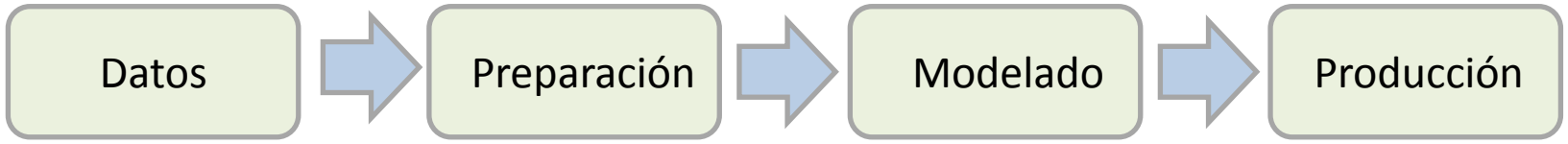
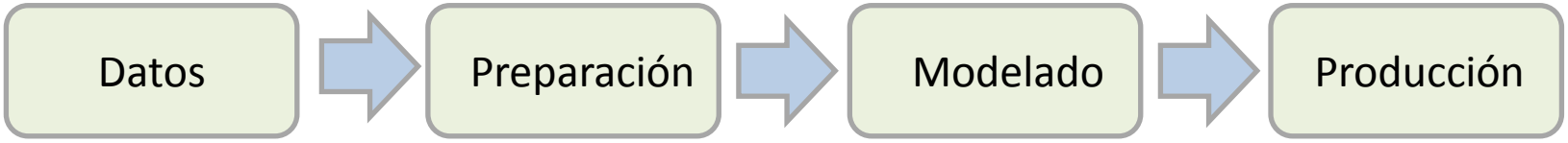
Aprendizaje Automático con Redes Neuronales

Contexto

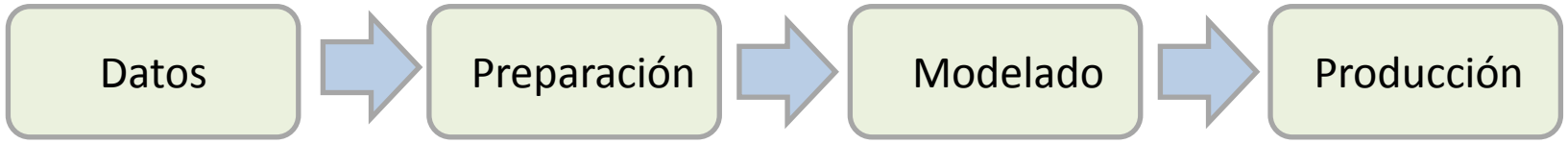


Contexto



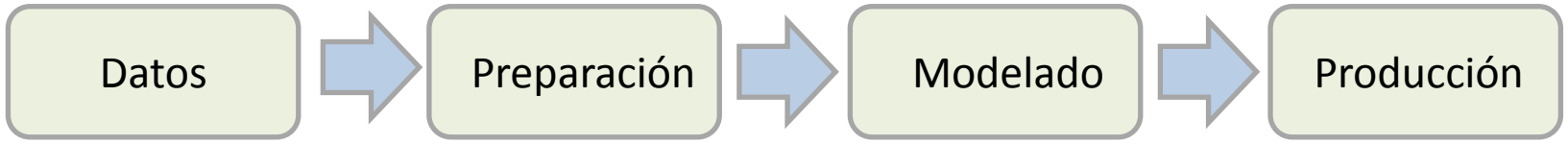
- Los datos representan una parte del mundo
- Son generados por distintos procesos
- Captura manual o automática (Sensores)
- Estructurados (bases de datos, Excel, csv)
- No Estructurados (imágenes, redes sociales, mails)

Contexto



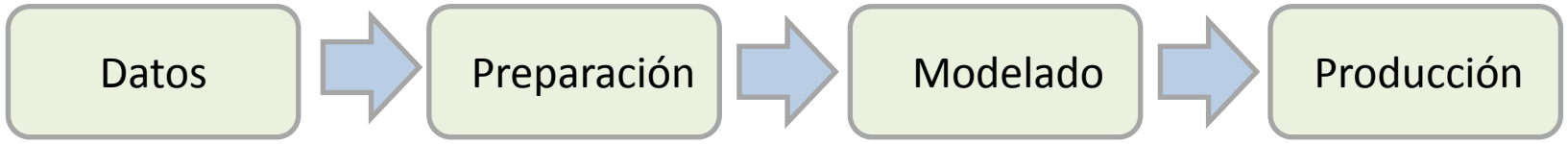
- Ajustar rangos
- Ajustar distribuciones
- Convertir categorías a números
- Convertir números en categorías
- Reducir la dimensión

Contexto



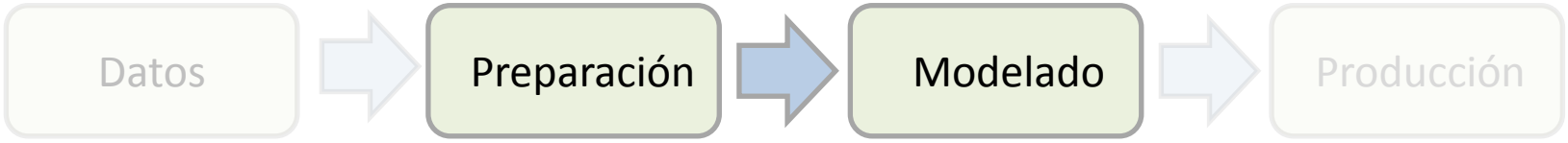
- Regresiones
- Árboles de decisión
- Redes Neuronales
- Support Vector Machine
- Clustering

Contexto



- Programa
- Descripción
- Segmentación

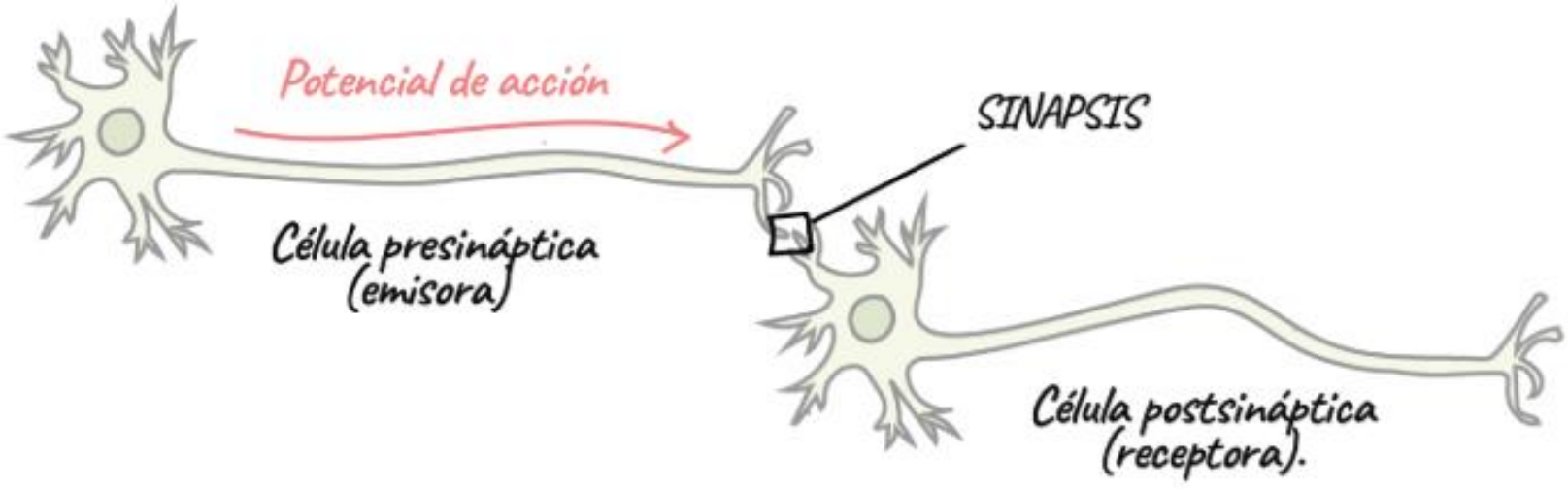
Foco del taller



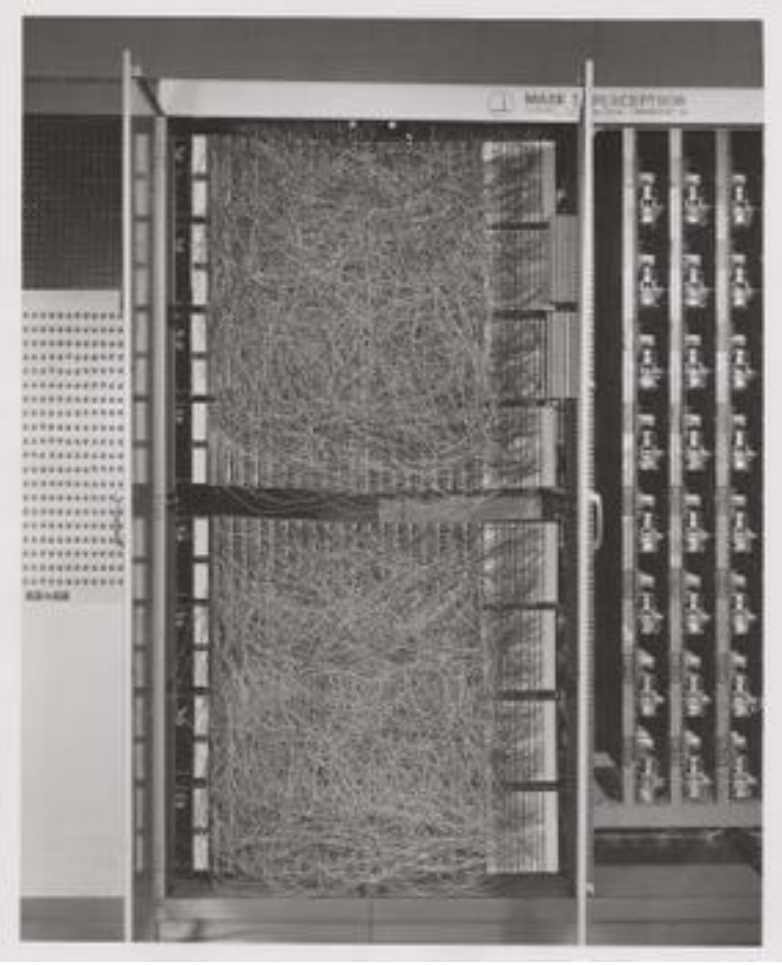
- Ajustar Rangos
- Reducir dimensión

- Aprendizaje Automático
- Supervisado
- Redes Neuronales

Un modelo muy simplificado de neuronas biológicas



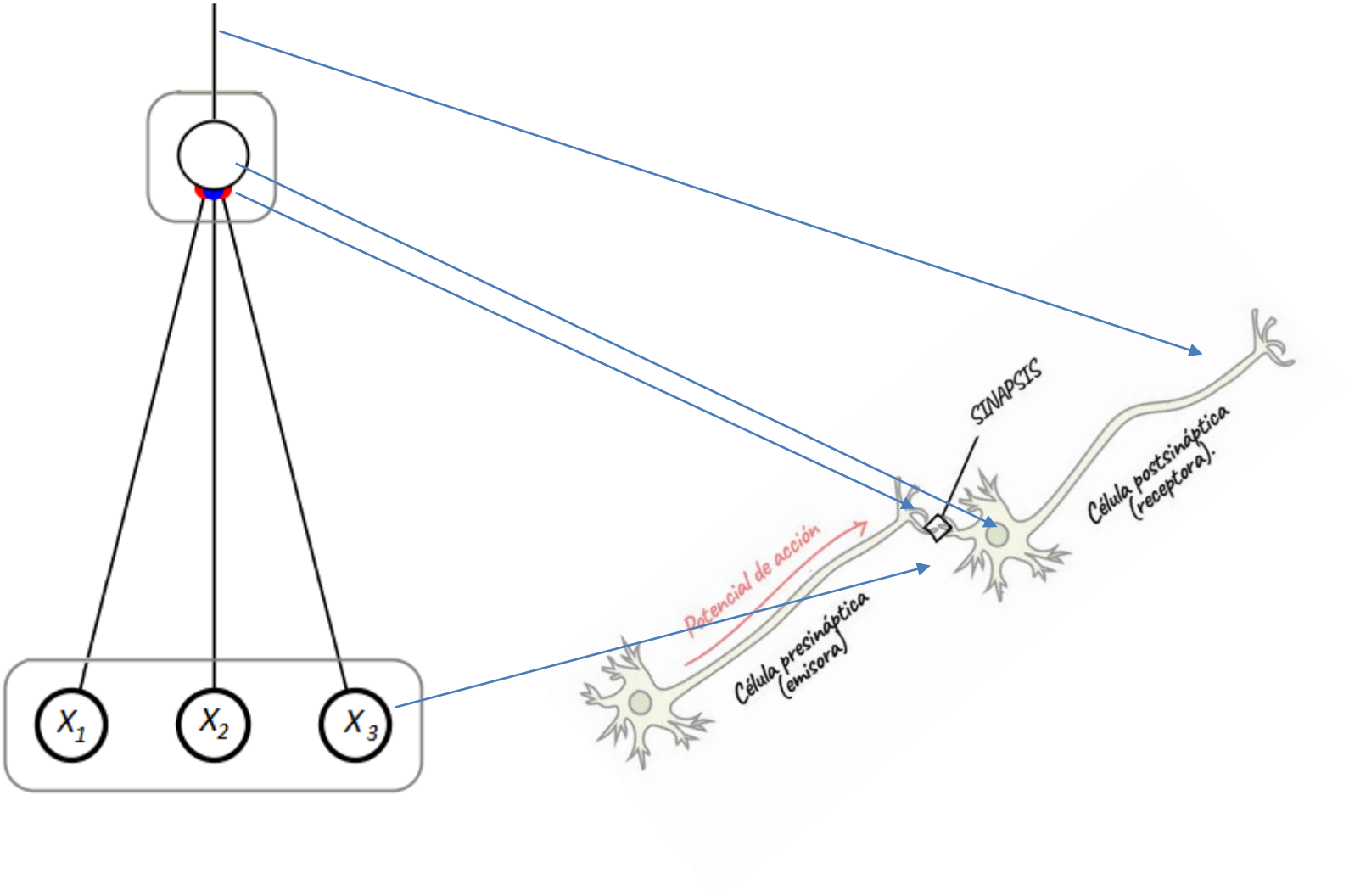
Perceptrón



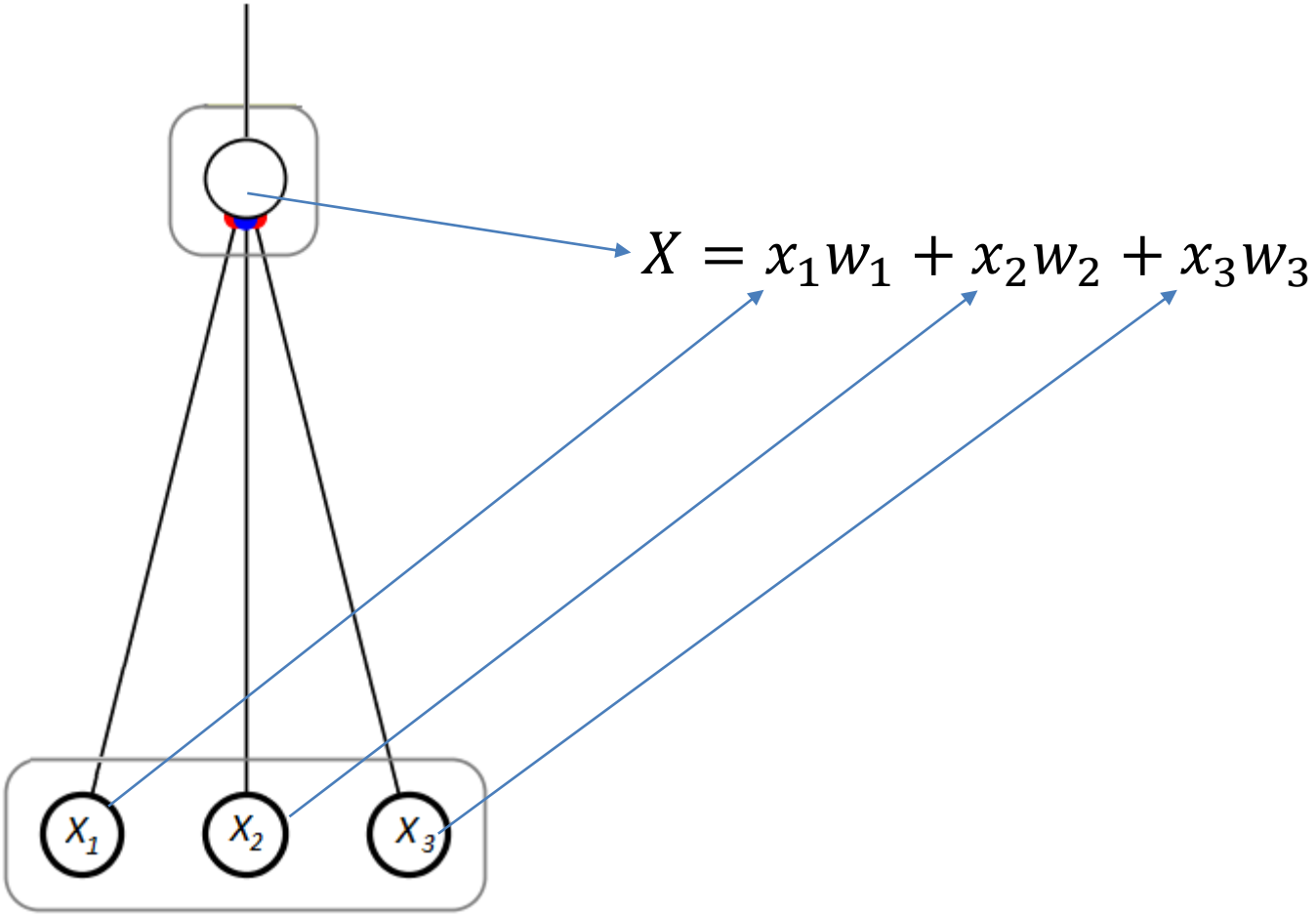
El primer Perceptrón era una máquina que tenía una cámara de 400 pixeles y se usaba para reconocimiento de imagenes

Funcionaba con motores y potenciómetros

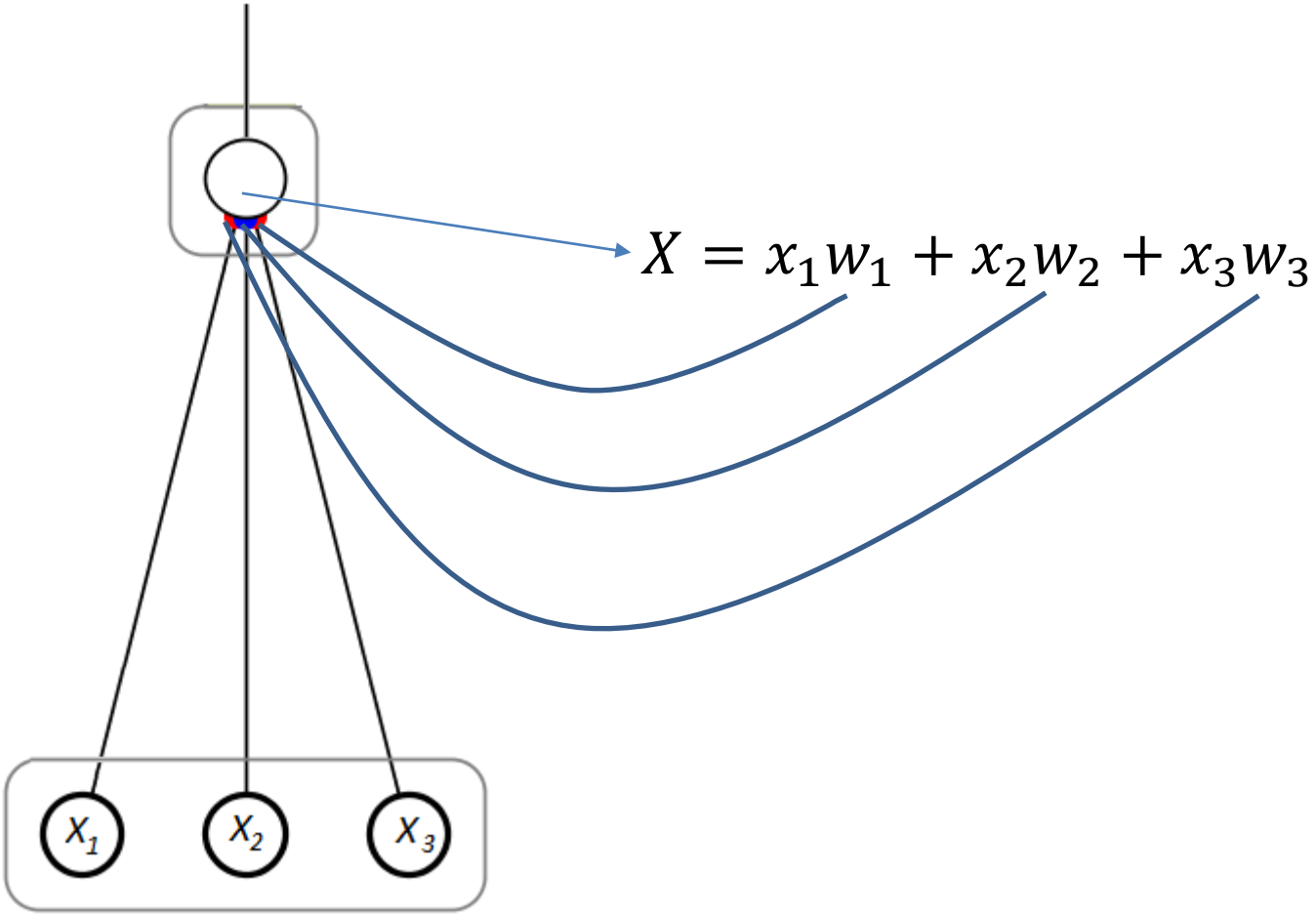
Perceptrón


X_1
X_2
X_3
Potencial de acción
Célula presináptica (emisora)
SINAPSIS
Célula postsináptica (receptora).

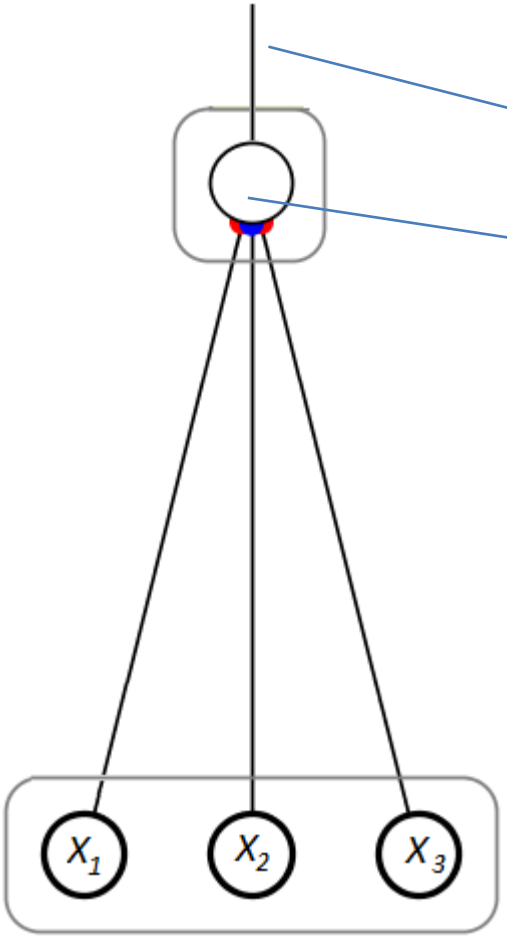
Perceptrón: información de entrada



Perceptrón: ponderación de la entrada



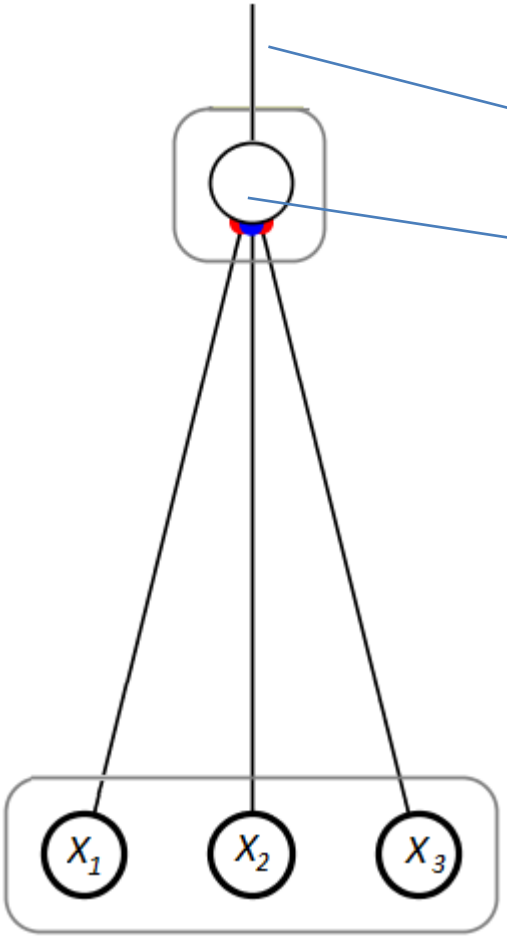
Perceptrón: procesamiento



$$Y = \emptyset(X)$$ ø es la función de activación

$$X = x_1 w_1 + x_2 w_2 + x_3 w_3$$

Perceptrón: procesamiento



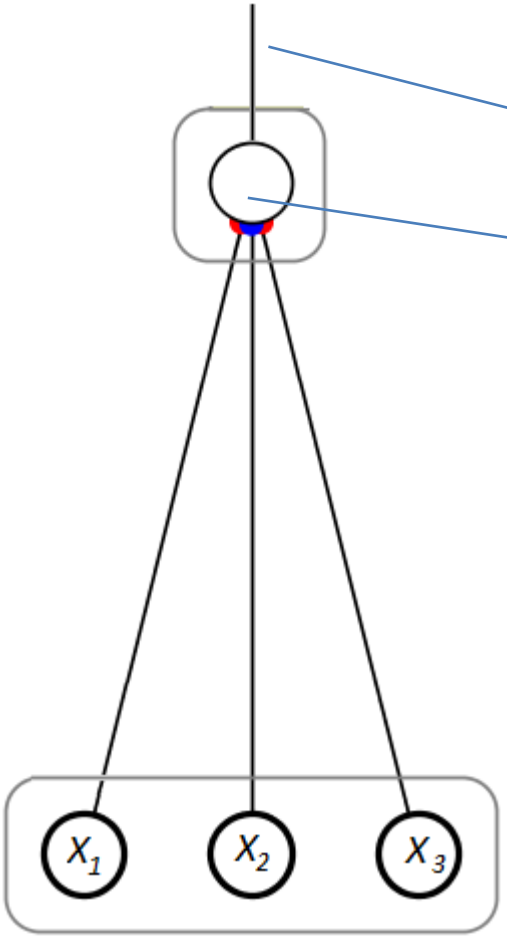
$$Y = \emptyset(X)$$

$$X = x_1 w_1 + x_2 w_2 + x_3 w_3$$

La función Ø es:

Si ***X*** es mayor a ***0*** entonces
el resultado será 1
Sino
el resultado será -1

Perceptrón: procesamiento



$$Y = \emptyset(X)$$

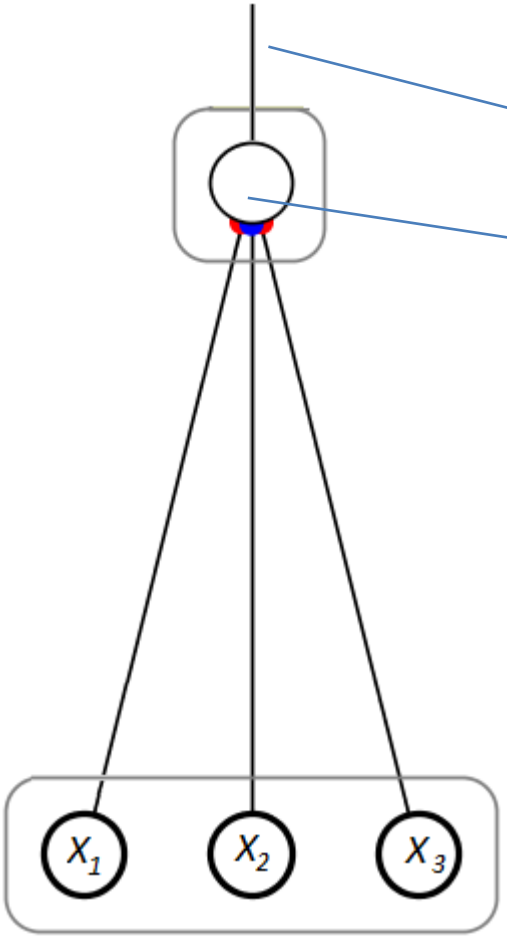
$$X = x_1 w_1 + x_2 w_2 + x_3 w_3$$

La función Ø es:

Si ***X*** es mayor a ***v*** entonces
el resultado será 1
Sino
el resultado será -1

$$\emptyset(X) = \begin{cases} 1 \; si \; X > v \\ -1 \; en \; otro \; caso \end{cases}$$

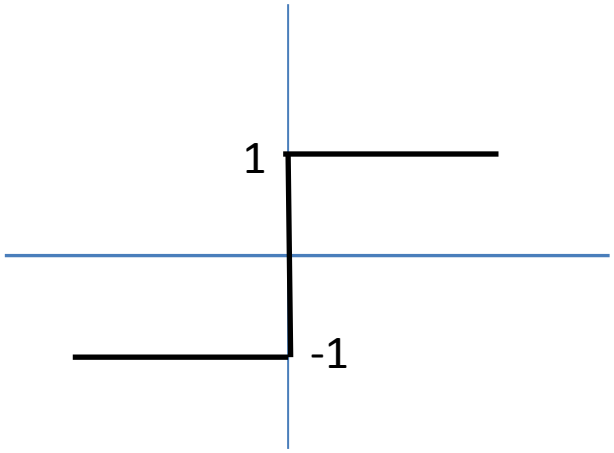
Perceptrón: procesamiento



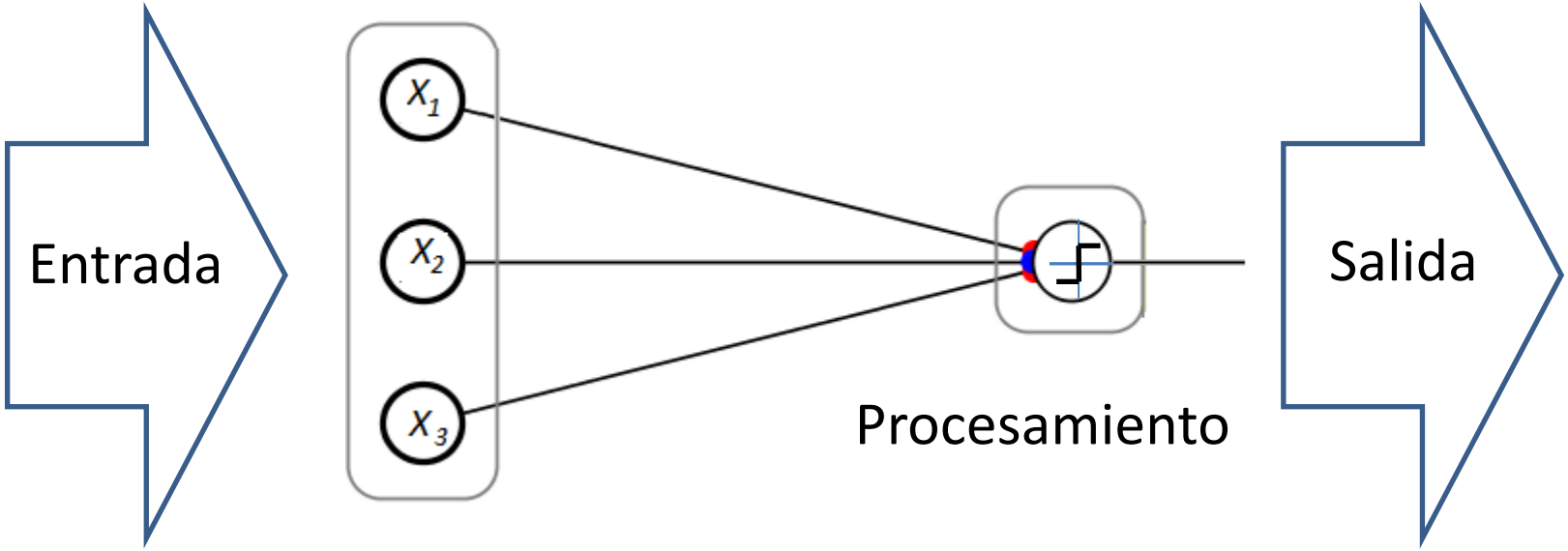
$$Y = \emptyset(X)$$

$$X = x_1 w_1 + x_2 w_2 + x_3 w_3$$

$$\emptyset(X) = \begin{cases} 1 \; si \; X > v \\ -1 \; en \; otro \; caso \end{cases}$$

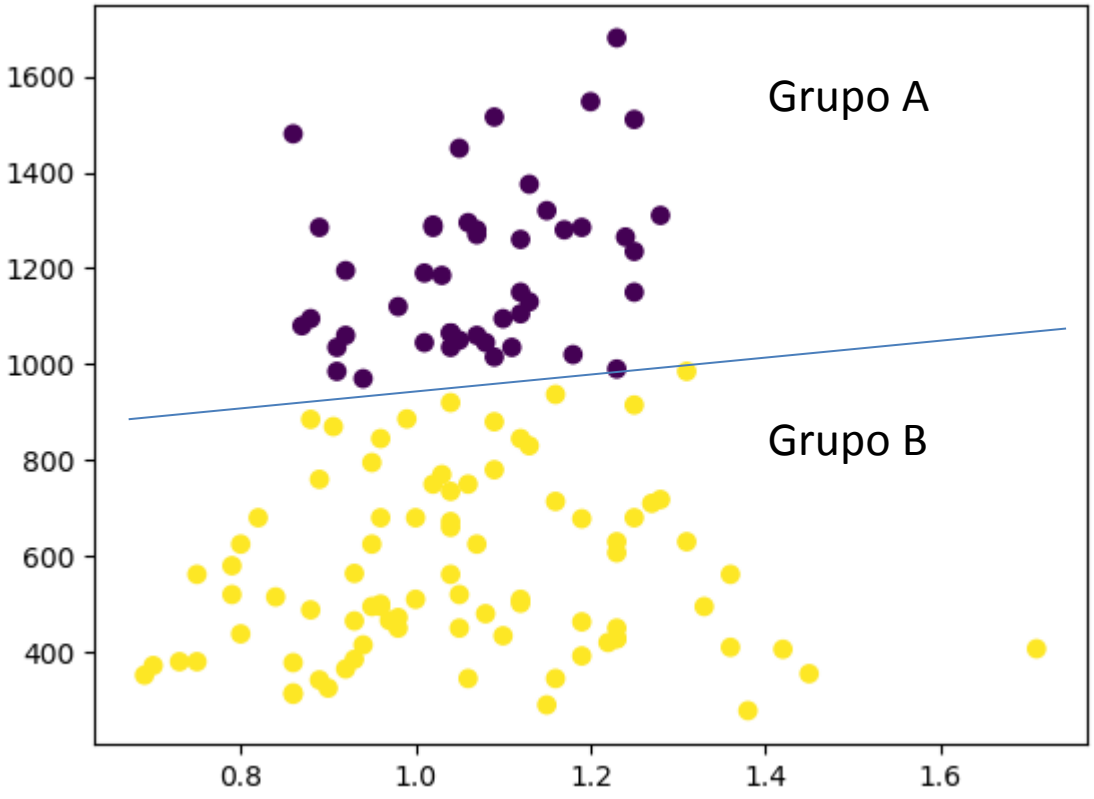


Perceptrón: Funcionamiento



Dado un conjunto de datos de **Entrada**, el Perceptrón lo **procesa** y emite una señal de **Salida**, que será -1 o 1

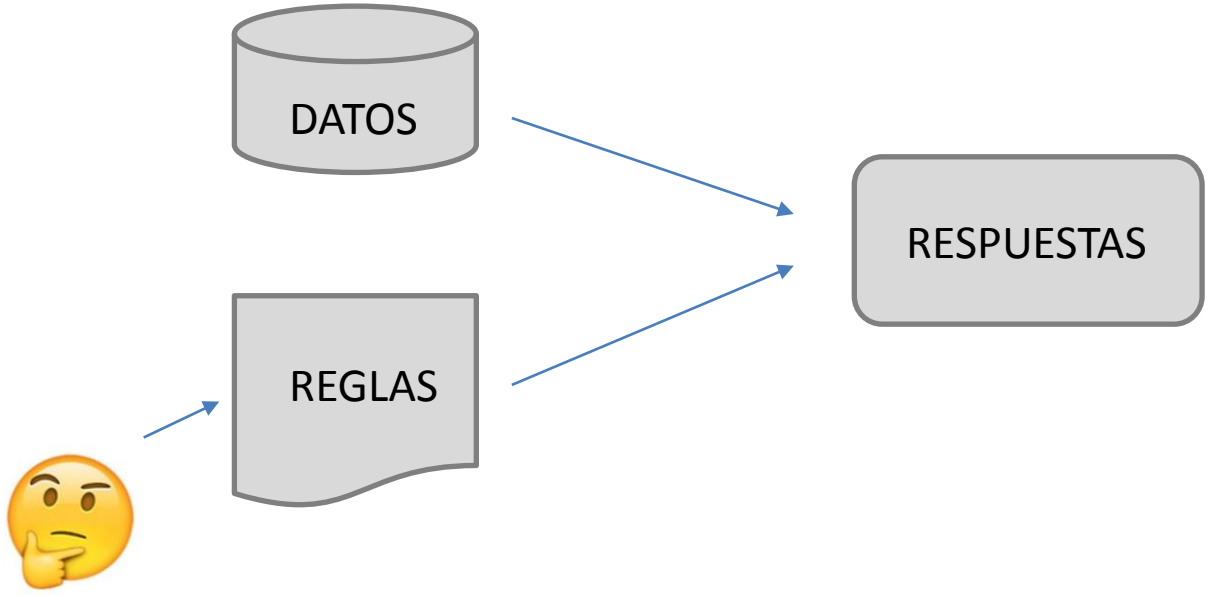
¿Qué puede hacer un Perceptrón?



Clasificar estos puntos en dos Grupos

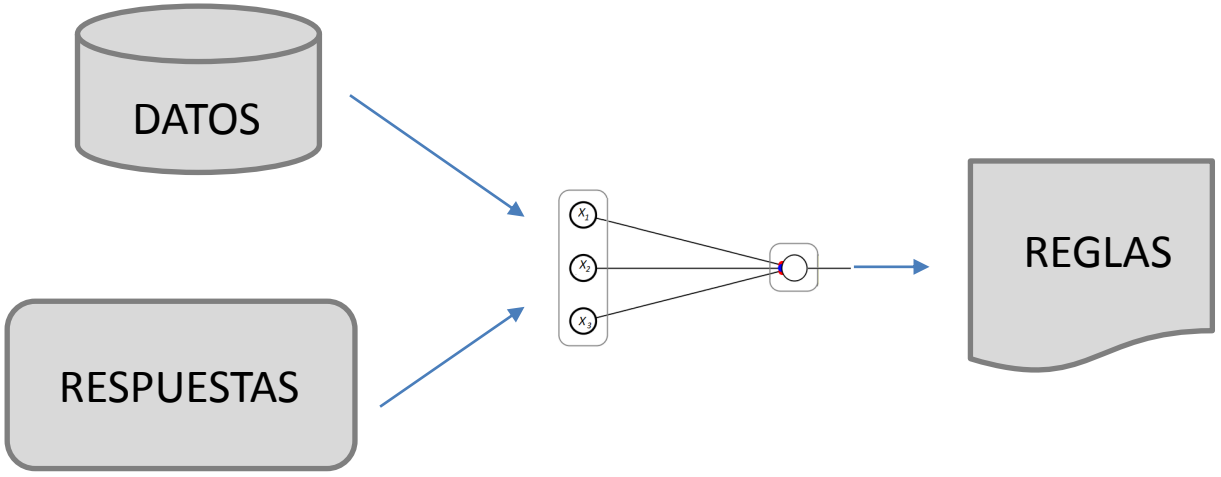
¿Cómo?

Ya que la respuesta del Perceptrón depende de los valores de los ponderadores w_i, una manera de elegir estos ponderadores, es analizar el problema y encontrar qué valores son los que producen la mejor solución.



Aprendizaje Automático

Una alternativa es utilizar un algoritmo de **aprendizaje automático** para obtener las reglas, usando los datos y las respuestas



Regla de Aprendizaje

Para que un Perceptrón aprenda hay que entrenarlo

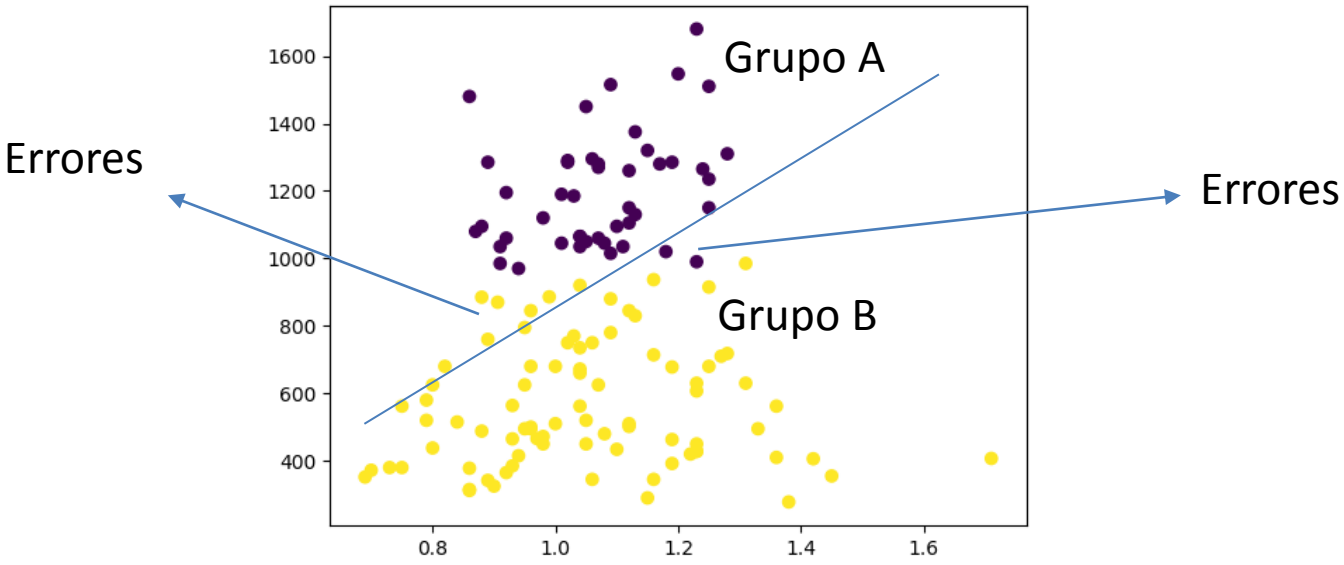
Entrenarlo significa:

1. Obtener la respuesta del perceptrón en un caso
2. Calcular el error
3. Con el error modificar los ponderadores para hacerlo más menor
4. Continuar de la misma manera para todos los casos

$$\Delta w_i = \eta\,(y_i - \hat{y}_i)\,x_{ij}$$

- Δw_i: Cuánto modifico w_i
- η: Coeficiente de Aprendizaje
- $(y_i - \hat{y}_i)$: Error
- x_{ij}: Entrada

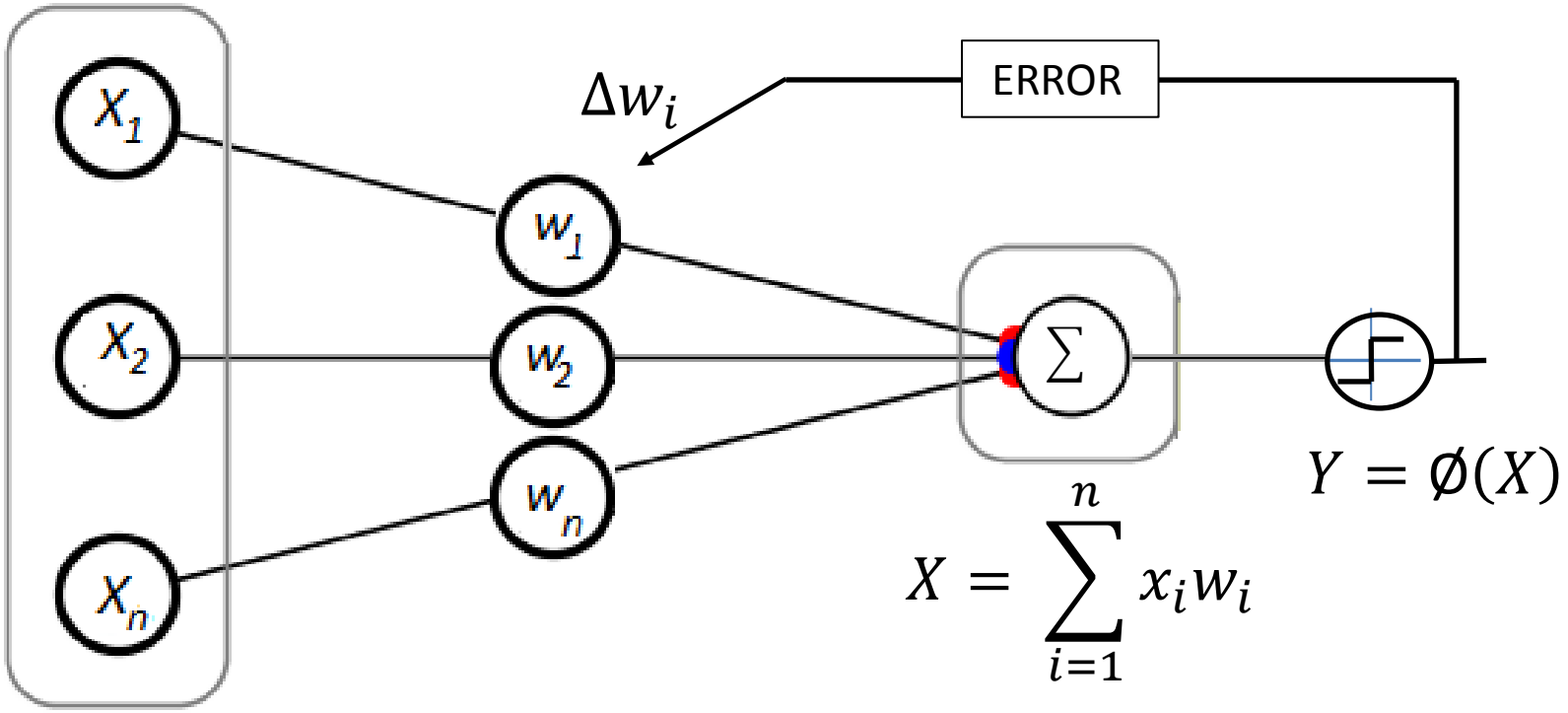
Error



El Error es la diferencia entre el valor esperado y el resultado

$$E_i = y_i - \hat{y}_i$$

Regla de Aprendizaje



$$w_i = w_i + \Delta w_i$$

$$\Delta w_i = \eta\,(y_j - \hat{y}_j)\,x_{ij}$$

Regla de Aprendizaje

$x_{ij} = 0.5 \quad y_j = -1 \quad \hat{y}_j = -1 \quad \eta = 1$

$$\Delta w_i = \eta\,(y_j - \hat{y}_j)\,x_{ij}$$

$$\Delta w_i = 1\,(-1 - -1)\,0.5$$

$$\Delta w_i = 0$$

Predicción Correcta

$x_{ij} = 0.5 \quad y_j = 1 \quad \hat{y}_j = -1 \quad \eta = 1$

$$\Delta w_i = \eta\,(y_j - \hat{y}_j)\,x_{ij}$$

$$\Delta w_i = 1\,(1 - -1)\,0.5$$

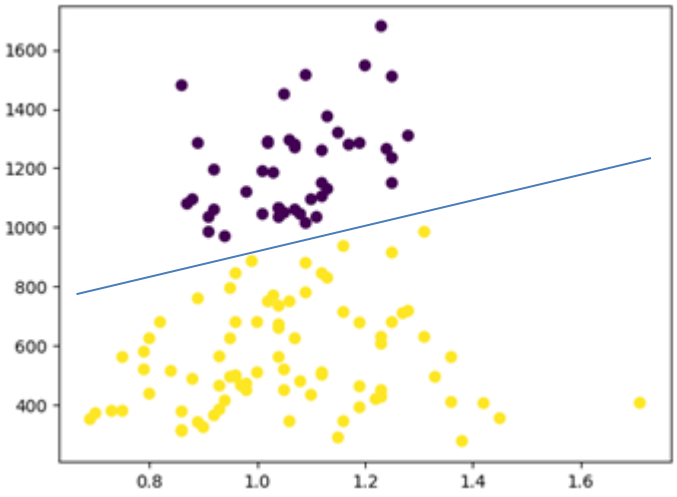
$$\Delta w_i = 1$$

Predicción Incorrecta

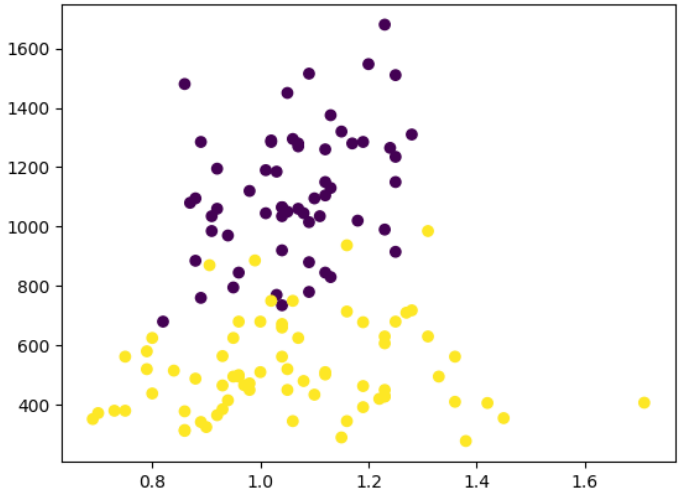
Propiedades del Perceptrón

Sólo funciona si las clases son ***linealmente separables***

Si los datos contienen clases no linealmente separables, el algoritmo **no converge nunca**

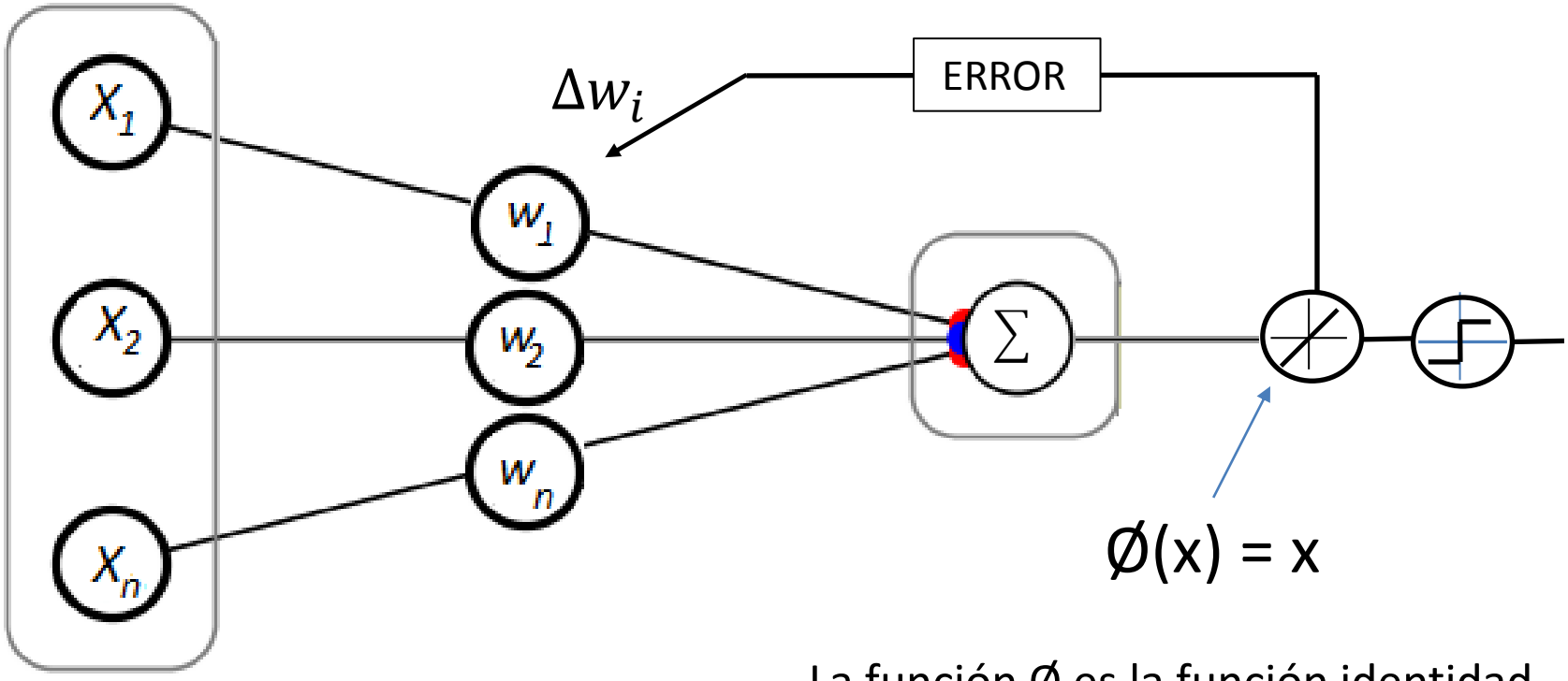


Linealmente Separables



NO Linealmente Separables

Adaline: ADAptative LINear Element



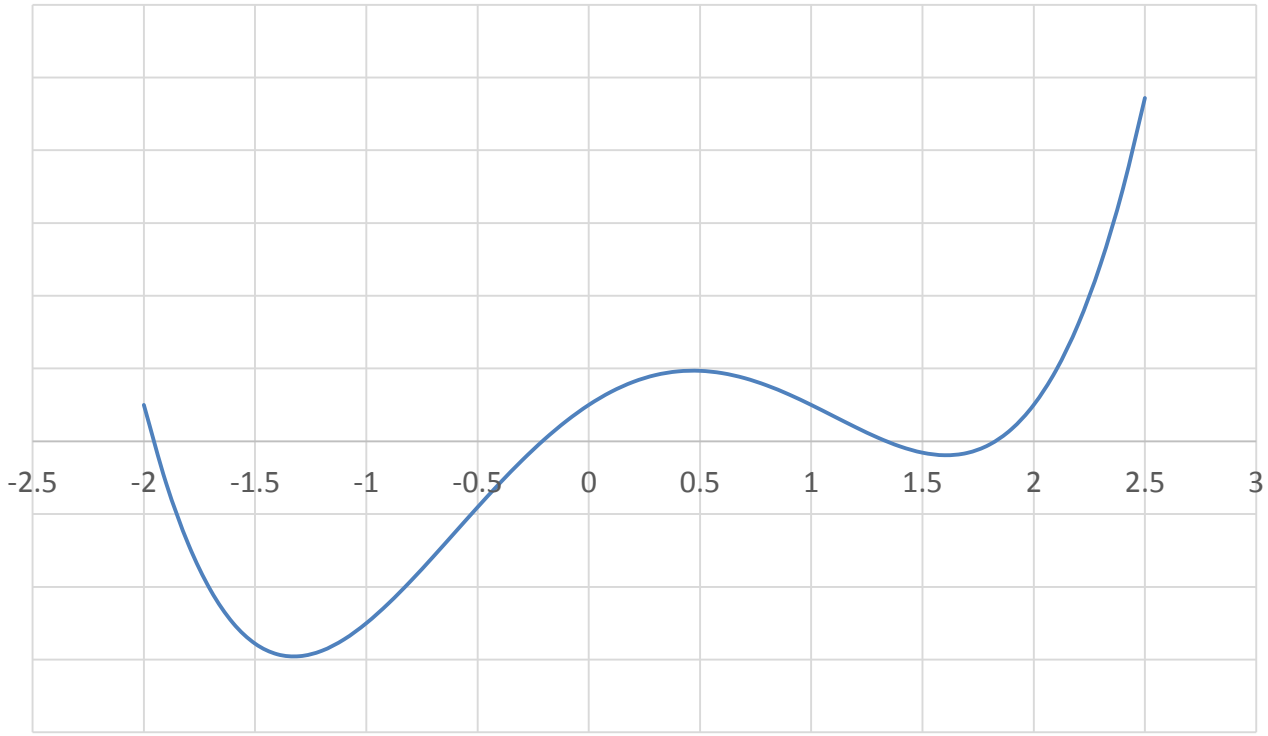
La función Ø es la función identidad

Adaline utiliza el error continuo, en contraste del perceptrón que utiliza el error entero (1 o -1)

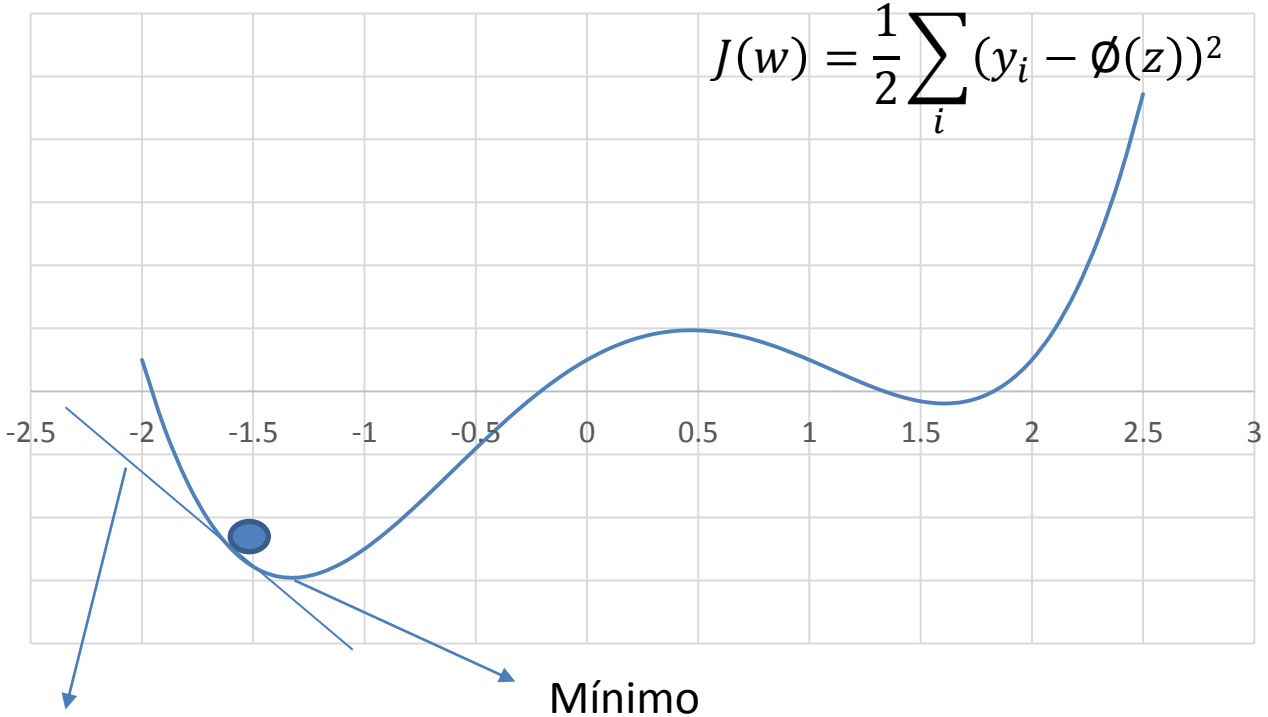
Función de Costo

$$J(w) = \frac{1}{2}\sum_{i}(y_i - \emptyset(z))^2$$

J(w) es la función a minimizar
z es la suma ponderada de las ***x***
y es la respuesta real



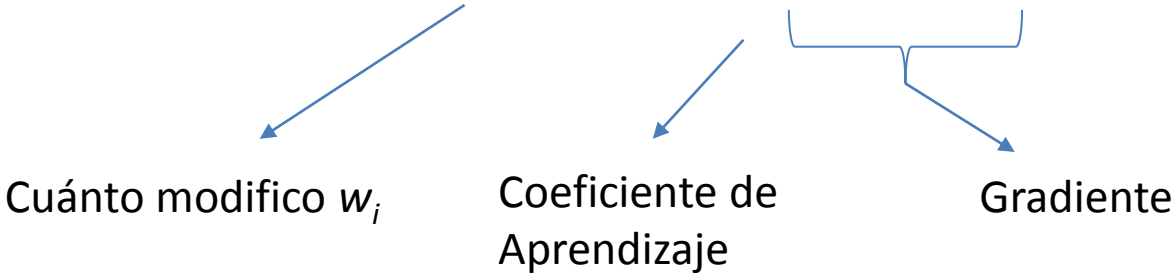
Gradiente Descendente



El Gradiente descendente es el algoritmo que se usa para encontrar el mínimo de la función de costo

Gradiente Descendente

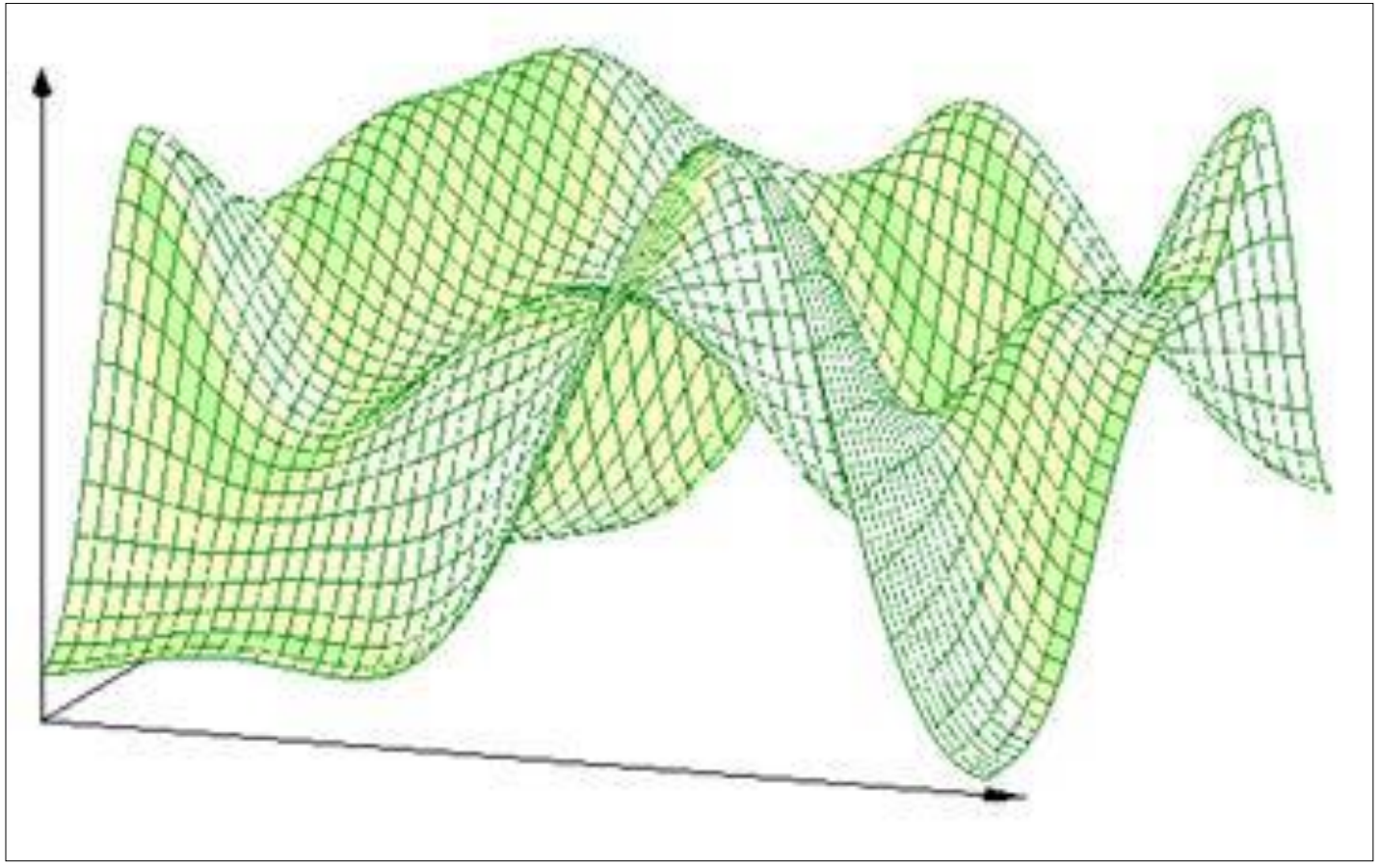
$$\Delta w = -\eta \nabla J(w)$$



$$\frac{\partial J}{\partial w_j} = -\sum_i (y_i - \emptyset(z_i))\, x_{i,j}$$

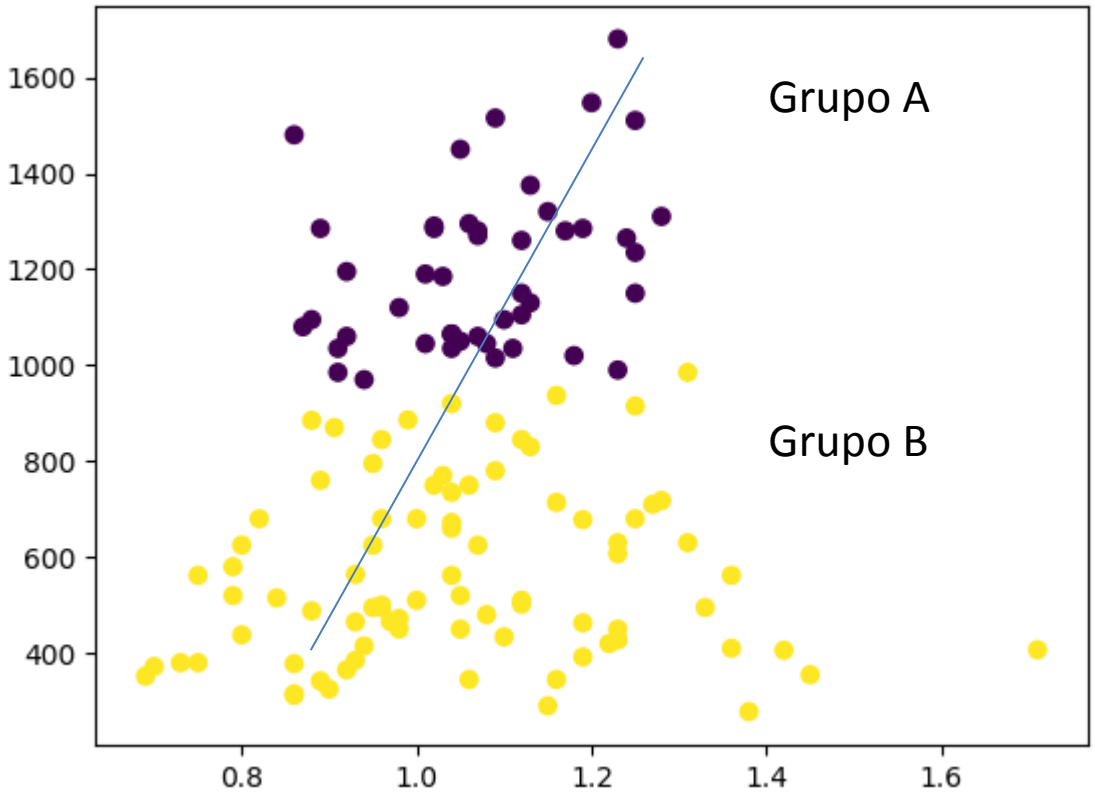
$$\Delta wj = -\eta \frac{\partial J}{\partial w_j} \quad \therefore \quad w = w + \Delta w$$

Gradiente descendente en 3D



Ajustando el modelo

Mediante el gradiente descendente, con cada nueva pasada se va ajustando el modelo

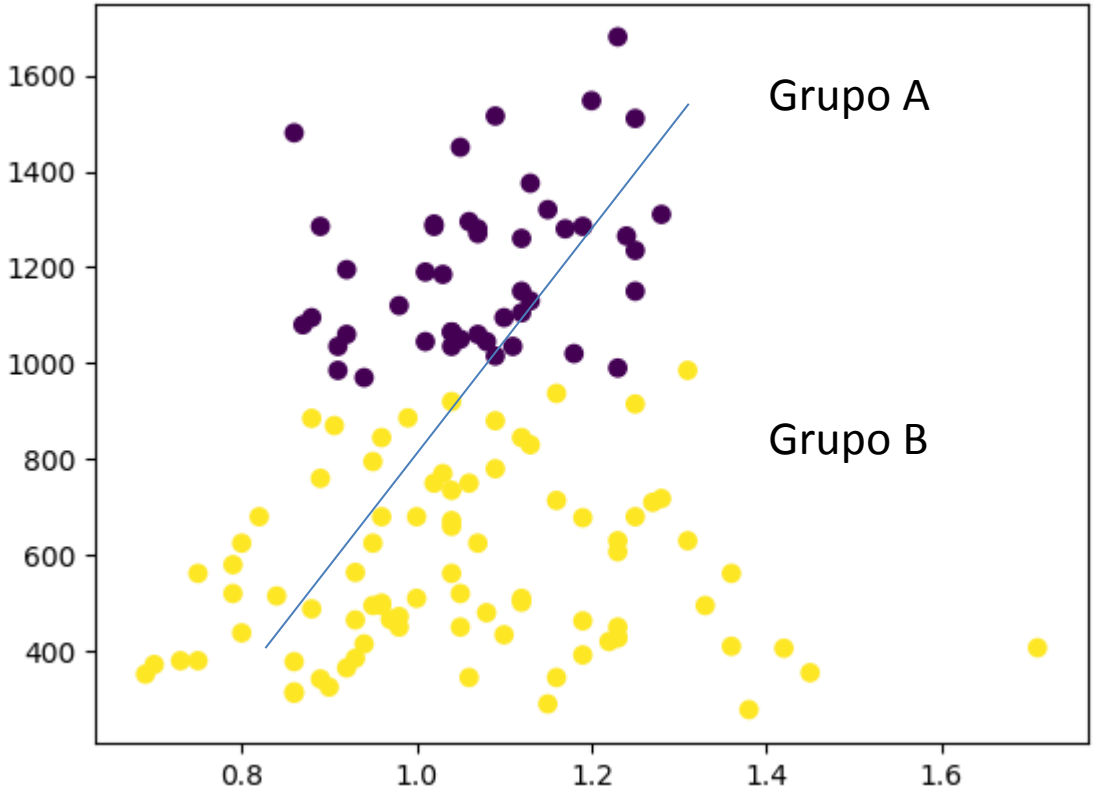


Comenzamos con valores al azar para los ponderadores w

Esto resulta en un modelo que seguramente no es muy bueno

Ajustando el modelo

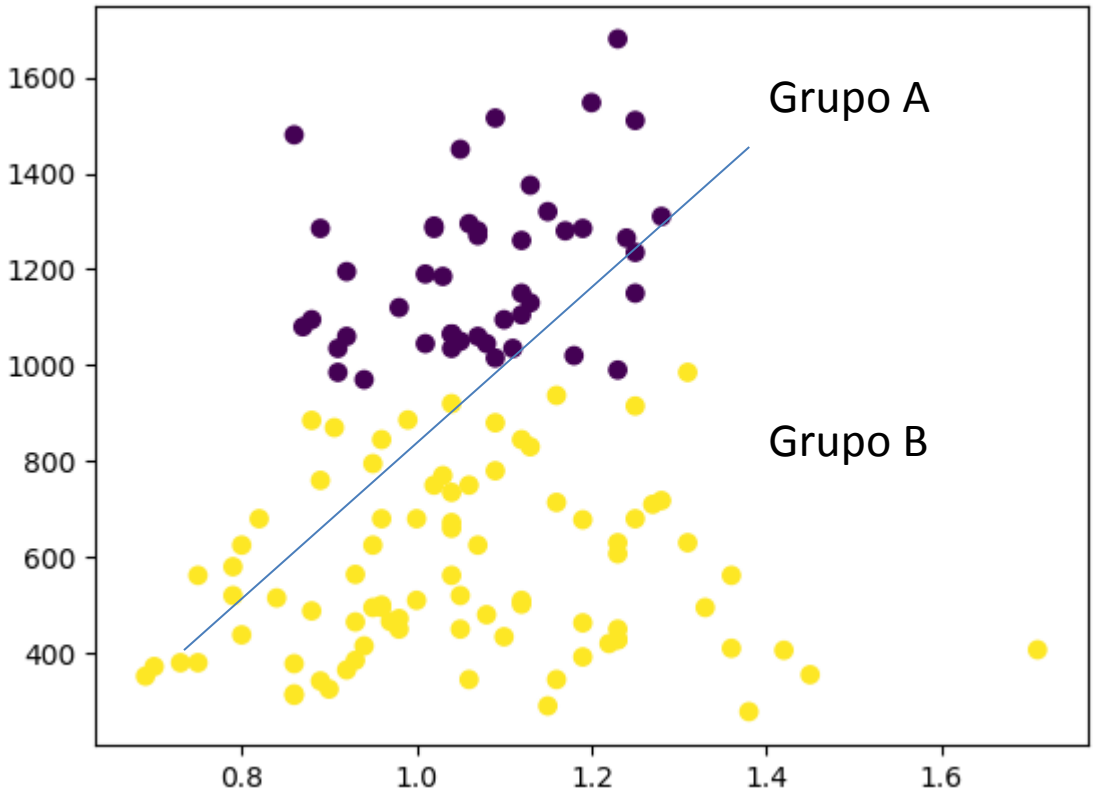
Mediante el gradiente descendente, con cada nueva pasada se va ajustando el modelo



Luego de la primera pasada, los w se ajustan para minimizar la función de costo

Ajustando el modelo

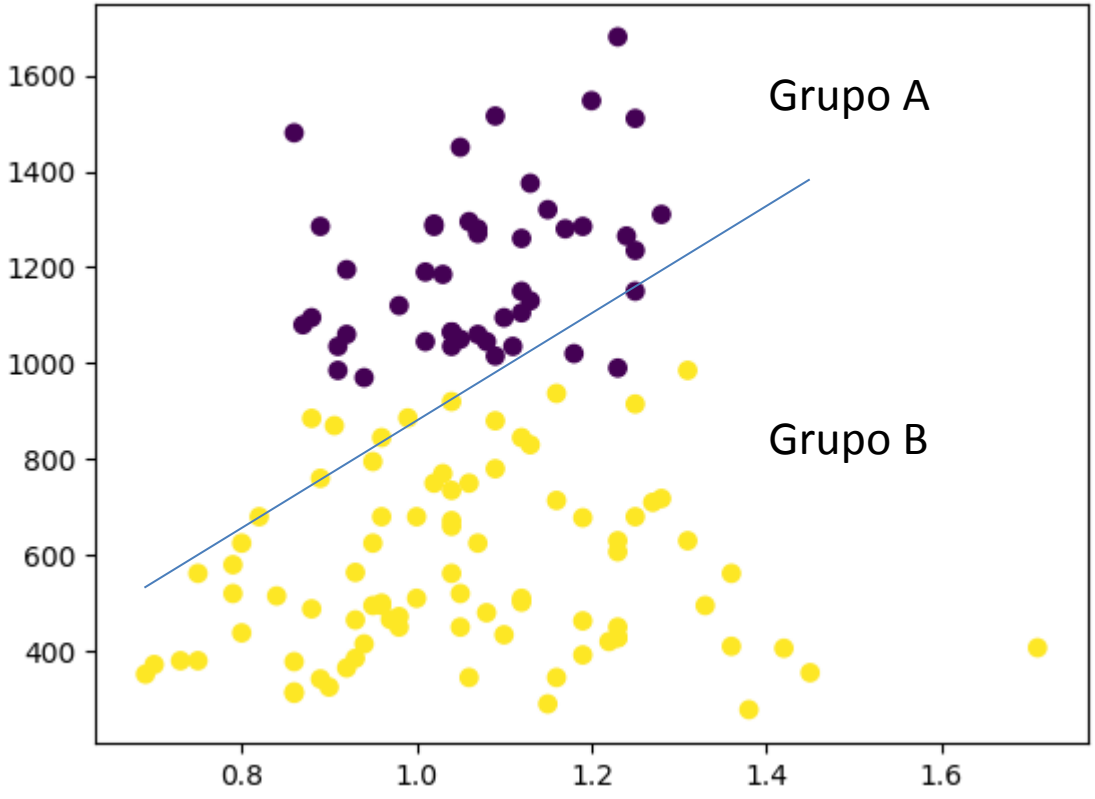
Mediante el gradiente descendente, con cada nueva pasada se va ajustando el modelo



Siguiente pasada

Ajustando el modelo

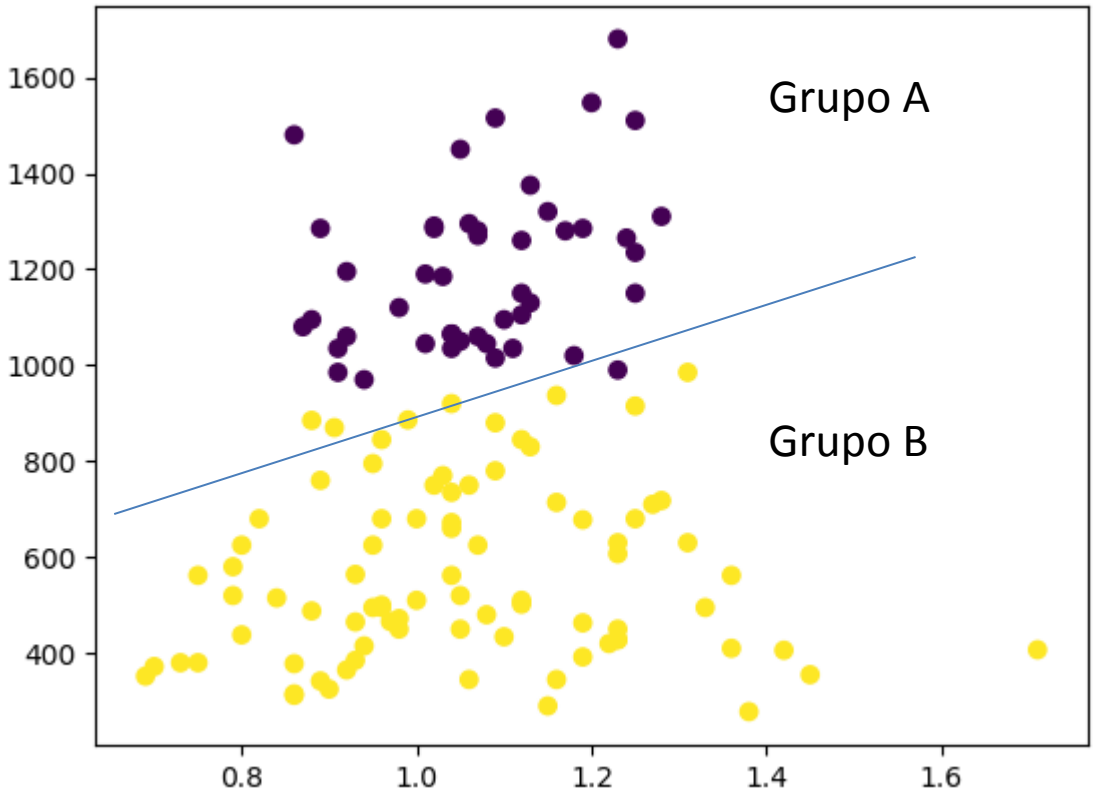
Mediante el gradiente descendente, con cada nueva pasada se va ajustando el modelo



Siguiente pasada

Ajustando el modelo

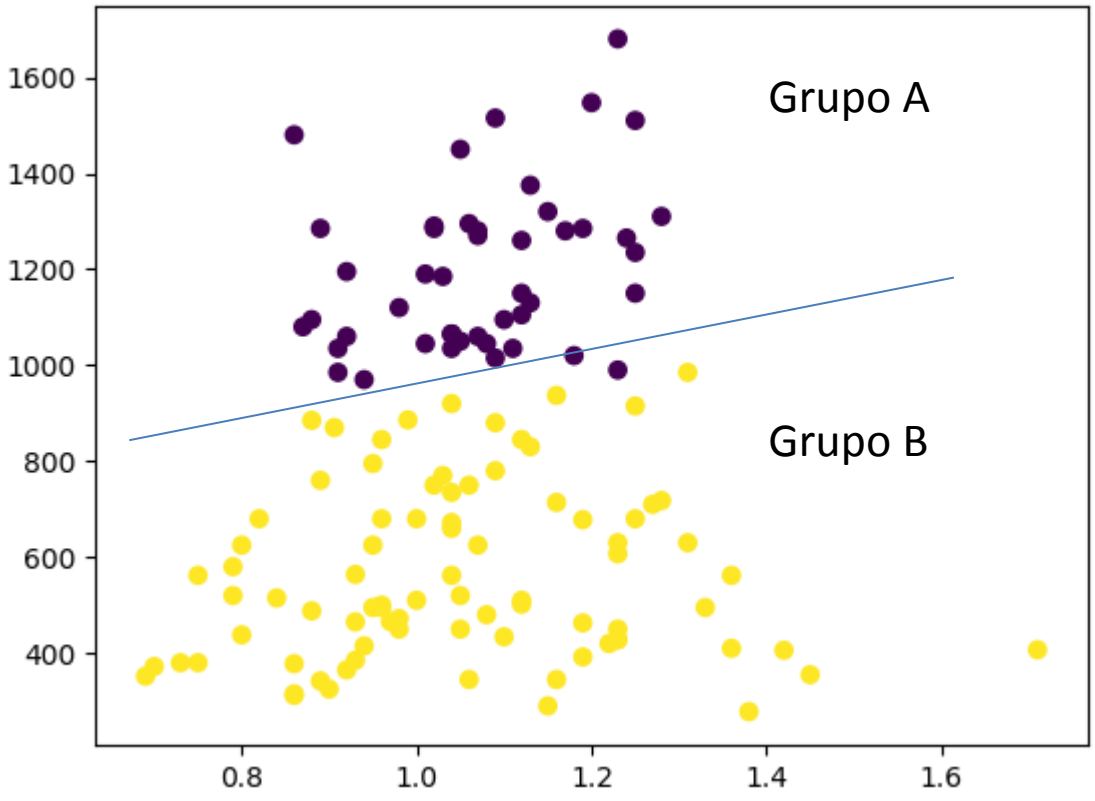
Mediante el gradiente descendente, con cada nueva pasada se va ajustando el modelo



Siguiente pasada

Ajustando el modelo

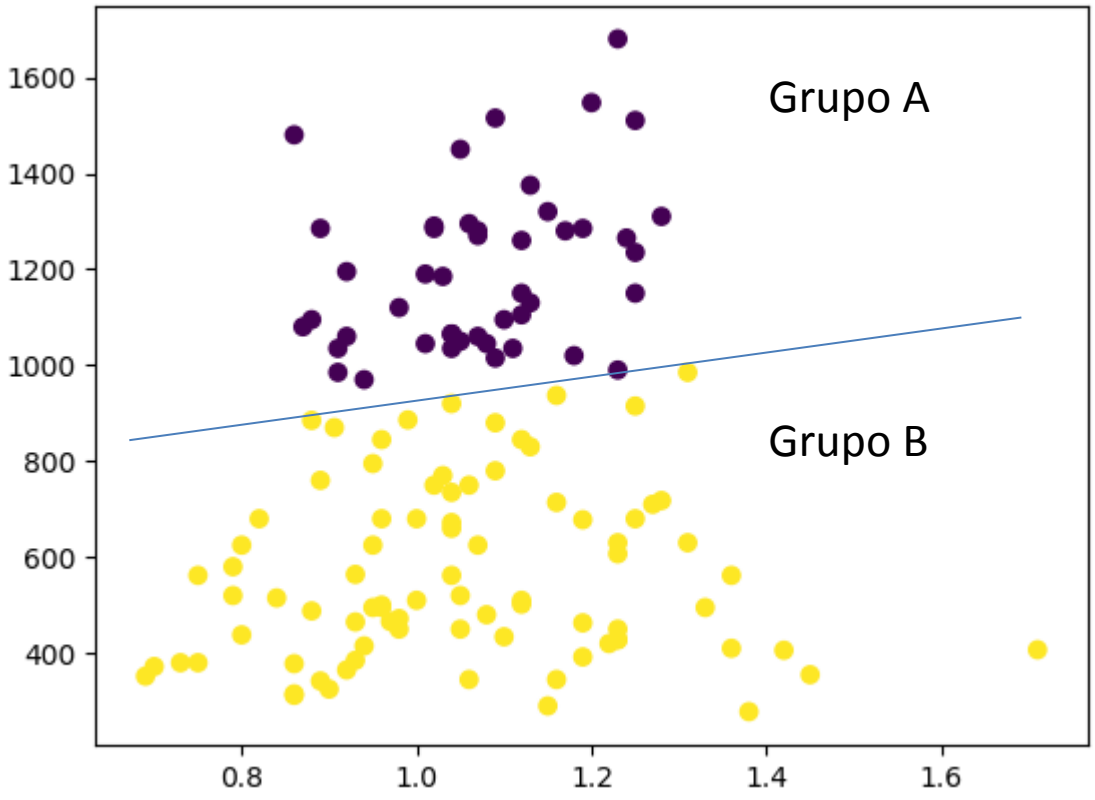
Mediante el gradiente descendente, con cada nueva pasada se va ajustando el modelo



Siguiente pasada

Ajustando el modelo

Mediante el gradiente descendente, con cada nueva pasada se va ajustando el modelo



Siguiente pasada

El algoritmo de entrenamiento finaliza porque el error es mínimo

Propiedades de Adaline

Sólo funciona si las clases son ***linealmente separables***

Si los datos contienen clases no linealmente separables, el algoritmo aún **puede converger**

Preparación de Datos

Para armar un modelo es necesario preparar los datos antes de procesarlos. En el caso de las redes neuronales se necesita que todas las variables estén en la misma escala, por ejemplo, de 0 a 1

Existen muchas técnicas para solucionar algunos problemas que tienen los datos reales, por ejemplo, valores muy alejados de la media (outliers), nulos, distribuciones sesgadas, etc.

En las prácticas de este taller se recibirán los datos escalados de 0 a 1, sin outliers, sin nulos y bien distribuidos.

Práctica

Para esta práctica utilizaremos datos que relacionan algunas mediciones de vinos con la región de donde provienen

La variable a predecir contiene 2 clases, por lo tanto se crean 2 variables de salida

Hue	Proline	Region
1.45	355	2
0.73	380	2
0.88	1095	1
1.04	1035	1
0.96	845	2

Hue	Proline	Region A	Region B
0.745	0.055	0	1
0.039	0.073	0	1
0.186	0.583	1	0
0.343	0.540	1	0
0.265	0.404	0	1

Utilizando el software libre simbrain construiremos un modelo para predecir la Región a la que pertenece el vino

Perceptrón Multicapa (MLP)

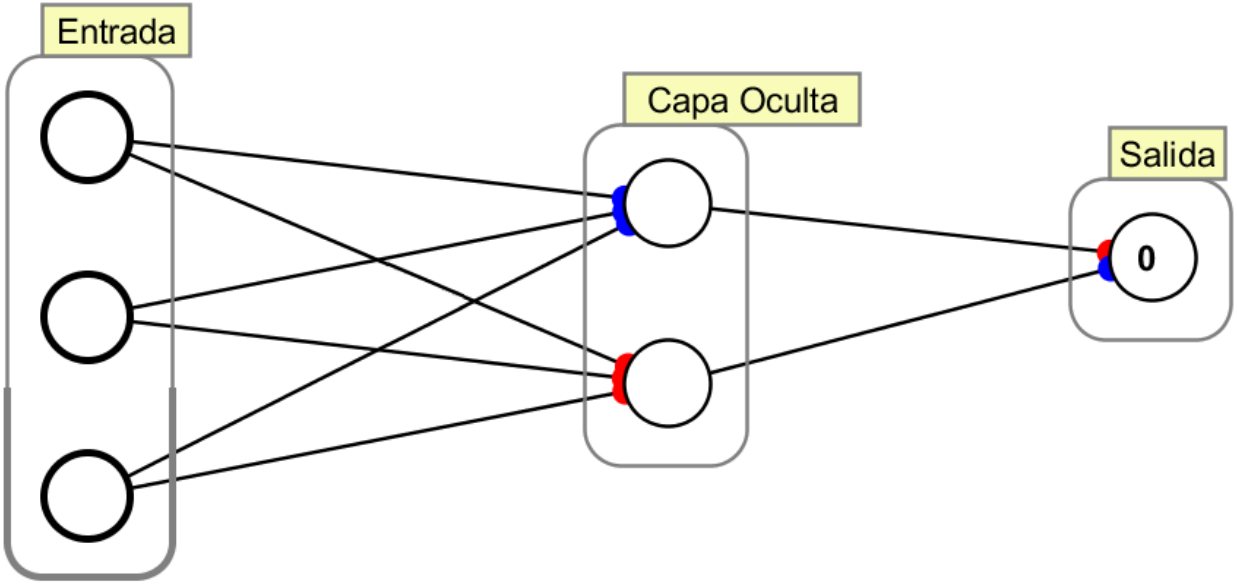
El Perceptrón y Adaline vistos hasta ahora, son capaces de generalizar si las entradas son similares

Por ejemplo, 01111000 y 01111001 darán respuestas similares

Pero estas similitudes físicas no siempre deberían tener respuestas similares. Hay veces en donde la similitud es abstracta.

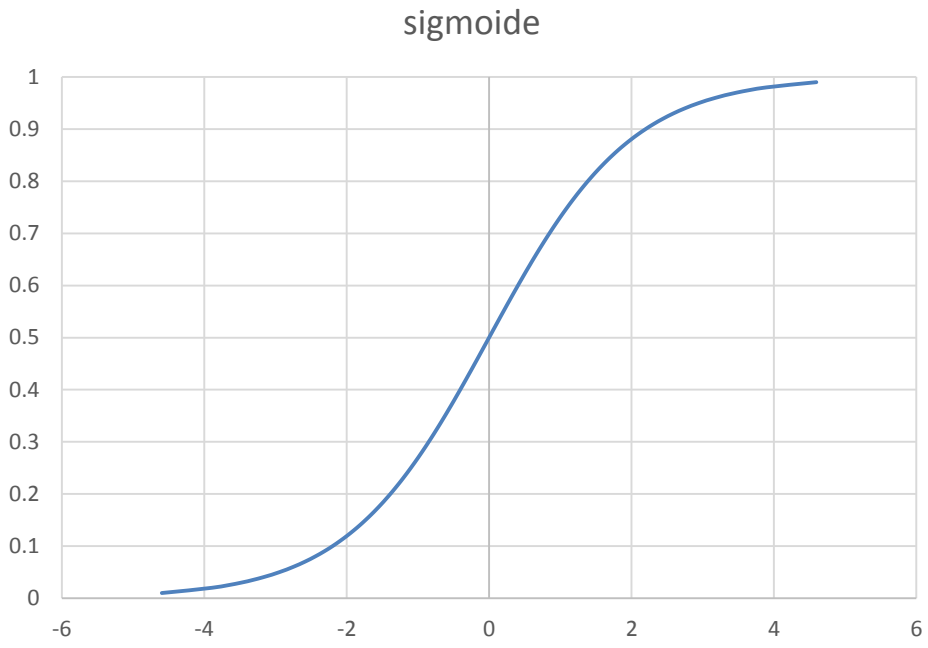
Los MLP tienen representaciones internas que los habilitan a tratar con similitudes abstractas

Perceptrón Multicapa (MLP)



Un MLP contiene una o más capas ocultas

Función Sigmoide



La función de activación Sigmoide es una de las más usadas en los MLP

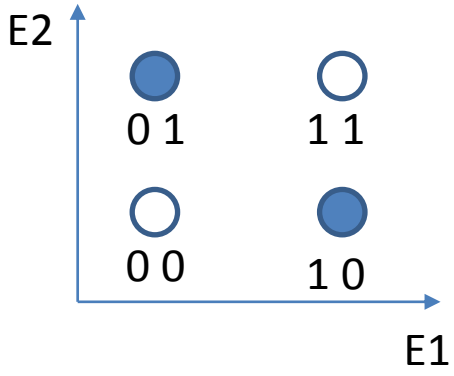
Tiene su salida acotada de 0 a 1 y tiene un comportamiento no lineal

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

Perceptrón Multicapa (MLP)

El problema XOR no puede resolverse con un Perceptrón pero sí con un MLP

E1	E2	Respuesta
0	0	0
0	1	1
1	0	1
1	1	0

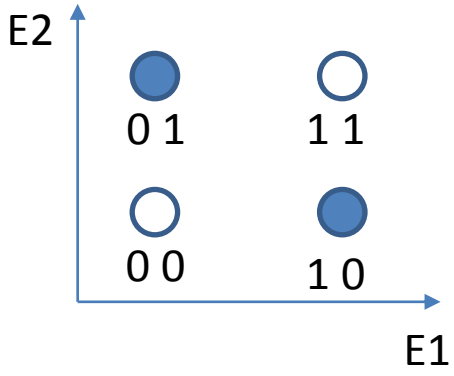


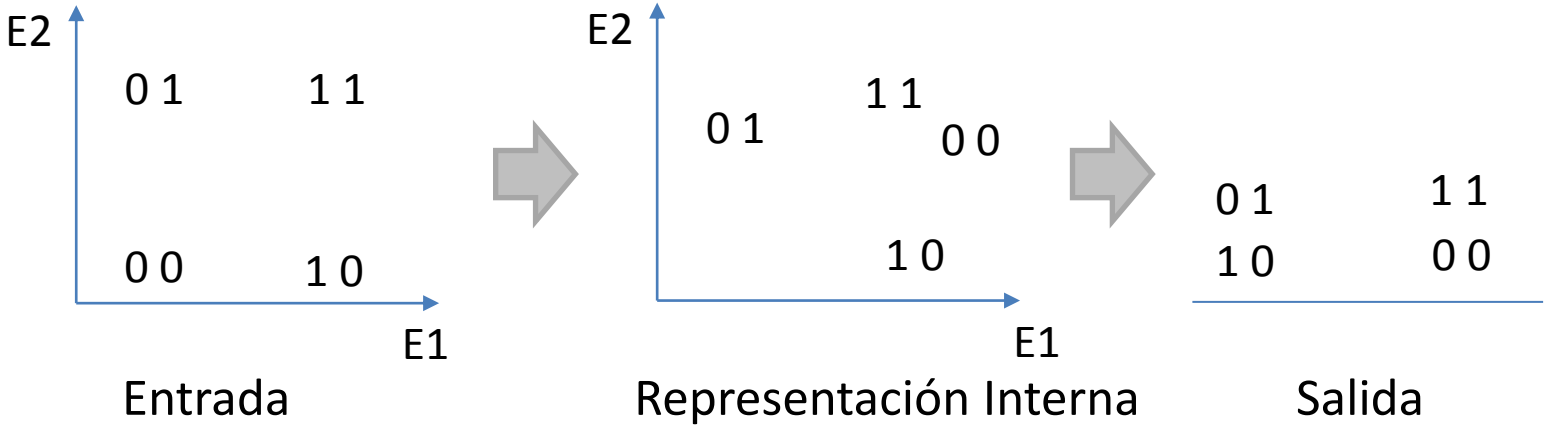
No existe una línea recta que pueda separar las respuestas 0 y 1

Perceptrón Multicapa (MLP)

El problema XOR no puede resolverse con un Perceptrón pero sí con un MLP

E1	E2	Respuesta
0	0	0
0	1	1
1	0	1
1	1	0





Regla de Aprendizaje de los MLP

Los MLP utilizan un algoritmo de aprendizaje llamado Backpropagation que calcula el error en la capa de salida y va *propagando* el error sobre las demás capas, para actualizar los ponderadores.

Práctica

Corazón: los datos contienen diagnósticos de pacientes con posibles problemas de corazón. Se trata de predecir si el paciente tiene o no problemas

Ofertas: se trata de una campaña de marketing de servicios financieros. La meta es crear un modelo que pronostique si el cliente aceptará o no la oferta

Dígitos: los datos tienen los dígitos de 0 a 9 en forma de imágenes de 120 pixeles. Se debe construir un modelo que identifique cada dígito, aún en presencia de ruido

A tener en cuenta

Las redes neuronales son demasiado plásticas por lo que hay que evitar overfitting

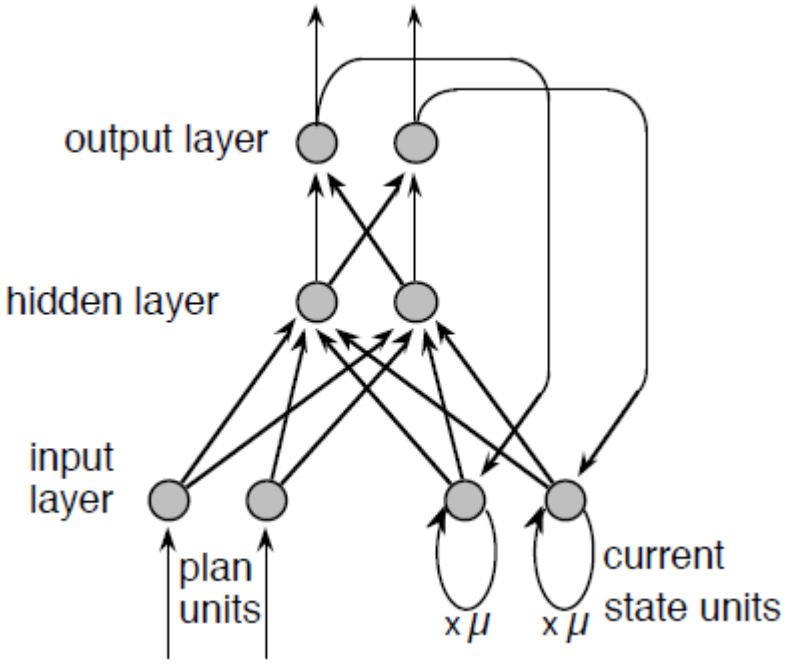
En la mayoría de los casos es necesario preparar los datos antes de trabajar con algoritmos de aprendizaje automático

En general es buena idea reducir la dimensionalidad

Existen muchas herramientas para tratar con datos. Python es un excelente ambiente de desarrollo. Pandas, NumPy, MatPlotLib, Scikit-learn, Keras, TensorFlow forman el complemento ideal

Redes Recurrentes

Las configuraciones que vimos hasta ahora se llaman multilayer feedforward networks, porque la información siempre va de las primera capa hacia la última y no forman lazos

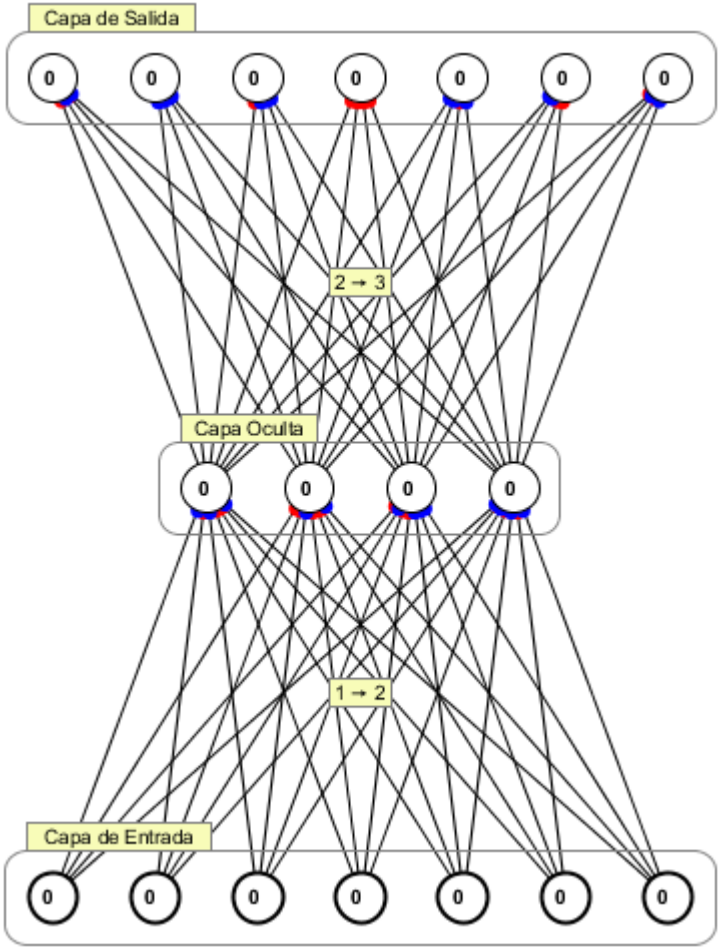


En otro tipo de configuración la información retrocede, son las redes **neuronales recurrentes**

Autoencoder

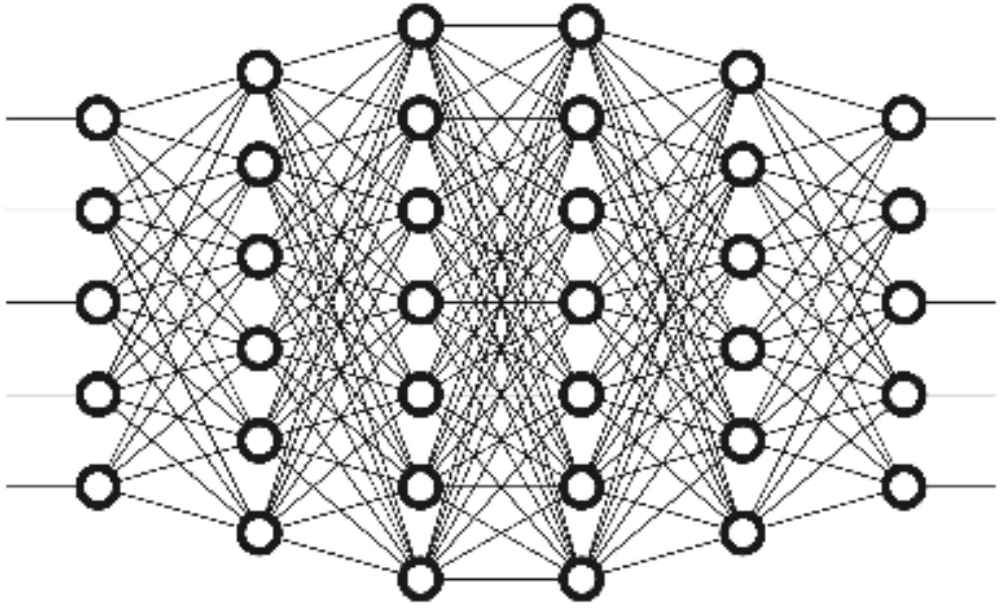
Un autoencoder copia la entrada a la salida

Si la capa oculta tiene menos nodos que la de entrada, la red será forzada a extraer la información más relevante de los datos



Deep Learning

Hace muy pocos años, gracias a los avances de hardware y software, se han realizado grandes logros con redes neuronales profundas



Polo Científico Tecnológico

Municipalidad de Trenque Lauquen

trenquelauquen.gov.ar/mct

trenquetecno@trenquelauquen.gov.ar

02392 15 53-1228