Práctica 1:
Regresión lineal

Grupo 13:

- David Ortiz Fernández
- Andrés Ortiz Loaiza

1 - Regresión lineal con una variable

En esta primera parte de la practica nos dedicaremos a realizar la implementación del algoritmo de regresión lineal para una única variable.
Primero de todo se cargarán los datos e dataset del fichero como se puede apreciar en el código del flujo principal de la práctica. Posteriormente se ha desarrollado el código asociado a la regresión para una única variable, en el que se obtiene el modelo lineal. Para minimizar el coste asociado se aplica descenso de gradiente alcanzando así el mínimo local del coste, y por tanto el modelo más ajustado, minimizando para ello simultáneamente los valores θ_0 y θ_1.

Código:

P1a.m: En este archivo se encuentra el flujo principal de la práctica.

```
%parametros iniciales
clear;
iter = 1500;
alpha = 0.01;

% valores de theta0 theta1 iniciales son 0
theta0 = 0;
theta1 = 0;
theta = zeros(2, 1); %vector de tamaño 2x1

data = load("ex1data1.txt");

x = data(:, 1); %primera columna de datos de entrenamiento
y = data(:, 2); %segunda columna de datos de entrenamiento

printf("Pintando datos de entrenamiento \n")

plot(x, y, "rx", "MarkerSize", 8,"linewidth", 2);
ylabel("Profit in $10,000s");
xlabel("Population of City in 10,000s");

printf("Pulse Enter para continuar\n")
pause;

% añadir 1s como primera componente de cada ejemplo de entrenamiento
m = length(x);
X = [ones(m, 1), x]; % para multiplicar Theta^T*X (producto vectorial)

%llamada a la funcion de coste con valores iniciales
```

```
costeInicial = fCost(X, y, theta);

[JS, thetaJ] = dGradiente(X, y, theta, alpha, iter);

printf("El valor que minimiza la función de coste es:  %f\n",min(JS));
printf("Valores Theta0: %f y Theta1: %f \n", thetaJ(1), thetaJ(2));
hold on;
printf("Pulse Enter para continuar\n")
pause;

printf("Pintando recta de regresion lineal... \n");

plot(x, X*thetaJ, "b-", "linewidth", 3)

hold off

predic = thetaJ(1)+ 15000*thetaJ(2);
printf("La prediccion para una poblacion de 15000 habitantes es:  %f\n",predic);

predic = thetaJ(1)+ 75000*thetaJ(2);
printf("La prediccion para una poblacion de 75000 habitantes es:  %f\n",predic);

printf("Pulse Enter para continuar.\n");
pause;


vTheta0 = linspace(-10, 10, 100);
vTheta1 = linspace(-1, 4, 100);
```

```
for i = 1:length(vTheta0)
    for j = 1:length(vTheta1)
    t = [vTheta0(i); vTheta1(j)];
    %Crear matriz de  funciones de costes
    Jniv(i,j) = fCost(X, y, t);
    end
end

%Hacemos la traspuesta para que la visualizacion sea correcta en los ejes
Jniv = Jniv';

% Grafica de contorno
printf("Pintando grafica de contorno... \n");
figure;
contour(vTheta0, vTheta1, Jniv, logspace(-2, 3, 20))
hold on;
plot(thetaJ(1), thetaJ(2), "rx", "MarkerSize", 8, "LineWidth", 4)
printf("Pulse Enter para continuar\n");
pause;
hold off

%Grafica surface
printf("Pintando grafica surface... \n");
figure;
surf(vTheta0, vTheta1, Jniv)
printf("Pulse Enter para finalizar\n");
pause;
```

dGradiente.m

En esta función se aplica el descenso de gradiente, si se suprimen los comentarios se pinta la evolución de la recta que vamos obteniendo al aplicar el descenso de gradiente.

```
function [JS , theta] = dGradiente(X, y, theta, alpha, iter)
  m = length(y);
  x = X(:,2);
  cont =0 ;

    for iter = 1:iter
      %utilizaremos para el calculo de la hipotesis los valores theta de la
      %iteracion anterior
        h = theta(1) + (theta(2)*x);
        theta0 = theta(1) - alpha * (1/m) * sum(h-y);
        theta1  = theta(2) - alpha * (1/m) * sum((h-y) .* x);

        theta = [theta0; theta1];
        %En una matriz JS se van almacenando los distintos valores de la funcion
        %de coste para quedarnos el valor minimo al final
        JS(iter) = fCost(X, y, theta);
        %%%%%%%%%%%%Modo depuración
        %  hold on;
        %  if(cont <= 100)
        %   cont++;
        %  else
        %    cont = 0;
        %    plot(x, X*theta, "b-", "linewidth", 3)
        %    printf("Pulse Enter para continuar\n")
        %    pause;
        %  endif;
     % hold off
    endfor
endfunction
```

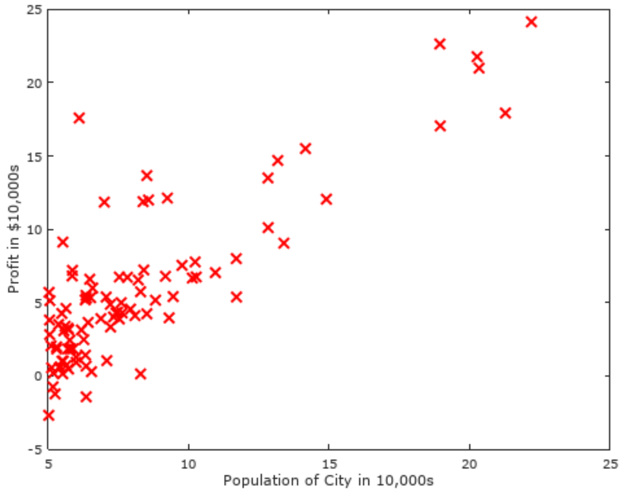
fCost.m

Función que calcula el coste para unos parámetros theta dados.

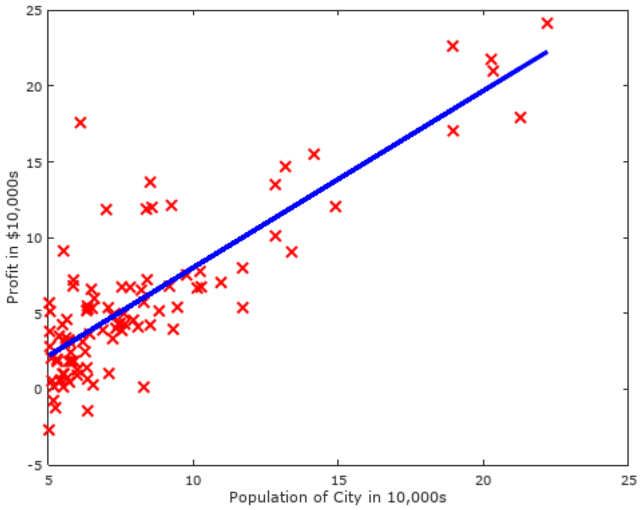
```
%Funcion que calcula el coste para unos parametros theta dados
function J = fCost(X, y, theta)
  %matriz m*2 por matriz 2*1, asi calculamos los m valores de la hipotesis para
  %los theta0 y theta1 dados
  h = X*theta;

  fCuadrado  = (h - y).^2;

  m = length(y);
  J = 1/(2*m) * sum(fCuadrado);
endfunction
```

Resultados:

En la siguiente imagen se puede observar la gráfica con los datos de entrenamiento:



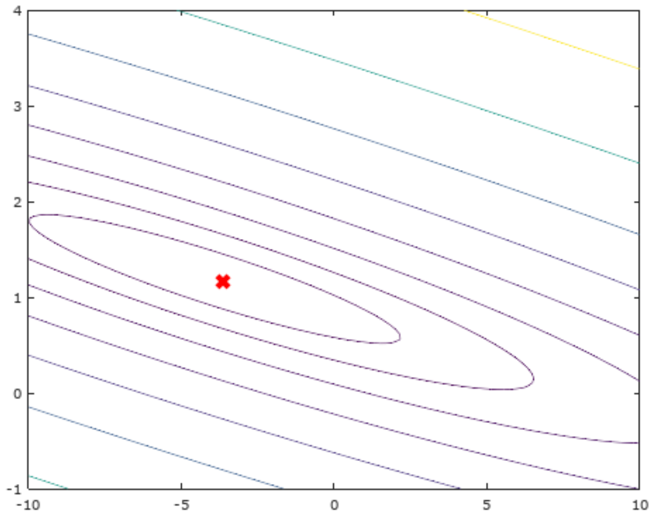
Aplicando el método de descenso de gradiente, con unas **1500** iteraciones y un valor de α = **0,01** obtenemos la siguiente recta, consiguiendo así un modelo de regresión bastante ajustado.



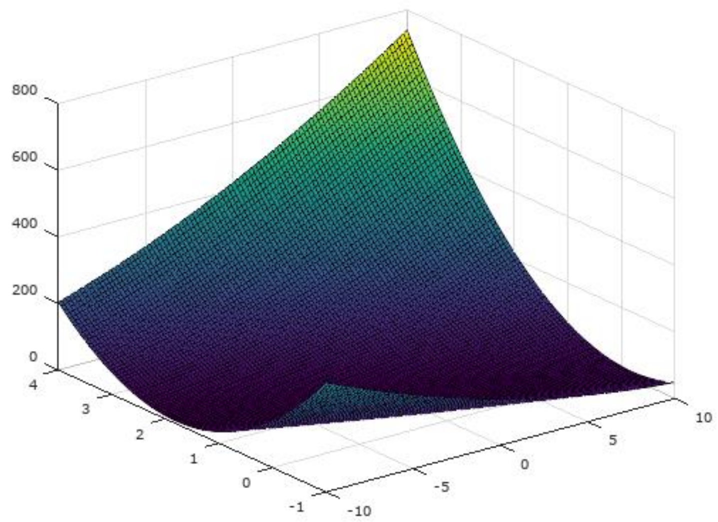
Los resultados de predicción para el modelo obtenido fueron los siguientes:

```
La prediccion para una poblacion de 15000 habitantes es:  17491.804964
La prediccion para una poblacion de 75000 habitantes es:  87473.545984
```

En la gráfica de contorno se muestra también el mínimo obtenido por el descenso de gradiente lo cual nos asegura la correcta aplicación de dicho método, como se puede observar en la siguiente imagen.



Grafica de superficie en 3D:



2 - Regresión lineal con varias variables

Esta parte de la practica está destinada a implementar regresión lineal multivariable para aplicarla a un dataset.
Como en la primera parte, primero se cargaron los datos del dataset, los cuales contenían el precio de unas casas en función de su superficie y habitaciones.

Para esta parte se ha desarrollado una función que se encarga de normalizar las *features.*

Posteriormente se aplicará regresión lineal multivariable, minimizando el coste utilizando para ello descenso de gradiente.

Por último, se aplicara el método de la ecuación normal, como alternativa al descenso de gradiente y como se observará posteriormente se conseguirán resultados muy parejos.

Código:

P1b.m: En este archivo se encuentra el flujo principal para el cálculo de la regresión lineal con varias variables.

```
clear;
%Cargamos los datos
data = load("ex1data2.txt");
X = data(:, 1:2); %features
y = data(:, 3); %target
m = length(y);

%valores sigma e iteraciones
alpha=zeros(4,1);
iters = 1500;

alpha(1) = 0.01;
alpha(2) = 0.1;
alpha(3) = 0.03;
alpha(4) = 0.3;

%normalizamos las features
[X,mu,sigma] = normalizaAtributo(X);
X = [ones(m, 1) X];

%descenso de gradiente para obtener los theta que minimizan la funcion de coste
theta = zeros(3, 1);
[JS4, thetaJ4] = dGradienteMul(X, y, theta, alpha(4), iters);
[JS3, thetaJ3] = dGradienteMul(X, y, theta, alpha(3), iters);
[JS2, thetaJ2] = dGradienteMul(X, y, theta, alpha(2), iters);
[JS1, thetaJ1] = dGradienteMul(X, y, theta, alpha(1), iters);
```

```
% Plot para  visualizar la evolucion de la funcion de coste para los distintos
%valores de alpha
figure;
xlabel('Numero de iteraciones');
ylabel('Coste J');
plot(1:numel(JS4), JS4, '-k', 'LineWidth', 2);
hold on;
plot(1:numel(JS3), JS3, '-g', 'LineWidth', 2);
hold on;
plot(1:numel(JS2), JS2, '-r', 'LineWidth', 2);
hold on;
plot(1:numel(JS1), JS1, '-b', 'LineWidth', 2);
hold on;

%Valores de ThetaJ
printf("Valores Theta0: %f ,Theta1: %f ,Theta2: %f \n", thetaJ4(1), thetaJ4(2)
        ,thetaJ4(3));
printf("Pulse Enter para continuar\n")
pause;

%calculo para 1.650 pies cuadrados y 3 habitaciones
precio = [1 (1650-mu(1))/sigma(1) (3-mu(2))/sigma(2)]*thetaJ1;
printf("El precio estimado para una casa de 1650 pies cuadrados y tres
        habitaciones  es: %f \n",precio)
%
printf("Pulse Enter para finalizar.\n");
pause;
```

normalizaAtributo.m: función que normaliza los atributos y retorna los valores mu y sigma.

```
function [X_norm, mu, sigma] = normalizaAtributo(X)
 X_norm=X;
  for i = 1:size(X,2) %una iteracción para cada atributo
      mu(i) = mean(X(:,i));
      sigma(i) = std(X(:,i));

      X_norm(:,i) = (X_norm(:,i)-mu(i))/sigma(i);
  endfor
endfunction
```

fCost.m Función que calcula el coste para unos parámetros theta dados.

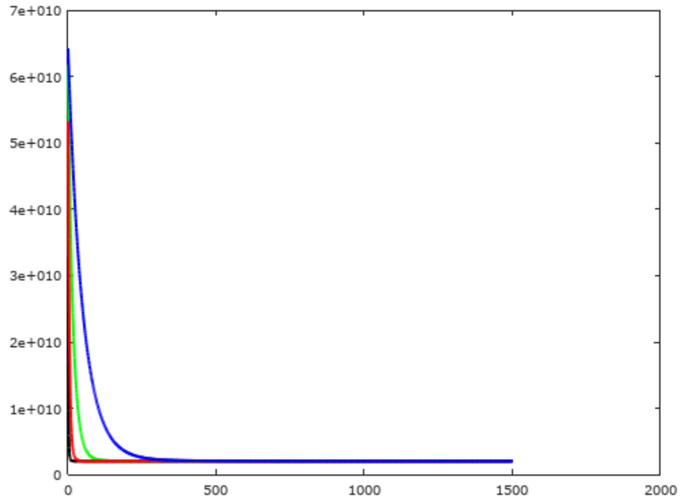
```
function J = fCost(X, y, theta)
  m = length(y);

  J = (1/(2*m))*(X*theta - y)' * (X*theta - y);
endfunction
```

dGradienteMul.m En esta función se aplica el descenso de gradiente para múltiples variables

```
function [JS, theta] = dGradienteMul(X, y, theta, alpha, num_iters)
m = length(y);

%repetimos hasta convergencia
  for iter = 1:num_iters
      %calculo derivada parcial
      theta = theta - (alpha/m) * (X' * (X * theta - y));
      %calculo de la función de coste
      JS(iter) = fCost(X, y, theta);
  endfor

endfunction
```

Resultados:

Grafica que muestra la evolución de la función de coste para diferentes valores de alpha (0.001, 0.1, 0.3, 0.03).



Valores Theta0: 340412.659574, Theta1: 110631.050279, Theta2: -6649.474271 obtenidos para Alpha 0.001.

El modelo obtenido tras aplicar regresión lineal multivariable devuelve la siguiente predicción:

El precio estimado para una casa de 1650 pies cuadrados y tres habitaciones es: 293101.056857

P1c.m: en este archivo está contenido el flujo que resuelve de nuevo el problema utilizando el método de la ecuación normal.

```
clear;
%% Load Data
data = load("ex1data2.txt");
X = data(:, 1:2);
y = data(:, 3);
m = length(y);

X = [ones(m, 1) X];

% calculo de ThetaJ con la ecuación normal
thetaJ = pinv(X'*X)*X'*y;

%Valores de ThetaJ
printf("Valores Theta0: %f ,Theta1: %f ,Theta2: %f \n", thetaJ(1), thetaJ(2),
        thetaJ(3));
printf("Pulse Enter para continuar\n")
pause;

%calculo para 1.650 pies cuadrados y 3 habitaciones
precio = [1 1650 3]*thetaJ;
printf("El precio estimado para una casa de 1650 pies cuadrados y tres
        habitaciones  es: %f \n",precio)

pause;
```

Resultados:

Valores Theta0: 89597.909542, Theta1: 139.210674, Theta2: -8738.019112.

El precio estimado para una casa de 1650 pies cuadrados y tres habitaciones utilizando el método de la ecuación normal es: 293081.464335

Como se puede apreciar los resultados predichos tanto con la ecuación normal como con el método de descenso de gradientes son muy parejos.