Prácticas de Aprendizaje Automático

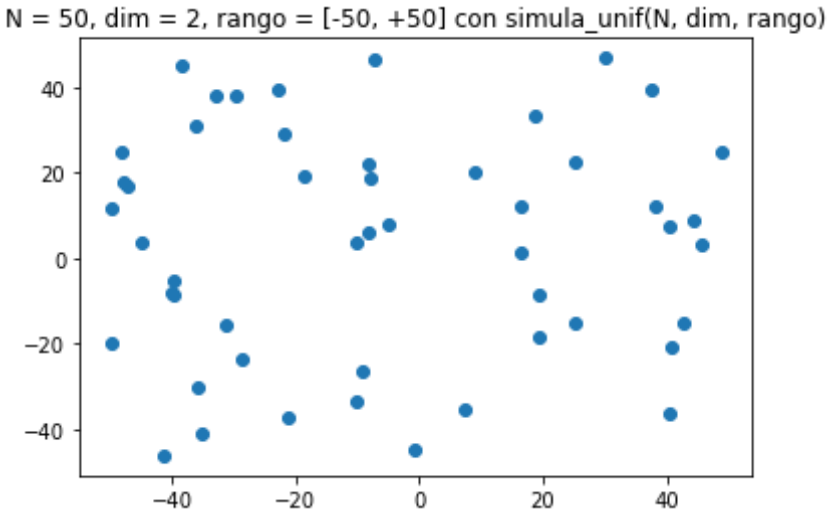
Trabajo 2: Complejidad de H y Modelos Lineales

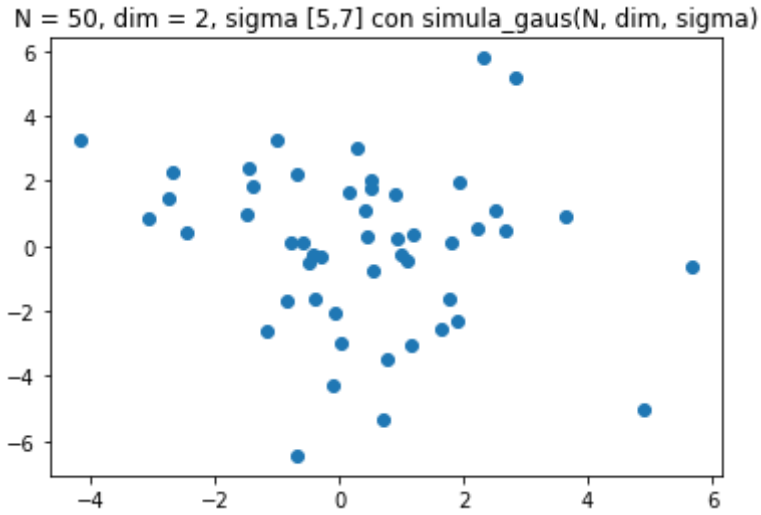
Pablo Mesejo y Jesús Giráldez
Universidad de Granada
Departamento de Ciencias de la Computación e Inteligencia Artificial

UNIVERSIDAD DE GRANADA

1. Ejercicio sobre la complejidad de H y el ruido

- Dibujar gráficas con nubes de puntos simuladas bajo ciertas condiciones



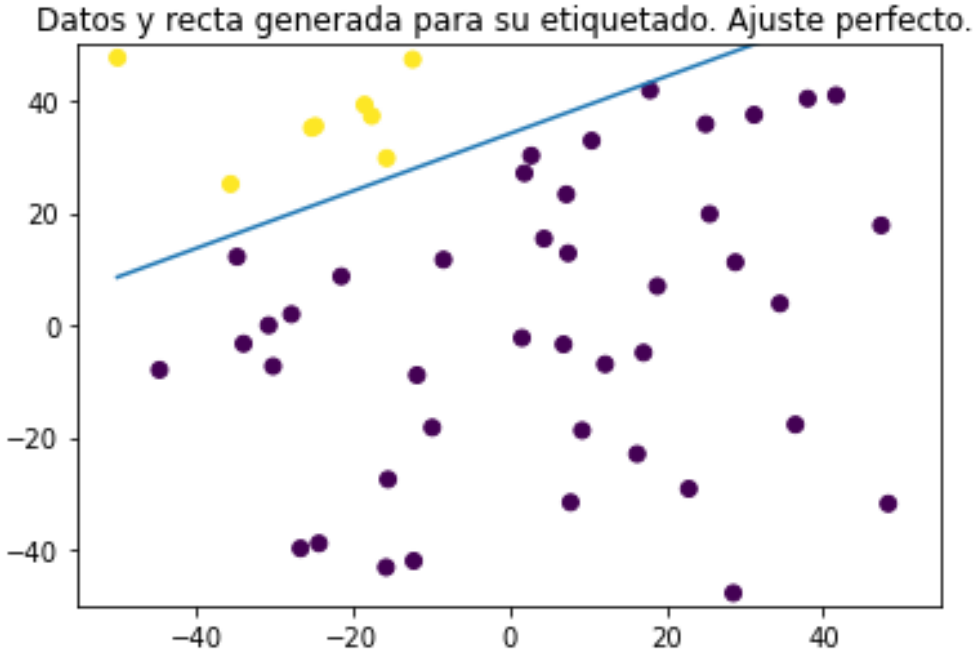


Recordad que H es nuestro *Hypothesis Set*: el conjunto de fórmulas candidatas para resolver nuestro problema. Al aplicar un algoritmo de aprendizaje, somos capaces de quedarnos con una única hipótesis final (*g*, que intenta aproximar *f*, la función objetivo desconocida).

Hypothesis Set	Learning Algorithm
ANNs	Backpropagation
Perceptron	PLA
Linear Regression	Pseudoinverse

1. Ejercicio sobre la complejidad de H y el ruido

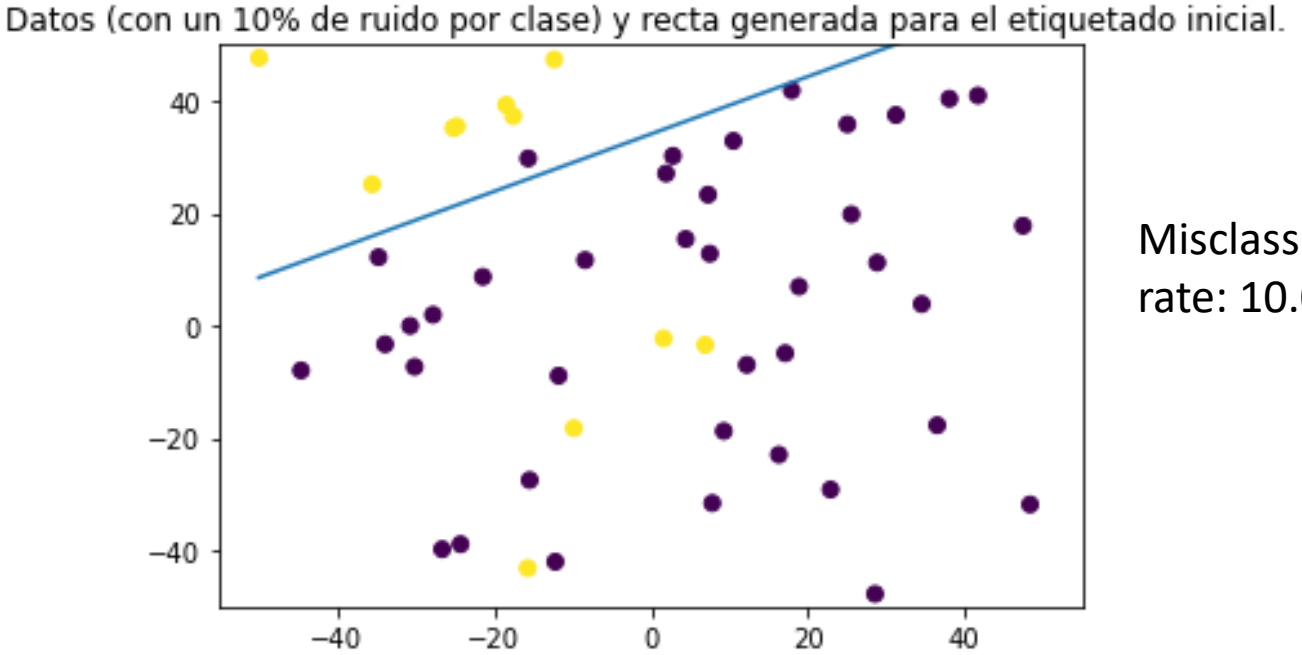
- Generar otra muestra de puntos 2D a la que vais a añadir una etiqueta usando el signo de la función **`f(x,y) = y-ax-b`**



Misclassification rate: 0.0%

1. Ejercicio sobre la complejidad de H y el ruido

- Modificar de forma aleatoria un 10% de las etiquetas positivas y otro 10% de las negativas



Misclassification rate: 10.0%

1. Ejercicio sobre la complejidad de H y el ruido

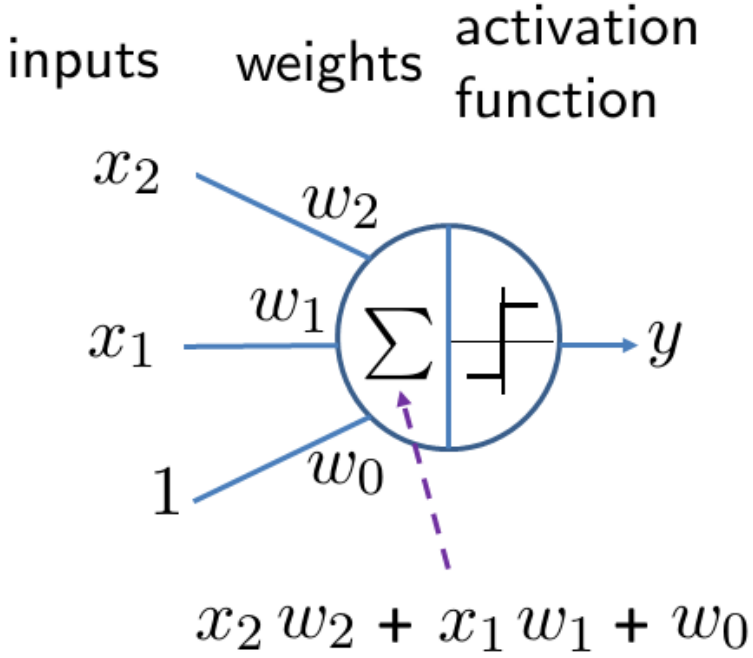
- Emplear otras funciones para definir la frontera de clasificación de los puntos de la muestra en lugar de una recta

$$f(x,y) = (x-10)^2 + (y-20)^2 - 400$$
$$f(x,y) = 0{,}5(x+10)^2 + (y-20)^2 - 400$$
$$f(x,y) = 0{,}5(x-10)^2 - (y+20)^2 - 400$$
$$f(x,y) = y - 20x^2 - 5x + 3$$

  - ¿Necesariamente funciones más complejas son mejores clasificadores (es decir, representan “mejores” bordes de decisión)?
  - ¿Es posible superar/mejorar ese 10% de error de clasificación?
  - ¿Qué pasa si repetimos el proceso con estas funciones más complejas (las empleamos para etiquetar los datos y luego metemos un 10% de ruido)? ¿Qué error de clasificación tenemos? ¿Es menor que ese 10%?

NOTA: se aconseja utilizar la función plot_datos_cuad proporcionada en template_trabajo2.py.

2. Modelos Lineales. Perceptron

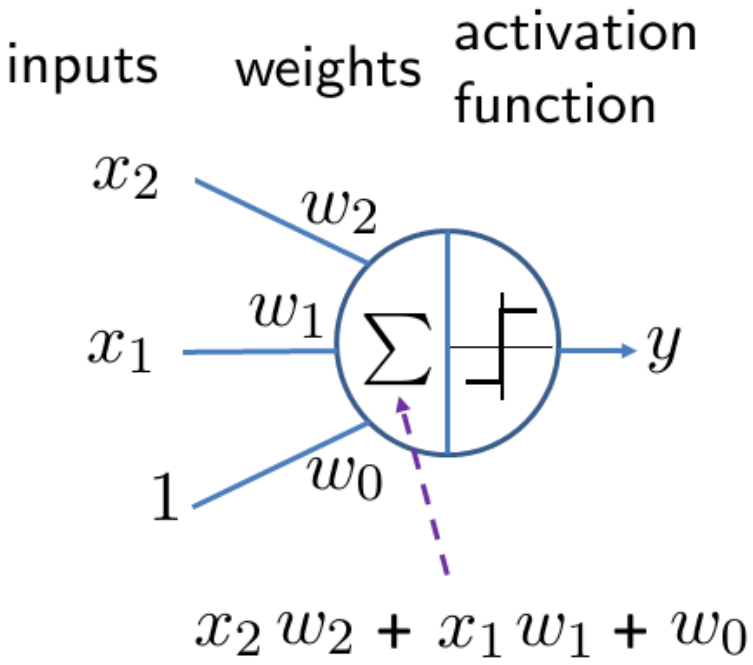


Referencias de apoyo sobre el perceptrón:

"Pattern Recognition and Machine Learning" (Christopher M. Bishop, 2006, págs. 192-196)

"Learning from Data" (Yaser S. Abu-Mostafa et al., 2012, págs. 5-8)

2. Modelos Lineales. Perceptron



$$h(\mathbf{x}) = \text{sign}\left(\left(\sum_{i=1}^{d} w_i\, x_i\right) + \quad w_0 \quad\right)$$

Introduce an artificial coordinate $x_0 = 1$:

$$h(\mathbf{x}) = \text{sign}\left(\sum_{i=0}^{d} w_i\, x_i\right)$$

In vector form, the perceptron implements

$$h(\mathbf{x}) = \text{sign}(\mathbf{w}^{\mathsf{T}}\mathbf{x})$$

http://work.caltech.edu/slides/slides01.pdf slide 11

2. Perceptron Learning Algorithm (PLA)

- Given the data set $(\mathbf{x}_n, y_n), n = 1,2,\cdots,N$
- Step.1: Fix $\mathbf{w}_{\text{ini}}=0$
- Step.2: Iterate on the $\mathcal{D}$-samples improving the solution:
- repeat

  For each $\boldsymbol{x}_i \in \mathcal{D}$ **do**

  if: $sign(\boldsymbol{w}^T\boldsymbol{x}_i) \neq y_i$ **then**

  update $\mathbf{w}$: $\mathbf{w}_{\text{new}} = \mathbf{w}_{\text{old}} + y_i\mathbf{x}_i$

  else continue

  End for
- Until No changes in a full pass on $\mathcal{D}$

$$E_{in}(\boldsymbol{w}) = \frac{1}{N}\sum_{i=1}^{N} \mathbf{1}_{sign(\boldsymbol{w}^T\boldsymbol{x}_i)\neq y_i}$$

2. Perceptron. Apuntes finales.

- Si **datos linealmente separables** → Convergencia garantizada
  De lo contrario → No convergerá

- Es un algoritmo que **suele requerir muchas iteraciones para converger** (en problemas linealmente separables)
  En nuestro caso, del orden de miles o decenas de miles…

- **Muy sensible a la inicialización**

2. Modelos Lineales. Logistic Regression

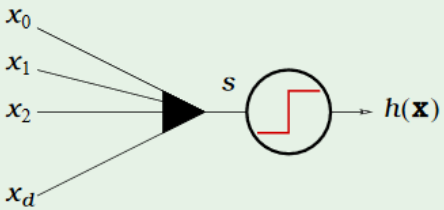
A third linear model

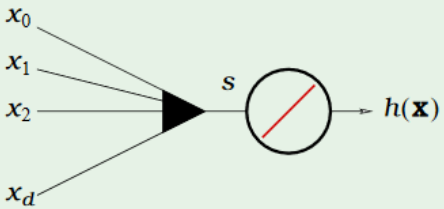
lineales en los parámetros del modelo

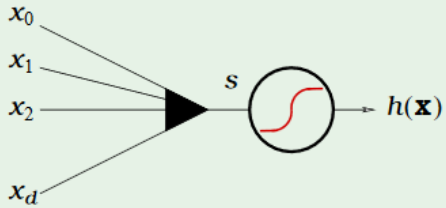
$$s = \sum_{i=0}^{d} w_i x_i$$

¡Ojo! Aunque usa el término “regresión” es una técnica de clasificación

linear classification	linear regression	logistic regression
$h(\mathbf{x}) = \text{sign}(s)$	$h(\mathbf{x}) = s$	$h(\mathbf{x}) = \theta(s)$







http://work.caltech.edu/slides/slides09.pdf slide 10

2. Modelos Lineales. Logistic Regression

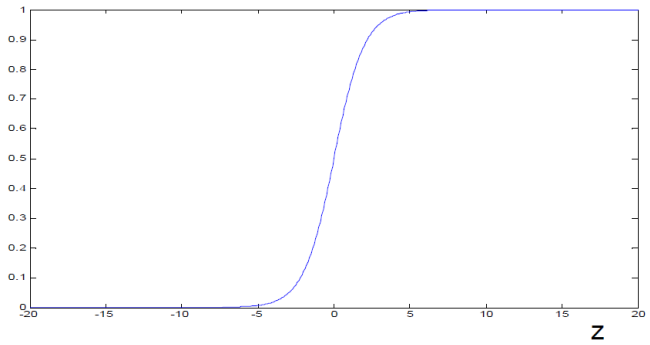
The LR classifier is defined as

$$\sigma\left(f(\mathbf{x}_i)\right) \begin{cases} \geq 0.5 & y_i = +1 \\ < 0.5 & y_i = -1 \end{cases}$$

where $\sigma(f(\mathbf{x})) = \frac{1}{1+e^{-f(\mathbf{x})}}$

The logistic function or sigmoid function

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



la salida está acotada entre 0 y 1 y, por tanto, se interpreta en términos probabilísticos

2. Modelos Lineales. Logistic Regression

Logistic regression algorithm

Ajustad cuidadosamente el learning rate y N

1: Initialize the weights at $t = 0$ to $\mathbf{w}(0)$

2: **for** $t = 0, 1, 2, \ldots$ **do**

3: Compute the gradient

$$\nabla E_{\text{in}} = -\frac{1}{N}\sum_{n=1}^{N}\frac{y_n \mathbf{x}_n}{1 + e^{y_n \mathbf{w}^{\intercal}(t)\mathbf{x}_n}}$$

4: Update the weights: $\mathbf{w}(t+1) = \mathbf{w}(t) - \eta \nabla E_{\text{in}}$

5: Iterate to the next step until it is time to stop

6: Return the final weights $\mathbf{w}$

inicializar los pesos a cero implica inicializar la probabilidad a 0.5

N: batch size

Parar el algoritmo cuando $||\mathbf{w}^{(t-1)} - \mathbf{w}^{(t)}|| < 0{,}01$, donde $\mathbf{w}^{(t)}$ denota el vector de pesos al final de la época t.

2. Modelos Lineales. Logistic Regression

$$E_{\text{in}}(\mathbf{w}) = \frac{1}{N}\sum_{n=1}^{N} \ln\left(1 + e^{-y_n \mathbf{w}^{\mathsf{T}}\mathbf{x}_n}\right)$$

"cross-entropy" error

Para comprender mejor de dónde proviene esta expresión, se recomienda consultar https://home.work.caltech.edu/slides/slides09.pdf y https://www.youtube.com/watch?v=qSTHZvN8hzs

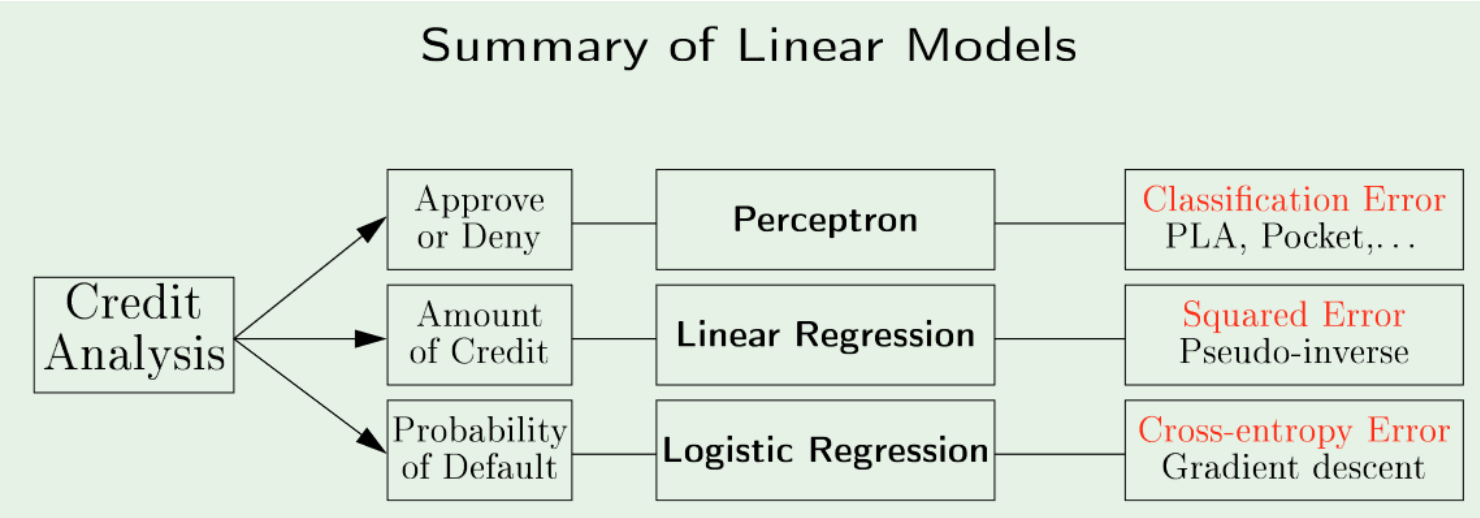
Comparativamente, recordad que en regresión lineal era

$$E_{\text{in}}(\mathbf{w}) = \frac{1}{N}\sum_{n=1}^{N} (\mathbf{w}^{\mathsf{T}}\mathbf{x}_n - y_n)^2$$

Error cuadrático medio
(y este error se podía minimizar en *closed-form* con la pseudoinversa)

2. Modelos Lineales. Logistic Regression

Contextualizando todo un poco…



https://home.work.caltech.edu/slides/slides09.pdf

Más información sobre Logistic Regression (Andrew Ng; utiliza una notación diferente):
https://www.youtube.com/playlist?list=PLNeKWBMsAzboR8vvhnlanxCNr2V7ITuxy

2. Modelos Lineales. Más ideas.

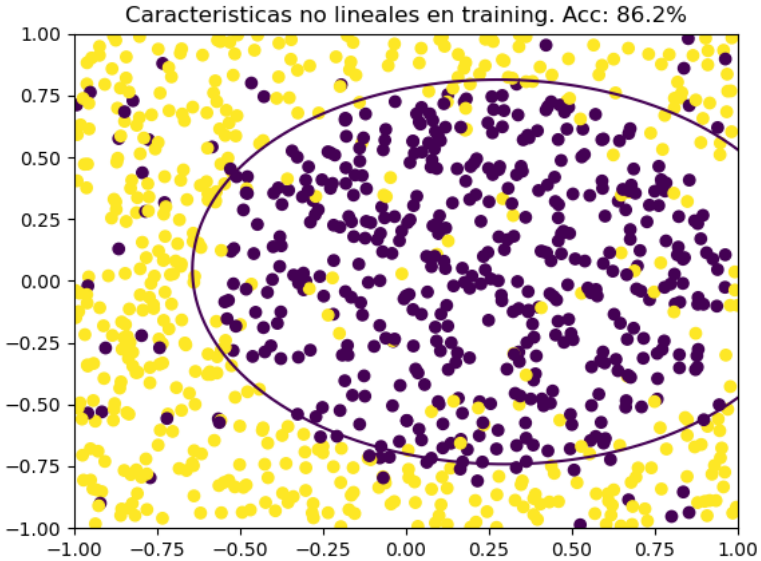
- Hablamos siempre de modelos lineales. Pero, ¿lineales en qué?
  - ¡En los pesos!
  - $w_0 + w_1x_1 + w_2x_2^2 = y$ es un modelo lineal, a pesar de que hay entradas al cuadrado! Lo que nos interesa es que los pesos no estén al cuadrado.

2. Modelos Lineales. Más ideas.

- Que un modelo sea lineal no está necesariamente ligado al hecho de que la frontera de decisión sea una línea recta.

Ejemplo: en la P1 transformamos nuestras características de entrada, y pudimos resolver satisfactoriamente un problema no linealmente separable con un modelo lineal (regresión lineal).

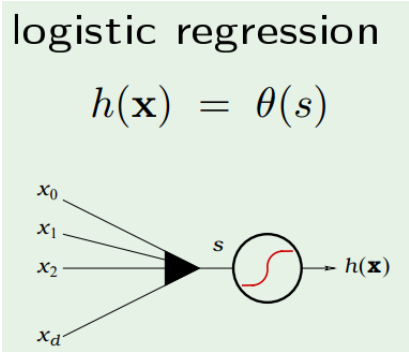
En el fondo, lo único que estábamos haciendo era $y = w_0 + w_1x_1 + w_2x_2 + w_3x_1x_2 + w_4{x_1}^2 + w_5{x_2}^2$ en lugar de $y = w_0 + w_1x_1 + w_2x_2$



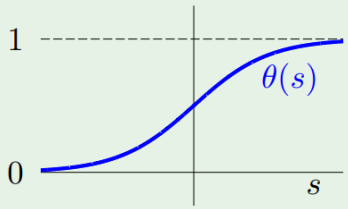
Una frontera de decisión no lineal puede ser producida por un modelo lineal.

2. Modelos Lineales. Más ideas.

- ¿Qué no sería un modelo lineal?
  - Por ejemplo, $y = w_0 \cdot x^{w_1}$
- ¿Cómo es posible que regresión logística sea un modelo lineal si usa una función de activación no lineal?

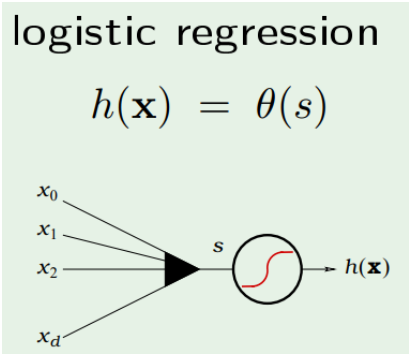


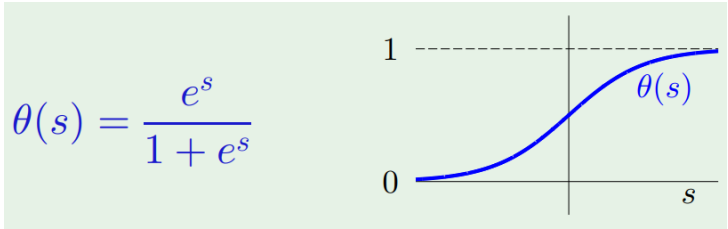
$$\theta(s) = \frac{e^s}{1+e^s}$$



2. Modelos Lineales. Más ideas.

- ¿Cómo es posible que regresión logística sea un modelo lineal si usa una función de activación no lineal?





Porque podemos escribir las predicciones en términos de una función lineal. Por ejemplo:

$$h(x) = \frac{1}{1+e^{-\boldsymbol{w}^T\boldsymbol{x}}} = 0.5 \rightarrow \ 1 = e^{-\boldsymbol{w}^T\boldsymbol{x}} \rightarrow \ln(1) = \ln(e^{-\boldsymbol{w}^T\boldsymbol{x}}) \rightarrow 0 = -\boldsymbol{w}^T\boldsymbol{x}$$

La salida siempre depende de la suma de los productos de entradas por pesos. No hay ninguna interacción del estilo $w_1x_1 \cdot w_2x_2$, que haría nuestro modelo no lineal.

2. Modelos Lineales. Más ideas.

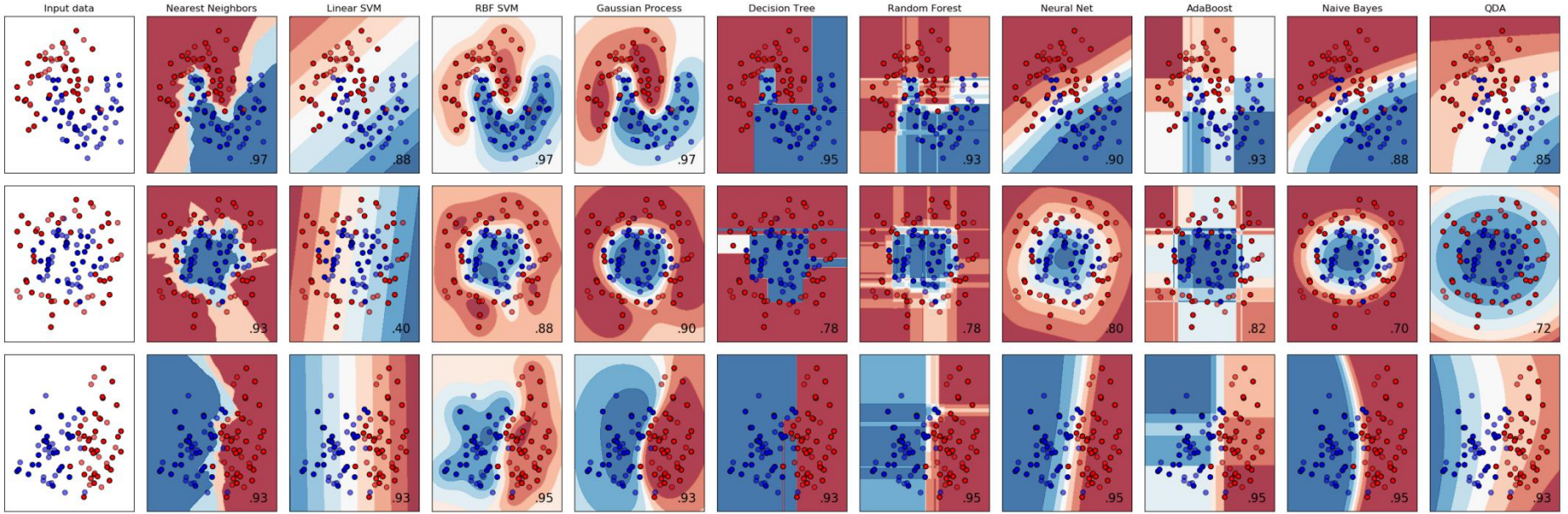
- Si regresión logística es lineal y, en cierto sentido, una red neuronal artificial es como la combinación de múltiples unidades logísticas… ¿por qué siempre se dice que una red neuronal es un modelo no lineal?
  - Porque es un modelo no lineal ☺
  - Al encadenar distintas capas de procesado (en donde la salida de una es entrada de otra) sí nos podemos encontrar con expresiones no lineales en los pesos. Ej.: red con dos capas, una única neurona por capa, sin bias y función de activación lineal.

2. Modelos Lineales. Más ideas.

- ¡Ojo!, pero ¿cómo sabemos exactamente si un modelo, sistema o función es lineal?
  - Porque verifica las propiedades de homogeneidad (o escalado) y superposición (o aditividad). En resumen, porque cumplen que $f(ax_1 + bx_2) = af(x_1) + bf(x_2)$
  - Se puede comprobar, por ejemplo que $f(w_1, w_2) = w_1 w_2 x_1$ no es lineal en los pesos, mientras que $f(w_1, w_2) = w_1 + w_2 x_1$ sí lo es.

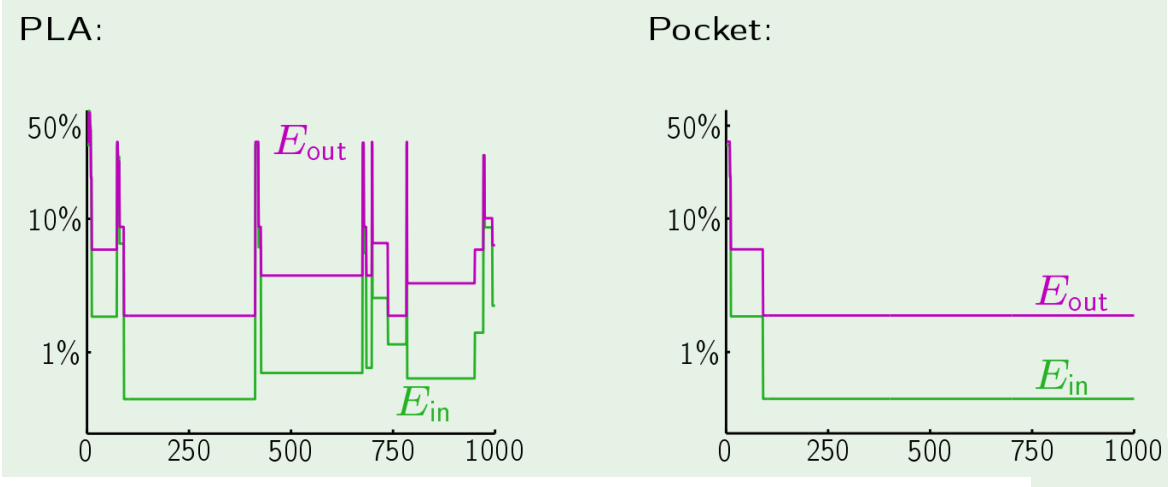
Algunos enlaces interesantes:
https://datascience.stackexchange.com/questions/12274/what-does-linear-in-parameters-mean
https://math.stackexchange.com/questions/127426/how-is-this-function-additive
https://datascience.stackexchange.com/questions/62328/regression-what-defines-linear-and-non-linear-models-or-functions

Interesante referencia para visualizar fronteras de decisión



https://scikit-learn.org/stable/auto_examples/classification/plot_classifier_comparison.html#sphx-glr-auto-examples-classification-plot-classifier-comparison-py

BONUS

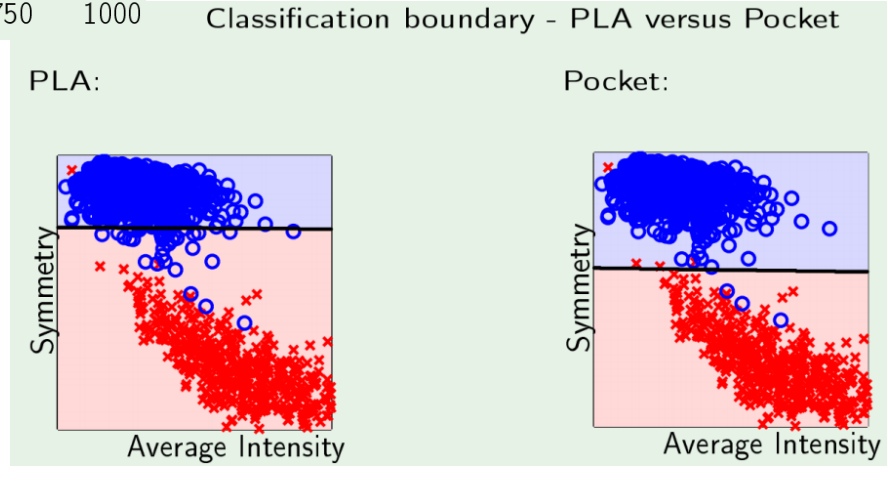


Clasificación de dígitos empleando:

- Regresión lineal
- Regresión logística
- PLA
- PLA-pocket

Y también vais a usar regresión lineal como inicialización para los otros métodos.

http://work.caltech.edu/slides/slides03.pdf slides 7 y 8



BONUS (2.d)

¿Cuál es la dimensión VC del perceptrón?

$$E_{out}(h) \leq E_{in}(h) + \sqrt{\frac{8}{N} log \frac{4 \cdot ((2N)^{d_{VC}} + 1)}{\delta}}$$

¿Cuál es la cardinalidad de H en test?

$$E_{out}(h) \leq E_{test}(h) + \sqrt{\frac{1}{2N} log \frac{2|H|}{\delta}}$$

Template

- Podéis partir, si queréis, del template que os hemos preparado
  - template_trabajo2.py

```python
# -*- coding: utf-8 -*-
"""
TRABAJO 2
Nombre Estudiante:
"""
import numpy as np
import matplotlib.pyplot as plt


# Fijamos la semilla
np.random.seed(1)


def simula_unif(N, dim, rango):
    return np.random.uniform(rango[0],rango[1],(N,dim))

def simula_gaus(N, dim, sigma):
    media = 0
    out = np.zeros((N,dim),np.float64)
    for i in range(N):
        # Para cada columna dim se emplea un sigma determinado. Es decir, para
        # la primera columna (eje X) se usará una N(0,sqrt(sigma[0]))
        # y para la segunda (eje Y) N(0,sqrt(sigma[1]))
        out[i,:] = np.random.normal(loc=media, scale=np.sqrt(sigma), size=dim)

    return out


def simula_recta(intervalo):
    points = np.random.uniform(intervalo[0], intervalo[1], size=(2, 2))
    x1 = points[0,0]
    x2 = points[1,0]
    y1 = points[0,1]
    y2 = points[1,1]
    # y = a*x + b
    a = (y2-y1)/(x2-x1) # Calculo de la pendiente.
    b = y1 - a*x1       # Calculo del termino independiente.

    return a, b


# EJERCICIO 1.1: Dibujar una gráfica con la nube de puntos de salida correspondiente

x = simula_unif(50, 2, [-50,50])
#CODIGO DEL ESTUDIANTE

x = simula_gaus(50, 2, np.array([5,7]))
#CODIGO DEL ESTUDIANTE

input("\n--- Pulsar tecla para continuar ---\n")
```