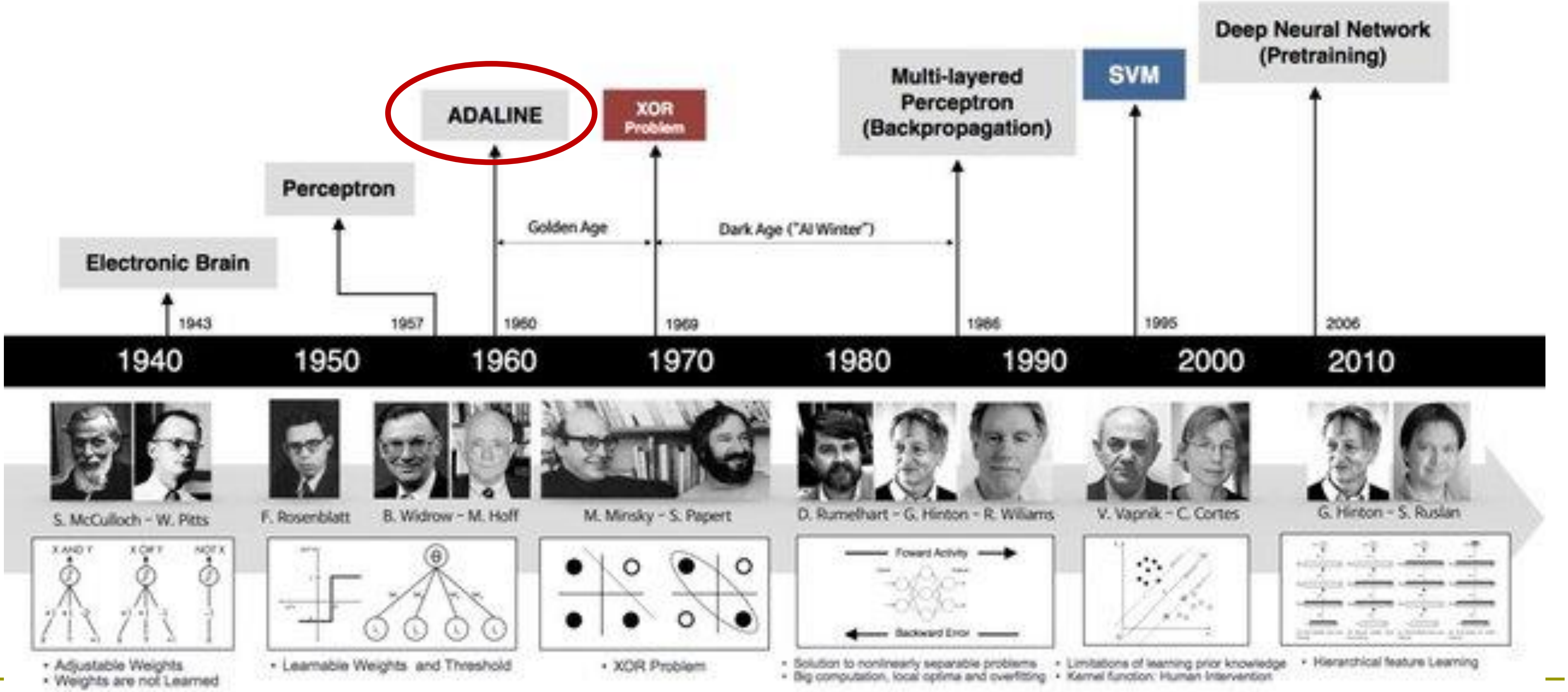
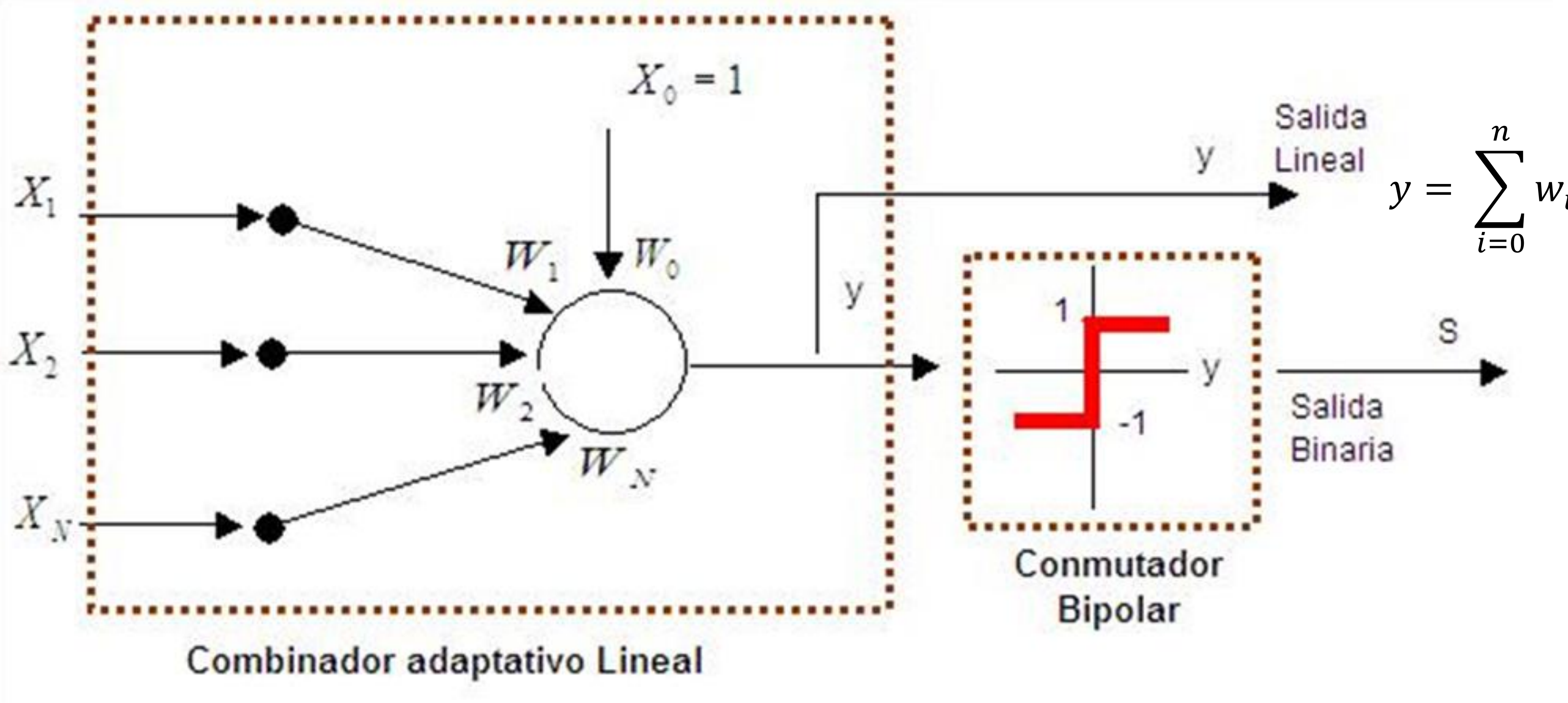
Historia de las Redes Neuronales



Red ADALINE (**ADA**ptative **LIN**ear **E**lement)

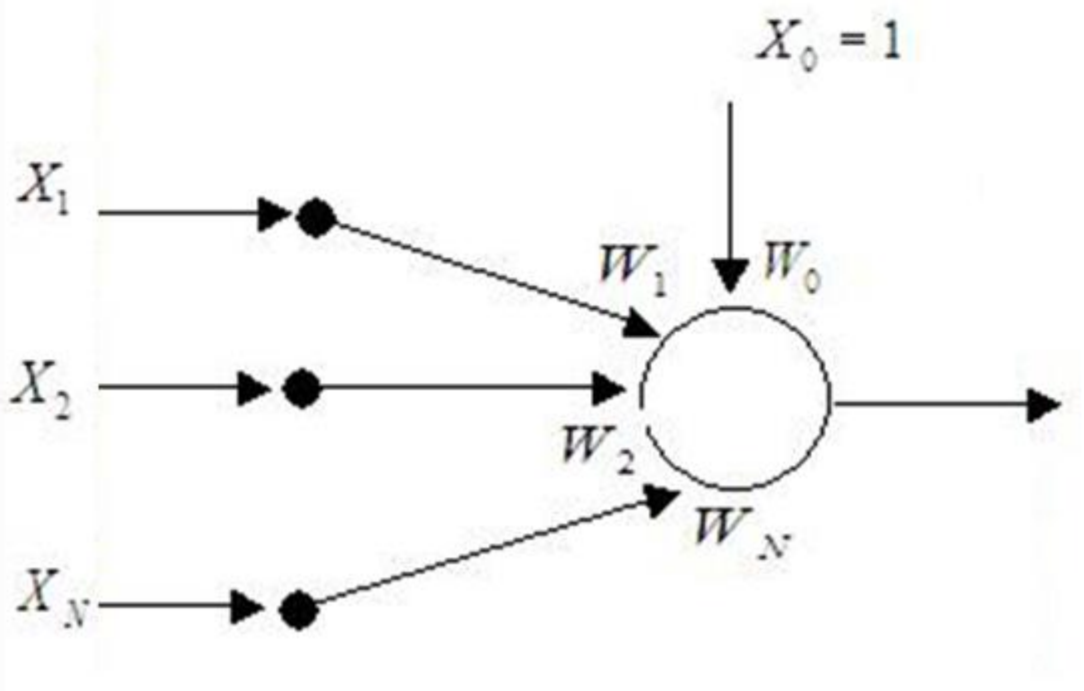
- Red neuronal formada por una sola neurona desarrollada por Widrow en 1960 al mismo tiempo que Rosenblatt trabajaba en el modelo del Perceptrón.
- Adapta los pesos de las conexiones **teniendo en cuenta el error** cometido al responder con respecto al valor esperado.
- Utiliza el **algoritmo LMS** (Least Mean Square) o **regla delta** para determinar los pesos de los arcos a fin de minimizar el error cuadrático medio.

Red ADALINE



$$y = \sum_{i=0}^{n} w_i x_i$$

Combinador Adaptativo Lineal

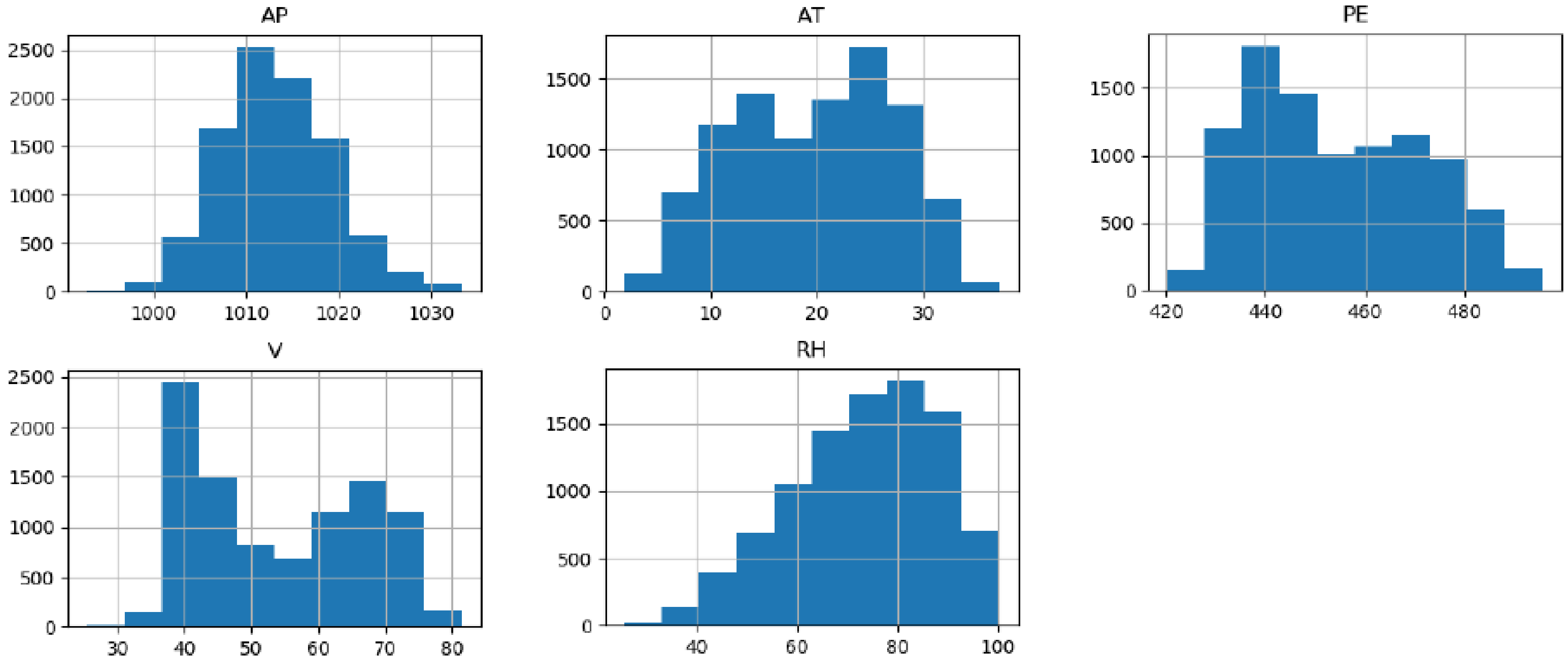


$$y = \sum_{i=0}^{n} w_i x_i$$

Ejercicio: Central de energía eléctrica

- El archivo **CCPP.csv** contiene 9568 datos de una Central de Ciclo Combinado recolectados entre 2006 y 2011.
- Objetivo: Predecir la producción neta de energía eléctrica por hora (PE) de la planta
- Las características relevadas son:
  - Temperatura ambiente (AT)
  - Presión ambiente (AP)
  - Humedad relativa (RH)
  - Vacío de escape (V)

Ejercicio: Central de energía eléctrica



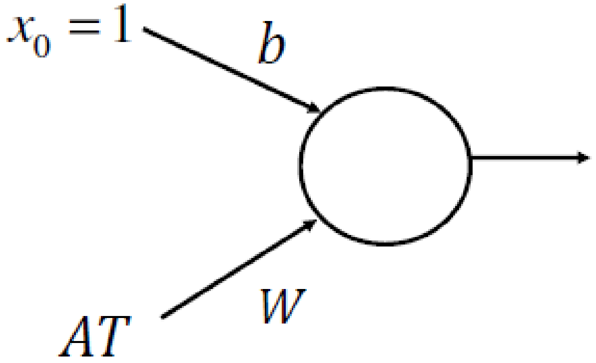
Ejercicio: Central de energía eléctrica

- Matriz de Correlación

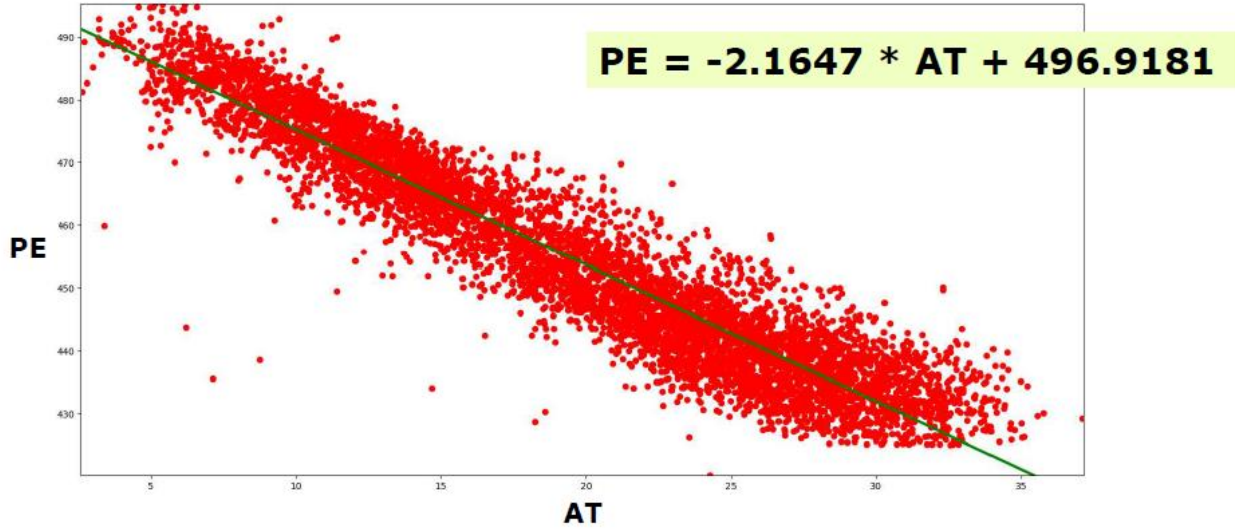
Index	AT	V	AP	RH	PE
AT	1	0.844107	-0.507549	-0.542535	-0.948128
V	0.844107	1	-0.413502	-0.312187	-0.86978
AP	-0.507549	-0.413502	1	0.0995743	0.518429
RH	-0.542535	-0.312187	0.0995743	1	0.389794
PE	-0.948128	-0.86978	0.518429	0.389794	1

EJERCICIO

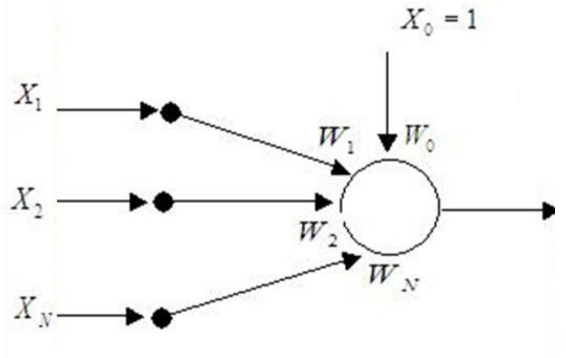
- A partir de los datos del archivo **CCPP.csv** utilice un combinador lineal para predecir la producción neta de energía eléctrica por hora (PE) de la planta a partir de la temperatura ambiente (AT)



$$PE = W * AT + b$$



Combinador Lineal



$$y = \sum_{i=0}^{n} w_i x_i$$

- Busca minimizar

$$\xi = < \varepsilon_k^2 > = {}^{1}\!/_{L} [\sum_{k=1}^{L} (d_k - \sum_{i=0}^{N} x_{ik} w_i)^2]$$

donde

L : Cantidad de ejemplos

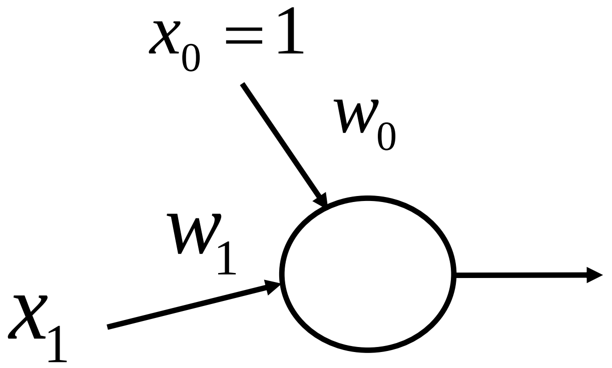
N : Cantidad de neuronas de entrada

< > representa promedio

d_k : salida esperada para el ejemplo x_k

Ejemplo: Entrene un combinador lineal utilizando los siguientes ejemplos: (2,3), (1,1), (-1,-3).

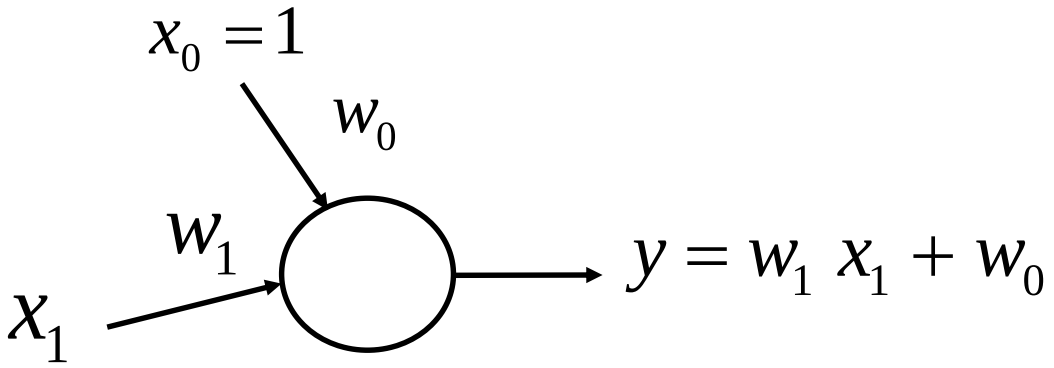
- El combinador lineal será de la forma



$$y = w_1 \, x_1 + w_0$$

Ejemplo: Entrene un combinador lineal utilizando los siguientes ejemplos: (2,3), (1,1), (-1,-3).

- El combinador lineal será de la forma

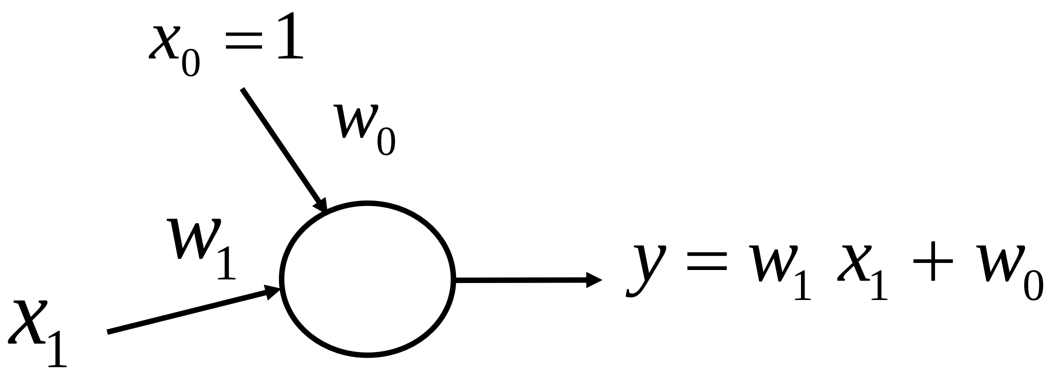


- Se busca determinar los valores de w_0 y w_1 que minimicen el error cuadrático medio

$$\xi = \frac{1}{3}\sum_{i=1}^{3}\left(d_i - y_i\right)^2$$

Ejemplo: Entrene un combinador lineal utilizando los siguientes ejemplos: (2,3), (1,1), (-1,-3).

- El combinador lineal será de la forma

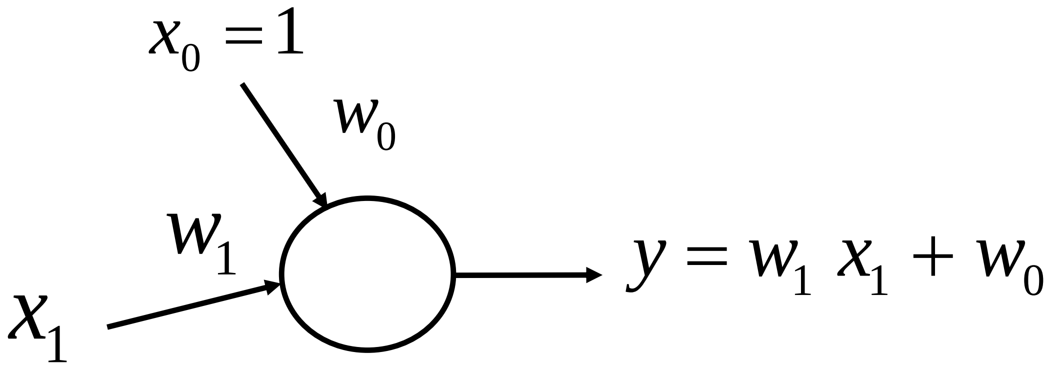


- Se busca determinar los valores de w_0 y w_1 que minimicen el error cuadrático medio

$$\xi = \frac{1}{3}\sum_{i=1}^{3}\left(d_i - y_i\right)^2 = \frac{1}{3}\sum_{i=1}^{3}\left(d_i - \sum_{j=0}^{1} w_j x_{ij}\right)^2$$

Ejemplo: Entrene un combinador lineal utilizando los siguientes ejemplos: (2,3), (1,1), (-1,-3).

- El combinador lineal será de la forma



- Se busca determinar los valores de w_0 y w_1 que minimicen el error cuadrático medio

$$\xi = \frac{1}{3}\sum_{i=1}^{3}\left(d_i - y_i\right)^2 = \frac{1}{3}\sum_{i=1}^{3}\left(d_i - \sum_{j=0}^{1} w_j x_{ij}\right)^2 = \frac{1}{3}\sum_{i=1}^{3}\left(d_i - (w_1 x_i + w_o)\right)^2$$

Ejemplo: Función de error a minimizar para los ejemplos: (2,3), (1,1), (-1,-3)

- Se busca minimizar

$$\xi = \frac{1}{3}\sum_{i=1}^{3}\left(d_i - \sum_{j=0}^{1} w_j x_{ij}\right)^2 = \frac{1}{3}\sum_{i=1}^{3}\left(d_i - (w_1 x_i + w_o)\right)^2$$

$$\xi = \frac{1}{3}\left(\left(3-(w_1 2 + w_o)\right)^2 + \left(1-(w_1 + w_o)\right)^2 + \left(-3-(w_1(-1) + w_o)\right)^2\right)$$

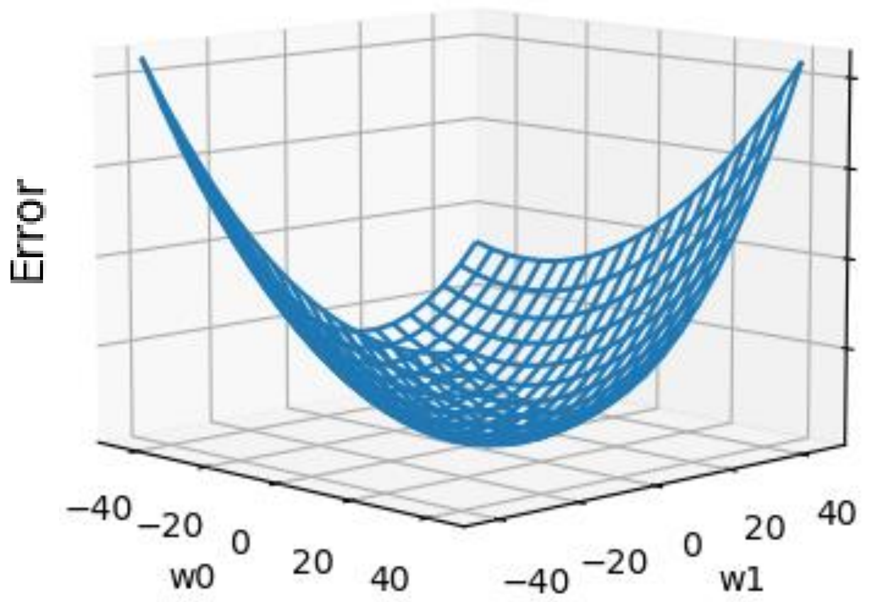
$$\xi = \frac{1}{3}(19 - 20w_1 - 2w_0 + 6w_1^2 + 4w_1 w_o + 3w_0^2)$$

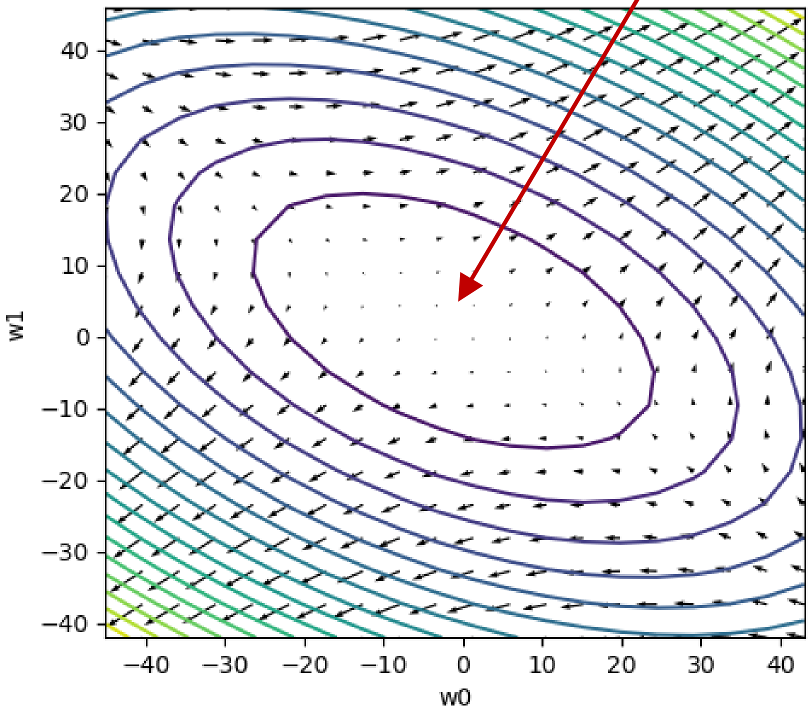
Ejemplo: Función de error a minimizar para los ejemplos: (2,3), (1,1), (-1,-3)

- Se busca minimizar la siguiente función

$$\xi = \frac{1}{3}\left(19 - 20w_1 - 2w_0 + 6w_1^2 + 4w_1w_0 + 3w_0^2\right)$$

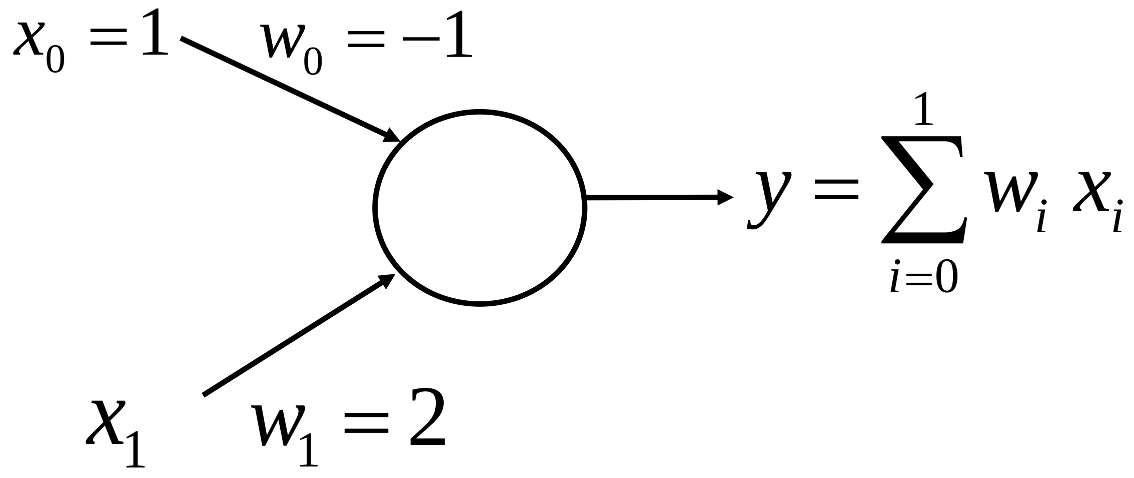
Mínimo en
w_0=-1, w_1=2

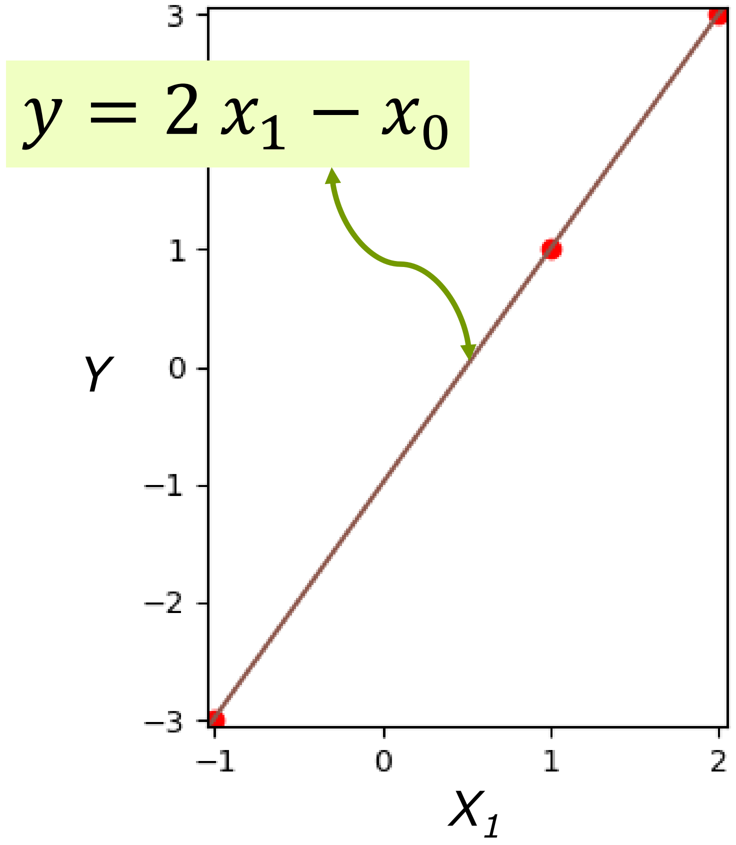




Ejemplo: Combinador lineal utilizando los siguientes ejemplos: (2,3), (1,1), (-1,-3).

- El Combinador Lineal a utilizar es el siguiente

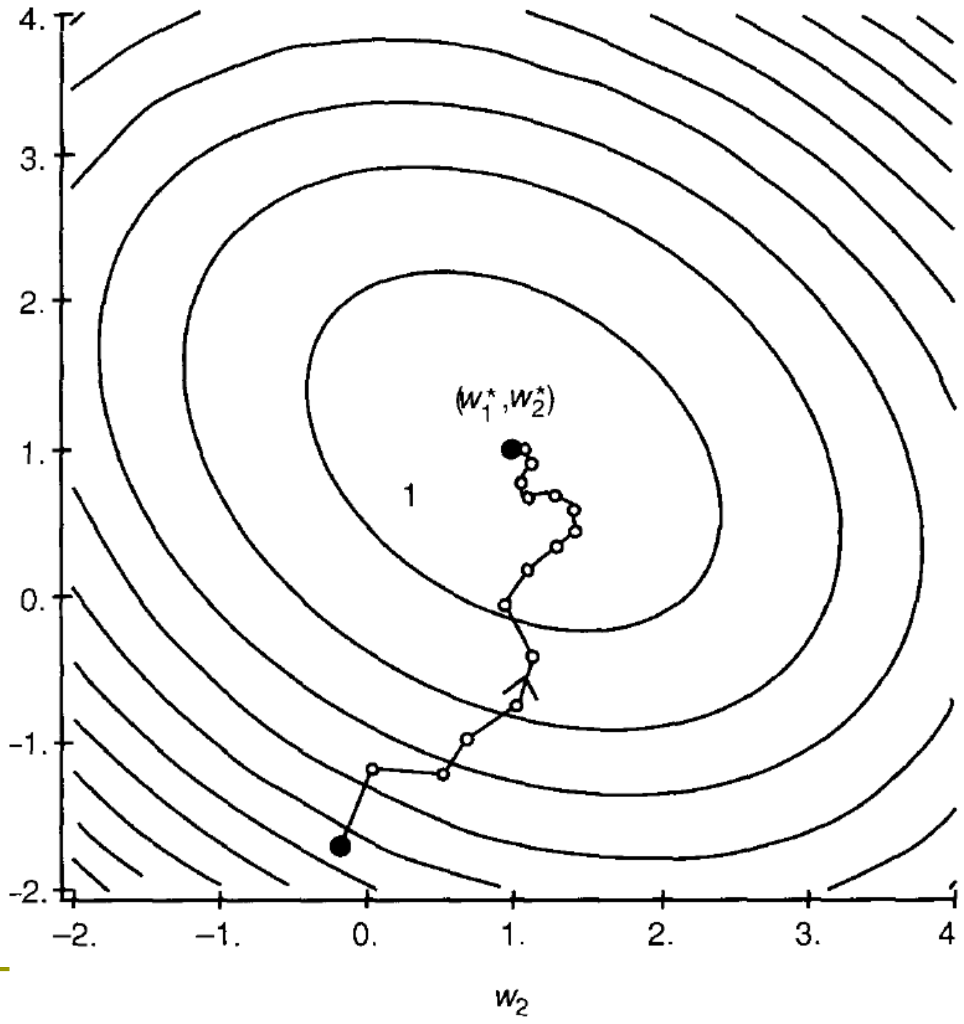




Entrenamiento del Combinador Lineal

- Se busca un método de entrenamiento que, a partir de los datos de entrada, permita calcular el vector W.

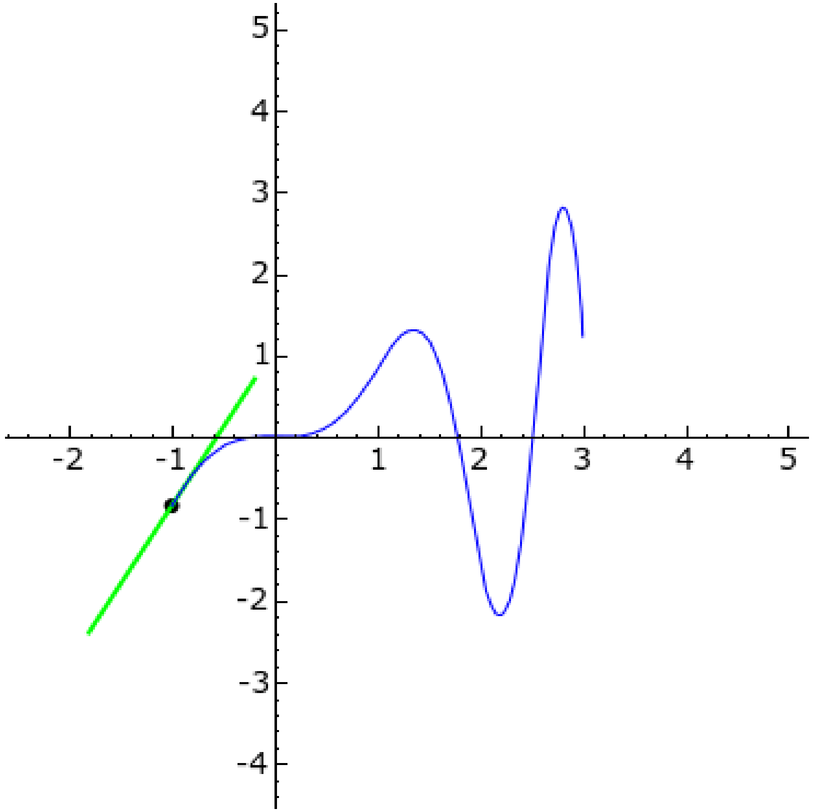
- Conceptos relacionados
  - Derivada
  - Derivada parcial
  - Vector gradiente



Derivada de una función

- Es una medida de la rapidez con la que cambia el valor de la función.

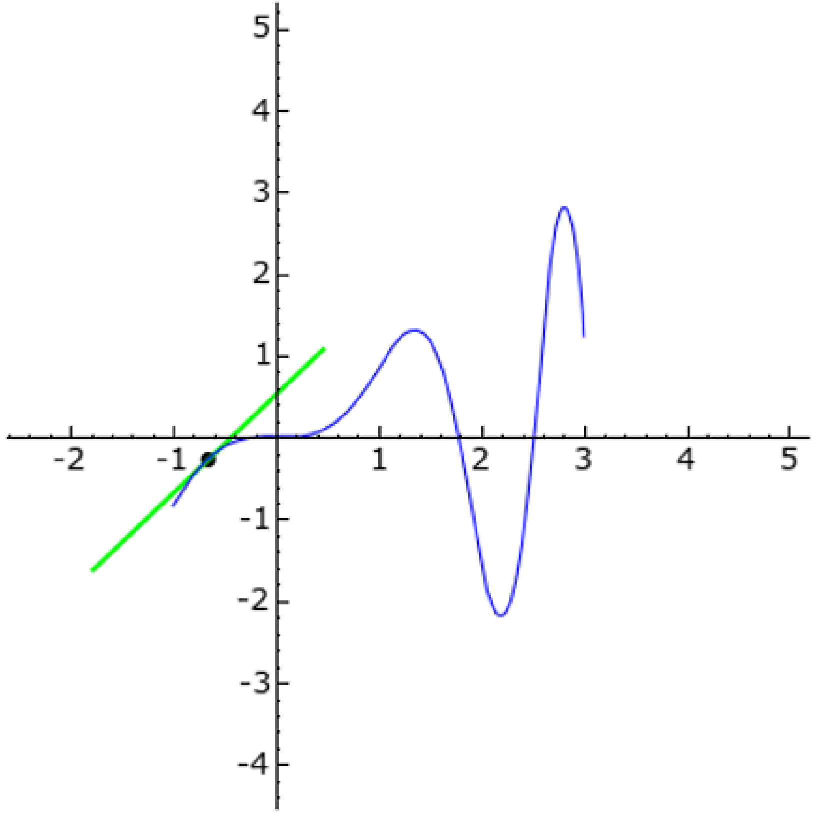
- Evaluada en un punto se corresponde con la pendiente de la recta tangente a la gráfica de la función en dicho punto.



Derivada de una función

- Es una medida de la rapidez con la que cambia el valor de la función.

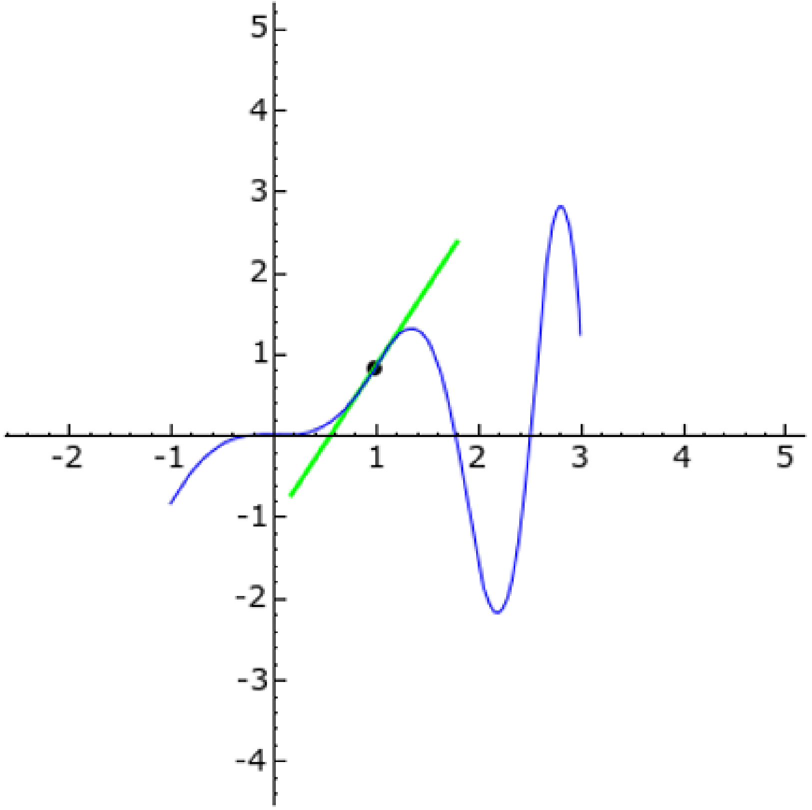
- Evaluada en un punto se corresponde con la pendiente de la recta tangente a la gráfica de la función en dicho punto.



Derivada de una función

- Es una medida de la rapidez con la que cambia el valor de la función.

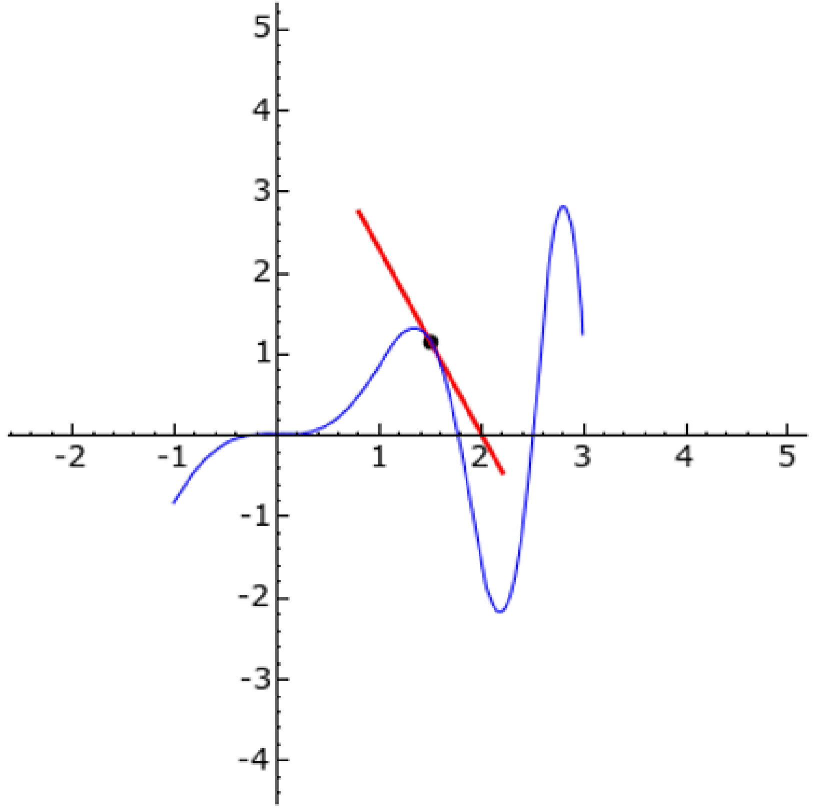
- Evaluada en un punto se corresponde con la pendiente de la recta tangente a la gráfica de la función en dicho punto.



Derivada de una función

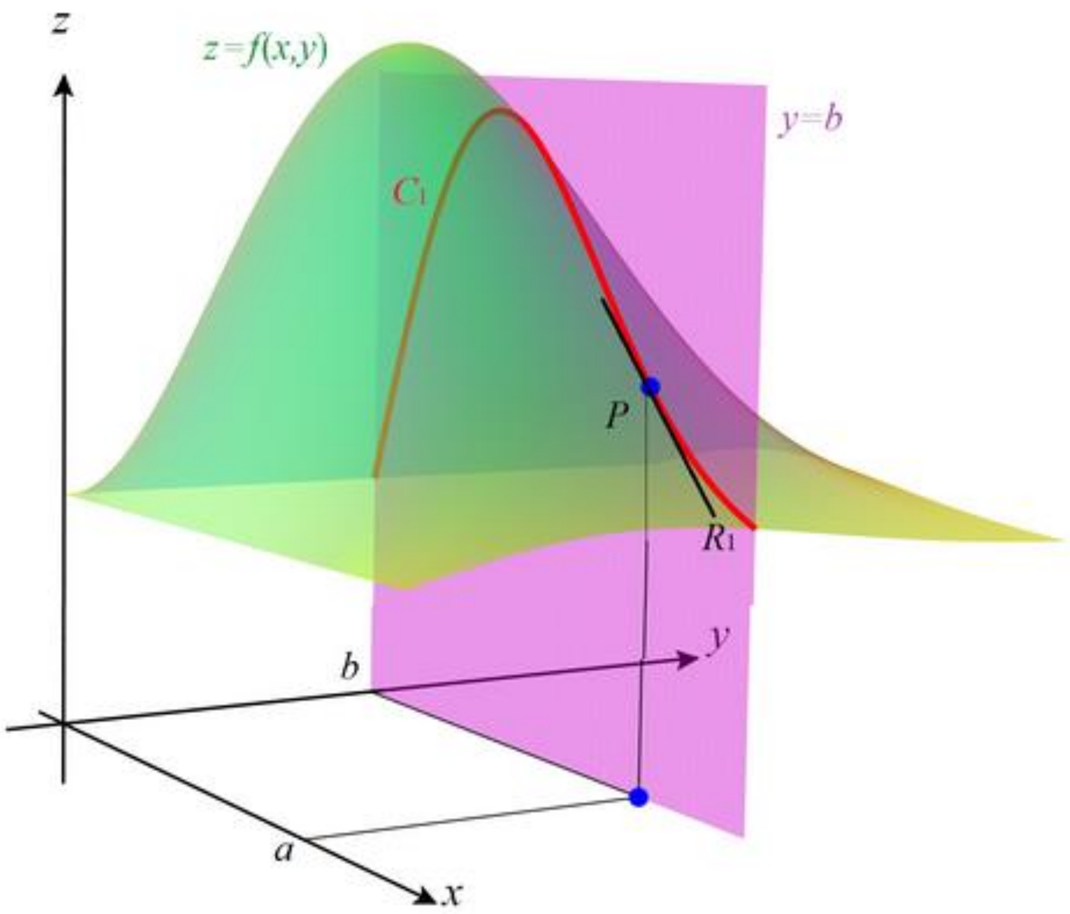
- Es una medida de la rapidez con la que cambia el valor de la función.

- Evaluada en un punto se corresponde con la pendiente de la recta tangente a la gráfica de la función en dicho punto.

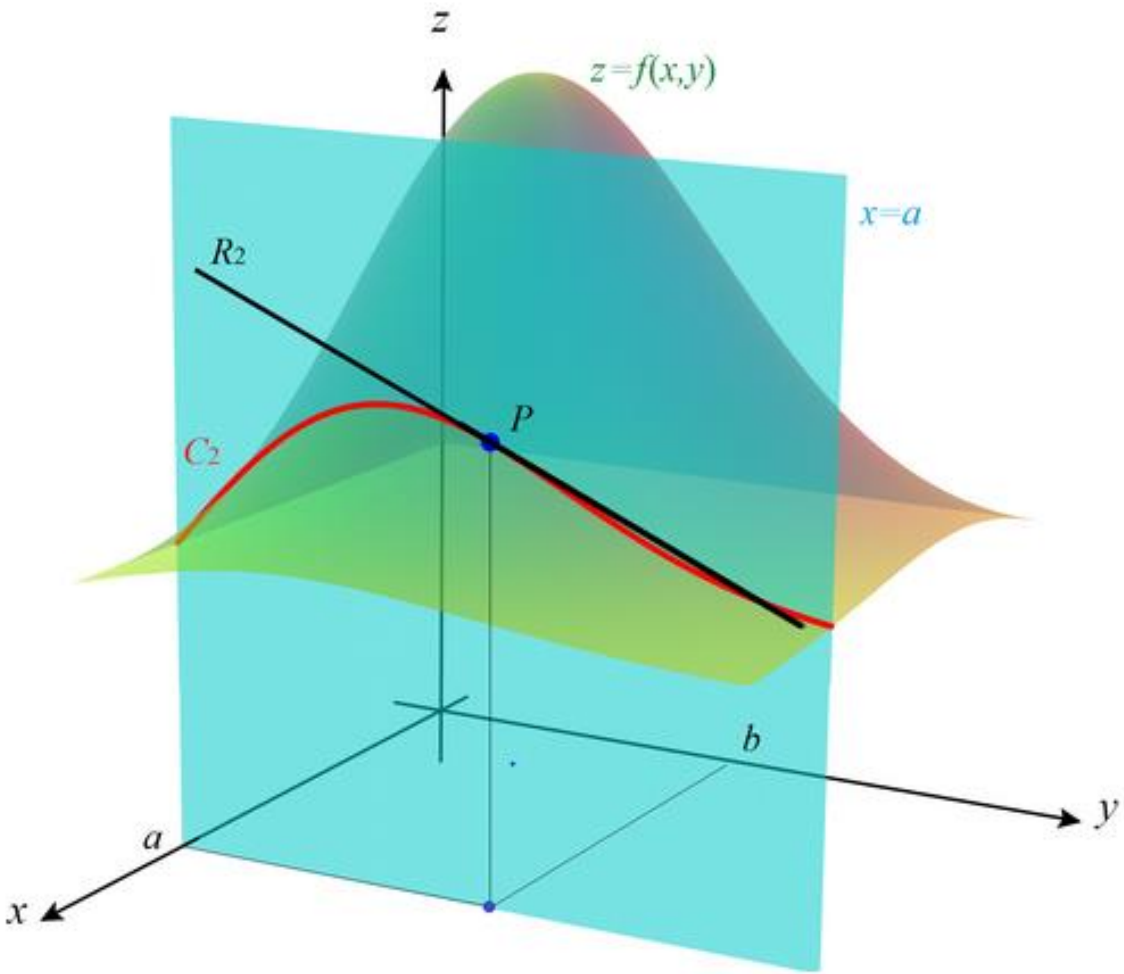


Derivadas parciales de $f(x,y)$

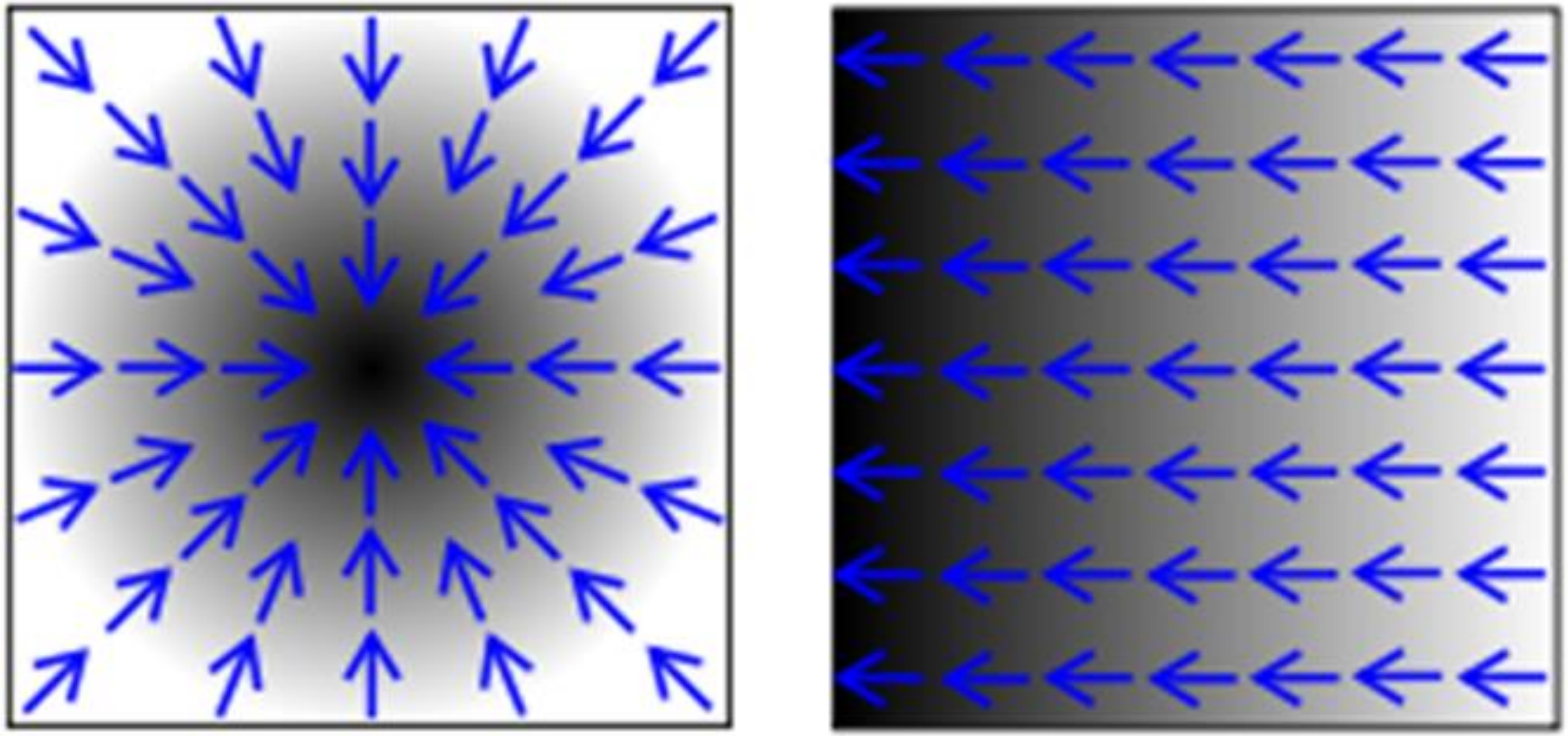
- **Con respecto a x**



- **Con respecto a y**



Gradiente



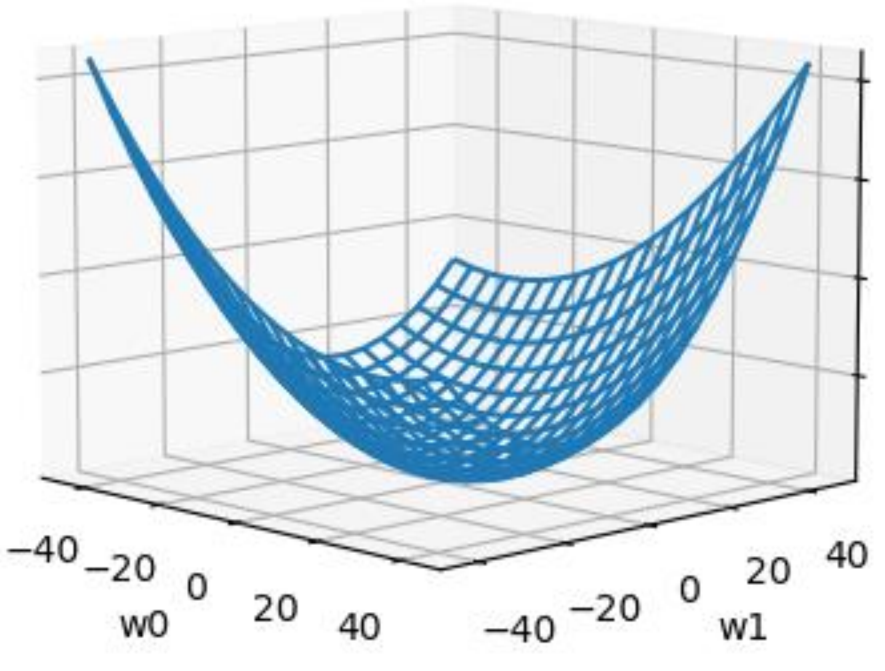
- En esta imagen, el campo escalar se aprecia en blanco y negro, los cuales representan valores bajos o altos respectivamente, y el gradiente correspondiente se aprecia por flechas azules.

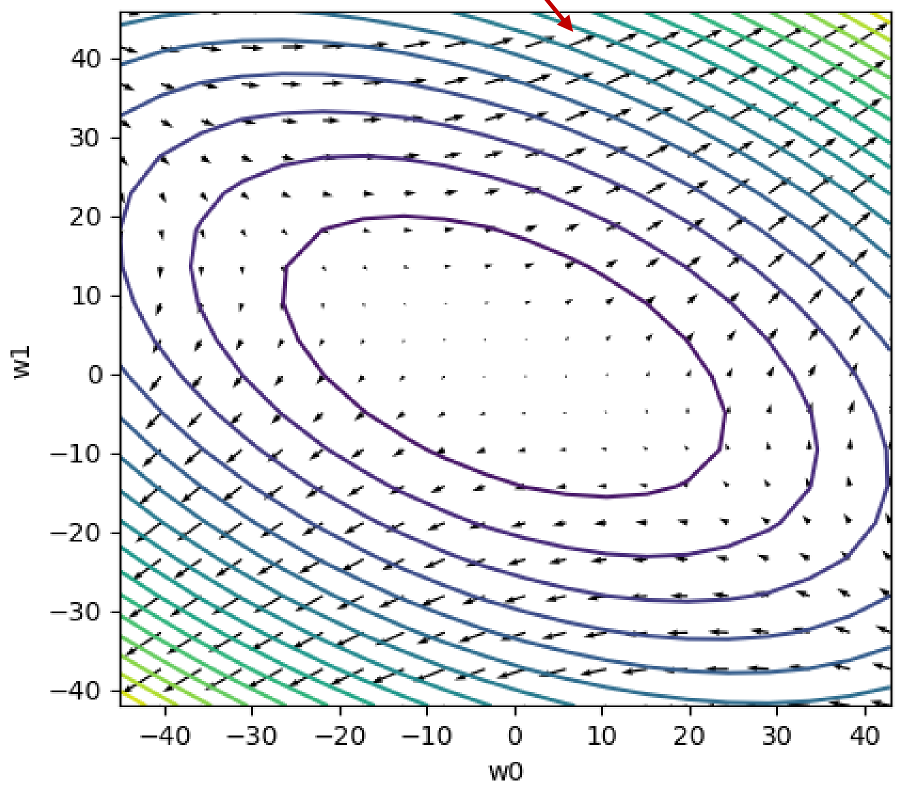
Gradiente

$$\nabla \xi = \left(\frac{\partial \xi}{\partial w_0}, \frac{\partial \xi}{\partial w_1} \right)$$

$$\xi = \frac{1}{3}\left(19 - 20w_1 - 2w_0 + 6w_1^2 + 4w_1 w_0 + 3w_0^2\right)$$

Vector numérico que resulta de evaluar las derivadas parciales en un punto dado de la función





Gradiente

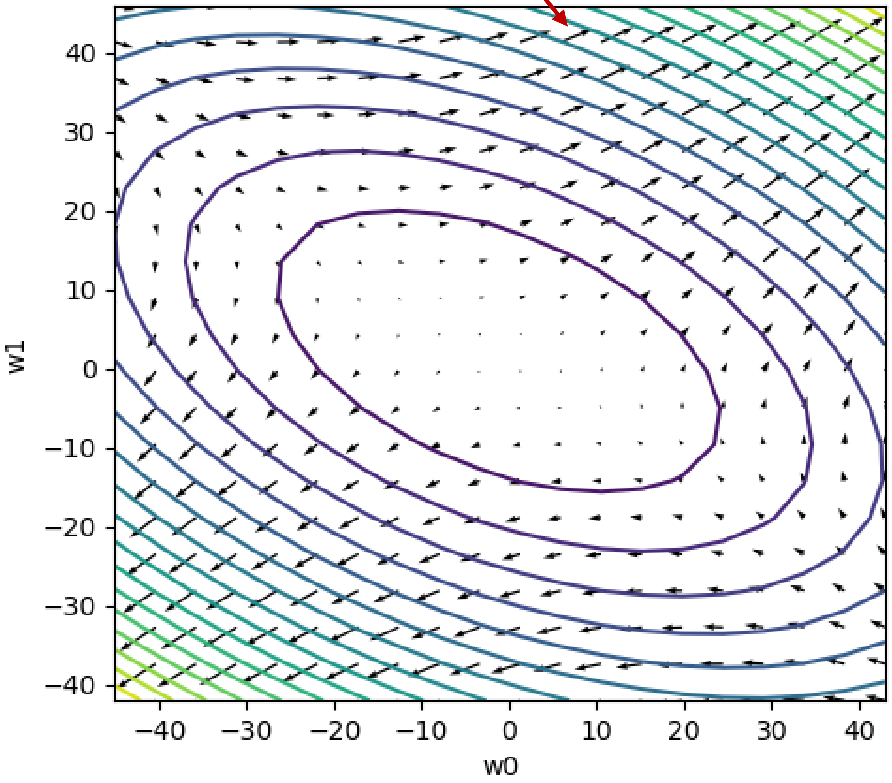
$$\xi = \frac{1}{3}\left(19 - 20w_1 - 2w_0 + 6w_1^2 + 4w_1w_0 + 3w_0^2\right)$$

$$\frac{\partial \xi}{\partial w_0} = \frac{1}{3}\left(-2 + 4w_1 + 6w_0\right)$$

$$\frac{\partial \xi}{\partial w_1} = \frac{1}{3}\left(-20 + 12w_1 + 4w_0\right)$$

$$\nabla \xi = \left(\frac{\partial \xi}{\partial w_0}, \frac{\partial \xi}{\partial w_1}\right) = \left(\frac{-2 + 4w_1 + 6w_0}{3}, \frac{-20 + 12w_1 + 4w_0}{3}\right)$$

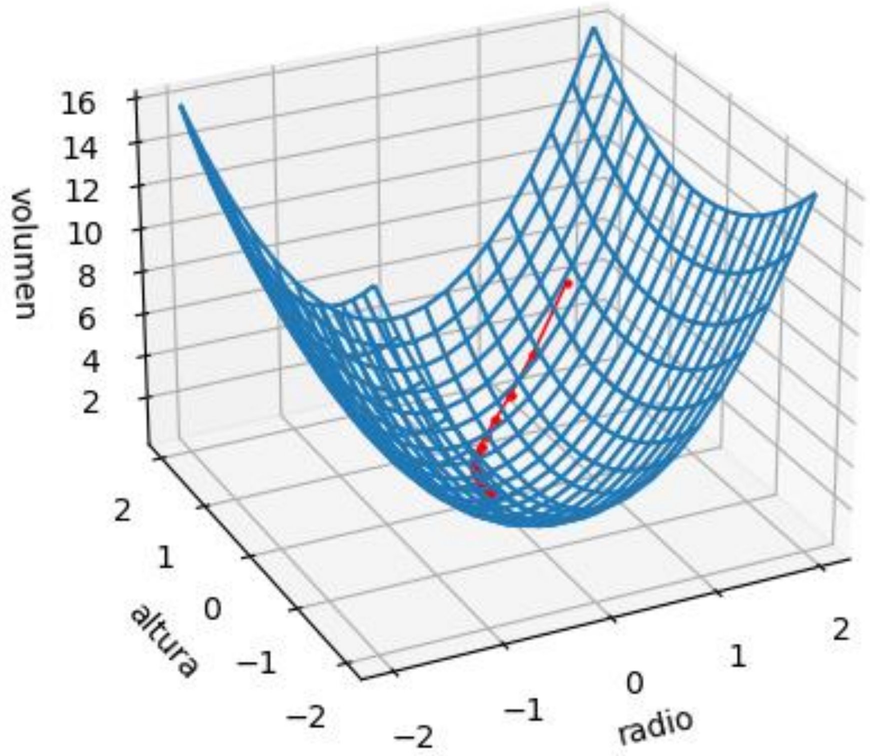
Vector numérico que resulta de evaluar las derivadas parciales en un punto dado de la función

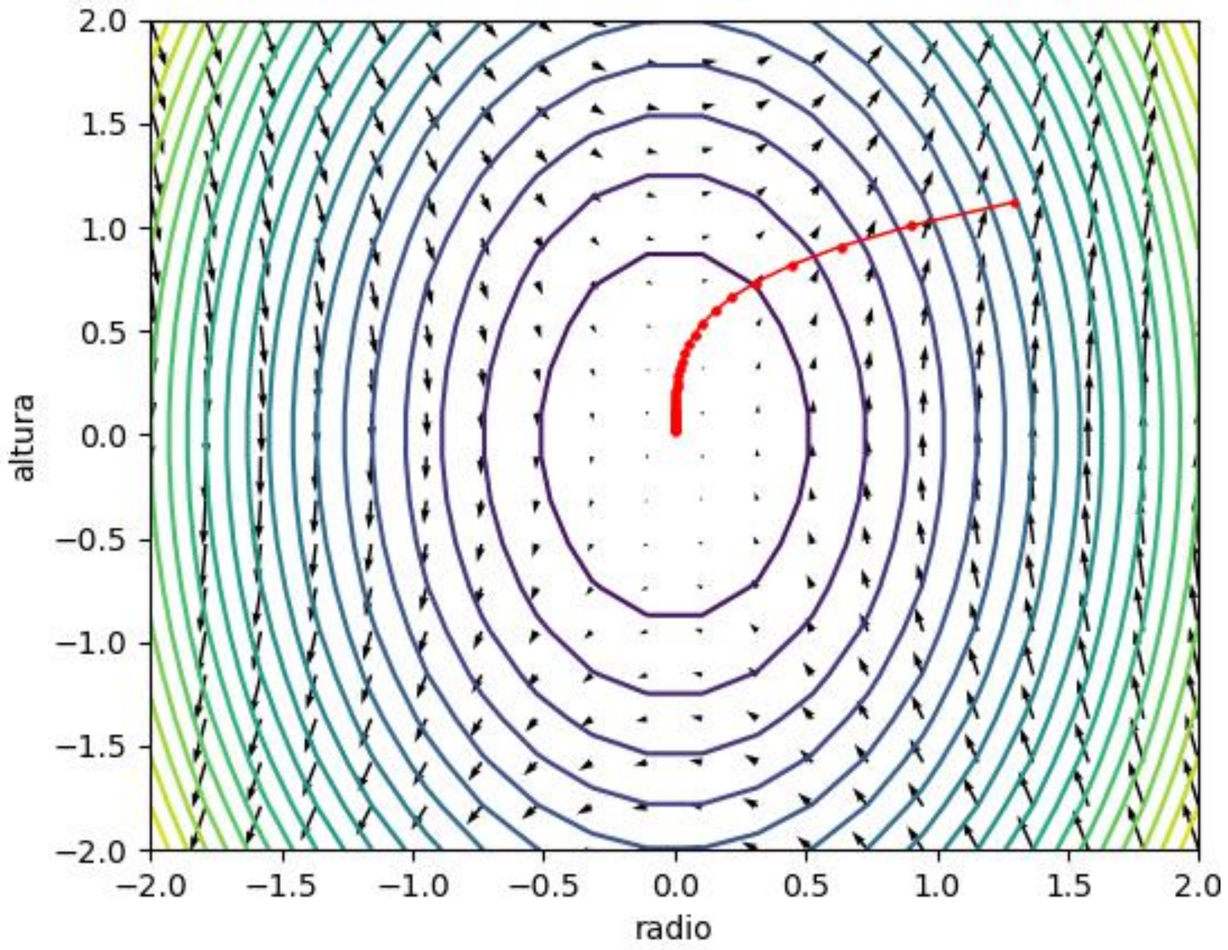


Minimización de funciones usando el gradiente

- Dada una función continua
  - Tomar un punto dentro del dominio de la función.
  - Calcular el vector gradiente de la función en ese punto.
  - Sumarle al punto anterior una fracción del gradiente negativo (para ir hacia el mínimo).
  - Repetir los dos pasos anteriores hasta que la diferencia entre evaluaciones consecutivas de la función sea inferior a una cierta cota.

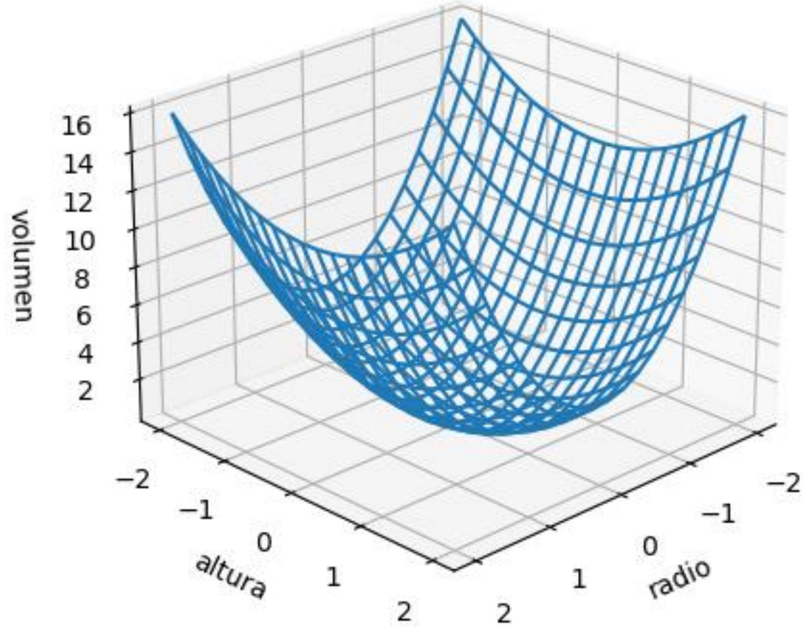
Ejemplo: Minimizar $f(x,y) = 3x^2+y^2$

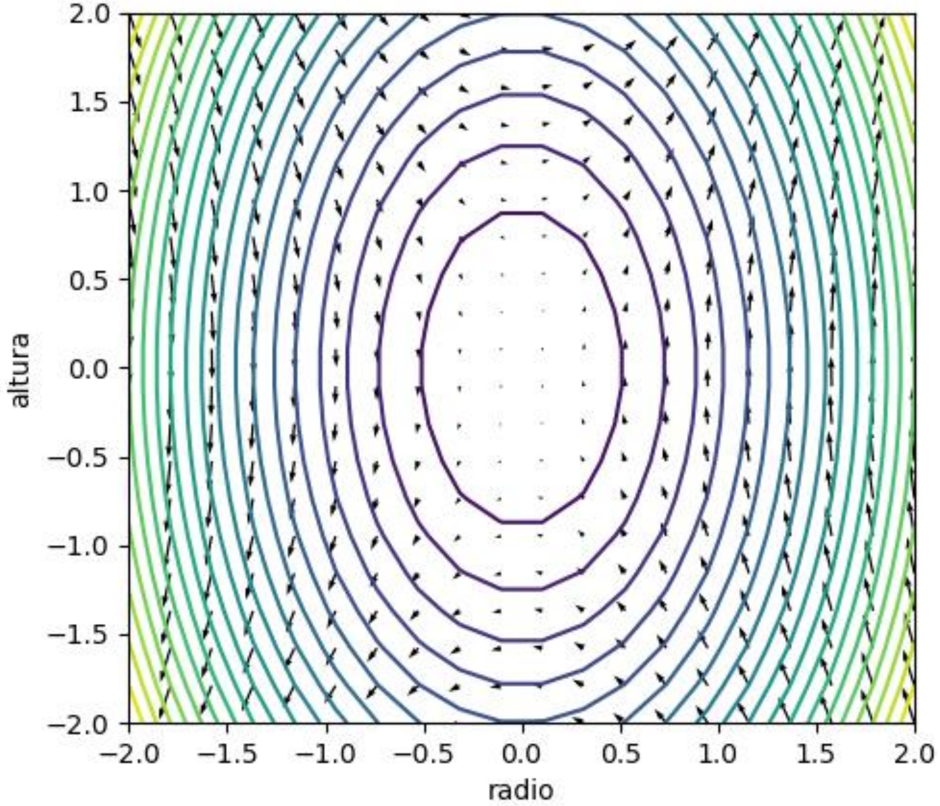




Función 1: $f(x,y) = 3x^2+y^2$

- Utilice **[x, y, h] = graficoGradiente(1)** para visualizar la función y elegir el pto.inicial



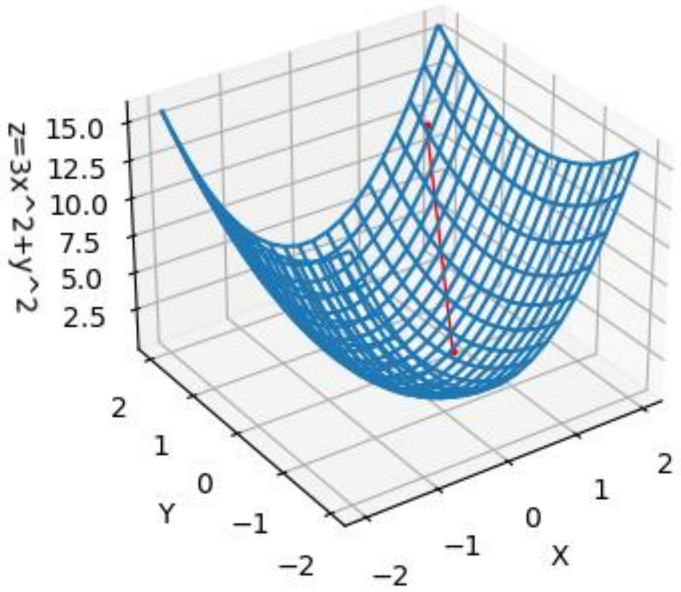


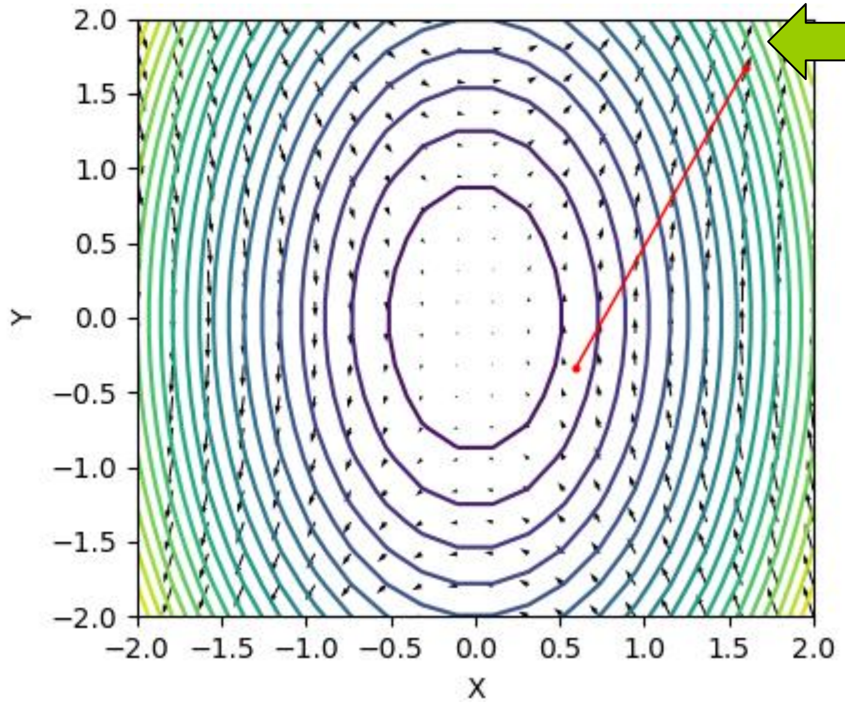
Función 1: desplazamiento en la figura

```
[x, y, h] = graficoGradiente(1)
z = 3*x**2 + y**2
PtoAnt = [x, y, z]
x = x-1
y = y-2  #--- cambiamos x e y
z = 3*x**2 + y**2;
graficarPaso(PtoAnt, [x, y, z], h)
```

gradienteFuncion_enClase.py

Función 1: $f(x,y) = 3x^2+y^2$



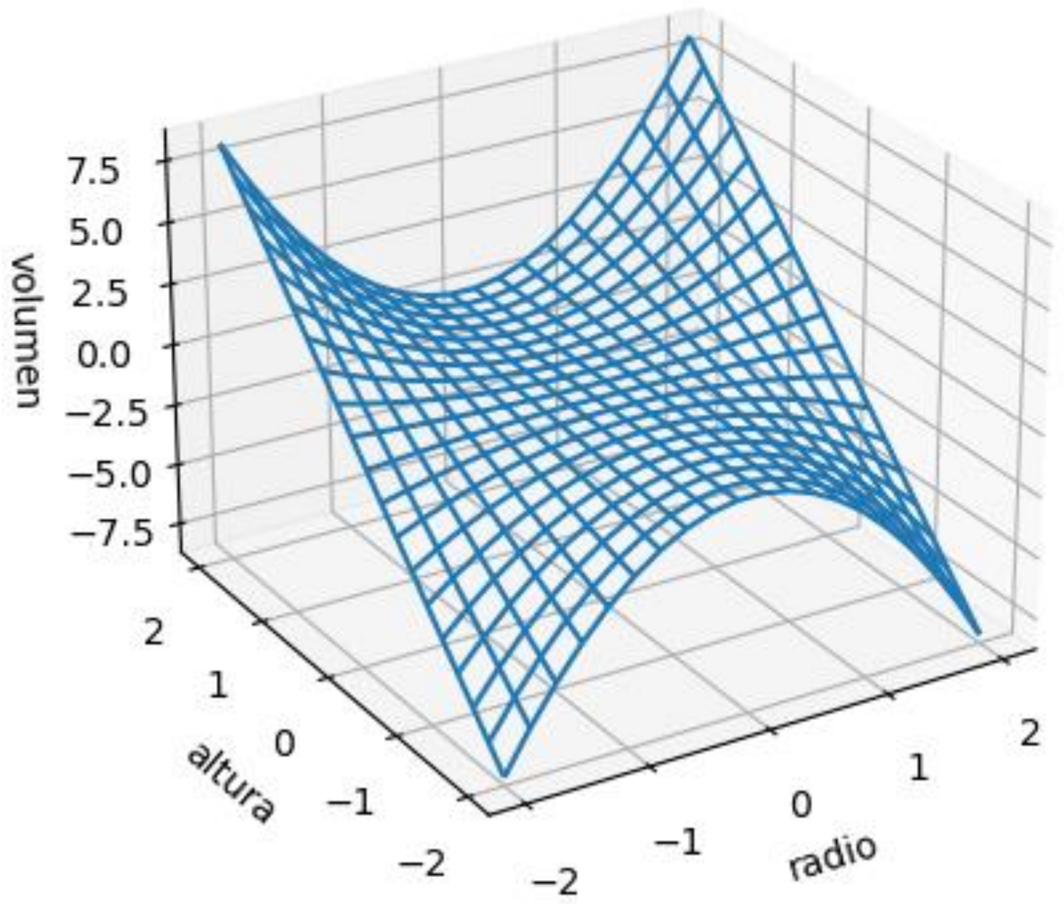


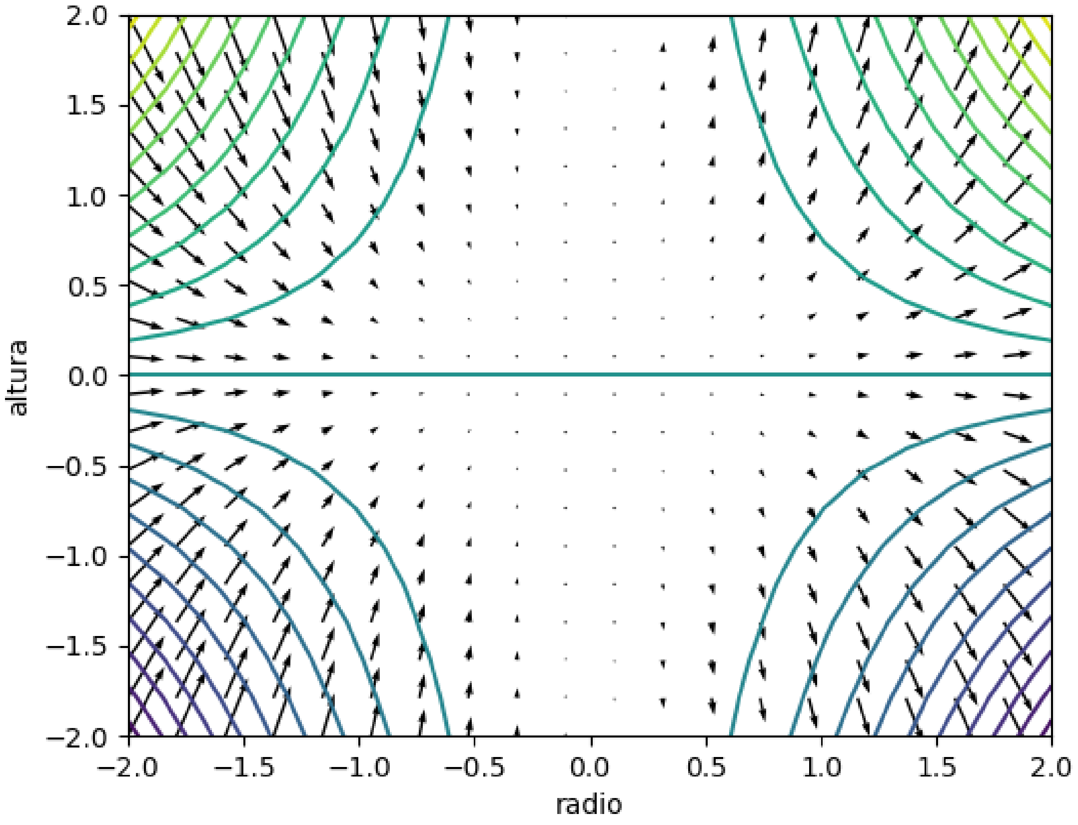
Posición inicial seleccionada

Escriba el código Python para minimizar f(x,y) usando la técnica de descenso por gradiente.

Función 2: Volumen del cono

$$V(r,h)=\frac{r^2h\pi}{3}$$

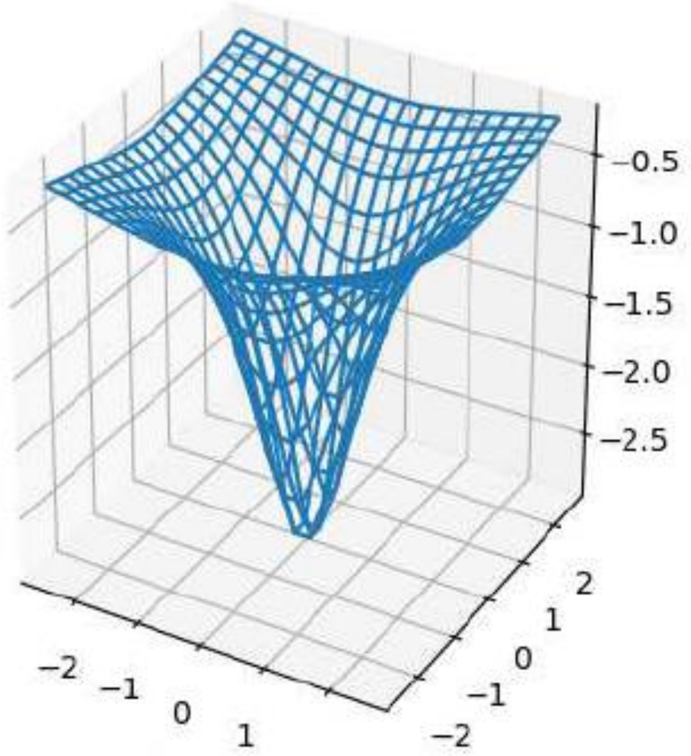




Función 3

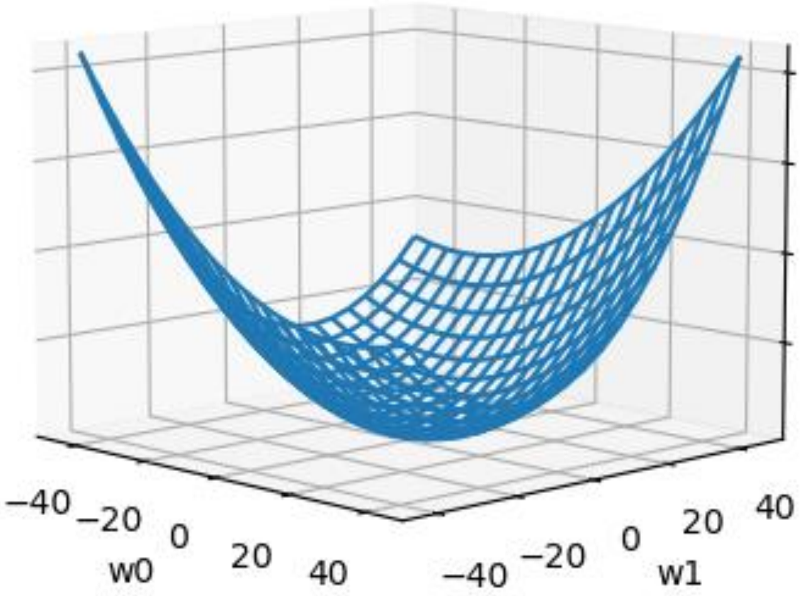
- Utilice la técnica de descenso por gradiente para hallar el mínimo de la función

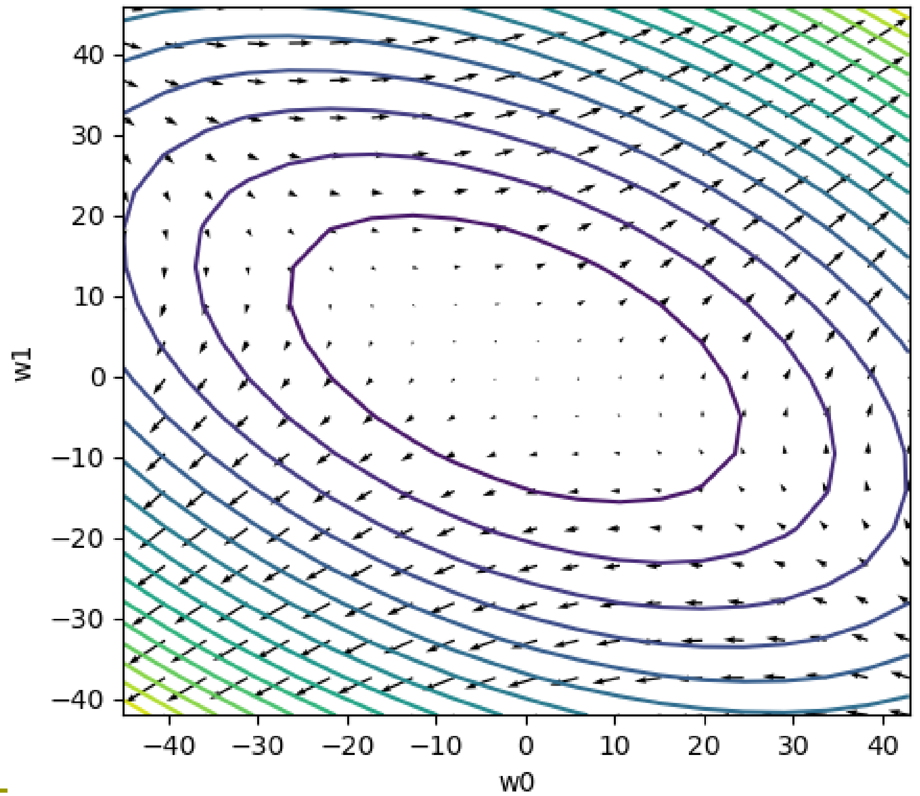
$$f(x,y) = \frac{-3}{x^2 + y^2 + 1}$$



Función 4: Error cuadrático medio

$$\xi = \frac{1}{3}\left(\left(3-2w_1-w_0\right)^2+\left(1-w_1-w_o\right)^2+\left(-3+w_1-w_0\right)^2\right)$$





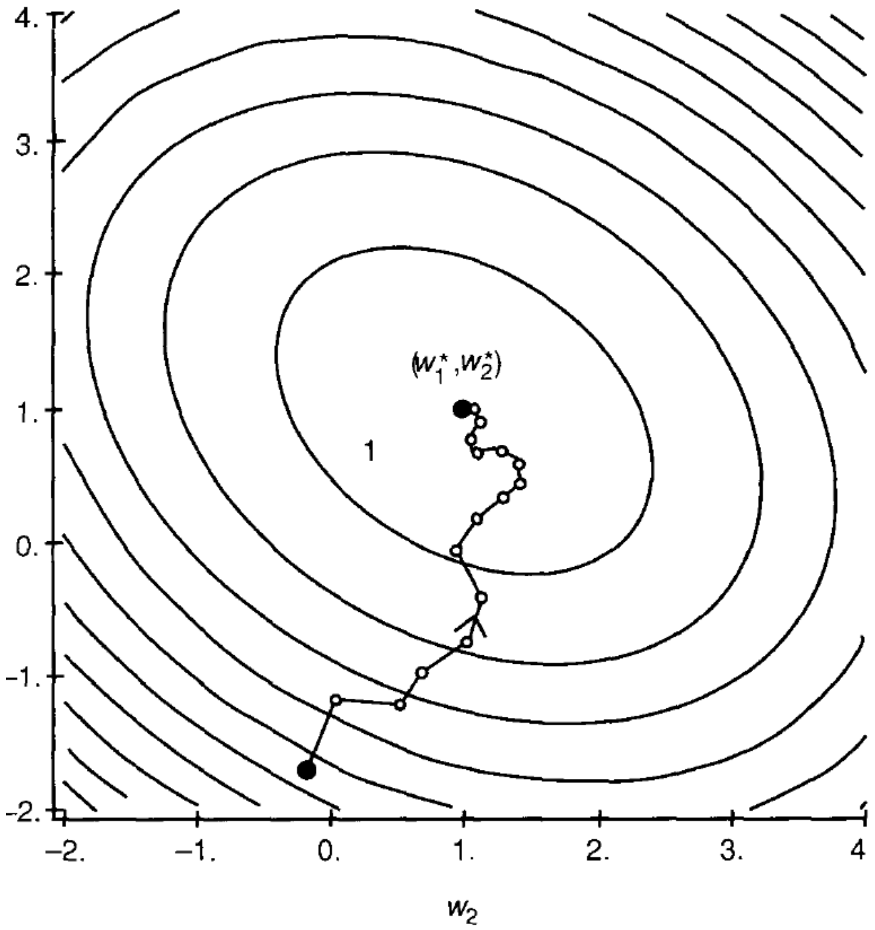
Técnica del descenso del gradiente estocástico

$$w(t+1) = w(t) + \Delta w(t)$$

$$w(t+1) = w(t) - \mu \nabla \xi(w(t))$$

- se utiliza

$$\xi = < \varepsilon_k^2 > \approx \varepsilon_k^2 = (d_k - \sum_{i=0}^{N} x_{ik} w_i)^2$$



Técnica del descenso del gradiente estocástico

$$w(t+1) = w(t) + \Delta w(t)$$

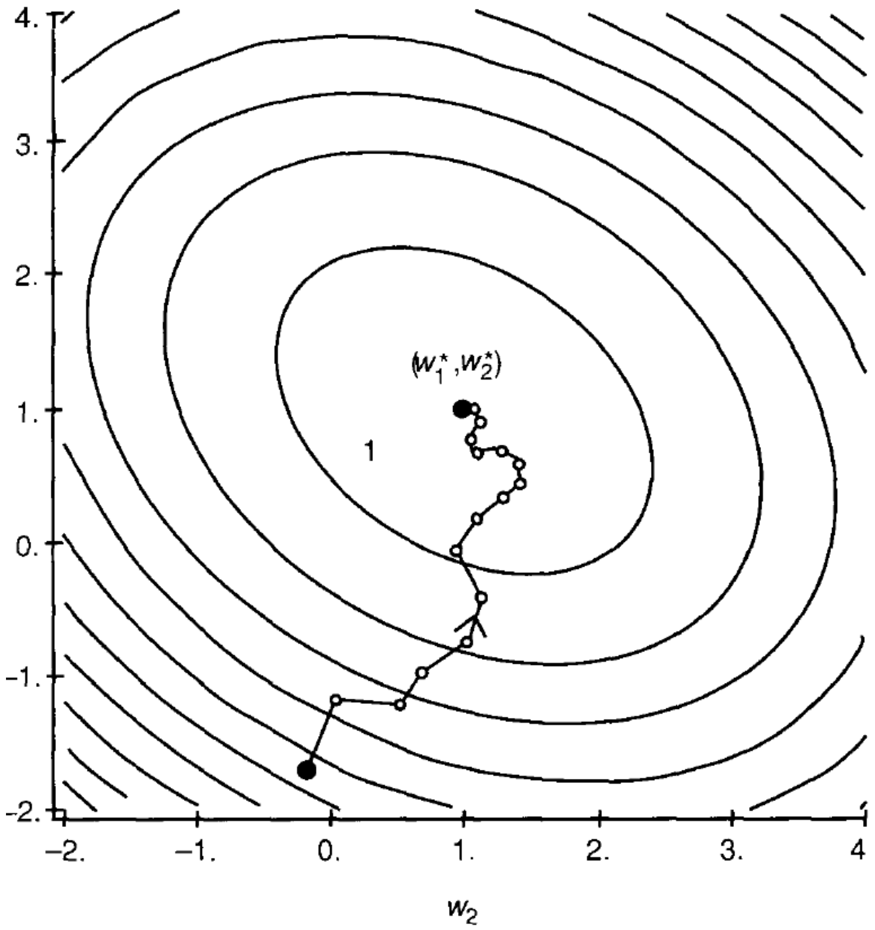
$$w(t+1) = w(t) - \mu \nabla \xi(w(t))$$

- se utiliza

$$\xi = < \varepsilon_k^2 > \approx \varepsilon_k^2 = (d_k - \sum_{i=0}^{N} x_{ik} w_i)^2$$

- veamos que

$$\nabla \varepsilon_k^2(t) = -2\varepsilon_k(t) x_k$$



Gradiente del error en el ejemplo x_k

$$\nabla \varepsilon_k^2(t) = \frac{\partial \varepsilon_k^2}{\partial w} = \left[\frac{\partial (d_k - y_k)^2}{\partial w_0}; \quad \ldots; \quad \frac{\partial (d_k - y_k)^2}{\partial w_n} \right]$$

$$\nabla \varepsilon_k^2(t) = \frac{\partial \varepsilon_k^2}{\partial w} = \left[-2(d_k - y_k)\frac{\partial y_k}{\partial w_0}; \quad \ldots; \quad -2(d_k - y_k)\frac{\partial y_k}{\partial w_n} \right]$$

$$\nabla \varepsilon_k^2(t) = \frac{\partial \varepsilon_k^2}{\partial w} = -2e_k \left[\frac{\partial y_k}{\partial w_0}; \quad \ldots; \quad \frac{\partial y_k}{\partial w_n} \right]$$

$$\nabla \varepsilon_k^2(t) = \frac{\partial \varepsilon_k^2}{\partial w} = -2e_k \left[x_{0k}, \quad x_{1k}, \quad \ldots, \quad x_{nk} \right] = -2e_k x_k$$

Entrenamiento del combinador lineal

- Para cada vector de entrada
  - Aplicar el vector de entrada, x_k
  - Calcular el gradiente utilizando

$$\nabla < \varepsilon_k^2 > \approx \nabla \varepsilon_k^2(t) = -2\varepsilon_k(t)x_k = -2(d_k - y_k)x_k$$

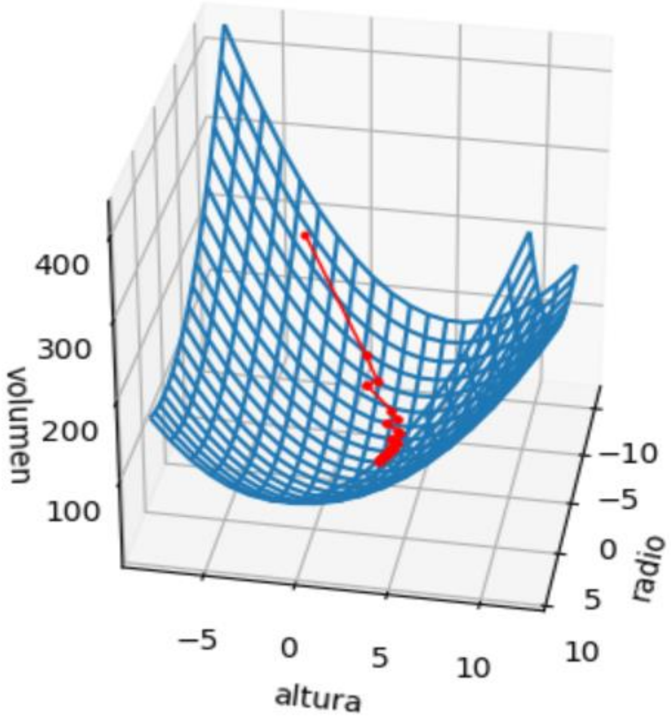
  - Actualizar el vector de pesos

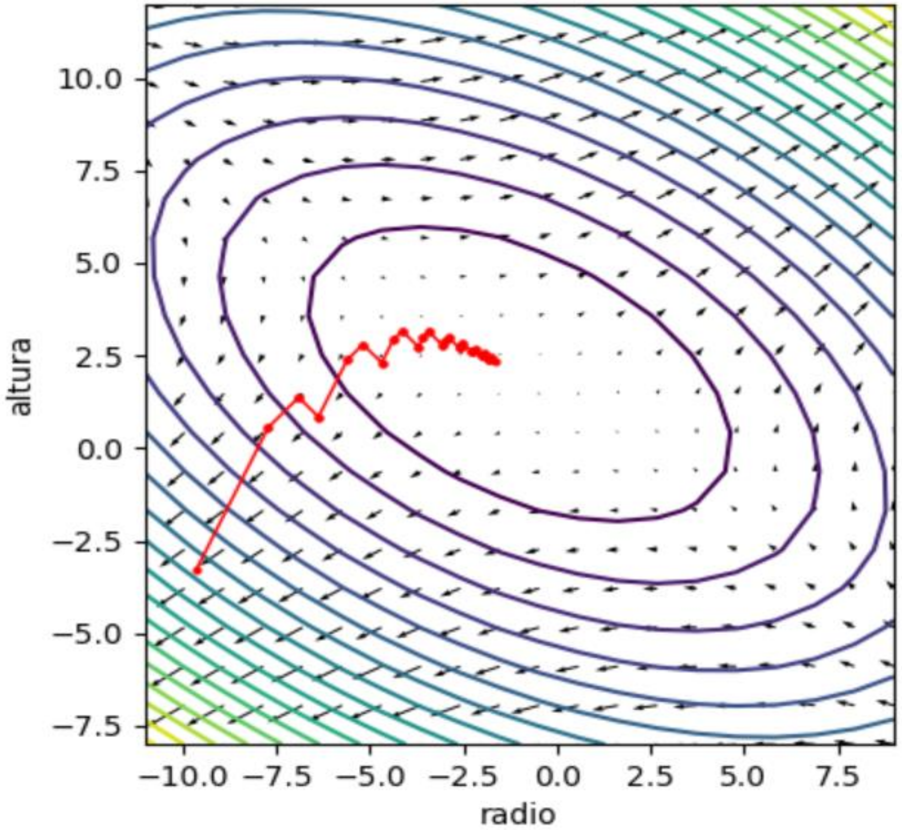
$$w(t+1) = w(t) + 2\mu(d_k - y_k)x_k$$

- Repetir todo hasta que el error sea aceptable

Minimización de la función de error usando el descenso de gradiente

gradienteFuncion4_CombinadorLineal.py





Ejecutar con distintos valores de alfa (velocidad de aprendizaje)

ClassNeuronaLineal.py

nl = NeuronaLineal(alpha=0.01, n_iter=50, cotaE=10E-07, random_state=None, draw=0, title=['X1','X2'])

- Parámetros de entrada
  - **alpha**: valor en el intervalo (0, 1] que representa la velocidad de aprendizaje.
  - **n_iter**: máxima cantidad de iteraciones a realizar.
  - **cotaE**: termina si la diferencia entre dos errores consecutivos es menor que este valor.
  - **random_state**: None si los pesos se inicializan en forma aleatoria, un valor entero para fijar la semilla
  - **draw**: valor distinto de 0 si se desea ver el gráfico y 0 si no. Sólo si es 2D.
  - **title**: lista con los nombres de los ejes para el gráfico. Se usa sólo si **draw** no es cero.

ClassPerceptron.py

```
nl = NeuronaLineal(alpha=0.01, n_iter=50)
nl.fit(X, T)
```

- Parámetros de entrada
  - **X** : arreglo de NxM donde N es la cantidad de ejemplos y M la cantidad de atributos.
  - **T** : arreglo de N elementos siendo N la cantidad de ejemplos
- Retorna
  - **w_** : arreglo de M elementos siendo M la cantidad de atributos de entrada
  - **b_** : valor numérico continuo correspondiente al bias.
  - **errors_**: errores cometidos en cada iteración.

ClassPerceptron.py

Y = nl.predict(X)

- Parámetros de entrada
  - **X** : arreglo de NxM donde N es la cantidad de ejemplos y M la cantidad de atributos.

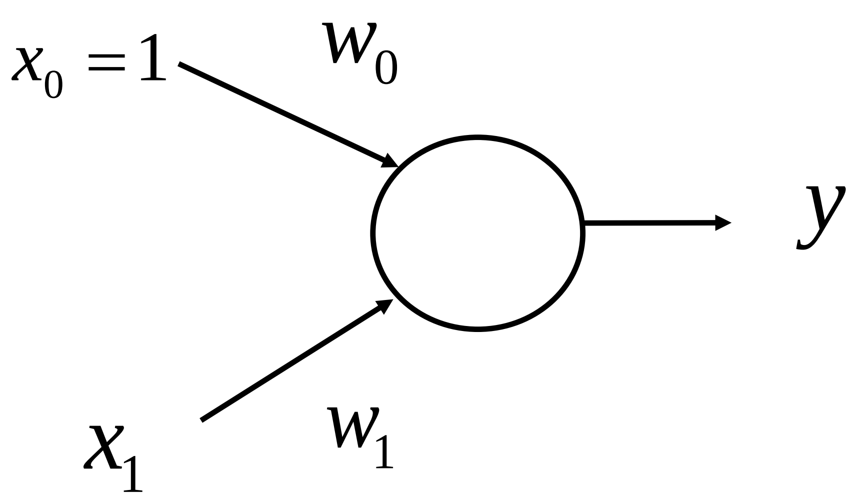
- Retorna: un arreglo con el resultado de aplicar el combinador lineal entrenado previamente con fit() a la matriz de ejemplos X.
  - **Y** : arreglo de N elementos siendo N la cantidad de ejemplos

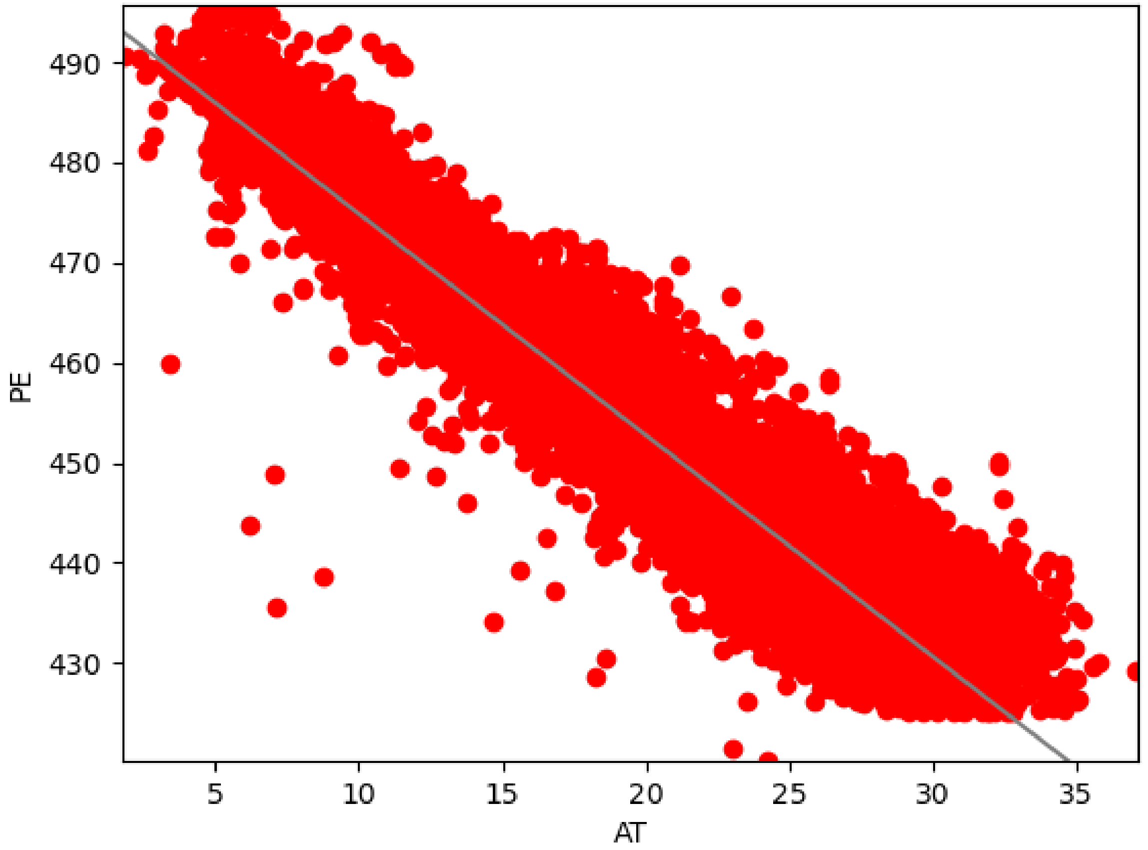
CombLineal_PtosNOalineados.ipynb

```
import numpy as np
from ClassNeuronaLineal import NeuronaLineal

Ptos = np.array([[1,1], [3,2], [5,5], [2,3], [4,3]])
X = Ptos[:,0].reshape(-1,1)
Y = Ptos[:,1]

alfa = 0.01
MAX_ITE = 500
Cota = 10e-06

nl = NeuronaLineal(alpha=0.01, n_iter=30, cotaE=10e-06, draw=1, title=['X','Y'])
nl.fit(X, Y)

#-- gráfico con la evolución del error ---
plt.plot(range(1, len(nl.errors_) + 1), nl.errors_, marker='o')
plt.xlabel('Iteraciones')
plt.ylabel('Cantidad de actualizaciones')
plt.show()
```

Producción de energía (CCPP.csv)

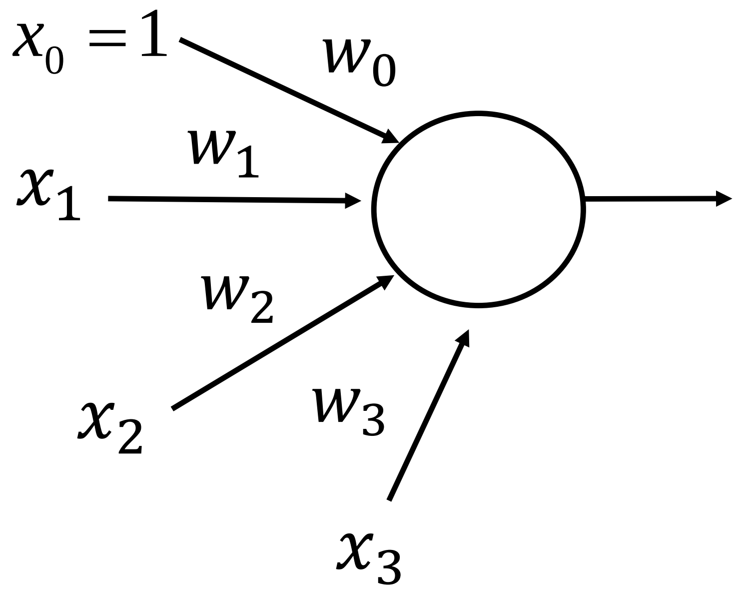
- Modifique el código anterior para predecir la producción de energía (atributo PE) a partir del valor de temperatura (atributo AT)





Ejercicio

- Entrene un combinador lineal que reciba tres dígitos binarios y devuelva el número decimal correspondiente.
- Ejemplo: (0 1 0) → 2



$$y = \sum_{i=0}^{3} w_i x_i$$

Comb_LINEAL_HOUSING_RN.py

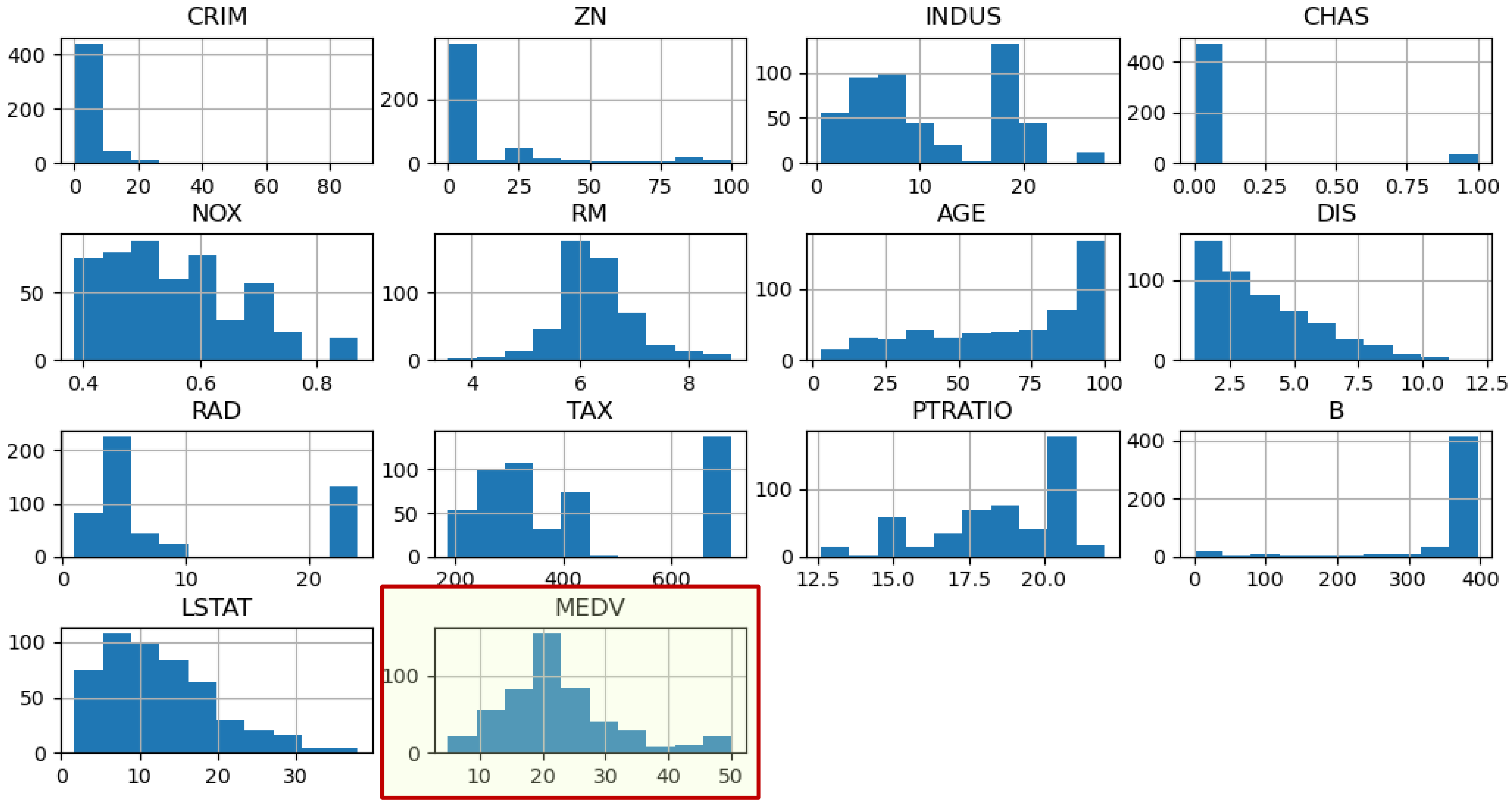
Boston Housing data set

- Contiene datos de las viviendas de 506 secciones censales de Boston del censo de 1970.
- Objetivo: Predecir el precio de una casa.
- Analizar los atributos antes de usarlos.
- Selección de atributos por correlación.

- CRIM - Crimen per cápita por ciudad
- ZN - proporción de terrenos residenciales
- INDUS - proporción de negocios no minoristas
- CHAS - 1 si el tramo limita el río, 0 si no
- NOX - concentración de óxidos nítricos
- RM – nro. promedio de habitaciones por vivienda
- AGE - proporción de unidades construidas antes de 1940
- DIS - Distancias promedio a centros de empleo
- RAD - accesibilidad a las autopistas radiales
- TAX - tasa de impuesto
- PTRATIO - colegios por localidad
- B - proporción de personas negras (año 1980!)
- LSTAT - porcentaje de personas de bajo estatus.
- MEDV - valor medio de las viviendas ocupadas por sus dueños

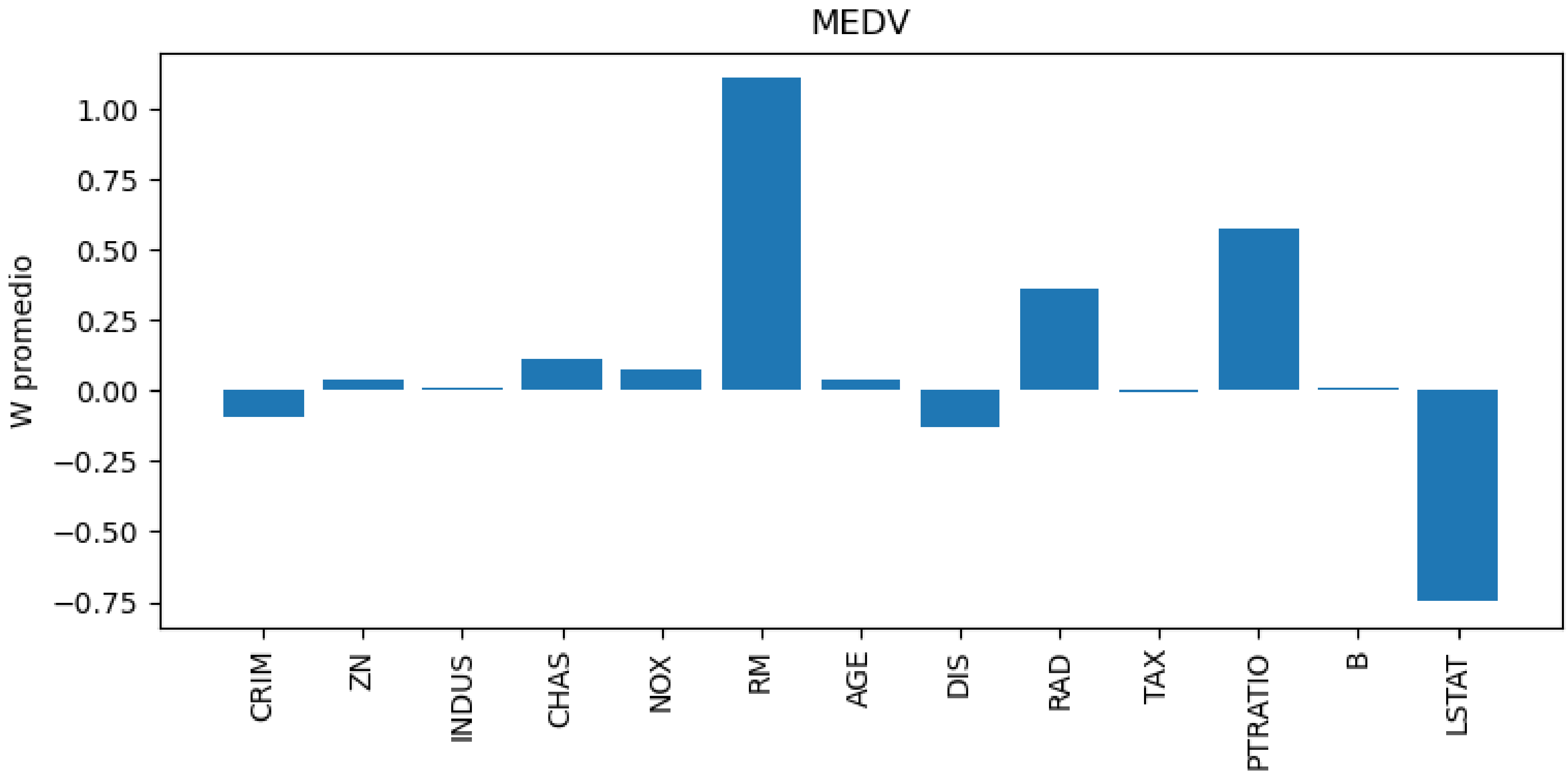
Comb_LINEAL_HOUSING_RN.py

Histogramas de los atributos relevados



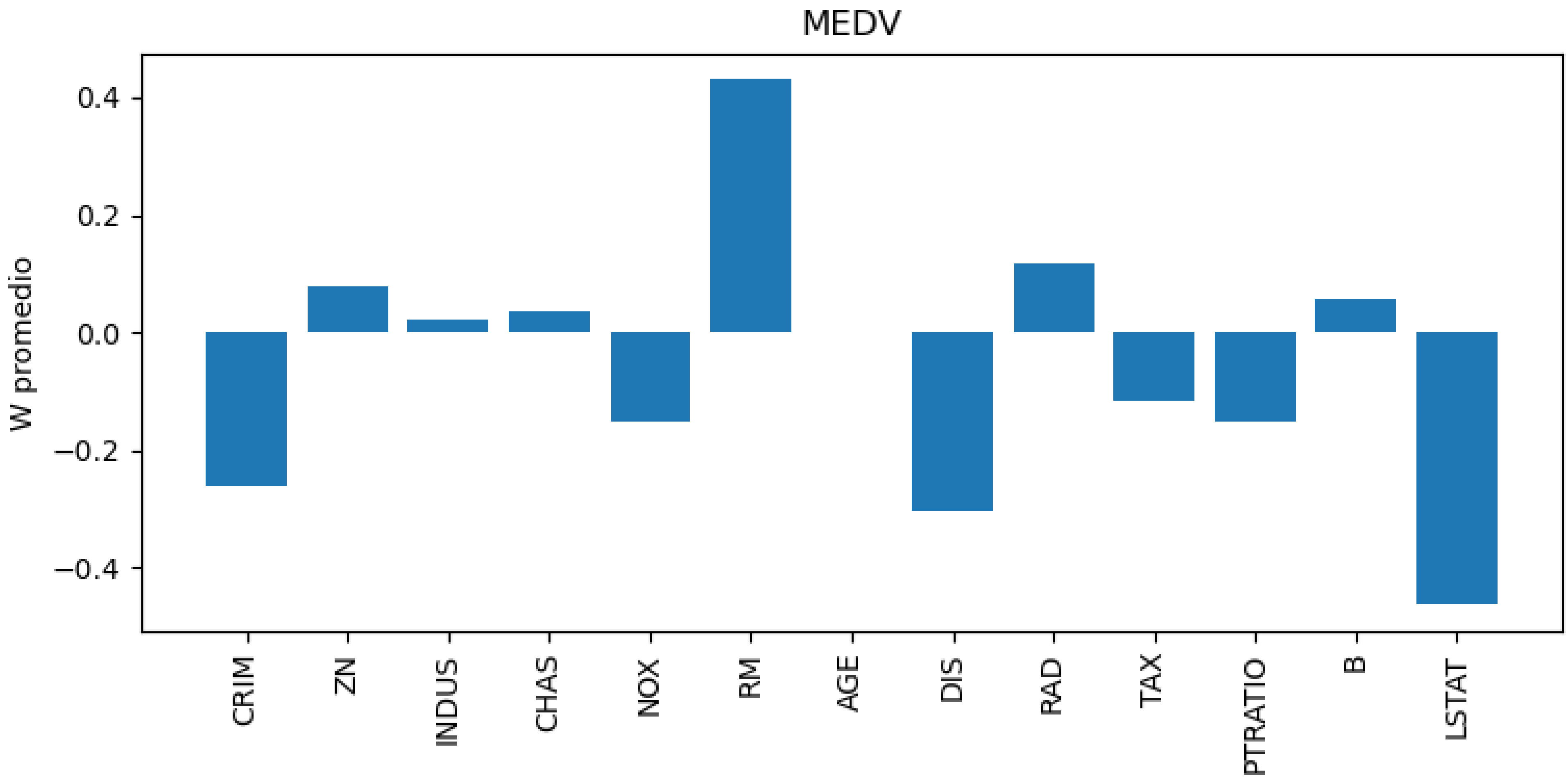
Comb_LINEAL_HOUSING_RN.py

Vector W – sin normalizar los ejemplos



Comb_LINEAL_HOUSING_RN.py

Vector W – ejemplos normalizados linealmente



Comb_LINEAL_HOUSING_RN.py

Vector W – normalización usando media y desvío