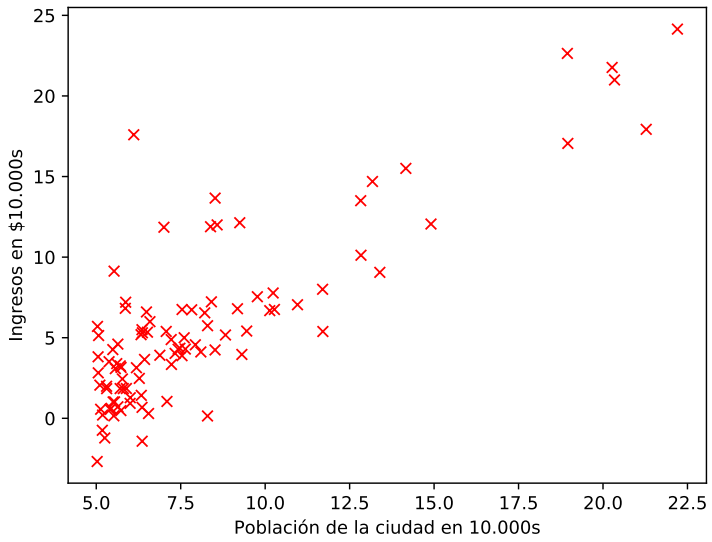
Práctica 1: Regresión lineal

Material proporcionado:

Fichero	Explicación
`ex1data1.csv`	Datos sobre los que aplicar regresión lineal con una variable.
`ex1data2.csv`	Datos sobre los que aplicar regresión lineal con varias variables.

1. Regresión lineal con una variable

En la primera parte de la práctica has de aplicar el método de regresión lineal sobre los datos del fichero `ex1data1.csv` que representan datos sobre los beneficios (segunda columna en el archivo) de una compañía de distribución de comida en distintas ciudades, en base a su población (primera columna en el archivo), como se muestra en esta figura:



Para leer el contenido del archivo csv puedes utilizar el método `read_csv` de Pandas, que devuelve un *DataFrame*, y luego convertirlo a un array de numpy con el método `to_numpy()`:

```python
import numpy as np
from pandas.io.parsers import read_csv

valores = read_csv("ex1data1.csv", header=None).to_numpy()
```

Para encontrar los parámetros θ que definen la recta que mejor se ajusta a los datos de entrenamiento, has de aplicar el método de descenso de gradiente. El objetivo de la regresión lineal es minimizar la función de coste:

$$J(\theta) = \frac{1}{2m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)} \right)^2$$

donde la hipótesis $h_\theta(x)$ viene dada por el modelo lineal:

$$h_\theta(x) = \theta_0 + \theta_1 x$$

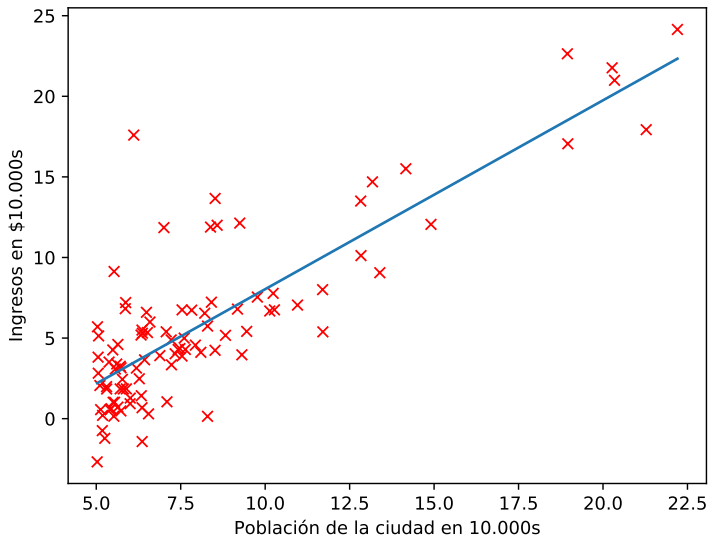
En el método de descenso de gradiente nos vamos acercando iterativamente al valor de θ que minimiza la función de coste $J(\theta)$ actualizando cada componente de θ con la expresión:

$$\theta_0 := \theta_0 - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})$$
$$\theta_1 := \theta_1 - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x^{(i)}$$

teniendo cuidado de actualizar simultáneamente todas las componentes θ_j, es decir, computando el valor de la hipótesis $h_\theta(x^{(i)})$ en cada iteración utilizando los valores de los parámetros θ_j obtenidos en la iteración anterior.

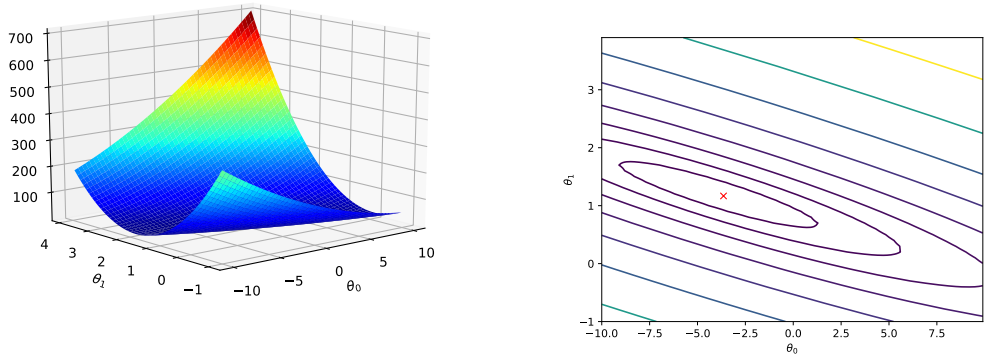
A modo de depuración es bueno que vayas visualizando por pantalla el valor de la función de coste $J(\theta)$ a medida que avanza el descenso de gradiente, para comprobar que efectivamente su valor decrece de manera continua. Para ello has de implementar una función auxiliar que calcule el coste. El coste inicial con $\theta_0 = 0$ y $\theta_1 = 0$ es aproximadamente de 32,07.

Aplicando el método de descenso de gradiente, con unas 1500 iteraciones y un valor de $\alpha = 0{,}01$ deberías obtener unos valores para θ que, por ejemplo, para una población de 70.000 habitantes ($x = 7$) predicen unos beneficios aproximados de \$45.282 ($y = 4{,}5282$), y que definen una recta como la que se muestra en esta figura:



1.1. Visualización de la función de coste

Para comprender mejor el comportamiento de la función de coste $J(\theta)$ puedes calcular su valor sobre una rejilla de valores de $\theta_0 = 0$ y $\theta_1 = 0$ y generar gráficas como estas:



donde se muestra el resultado de llamar a las funciones *plot_surface* y *contour* de matplotlib con los valores de la función de coste en el intervalo $\theta_0 \in [-10, 10]$ y $\theta_1 \in [-1, 4]$. Para generar la rejilla de valores de $\theta_0 = 0$ y $\theta_1 = 0$ en esos intervalos se puede utilizar la función `numpy.meshgrid()`:

```
def make_data(t0_range, t1_range, X, Y):
    """Genera las matrices Theta0, Theta1, Coste para generar un plot en 3D
       del coste para valores de theta_0 en el intervalo t0_range y
       valores de theta_1 en el intervalor t1_range
    """

    step = 0.1
    Theta0 = np.arange(t0_range[0], t0_range[1], step)
    Theta1 = np.arange(t1_range[0], t1_range[1], step)
    Theta0, Theta1 = np.meshgrid(Theta0, Theta1)
    Coste = np.empty_like(Theta0)
    for ix, iy in np.ndindex(Theta0.shape):
        Coste[ix, iy] = coste(X, Y, [Theta0[ix, iy], Theta1[ix, iy]])
    return [Theta0, Theta1, Coste]
```

La gráfica de contorno de la figura se ha generado utilizando una escala logarítmica para el eje z, donde se representa $J(\theta)$, con 20 intervalos entre 0.01 y 100, lo que se consigue pasando `np.logspace(-2, 3, 20)` como cuarto argumento de la función *contour*. En la gráfica de contorno se muestra también el mínimo obtenido por el descenso de gradiente, y se podrían mostrar otros puntos intermedios obtenidos en el proceso.

2. Regresión con varias variables

El objetivo de la segunda parte de la práctica es aplicar el método de regresión lineal a los datos del archivo `ex1data2.csv` que contienen datos sobre el precio de casas vendidas en Portland, Oregon, incluyendo para cada casa el tamaño en pies cuadrados, el número de habitaciones y el precio.

Como el rango de los distintos atributos es muy diferente (unidades en el caso del número de habitaciones y miles en el caso de la superficie) para acelerar la convergencia al aplicar el método de descenso de gradiente, es necesario que normalices los atributos, sustituyendo cada valor por el cociente entre su diferencia con la media y la desviación estándar de ese atributo en los ejemplos de entrenamiento. Para ello debes implementar una función que reciba una matriz X con los ejemplos de entrenamiento, uno por fila, y devuelva otra matriz X_norm, de las mismas dimensiones, con los valores normalizados, además de un vector *mu* con la media de cada atributo y otro vector *sigma* con la desviación estándar de cada atributo. Los vectores *mu* y *sigma* son necesarios para poder utilizar el modelo ajustado por el método de descenso de gradiente para hacer predicciones sobre otros ejemplos. Dados el número de habitaciones y la superficie de una nueva casa, si se quiere predecir el precio utilizando el modelo resultado del entrenamiento, es necesario normalizar esos atributos utilizando los mismos valores de media y desviación estándar con los que se normalizaron los ejemplos de entrenamiento utilizados para obtener el modelo. En numpy, la función `mean` computa la media y la función `std` la desviación estándar.

Aplica el método de descenso de gradiente a los datos normalizados para obtener el valor de los parámetros θ que minimizan la función de coste. Has de realizar una implementación vectorizada del algoritmo de descenso de gradiente que funcione independientemente del número de atributos n.

2.1. Implementación vectorizada del descenso de gradiente

En primer lugar, es necesario añadir un 1 como primera componente de cada ejemplo de entrenamiento x, de forma que el valor de la hipótesis $h_\theta(x)$ se pueda obtener como el producto de dos vectores:

$$h_\theta(x) = \theta_0 + \theta_1 x_1 = \theta^T x$$

Puedes utilizar la función `hstack()` de numpy para añadir una columna de 1's a la matriz con los ejemplos de entrenamiento:

```
valores = read_csv("ex1data2.csv", header=None).to_numpy().astype(float)
X = datos[:, :-1]
Y = datos[:, -1]
m = np.shape(X)[0]
n = np.shape(X)[1]
X = np.hstack([np.ones([m, 1]), X])
```

Una vez modificada la matriz X de esta forma, es posible calcular la función de coste en forma vectorizada (en notación matricial):

$$J(\theta) = \frac{1}{2m}(X\theta - \vec{y})^T(X\theta - \vec{y})$$

donde

$$X = \begin{bmatrix} — & (x^{(1)})^T & — \\ — & (x^{(2)})^T & — \\ & \vdots & \\ — & (x^{(m)})^T & — \end{bmatrix} \quad \vec{y} = \begin{bmatrix} y^{(1)} \\ y^{(2)} \\ \vdots \\ y^{(m)} \end{bmatrix}$$

y el producto $X\theta$ resulta en

$$X\theta = \begin{bmatrix} (x^{(1)})^T\theta \\ (x^{(2)})^T\theta \\ \vdots \\ (x^{(m)})^T\theta \end{bmatrix} = \begin{bmatrix} \theta^T(x^{(1)}) \\ \theta^T(x^{(2)}) \\ \vdots \\ \theta^T(x^{(m)}) \end{bmatrix}$$

a partir de lo cual es sencillo obtener el valor de la función de coste sin necesidad de bucles, ayudados de las operaciones elemento a elemento entre matrices de numpy y la función sumatorio (`sum`).

Para completar el algoritmo de descenso de gradiente, has de calcular el gradiente de la función de coste. El gradiente es un vector de la misma longitud que θ donde la componente j (para $j = 0, 1, \ldots, n$) viene dada por la expresión:

$$\frac{\partial J(\theta)}{\partial \theta_j} = \frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})x_j^{(i)}$$

Para vectorizar esta operación escribimos el gradiente en forma de vector:

$$\begin{aligned} \begin{bmatrix} \frac{\partial J(\theta)}{\partial \theta_0} \\ \frac{\partial J(\theta)}{\partial \theta_1} \\ \vdots \\ \frac{\partial J(\theta)}{\partial \theta_n} \end{bmatrix} &= \frac{1}{m}\begin{bmatrix} \sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})x_0^{(i)} \\ \sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})x_1^{(i)} \\ \vdots \\ \sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})x_n^{(i)} \end{bmatrix} \\ &= \frac{1}{m}\sum_{i=1}^{m}\left((h_\theta(x^{(i)}) - y^{(i)})x^{(i)}\right) \\ &= \frac{1}{m}X^T(h_\theta(x) - y) \end{aligned}$$

donde $x^{(i)}$ es un vector mientras que $(h_\theta(x^{(i)}) - y^{(i)})$ es un número; y $(h_\theta(x)) - y)$ es el vector que viene dado por:

$$h_\theta(x) - y = \begin{bmatrix} h_\theta(x^{(1)}) - y^{(1)} \\ h_\theta(x^{(2)}) - y^{(2)} \\ \vdots \\ h_\theta(x^{(m)}) - y^{(m)} \end{bmatrix}$$

que nos permite obtener el vector gradiente sin necesidad de realizar ningún bucle.

A continuación, debes experimentar con el efecto de utilizar diferentes valores para la tasa de aprendizaje y construir una gráfica donde se muestre la evolución de la función de coste $J(\theta)$ a medida que avanza el descenso de gradiente, con distintos valores de tasa de aprendizaje (0.3, 0.1, 0.03, 0.01, ...).

2.2. Ecuación normal

Has de resolver de nuevo el problema utilizando el método de la ecuación normal que obtiene en un sólo paso el valor óptimo para θ con la expresión:

$$\theta = (X^TX)^{-1}X^T\vec{y}$$

y recuerda que en este caso no has de normalizar los atributos.

Por último, demuestra que tus cálculos son correctos comprobando que el modelo obtenido con descenso de gradiente hace las mismas predicciones que el que resulta de la ecuación normal, aplicándolo por ejemplo a una casa con una superficie de 1.650 pies cuadrados y 3 habitaciones.

3. Entrega de la práctica

La práctica debe entregarse utilizando el mecanismo de entregas del campus virtual. Se entregará un único fichero en formato pdf que contenga la memoria de la práctica, incluyendo el código desarrollado y los comentarios y gráficas que se estimen más adecuados para explicar los resultados obtenidos.