.8)$$

Calculo del número de parámetros

El número de parámetros contenidos en una red neuronal se calcula sumando el número de pesos y los sesgos por capa:

$$\phi = \{w_1, \dots, w_n, b\} \quad (3.9)$$

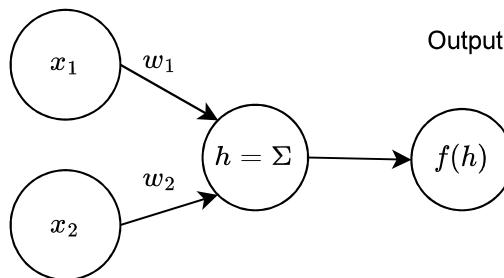
$$|\phi| = n + 1 \quad (3.10)$$

Donde ϕ será el número de parámetros calculables o modificables en el perceptrón o conjunto de perceptrones durante el entrenamiento.

3.2.2 Ejercicios

1. A partir del siguiente perceptrón de dos entradas cuya función de activación, $f(h)$, es la función escalón y el sesgo b es nulo:

Input



- (a) Suponga una entrada $X = [0.8, 0.2]$ con pesos $W = [1.0, 0.8]$. Calcule el logit (h) y la salida del perceptrón (\hat{y}).

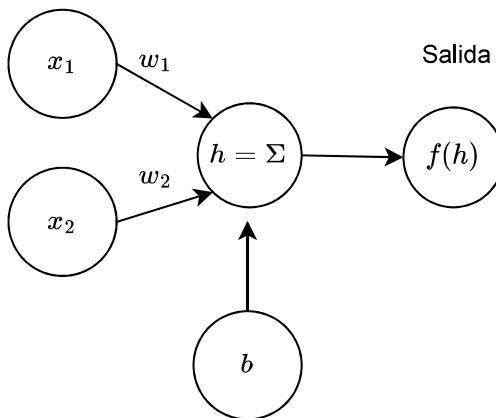
- (b) Suponga que tenemos un conjunto de datos de dos ejemplos de un determinado fenómeno. El conjunto contiene la entrada y el valor de salida de acuerdo a la siguiente tabla:

Ejemplo	x_1	x_2	y
1	-0.5	-1	0
2	-1	0.5	1

Determine por prueba y error que valores para W satisfacen lo asentado en dicha tabla.

2. Suponga el perceptrón de la siguiente figura cuya función de activación es la función escalón.

Entrada



- (a) Donde $X = [0.4, 0.8]$, $W = [0.2, 0.5]$ y $b = 0.3$. Determine la salida del perceptrón, \hat{y} .

3.2.3 Soluciones a los ejercicios

3.3 Redes neuronales simples

Una red neuronal simple es una extensión del perceptrón de Rosenblatt que utiliza una función de activación continua y derivable. La función sigmoide fue la más utilizada en los primeros años. En esta sección, revisaremos algunas funciones de activación populares.

3.3.1 Funciones de activación

Las funciones de activación se pueden agrupar de forma general en dos conjuntos denominados comúnmente como:

- funciones de activación discretas: Estas acotan los valores de salida de las neuronas a un conjunto finito de valores, generalmente usan valores binarios (0 o 1).
- funciones de activación continuas: De la misma forma, con estas es posible acotar los valores de salida y estos pueden tomar cualquier valor dentro del intervalo, generalmente los intervalos son $[0, 1]$ o $[-1, 1]$. Dentro de este conjunto se pueden utilizar distintos tipos de funciones ya sean lineales¹ o no lineales².

Función lineal

Una de las funciones más sencillas a la que se proporciona un valor de entrada (x) y a su salida (y) produce un valor proporcional a la entrada, ver figura 3.3. La función se escribe de la siguiente forma:

¹Funciones lineales en pytorch

²Funciones no lineales en pytorch

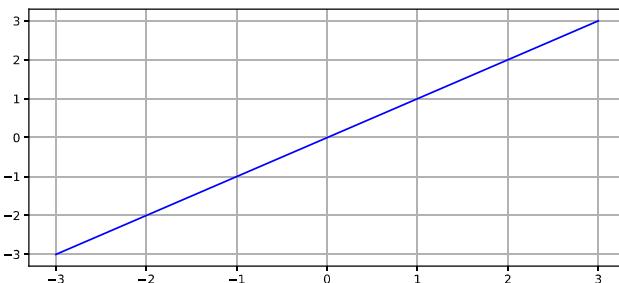


Figure 3.3: Función lineal

$$y = f_{lin}(x) = x \quad (3.11)$$

Una de las características más relevantes de estas funciones es que al colocar varias neuronas lineales en serie la salida siempre será un valor lineal. Desde una perspectiva más formal, las neuronas lineales funcionan como buenos clasificadores en datos linealmente separables.

Función escalón

La función escalón unitario o Heaviside, es una función matemática que a su salida (y) devuelve un valor constante de 1 si la entrada (x) es mayor o igual que 0 y un valor constante de 0 si es menor que cero, ver figura 3.4. La función se define como:

$$y = f_{sig}(x) = \begin{cases} 0 & \text{if } x < 0 \\ 1 & \text{if } x \geq 0 \end{cases} \quad (3.12)$$

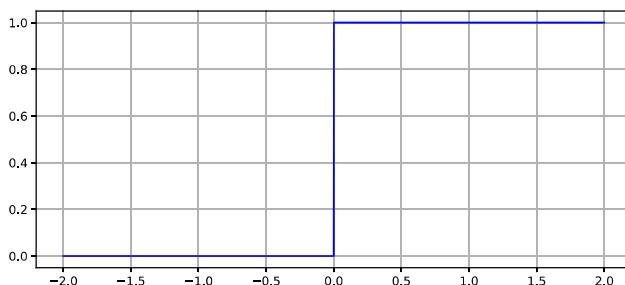


Figure 3.4: Función escalón unitario o Heaviside

Esta función puede ser vista como un ‘todo o nada’, es decir, una salida binaria que representa la presencia o ausencia de un evento.

Función sigmoidal

Comúnmente, el uso de funciones de activación no lineales y derivables es necesario para buscar soluciones a problemas más complejos que no pueden ser resueltos haciendo uso de funciones lineales. La función sigmoidal se utiliza a menudo como una función de activación en las redes neuronales para introducir no linealidad en la red y esto a su vez permite que las neuronas de la red sean activadas de manera no lineal en función de la entrada, lo que les concede la capacidad de aprender patrones complejos en los datos de entrada. El hecho de que esta y otras funciones de

activación sean derivables toma relevancia ya que algunas estrategias en el entrenamiento emplean algoritmos basados en la optimización de gradientes.

$$y = f_{\text{sigmoid}}(x) = \frac{1}{1 + e^{-x}} \quad (3.13)$$

Uno de los inconvenientes que puede presentar esta función es que tiende saturarse en sus extremos, lo que puede dificultar el entrenamiento de las redes.

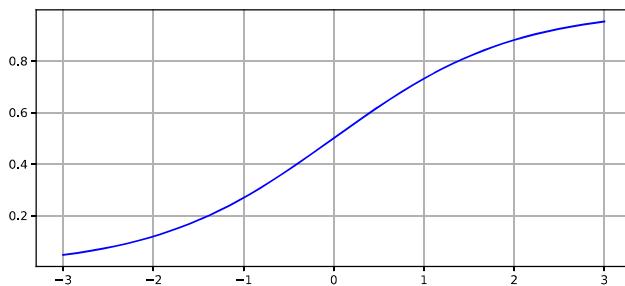


Figure 3.5: Función sigmoide

Función tangente hiperbólica

La función tangente hiperbólica, también conocida como "tanh", es una función matemática no lineal que toma una entrada (x) y produce una salida (y) en el rango de -1 a 1 (ver figura 3.6). La función tanh se define como:

$$y = f_{\text{tanh}}(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \quad (3.14)$$

A diferencia de la función sigmoide, la función tanh es simétrica alrededor del origen, lo que significa que tiene una salida negativa para entradas negativas y una salida positiva para entradas positivas. Esto puede ser útil en algunos casos en los que se desea una salida simétrica en torno a cero.

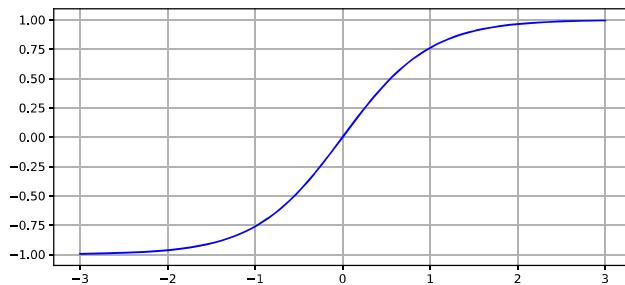


Figure 3.6: Función tangente hiperbólica

Función activación lineal rectificada (ReLU)

La función de activación activación lineal rectificada (ReLU, por sus siglas en inglés *Rectified Linear Unit*) es una función no lineal, que toma una entrada (x) y produce una salida (y) de valor cero si la entrada es negativa, y si la entrada es positiva, devuelve la entrada misma, ver figura 3.7. La función ReLU se define como:

$$y = f_{relu}(x) = \begin{cases} 0 & \text{if } x < 0 \\ x & \text{if } x \geq 0 \end{cases} \quad (3.15)$$

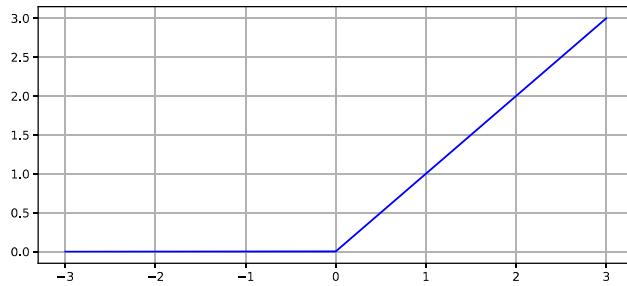


Figure 3.7: Funcion ReLU

Existen formas alternativas de definir la función ReLU:

$$y = f_{ReLU}(x) = x^+ = \max(0, x) \quad (3.16)$$

Una de las ventajas es que al no saturarse en sus extremos como la función sigmoide, permitiendo que fluya mayor cantidad de información a través de las neuronas; sin embargo, para los valores negativos esta función no permite que las neuronas se activen, que puede verse como una perdida de información.

Resumen

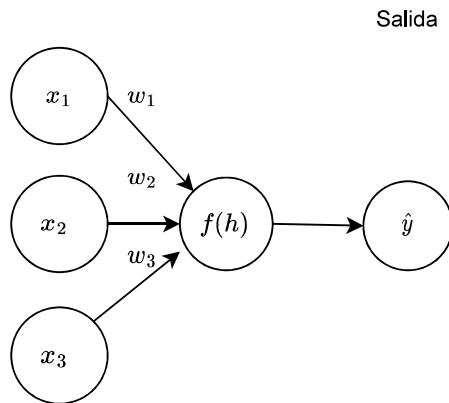
En la tabla se resumen algunas funciones de activación comúnmente empleadas para la construcción de arquitecturas de redes neuronales.

Nombre	Función
Función lineal	$y = f_{lin}(x) = x$
Función escalón	$y = f_{sig}(x) = \begin{cases} 0 & \text{if } x < 0 \\ 1 & \text{if } x \geq 0 \end{cases}$
Función sigmoide	$y = f_{sigmoid}(x) = \frac{1}{1+e^{-x}}$
Función tangente hiperbólica	$y = f_{tanh}(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$
Función <i>Rectified Linear Unit</i> (ReLU)	$y = f_{relu}(x) = \begin{cases} 0 & \text{if } x < 0 \\ x & \text{if } x \geq 0 \end{cases} = \max(0, x)$

Table 3.1: Funciones de activación

3.3.2 Ejercicios

1. Usando la red del siguiente diagrama:



donde

$$W = [1.0, 0.9, 0.4], X = [0.4, 0.6, 0.8]^T, b = 0.1$$

y la función de activación es una sigmoide. Responda las siguientes preguntas.

- (a) Calcule la salida de la red, \hat{y} .
- (b) Cambie la función de activación por una función tangente hiperbólica y calcule de nuevo la salida.
- (c) Cambie las entradas al siguiente vector y aplique la función sigmoide.

$$X = \begin{bmatrix} 0.4 \\ 0.6 \\ 0.4 \end{bmatrix}$$

3.4 Redes neuronales de varias salidas

Una red neuronal con varias salidas contiene dos o más perceptrones en una sola capa y estos a su vez tienen su propia salida, estas arquitecturas de redes se limitan a recibir las señales de entrada y redistribuyéndolas a la salida, presentando así múltiples salidas en la red neuronal.

Generalmente pueden verse como 3 capas, en donde la primera capa es la correspondiente al vector de entradas x , una capa oculta que contiene a los perceptrones que integran la red y finalmente, la capa de salida que será el vector resultante de las operaciones realizadas en la capa oculta. Esto podría interpretarse como una red multicapa; sin embargo, la red aquí descrita solo contiene una capa de perceptrones.

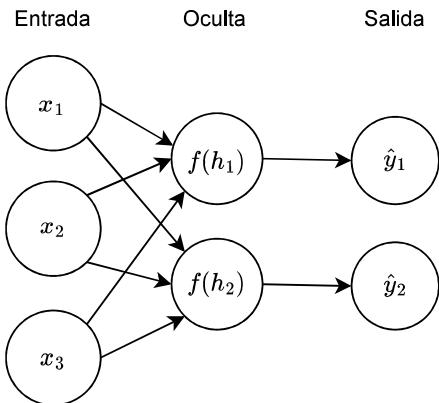
El cálculo de la salida de estos perceptrones es muy similar al realizado para una única salida y lo podemos escribir como:

$$\hat{y}_j = f \left(\sum_i w_{i,j} x_i + b \right), \quad (3.17)$$

donde \hat{y}_j es la salida para la neurona j de interés.

3.4.1 Ejercicios

1. Tomando en cuenta el siguiente diagrama de red neuronal simple de múltiples salidas sin sesgos:



, donde

$$X = [0, 4, 1],$$

$$W = \begin{bmatrix} 0.5 & 0.6 \\ 0.9 & 0.5 \\ 0.3 & 0.9 \end{bmatrix},$$

$$\eta = 15,$$

$$y = [5, 5],$$

funciones de activación *sigmoideas* y error cuadrático medio como pérdida calcule:

- (a) el vector de predicciones, \hat{Y}
- (b) el vector de términos de error, δ
- (c) la matriz de incrementos, ΔW
- (d) actualice los pesos.

3.5 Redes neuronales multicapa

Las redes neuronales multicapa, de forma general presentan una arquitectura similar a la de las redes con varias salidas, capa de entrada, oculta y salida, pero estas cuentan con 2 o más capas ocultas donde se realiza el procesamiento de la información alimentada a la red.

3.5.1 Pesos

Como se presentó anteriormente, los pesos pueden estar contenidos en un vector; ahora en los perceptrones multicapa existen múltiples conexiones sinápticas entre las distintas capas de la red, mismos que se pueden contener en un matriz y estos se etiquetan de la siguiente forma: $w_{i,j}$ donde i representa la unidad de entrada y j denota el perceprón al que esta asociada esa conexión³.

$$h_j = \begin{bmatrix} w_{1,1} & w_{1,2} \\ w_{2,1} & w_{2,2} \\ w_{3,1} & w_{3,2} \end{bmatrix} \quad (3.18)$$

³El tamaño de la matriz de pesos es dado por el numero de entradas i y el numero de neuronas siguientes a las que se conecta j , así la matriz de pesos tendrá dimensiones $i \times j$

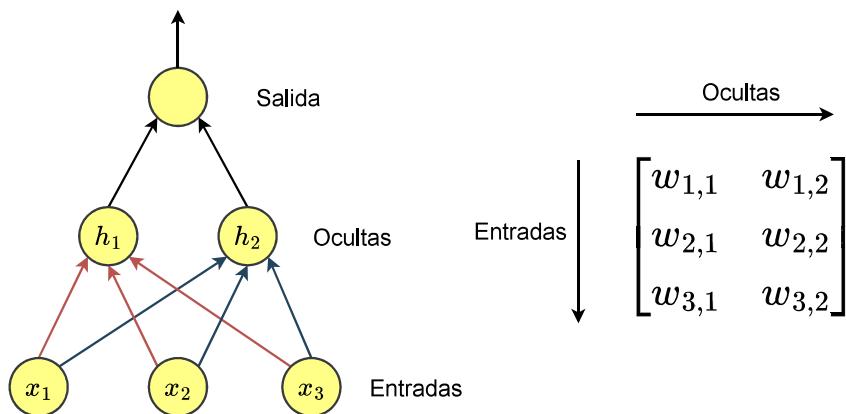


Figure 3.8: Red neuronal multicapa (izquierda) y matriz de pesos (derecha)

3.5.2 Capas ocultas

Las capas ocultas como se muestra en la figura 3.8, son aquellas que están contenidas entre las capas de entrada y salida en una red neuronal.

La entrada a un perceptrón para una capa oculta j es definida por la siguiente ecuación:

$$h_j = \sum_i w_{i,j} x_i \quad (3.19)$$

donde h_j representa el valor que recibirá en la entrada la neurona estudiada, que será el resultado de la suma de las salidas generadas por las neuronas de la capa anterior (i) multiplicadas por los pesos ($w_{i,j}$).

O bien, reescribiendo los pesos de una capa oculta de forma matricial como en la ecuación 3.18, se tiene que para la capa j los valores de entrada en h_j serán:

$$h_j = \begin{bmatrix} w_{1,j} \\ w_{2,j} \\ w_{3,j} \end{bmatrix} \begin{bmatrix} x_1 & x_2 & x_3 \end{bmatrix} \quad (3.20)$$

3.5.3 Salida

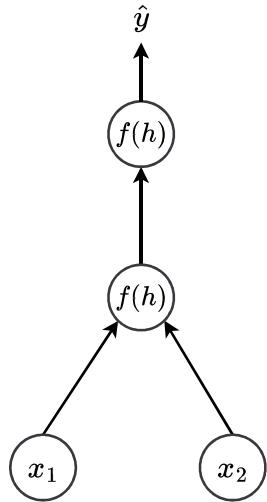
La salida de la red neuronal multicapa, es calculada mediante un proceso llamado propagación hacia adelante o *feedforward*, una vez alimentada una señal de entrada se evalúan capa a capa las salidas de los perceptrones hasta llegar a la capa de salida y esta se puede escribir de la siguiente forma:

$$o_k = f \left(\sum_j w_{jk} a_j + b_k \right) \quad (3.21)$$

donde o_k es el valor de la capa de salida k , f la función de activación, w_{jk} los pesos sinápticos existentes entre la capa oculta k y la de salida, a_j la salida calculada por los perceptrones de la capa oculta y el sesgo b .

3.5.4 Ejercicios

1. Tomando en cuenta la siguiente red neuronal multicapa:



donde:

$$X = [0.1, 0.2], W^1 = \begin{bmatrix} -0.5 \\ 0.4 \end{bmatrix}, W^2 = [0.3]$$

- (a) Calcule la salida de la capa oculta
- (b) Calcule la predicción de la red