Práctica 1

Pablo Arranz Ropero
Juan Alberto Camino Sáez
Grupo 2

Práctica 1: Regresión lineal

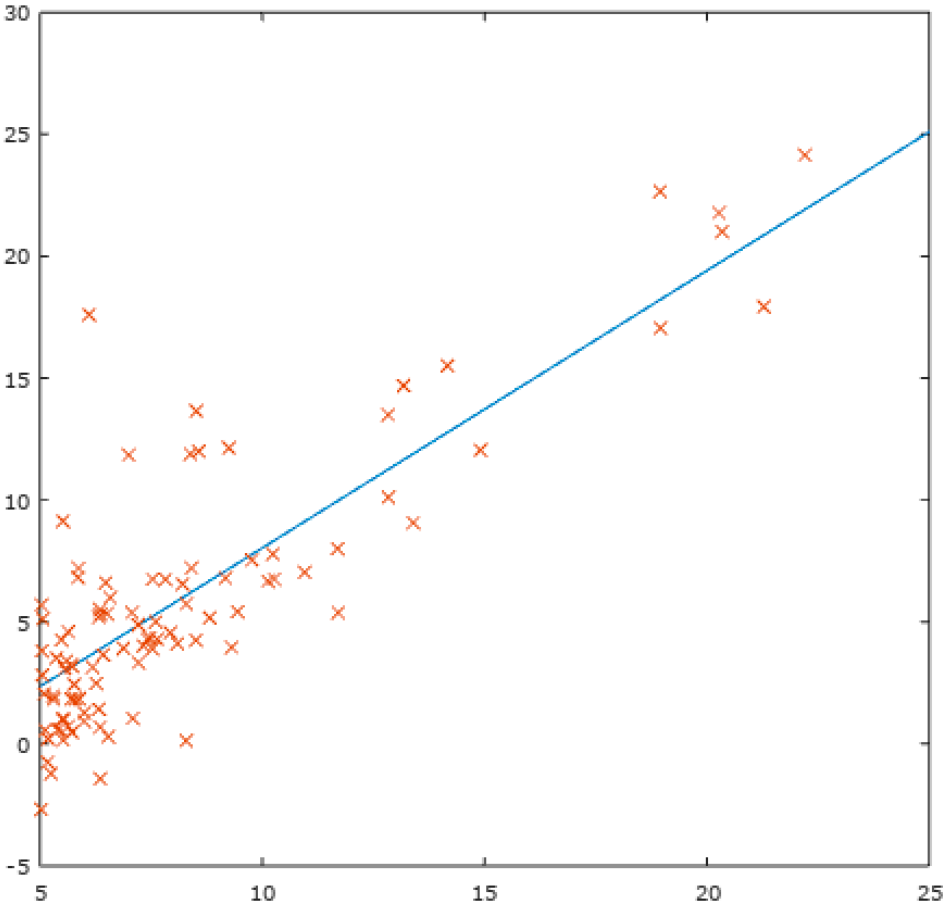
En esta práctica, que se divide en dos apartados, se trata de aplicar regresión lineal a un conjunto de datos, de forma que el primer caso es regresión con una variable y el segundo, regresión multivariable.

Regresión lineal con una variable

En el primer caso, tenemos los datos en el fichero *ex1data1.txt*, en el cual se encuentran los beneficios de una compañía de comida en distintas ciudades en base a la población que tienen.
Para realizar este caso, hemos creado un fichero llamado *regresionunivariable.m,* en el cual le pasamos el ratio de aprendizaje, y esta función se encarga de conseguir el modelo lineal en base a los datos de ese fichero, aplicando descenso de gradiente de θ_0 y θ_1 para alcanzar el mínimo posible de la función de coste. Este es el código:

```
1.  function [theta0, theta1] = regresionunivariable(ratioaprendizaje)
2.
3.    datos = load("data-p1/ex1data1.txt");
4.
5.    theta0 = 0;
6.    theta1 = 0;
7.    diffuncost = 1
8.    m = rows(datos);
9.
10.   while diffuncost > 0.0001
11.
12.     #calculo funcion coste
13.     sumatorio = sum((theta0 + (theta1*datos(:,1)) - datos(:,2)).^2);
14.     funcostanterior = (1/(2*m))*sumatorio;
15.
16.     sumatoriotheta0 = sum((theta0 + (theta1*datos(:,1))) - datos(:,2));
17.     sumatoriotheta1 = sum(((theta0 + (theta1*datos(:,1))) - datos(:,2)).*datos
    (:,1));
18.
19.     diftheta0 = (ratioaprendizaje/m) * sumatoriotheta0;
20.     diftheta1 = (ratioaprendizaje/m) * sumatoriotheta1;
21.
22.     theta0 = theta0 - diftheta0;
23.     theta1 = theta1 - diftheta1;
24.
25.     #calculo funcion coste
26.     sumatorio = sum((theta0 + (theta1*datos(:,1)) - datos(:,2)).^2);
27.     funcost = (1/(2*m))*sumatorio;
28.
29.     diffuncost = funcostanterior - funcost;
30.
31.   endwhile
32.
33. endfunction
```

Ejecutando el algoritmo de descenso de gradiente con un ratio de aprendizaje de 0.01, θ_0 tiene un valor de -3.3455 y θ_1 de 1.1377, obteniendo de esta manera el siguiente modelo

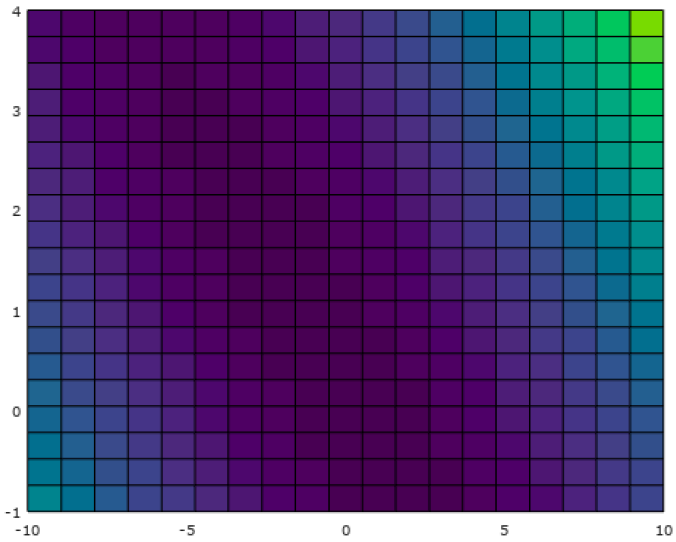


A continuación, fuimos comprobando el valor de la función de coste para diferentes valores de θ_0 (entre -10 y 10) y θ_1 (entre -1 y 4) para comprobar, aplicando las funciones surface y contour en Octave, si el valor obtenido al aplicar el descenso de gradiente era razonablePara poder aplicarlo correctamente, tuvimos que asignar las siguientes operaciones en consola para disponer de estos datos:

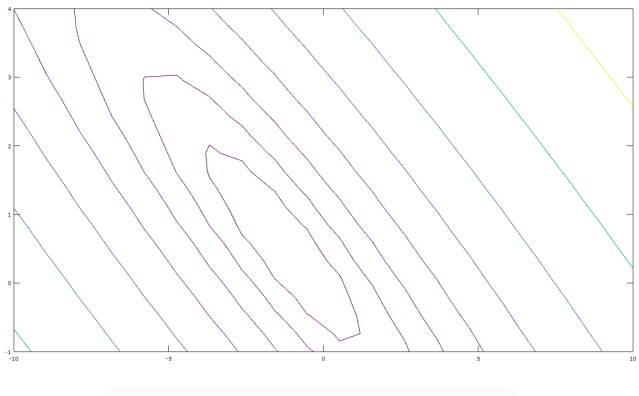
```
1.  theta0 = linspace(-10, 10, 20)
2.  theta1 = linspace(-1, 4, 20)
3.  theta(:, 1)= theta0
4.  theta(:, 2)= theta1
5.  datos = load("data-p1/ex1data1.txt");
6.  X(:,2) = datos(:, 1)
7.  X(:,1) = 1
8.  theta = theta'
9.  Y = datos(:,2)
10.
11. #calculo de las J
12.
13.   for i = 1:20
14.   for j = 1:20
15.   tmp(1,1)=theta(1,i);
16.   tmp(2,1)=theta(2,j);
17.   J(i,j)=(1/(2*rows(datos)))*((X*tmp)-Y)'*((X*tmp)-Y)
18.   endfor
```

```
19.   endfor
20.
21. surface(theta(1,:),theta(2,:),J)
22. contour (theta(1,:),theta(2,:),J,logspace(-2,3,20))
```

Esta son las gráficas obtenidas con *surface* (en dos dimensiones, pero se aprecia como la función de coste se va acercando al mínimo, siendo el mínimo la zona más oscura) y contour, respectivamente.



surface



contour

Por último, obtuvimos los beneficios estimados de una compañía de comida en una población de 15000 y 75000 personas, aplicando el modelo de regresión obtenido, obteniendo
15000 (1.5 * 10,000)→ -1.6388
75000 (7.5 * 10,000) → 5.1876

Regresión lineal multivariable

En este caso, a partir de los datos del fichero *ex1data2.txt* que contiene los precios de las casas en base a su superficie (en pies cuadrados) y en su número de habitaciones, había que conseguir el modelo de regresión. En este caso, tenemos un fichero *normalizaAtributo.m*, que se encarga de, dados unos datos, normalizarlos para que no haya tanta diferencia entre los datos de entrada y de esta forma poder aplicar un buen descenso de gradiente, además de devolver mu y sigma. Este es el código:

```
function [X_norm, mu, sigma] = normalizaAtributo(X)

    for i = 1:columns(X)
        mu(i) = mean(X(:,i));
        sigma(i) = std(X(:,i));
        if(sigma(i) == 0)
            X_norm(:,i) = ones(rows(X))(:,i);
        else
            X_norm(:,i) = (X(:,i) - mu(i)) / sigma(i);
        endif
    endfor

endfunction
```

Una vez normalizados, obtuvimos los valores de θ aplicando, por un lado, el descenso de gradiente, y por otro normalización, para comprobar que en ambos casos salían valores de θ semejantes, ya que ambas formas son equivalentes, diferenciándose solo es las ventajas y desventajas que tienen.
El descenso de gradiente se aplica en un fichero llamado *regresionmultidescgrad.m,* y a continuación se muestra el código:

```
function [theta, sigma, mu] = regresionmultidescgrad(ratioaprendizaje)

  datos = load("data-p1/ex1data2.txt");

  columnas = columns(datos) + 1;
  for i = 0 : columnas - 2
    datos(:,columnas - i)=datos(:,(columnas - i) - 1)
  endfor
  datos(:,1)=1

  [X_norm, mu, sigma] = normalizaAtributo(datos);

  Y = X_norm (:,columns(X_norm));
  X_norm(:,columns(X_norm)) = [];
  X = X_norm;

  theta = zeros(1, columns(X_norm))
  diffuncost = 1
  m = rows(X_norm);

  while diffuncost > 0.0001

    #calculo funcion coste
    funcostanterior = (1/(2*m))*(((X_norm*theta') - Y)' * ((X_norm*theta') - Y
));

    printf("Funcost: %f \n", funcostanterior)
```

```
    #recalculo theta
    sumatorio = ((X_norm*theta') - Y)' * X_norm;
    dif = (ratioaprendizaje/m) * sumatorio;
    theta = theta - dif;

    #calculo funcion coste
    funcost = (1/(2*m))*(((X_norm*theta') - Y)' * ((X_norm*theta') - Y));

    diffuncost = funcostanterior - funcost;

  endwhile

endfunction
```

El descenso de gradiente nos devolvía unos valores de θ siendo:

- $\theta_0 = -7.3353x10^{-17} \approx 0$
- $\theta_1 = 0.44549$
- $\theta_2 = 0.15848$

También devolverá unos valores de σ iguales a:

- $\sigma_0 = 0$
- $\sigma_1 = 794.70$
- $\sigma_2 = 0.76098$
- $\sigma_3 = 125040$

Y los siguientes valores de μ:

- $\mu_0 = 1$
- $\mu_1 = 2000.7$
- $\mu_2 = 3.1702$
- $\mu_3 = 3.4041$

Con estos datos y unas x de entrada normalizamos dichas x, las aplicamos en la hipótesis junto con las θ y desnormalizamos el resultado con σ_3 y μ_3.

La forma normal, sin embargo, se aplica en otro fichero llamado *regresionmultiecnormal.m,* con el siguiente código:

```
function [theta] = regresionmultiecnormal()

    datos = load("data-p1/ex1data2.txt");

    columnas = columns(datos) + 1;
    for i = 0 : columnas - 2
        datos(:,columnas - i)=datos(:,(columnas - i) - 1)
    endfor
    datos(:,1)=1

    Y = datos (:,columns(datos));
    datos(:,columns(datos)) = [];
    X = datos;

    theta = pinv(X'*X)*X'*Y

endfunction
```

En este caso se consiguen $\theta_0 = 89598$, $\theta_1 = 139.21$ y $\theta_2 = -873.8$