Tema 03. Redes de Neuronas

Autor: Ismael Sagredo Olivenza

3.1 Introducción a las redes de neuronas

Al intentar construir máquinas inteligentes surge de forma inmediata como modelo el cerebro humano. Esta tecnología se engloba dentro de la **IA Subsimbolica** que diseña sistemas genéricos, en este caso, intentando modelizar el funcionamiento dle propio cerebro humano.

Las redes de neuronas se basan en los trabajos realizados por **Warren McCulloch** y **Walter Pittis** en 1943.

Apoyándose en la teoría de la computación de Allan Touring y los estudios sobre la neurología, diseñaron un sistema basado en neuronas artificiales que simulaba el funcionamiento de las neuronas biológicas

Frank Rosenblatt (1957) generalizó el modelo de neuronas artificiales de McCulloch-Pittis añadiéndole aprendizaje y creando el Perceptron.

En 1986 se desarrollo del algoritmo de **Retropropagación** (Rumelhart, Hinton y Willians en 1986) y a partir de entonces las Redes de Neuronas se han ido aplicando a diferentes problemas

Este algoritmo hizo que las redes aprendieran con más de una capa.

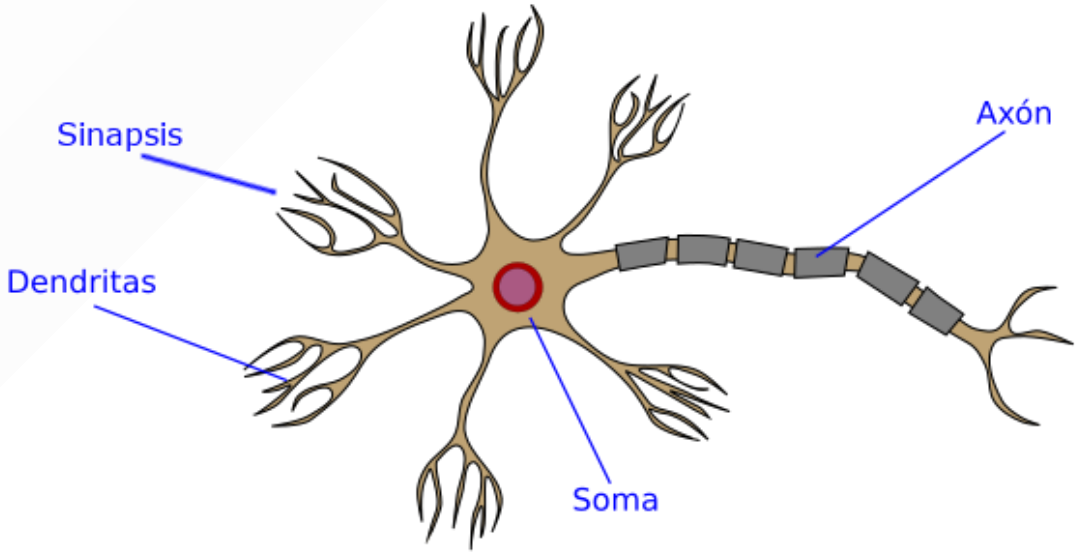
3.1.1 La neurona biológica

La neurona se considera como **la unidad funcional básica del sistema nervioso**, que en los organismos vivos es el encargado de su mantenimiento vital. El sistema nervioso tiene como funciones principales el funcionamiento global del ser vivo y su relación con el entorno.

Las células nerviosas cuentan, como el resto de las células, con un cuerpo celular y un **núcleo** donde se encuentra la **información genética** de la célula o ADN. Pero además cuentan con una serie de prolongaciones con las que se intercomunican entre sí.

3.1.2 Partes de la neurona biológica

- **El axón** es la ramificación de salida de la neurona. A través de él se propagan los impulsos nerviosos.
- **Las dentritas** son ramificaciones de entrada para la célula
- Entre dentrita y axón se encuentra la **sinapsis** que es un líquido semiconductor que permite circular un cierto grado del impulso



Cuando el impulso llega a los extremos del axón, éste libera unas sustancias químicas denominadas **neurotransmisores** que se generan en los terminales del axón denominados **botones sinápticos**. Estos neurotransmisores intentan cruzan el espacio sináptico. Pero este le ofrece una resistencia dieléctrica que debe ser superada por los neurotransmisores para poder alcanzar las dendritas de la neurona receptora.

A pesar de lo que pueda parecer, la capacidad de cómputo principal de las neuronas no está en su núcleo sin en sus **interconexiones**. Es decir en la Sinapsis. La potencia de cerebro humano se obtiene por el número de interconexiones que se producen entre las neuronas. Una neurona se conecta con unas 10.000 neuronas y en el cerebro humano hay varios cientos de millones de ellas.

3.2 Redes de neuronas artificiales

Al igual que el cerebro humano, sustentan su capacidad de cálculo en las interconexiones. Pero debemos tener en cuenta que la neurona artificial es una analogía matemática y que su complejidad sináptica es mucho menor que la biológica.

Definimos Neurona Artificial como un elemento que posee un estado interno (S) denominado **nivel de activación** que simula la resistencia dieléctrica de la sinapsis. Además, recibe un conjunto de señales que le permiten cambiar de estado a través de una conexión con otras neuronas, que influyen en el nivel de activación con señales positivas y negativas (Simulando los **neurotransmisores**)

- Su potencia de cálculo reside en las interconexiones.
- La neurona artificial es una analogía matemática por lo que su complejidad sináptica es mucho menor.
- Una **neurona artificial** es un elemento informático que posee un estado interno (S) denominado **nivel de activación** que simula la complejidad dieléctrica de la sinapsis.
- Recibe un conjunto de señales que le permiten cambiar de estado. (Simula los **neurotransmisores**)

La función matemática que permite cambiar el estado de activación dependiendo de las entradas se denomina **función de activación**

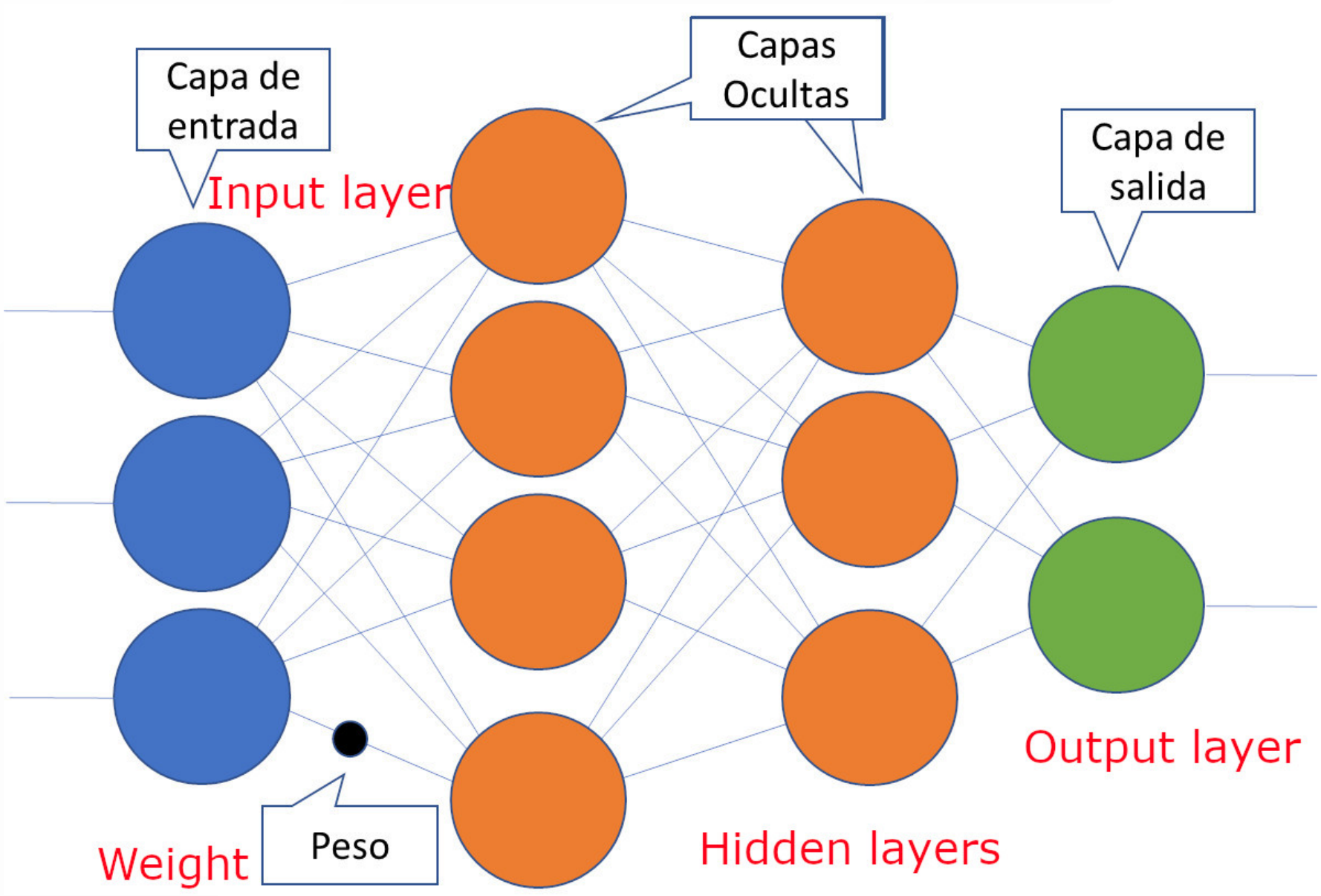
Cada entrada de la neurona es **multiplicada por un peso** que es modificable durante el aprendizaje. Cada peso permite que llegue una cantidad mayor o menor de la señal procedente de otra neurona.

Una red neuronal es una colección de neuronas cuya salida está conectada a las entradas de otras neuronas formando un **grafo dirigid** G(V,A) donde los vértices (V) son las neuronas y las aristas (A) son las conexiones entre ellas.

3.2.1 Tipos de neuronas

- **Neuronas de entrada**: Son las encargadas de recibir los datos del exterior de la red e inician la propagación de dichos datos por toda la estructura
- **Neuronas ocultas**: neuronas que permiten enlazar las neuronas de entrada y las neuronas de salida, aunque también pueden estar conectadas a otras neuronas ocultas
- **Neuronas de salida**: Son las que muestran al exterior los resultados obtenidos después de que la entrada haya recorrido toda la red

Capa: colección de neuronas que tienen una función común


Capa de entrada
Input layer
Capas Ocultas
Capa de salida
Output layer
Hidden layers
Weight
Peso

Normalmente en cada neurona se hace la suma ponderada de los pesos para calculara su activación (aunque tambien se usa max y min)

Podemos definir por lo tanto de forma matemática una red de neuronas como una ecuación vectorial, de la siguiente forma:

$$\vec{S} = F(F(F(\vec{X} \cdot W_1) \cdot W_2) \dots W_i)$$

Si la **función de activación es lineal**, añadir más capas en la red es irrelevante y se podría crear una red de una sola capa oculta equivalente.

Pero si la función de activación no es lineal (lo habitual), entonces añadir más capas si tiene sentido.

3.2.1 Funciones de activación más usadas

- La función escalón donde α es el valor umbral de activación:

$$y(x) = \begin{cases} 1 & si \quad x \geq \alpha \\ \\ -1 \quad o \quad 0 & si \quad x \leq \alpha \end{cases}$$

- función sigmoidal (logística)

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

- La tangente hiperbólica:

$$y(x) = tanh(x) = \frac{senh(x)}{cosh(x)} = \frac{1 - e^{-2x}}{1 + e^{-2x}}$$

- La función rectificadora (ReLU):

$$y(x) = max(0, x)$$

Las funciones sigmoidal e hiperbólicas son diferenciables y monótonas y son no lineales, valores que nos interesan.

3.3 Perceptron multicapa

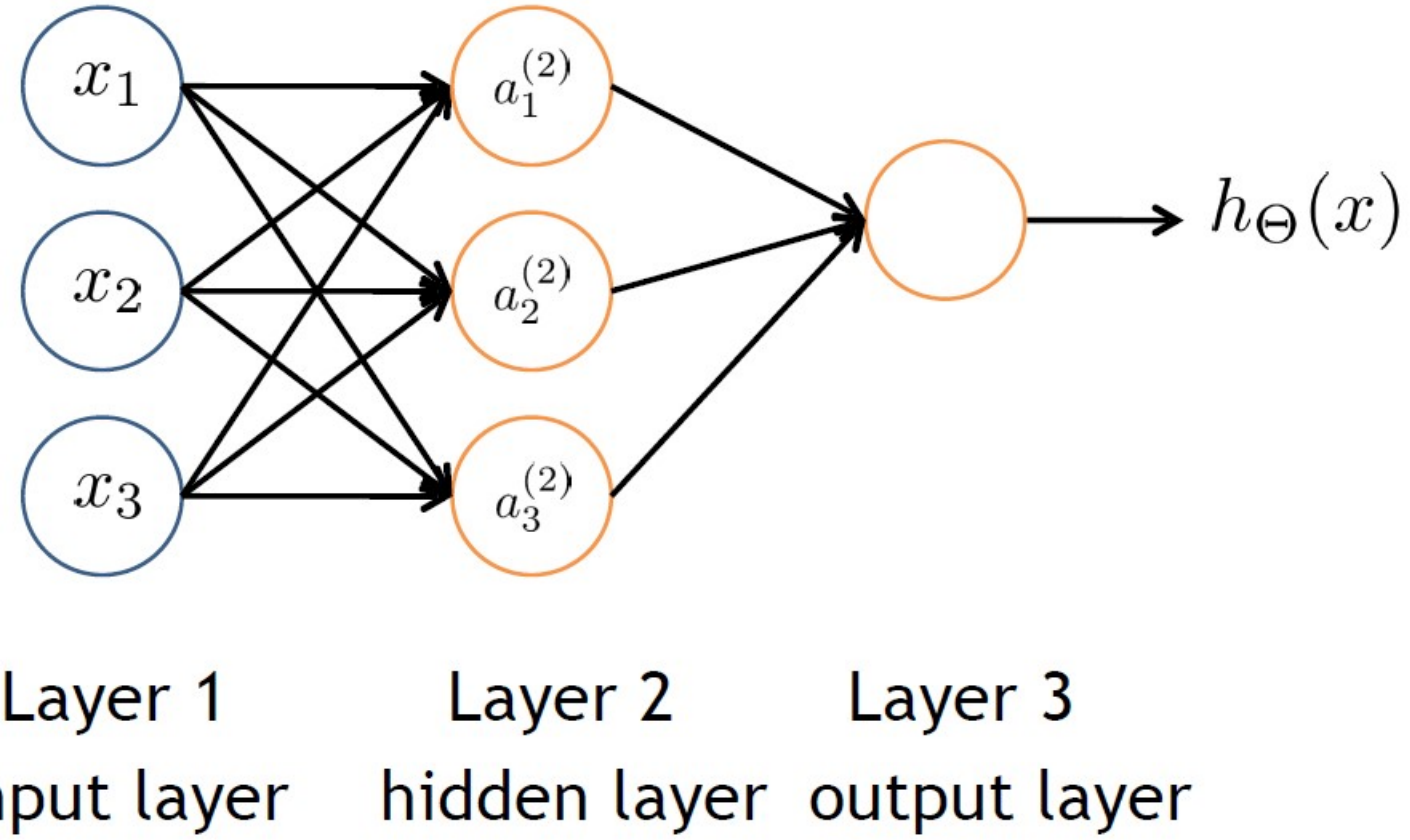
Es una generalización multicapa del **perceptrón** definido por Rosenblattd (1962). Usa como función de entrada la suma ponderada y como función de activación la que el usuario establezca (inicialmente el perceptron de Rosenblattd usaba la función escalón)

Rumelhartt, Hilton y Willians en 1989 presentaron un mecanismo que permitía dicha propagación del error por todos los pesos de las conexiones, permitiendo crear modelos de redes de más de una capa.

El **perceptrón multicapa** está demostrado que es un **aproximador universal** de cualquier función.

3.3.1 Pero antes de entrenar la red...

Veamos cuál es la primera fase de una red neuronal: la **propagación hacia delante (feedforward)** .



Sea a_i^j la activación de la unidad i en la capa j y Θ^j la matriz de pesos que controla la activación desde la capa j a la capa j+1.

La **capa de entrada** tendrá tantas neuronas como datos de entrada.

La **capa oculta** podría tener cualquier número de neuronas, pero para este ejemplo supondremos que tiene el mismo número que la entrada.

Para cada conexión desde la entrada a la capa oculta hay un peso w que conecta cada neurona de la capa de entrada con otra neurona de la capa oculta. La activación es la suma de la entrada multiplicada por los pesos y , añadiendo un parámetro más llamado **umbral/bias**. Es como si hubiera otra entrada siempre a 1

En el ejemplo anterior

$$a_1^2 = g(\Theta_{1,1}^1 x_1 + \Theta_{1,2}^1 x_2 + \Theta_{1,3}^1 x_3 + b_1^1)$$

$$a_2^2 = g(\Theta_{2,1}^1 x_1 + \Theta_{2,2}^1 x_2 + \Theta_{2,3}^1 x_3 + b_2^1)$$

$$a_3^2 = g(\Theta_{3,1}^1 x_1 + \Theta_{3,2}^1 x_2 + \Theta_{3,3}^1 x_3 + b_3^2)$$

El umbral o vias es como si establecemos ($x_0 = 1$)

$$a_1^2 = g(\Theta_{1,0}^1 + \Theta_{1,1}^1 x_1 + \Theta_{1,2}^1 x_2 + \Theta_{1,3}^1 x_3)$$

$$a_2^2 = g(\Theta_{2,0}^1 + \Theta_{2,1}^1 x_1 + \Theta_{2,2}^1 x_2 + \Theta_{2,3}^1 x_3)$$

$$a_3^2 = g(\Theta_{3,0}^1 + \Theta_{3,1}^1 x_1 + \Theta_{3,2}^1 x_2 + \Theta_{3,3}^1 x_3)$$

Para calcular a_1^3 (La capa de salida))...

$$a_1^3 = g(\Theta_{1,0}^2 + \Theta_{1,1}^2 a_1^2 + \Theta_{1,2}^2 a_2^2 + \Theta_{1,3}^2 a_3^2)$$

Nótese que a_0^2 no se calcula porque es como si la activación siempre estuviese a 1.

Feedforwar iterativo:

```
#W weights, X input, A! activation of Hidden layer, G function
for j in range(len(A1))
  for i in range(len(X))
    A1[j] = W[j,i]*X[i]
  A1[j]=g(A1[j])

#Same for A2, A3.... An
```

3.3.2 Forward propagation (feedforward)

Si generalizamos para cualquier número de capas y número de neuronas por capa:

$$a_i^j = g((\sum_{k=1}^{|a^{j-1}|} \Theta_{i,k}^j a_k^{j-1}) + b_i^j) \forall i \in [1..|a^j|], j \in [1..|C|]$$

Donde C son las capas del modelo, |C| es el número de capas, $|a_j|$ es el número de neuronas en la capa j, $|a^{j-1}|$ número de neuronas en la capa anterior y g la función de activación (ReLU, sigmoidal, etc).

3.3.3 Implementación vectorial

Input = 3, Hidden = 5, Output = 3
$a^1 = [1, x_1, x_2, x_3]$
$a^2 = g(\Theta^1 a^1)$ => Θ^1 shape [5x4], a^1 shape [4x1]
a^2 shape [5x1]
Add bias a^2 shape [5x1] => shape [6x1]
$a^3 = g(\Theta^2 a^2)$ => Θ^2 shape [3x6], a^2 shape [6x1]
a^3 shape [3x1]

Θ^j shape is a^{j+1} shape[0], a^j shape[0]

Nota: para realizar la multiplicación de matrices, los vectores deben estar en formato vertical (a^T).

3.3.4 ¿Cómo codifico la salida?

Todos los valores de una NN son números reales, ¿cómo creo un clasificador?

En teoría, cualquier codificación de la salida real sería válida, en la práctica, se utiliza como codificación que cada neurona de salida de la red representa una clase.

Otras condifcaciones: La codificación binaria, por ejemplo, produce peores resultados y no expresa la salida en forma de probabilidad, lo cual es bastante útil para razonar sobre la salida de la red a posteriori.

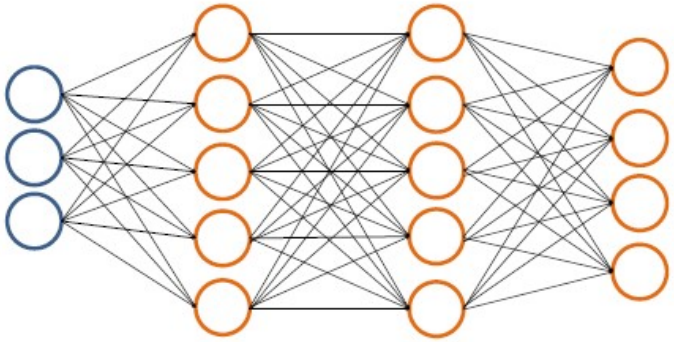
Multiple output units: One-vs-all

Pedestrian

Car

Motorcycle

Truck

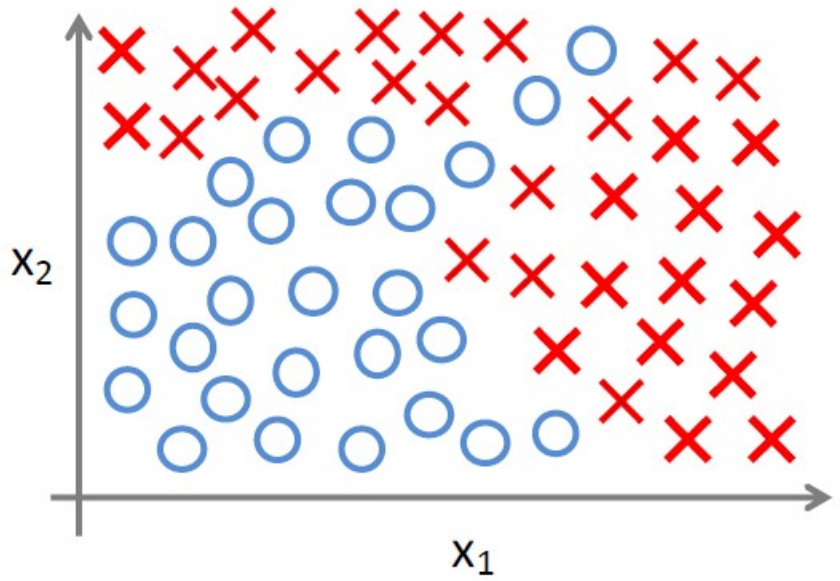


$h_\Theta(x) \in \mathbb{R}^4$

Want $h_\Theta(x) \approx \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}$, $h_\Theta(x) \approx \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}$, $h_\Theta(x) \approx \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}$, etc.

when pedestrian when car when motorcycle

Non-linear classification



$$g(\theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_1 x_2 + \theta_4 x_1^2 x_2 + \theta_5 x_1^3 x_2 + \theta_6 x_1 x_2^2 + \dots)$$

x_1 = size
x_2 = # bedrooms
x_3 = # floors
x_4 = age
$\dots$
x_{100}

$O(n^2)$ quadratic features: x_1^2, x_1x_2, x_1x_3, ...
$O(n^3)$ cubic features: 170,000

3.3.5 Normalización

La red neuronal funciona mejor con datos normalizados, ya sea en el rango 0..1 o en el rango -1..1. Por lo tanto, es un requisito previo que los valores de entrada de la red estén normalizados, utilizando z-Score o dividiendo por el valor máximo. Podemos utilizar Sklearn para normalizar los datos

```
from sklearn import preprocessing
import numpy as np
x_array = np.array([2,3,5,6,7,4,8,7,6])
normalized_arr = preprocessing.normalize([x_array])
print(normalized_arr) ## [[0.11785113 0.1767767  0.29462783 0.35355339 0.41247896 0.23570226
## 0.47140452 0.41247896 0.35355339]]
...
preprocessing.MinMaxScaler()
```