Memoria Técnica. Práctica 1

Sergio Gavilán Fernández sgavil01@ucm.es

Alejandro Villar Rubio alvill04@ucm.es

# Regresión lineal con una variable

## Código

import numpy as np

from pandas.io.parsers import read\_csv

import matplotlib.pyplot as plt

# Carga el fichero csv especificado y lo devuelve en un array de numpy

def carga\_csv(file\_name):

    valores = read\_csv(file\_name, header=None).values

    # suponemos que siempre trabajaremos con float

    return valores.astype(float)

def axis\_lim\_grafica(subPlt, minX, maxX, minY, maxY):

    subPlt.set\_xlabel('Población de la ciudad en 10.000s')

    subPlt.set\_ylabel('Ingresos en $10.000s')

    subPlt.set\_xlim([minX - 1, maxX + 1])

    subPlt.set\_ylim([minY - 4, maxY + 0.3])

# Dibuja la gráfica principal donde se mostrará la función h

def dibuja\_grafica(subPlt, fArray, funH, theta):

    X = np.linspace(5.0, 22.5, len(fArray), endpoint=True)

    subPlt.scatter(fArray[:, 0], fArray[:, 1], s=50, c='red', marker="x")

    axis\_lim\_grafica(subPlt, 5.0, 22.5, 0, 25)

    subPlt.plot(fArray[:, 0], funH)

    max\_value = np.amax(fArray[:, 0])

    t0 = theta[0]

    t1 = theta[1]

    subPlt.annotate(r'$h(x)={}+{}x$'.format(t0, t1) ,

        xy=(max\_value, h(max\_value, theta)), xycoords='data',

        xytext=(5, 26.5), fontsize=10,

        arrowprops=dict(arrowstyle="->", connectionstyle="arc3,rad=-.2"))

def axis\_lim\_costes(subPlt, minX, maxX, minY, maxY):

    subPlt.set\_xlabel('Número de iteraciones')

    subPlt.set\_ylabel('Coste')

    subPlt.set\_xlim([minX - 50, maxX + 200])

    subPlt.set\_ylim([minY - 0.5, maxY + 0.5])

# Dibuja la gráfica donde se verá mostrará el valor del coste en función del número de iteraciones

def dibuja\_costes(subPlt, numCasos, costeArray):

    X = np.linspace(0, numCasos, numCasos, endpoint=True)

    axis\_lim\_costes(subPlt, 0, numCasos, costeArray[len(costeArray) - 1], costeArray[0])

    subPlt.plot(range(numCasos), costeArray)

# Función principal de la práctica que realiza el algoritmo de "Descenso de Gradiente"

def descenso\_gradiente(casos, alpha=0.01, iter=1500):

    # Inicialización de los valores de theta a 0

    theta = np.zeros(2)

    # Inicialización de un array que guarda el historial de los costes

    costeArray = np.zeros(iter)

    # m es el número de ejemplos

    m = len(casos)

    # Bucle de "iter" iteraciones (por defecto son 1500) donde calculamos el valor de theta que minimice la función de coste

    for i in range(iter):

        temp0 = theta[0] - alpha \* (1 / m) \* np.sum(h(casos[:,0], theta) - casos[:,1], axis=0)

        temp1 = theta[1] - alpha \* (1 / m) \* np.sum((h(casos[:,0], theta) - casos[:,1]) \* casos[:, 0], axis=0)

        theta[0] = temp0

        theta[1] = temp1

        funH = h(casos[:,0], theta)

        costeArray[i] = (1 / (2 \* m)) \* np.sum(np.square(h(casos[:,0], theta) - casos[:,1]), axis=0)

        plt.clf()

        print(costeArray[i])

    # Llamada a métodos para dibujar las gráficas

    fig, subPlot = plt.subplots(1, 2, figsize=(12, 5))

    dibuja\_grafica(subPlot[0], casos, funH, theta)

    dibuja\_costes(subPlot[1], iter, costeArray)

    plt.show()

# Función h

def h(x, theta):

   return theta[0] + x \* theta[1]

def main(file\_name):

    a = carga\_csv(file\_name)

    descenso\_gradiente(casos=a)

main("ex1data1.csv")

## Desarrollo y Resultados

Como se puede observar en la *Ilustración 1.* tenemos dos gráficas donde aparecen las soluciones a esta primera parte: la gráfica de la izquierda (A) se representa la hipótesis que viene dada por el modelo lineal:

y la de la derecha (B) donde se observa la función de coste:

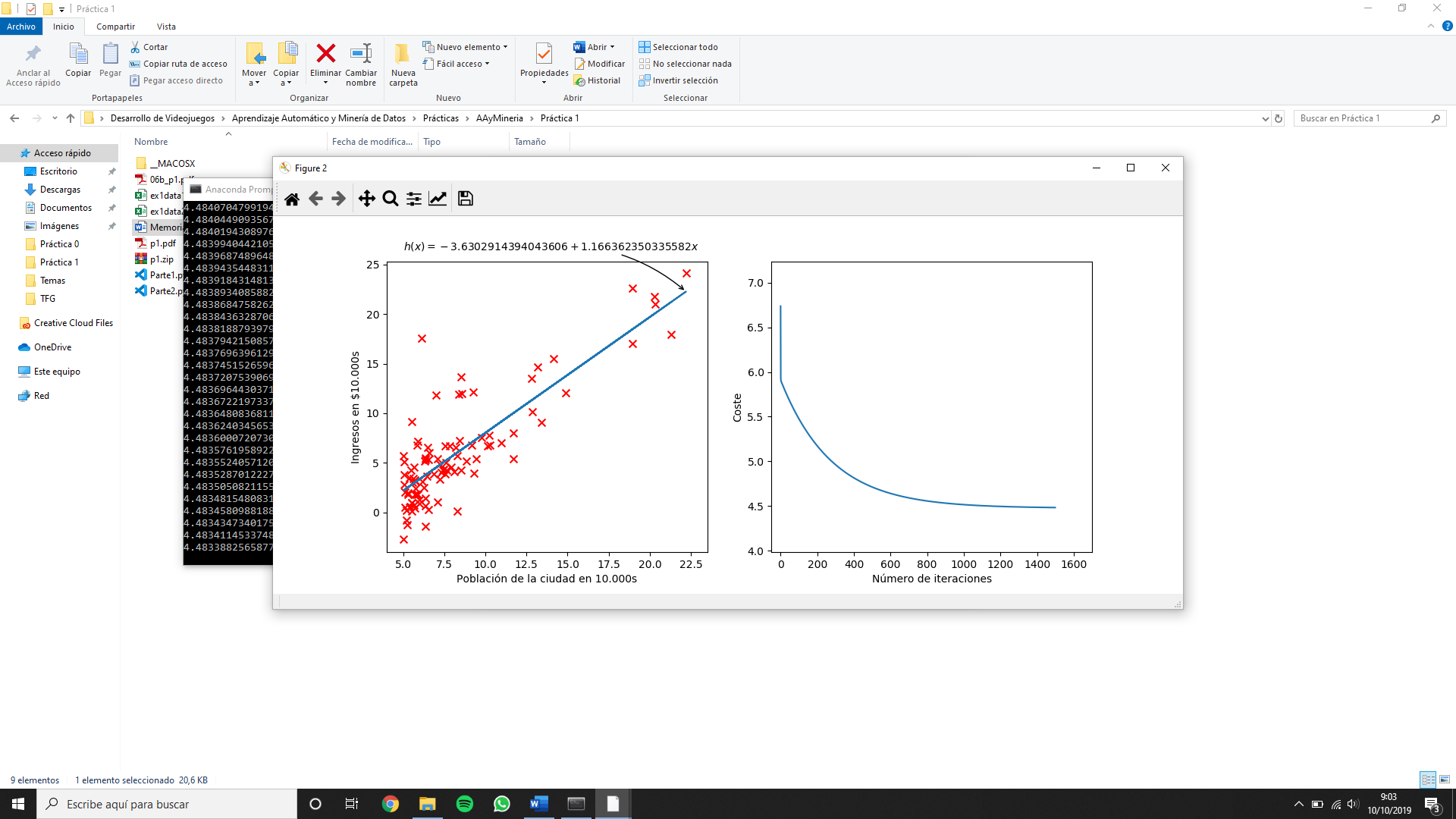


Ilustración . Representación gráfica de la solución de la Parte 1

En la gráfica A tenemos representados los datos del fichero *ex1data.csv* lo más similar posible al enunciado de la práctica, donde estos aparecen con una “x”. El eje X está formado por aquellos valores de *Población de la ciudad en 10.000s* que se encuentran en un intervalo de [5.0, 22.5] y el eje Y mide la variable de *Ingresos en $10.000s* dentro del intervalo [-4.0, 25.0].   
El objetivo principal de esta gráfica es representar la hipótesis. Para ello se ha creado el método ***descenso\_gradiente***donde nos vamos acercando iterativamente al valor de que minimiza la función de coste actualizando cada componente de con la expresión:

Como se puede observar, el resultado de aplicar estas expresiones nos da una solución mostrada en la parte superior de esta misma gráfica:

En la gráfica B se puede ver representada la función de coste . En el eje X tenemos el *Número de iteraciones* que se mueve en un intervalo de [0, 1500] y el eje Y se corresponde con el *Coste* en función de las iteraciones, en este caso tenemos un intervalo de [4.5, 7.0], en función del número de iteraciones. Como se puede ver el coste va disminuyendo a medida que se realizan más iteraciones debido a que el algoritmo de ***descenso\_gradiente***está encontrando la hipótesis que lo minimiza.

Nota: no se ha añadido un 1 como primera componente de cada ejemplo de entrenamiento x.

# Regresión lineal con varias variables

## Código

## Desarrollo y Resultados