**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.

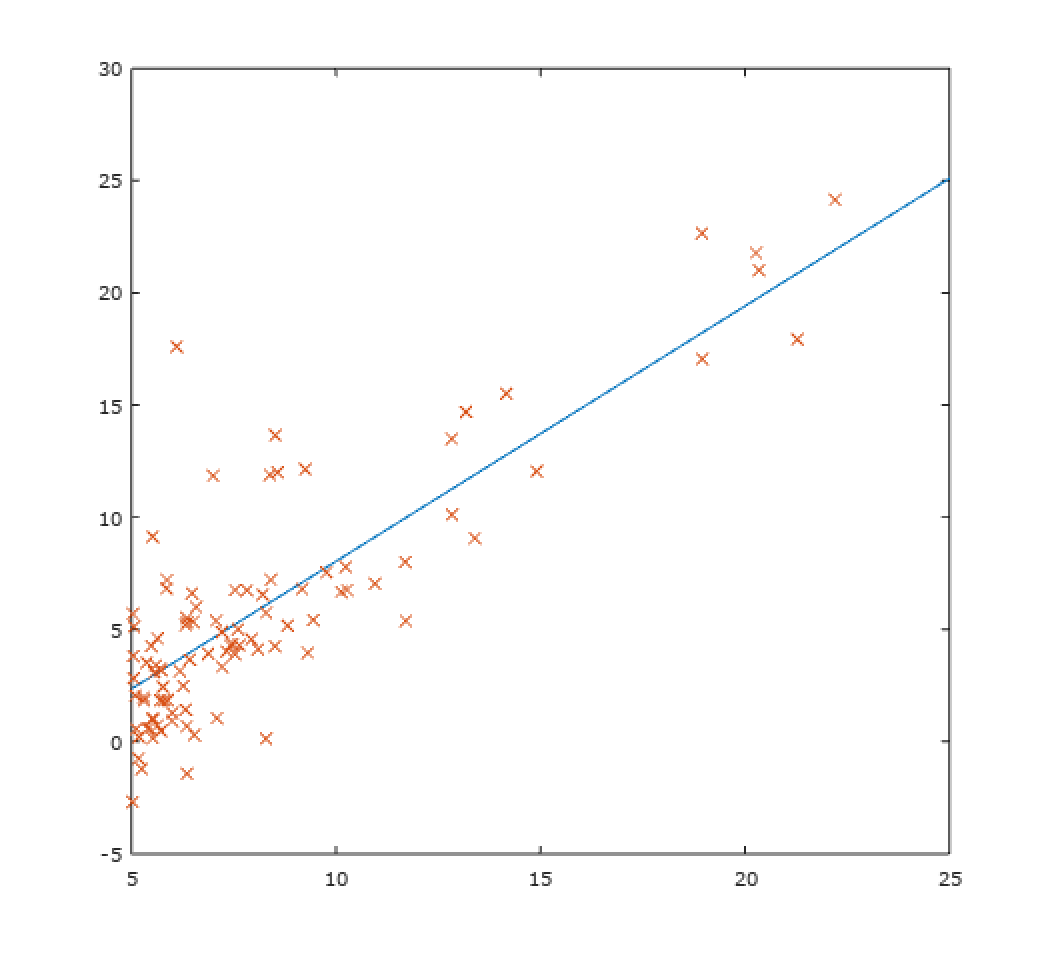
**Regresión lineal con una variable**

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 theta0 y theta1 para alcanzar el mínimo posible de la función de coste. Este es el código:

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

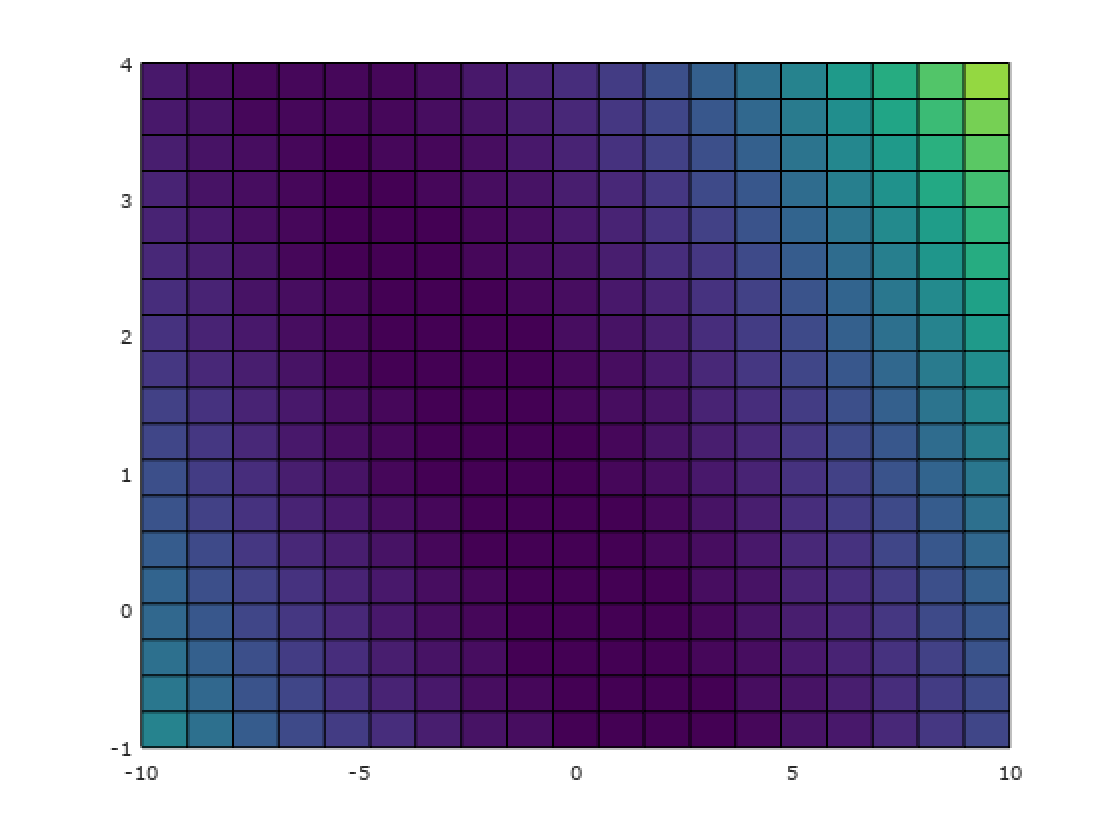
En base a theta0 y theta1 obtenidos al aplicar el modelo de regresión, theta0 tiene un valor de RELLENAR y theta1 de RELLENAR, obteniendo de esta manera el siguiente modelo

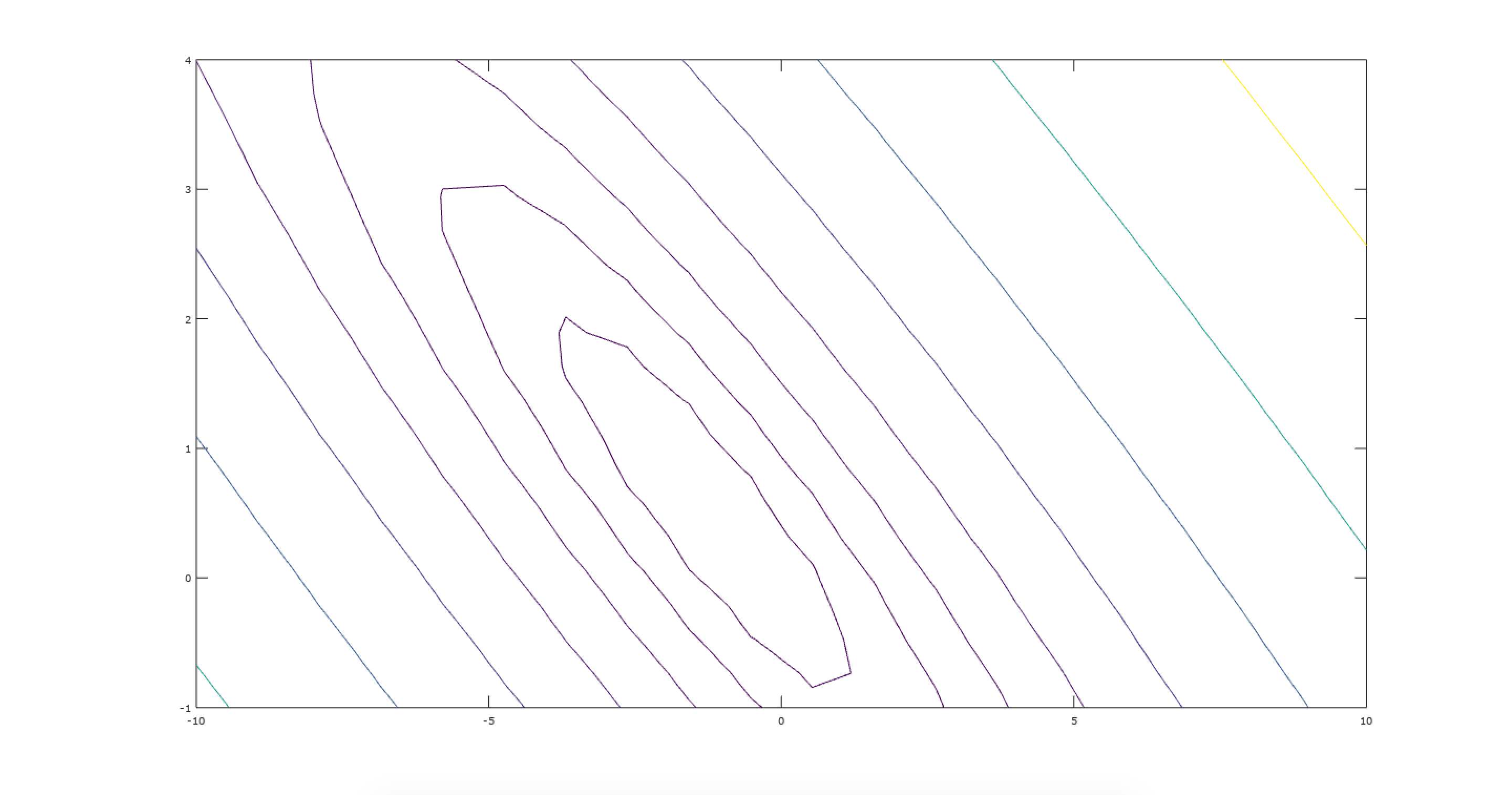
****

A continuación, fuimos comprobando el valor de la función de coste y obtuvimos una gráfica en base a los cambios que se producían en el valor de la función de coste a partir de los valores de theta0 y theta1 que se iban obteniendo en el descenso de gradiente, aplicando surface y contour en Octave. Para 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)
11. #calculo de las J
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
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) 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 10,000s)🡪 -1.6388

75000 (7.5 10,000s) 🡪 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:

1. function [X\_norm, mu, sigma] = normalizaAtributo(X)
3. **for** i = 1:columns(X)
4. mu(i) = mean(X(:,i));
5. sigma(i) = std(X(:,i));
6. **if**(sigma(i) == 0)
7. X\_norm(:,i) = ones(rows(X))(:,i);
8. **else**
9. X\_norm(:,i) = (X(:,i) - mu(i)) / sigma(i);
10. endif
11. endfor
13. endfunction

Una vez normalizados, obtuvimos los valores de theta aplicando, por un lado, el descenso de gradiente, y por otro normalización, para comprobar que en ambos casos salían valores de theta 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:

1. function [theta] = regresionmultidescgrad(ratioaprendizaje)
3. datos = load("data-p1/ex1data2.txt");
5. columnas = columns(datos) + 1;
6. **for** i = 0 : columnas - 2
7. datos(:,columnas - i)=datos(:,(columnas - i) - 1)
8. endfor
9. datos(:,1)=1
11. [X\_norm, mu, sigma] = normalizaAtributo(datos);
13. Y = X\_norm (:,columns(X\_norm));
14. X\_norm(:,columns(X\_norm)) = [];
15. X = X\_norm;
17. theta = zeros(1, columns(X\_norm))
18. diffuncost = 1
19. m = rows(X\_norm);
21. X\_norm = X\_norm';
23. **while** diffuncost > 0.0001
25. #calculo funcion coste
26. funcostanterior = (1/(2\*m))\*(((theta\*X\_norm)' - Y) \* ((theta\*X\_norm)' - Y)');
28. #recalculo theta
29. sumatorio = ((theta\*X\_norm)' - Y)' \* X\_norm';
30. dif = (ratioaprendizaje/m) \* sumatorio;
31. theta = theta - dif;
33. #calculo funcion coste
34. funcost = (1/(2\*m))\*(((theta\*X\_norm)' - Y) \* ((theta\*X\_norm)' - Y)');
36. diffuncost = funcostanterior - funcost;
38. endwhile
39. sigma(:,columns(sigma)) = [];
40. mu(:,columns(mu)) = [];
42. theta = (theta .\*  sigma) + mu;
44. endfunction

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

1. function [theta] = regresionmultiecnormal()
3. datos = load("data-p1/ex1data2.txt");
5. columnas = columns(datos) + 1;
6. **for** i = 0 : columnas - 2
7. datos(:,columnas - i)=datos(:,(columnas - i) - 1)
8. endfor
9. datos(:,1)=1
11. Y = datos (:,columns(datos));
12. datos(:,columns(datos)) = [];
13. X = datos;
15. theta = pinv(X'\*X)\*X'\*Y
17. endfunction

En ambos casos se consiguen unos thetas similares, siendo theta0= RELLENAR, theta1 = RELLENAR y theta2= RELLENAR