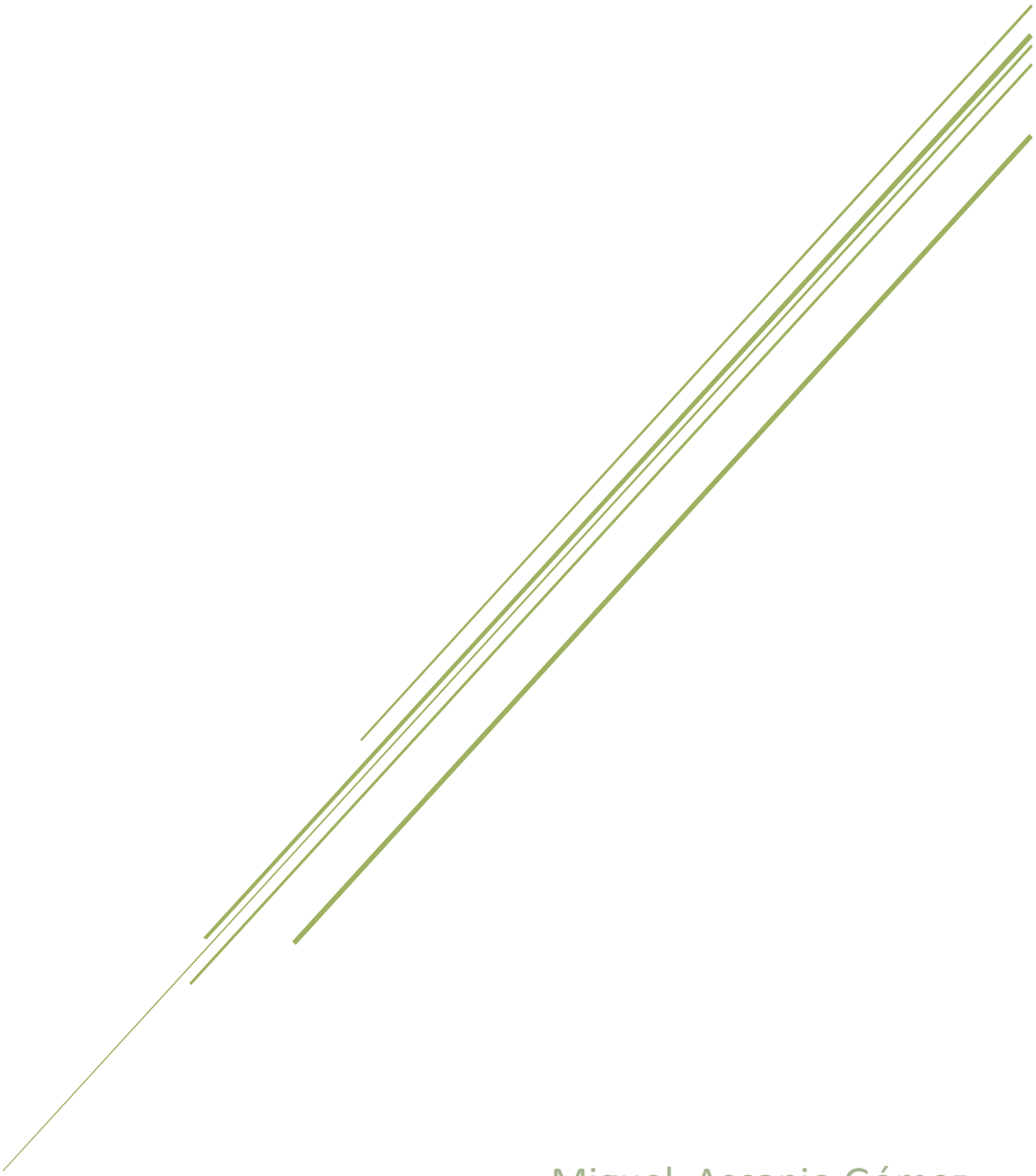
APRENDIZAJE AUTOMÁTICO

Práctica 1



Miguel Ascanio Gómez

Esther Ávila Benito

En esta práctica se ha implementado en Octave un método que aplica regresión lineal mediante el método de descenso de gradiente con variables dadas.

En una primera versión, aplicaremos la regresión lineal a una única variable para relacionar los beneficios de una empresa en distintas ciudades con la población.

En la segunda parte la regresión se aplicará sobre varias variables con el fin de relacionar el precio de una casa con el número de habitaciones y su superficie.

Todas las funciones empleadas son vectorizadas, por tanto, son reutilizables con una o más variables.

REGRESIÓN LINEAL CON UNA VARIABLE

El código de la versión que trabaja con una variable es el siguiente:

```
# LECTURA DE DATOS

s = load("ex1data1.txt");
xEntrenamiento = s(:,1);
yEntrenamiento = s(:,2);
m = length(xEntrenamiento);
# Poner una fila de unos en las x
xEntrenamientoConUnos = [ones(length(xEntrenamiento),1), xEntrenamiento];

tetaVector = [0;0];

cost = inf;
costeAnterior = 0;
i = 1;
vectorCoste = [];


while(abs(cost - costeAnterior) > 0.00003)

        valoresH = (tetaVector' * xEntrenamientoConUnos');
        tetaVector = teta2(tetaVector, m, valoresH, 0.01, xEntrenamientoConUnos, yEntrenamiento);

        costeAnterior = cost;
        cost = coste(tetaVector, m, valoresH, xEntrenamientoConUnos, yEntrenamiento);

        vectorCoste = [vectorCoste, cost];
        i++;

endwhile


# Representación
x = [min(xEntrenamiento):0.1:max(xEntrenamiento)];
xConUnos = [ones(1, length(x)); x];
y = (tetaVector' * xConUnos);

plot(xEntrenamiento, yEntrenamiento, 'ro','markersize',5);
hold on;
plot(x,y);
hold off;
```

FUNCIONES EMPLEADAS:

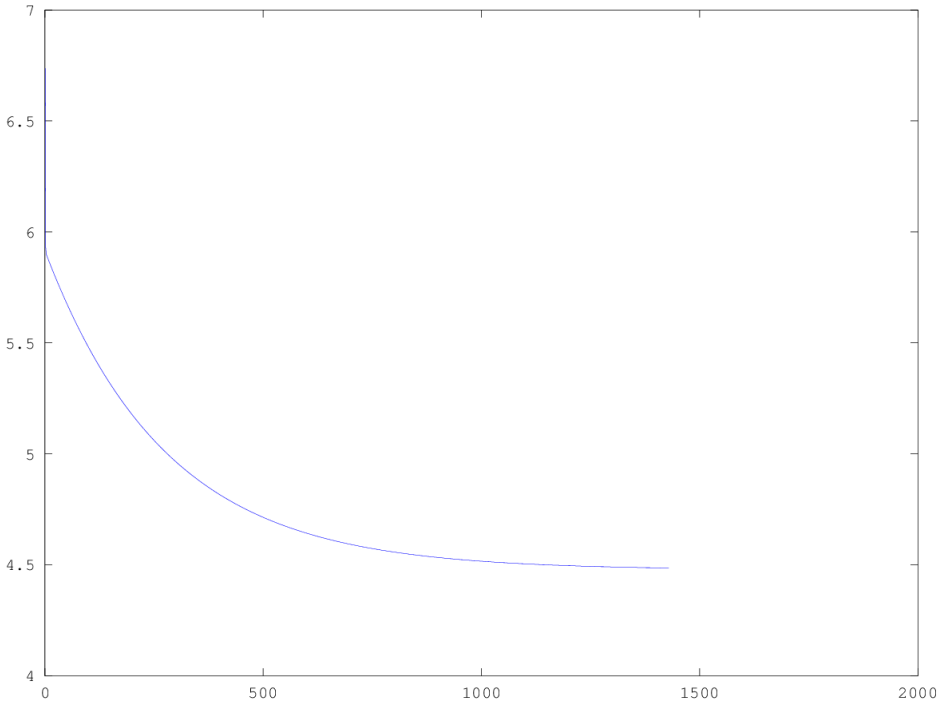
- Función de coste a minimizar

```
function r = coste(tetaVector, m, valoresH, xEntrenamiento, yEntrenamiento)

r = 1 / (2*m) * (xEntrenamiento * tetaVector - yEntrenamiento)' * (xEntrenamiento * tetaVector -
yEntrenamiento);

endfunction;
```

- Función “theta”: calcula los valores que minimizan la función de coste

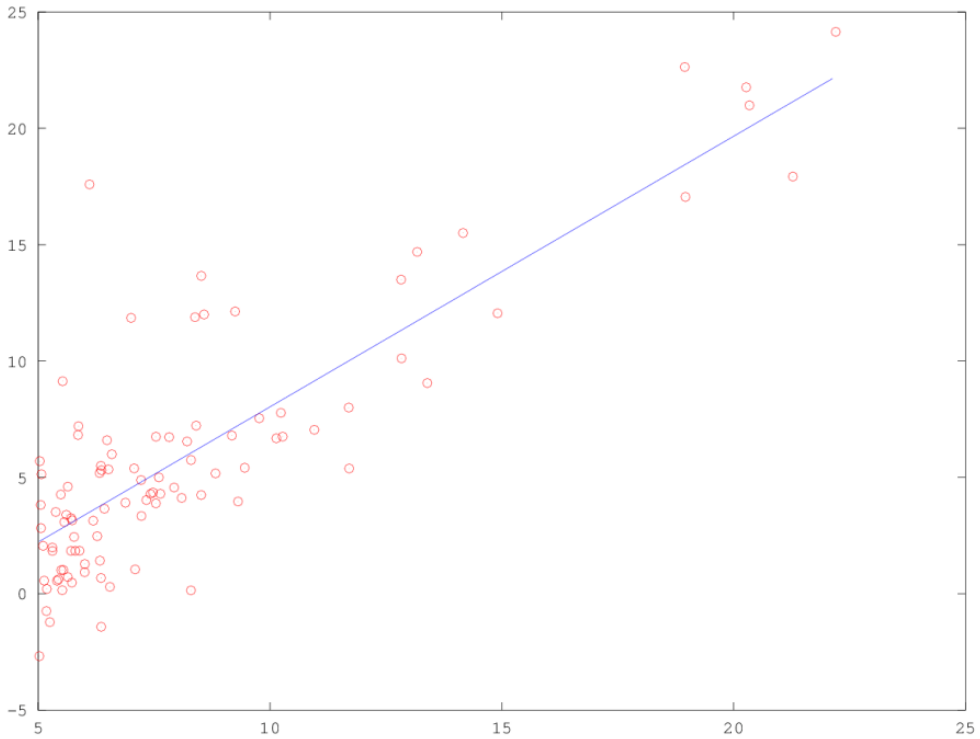
```
function r = teta2(teta, m, valoresH, alpha, xEntrenamientoConUnos, yEntrenamiento)
warning("off", "Octave:broadcast");
a = (valoresH.-yEntrenamiento') .* xEntrenamientoConUnos';
warning("on", "Octave:broadcast");

r = teta - ((alpha/m) * sum(a'))';

endfunction
```

GRÁFICAS



1 .- Función de coste

En la función de coste se observa que inicialmente los valores de teta dan como resultado una función hipótesis mala (el coste es alto), pero rápidamente empieza a bajar, de manera cada vez más lenta, hasta que la diferencia entre los costes de dos iteraciones consecutivas es menor que 0.00003 (condición del while). La gráfica anterior es para alpha = 0.01.



2.- Regresión lineal

REGRESIÓN LINEAL CON VARIAS VARIABLES

```
s = load("ex1data2.txt");

# Pruebas de distintos alpha

valoresAlpha = [0.3,0.1,0.03,0.01,0.003,0.001,0.0003,0.0001];

for alpha = valoresAlpha

m = length(s(1,:))-1;
tam = length(s(:,1));

X = s(:,1:m);
[X, mu,sigma] = normalizaAtributo(X);
X = [ones(tam,1), X];
Y = s(:,m+1);

tetaVector = zeros(m+1,1);

cost = inf;
costeAnterior = 0;

while(abs(cost - costeAnterior) > 0.0003)

        valoresH = (tetaVector' * X');

        tetaVector = teta2(tetaVector, m, valoresH, alpha, X, Y);

        costeAnterior = cost;
        cost = coste(tetaVector, m, valoresH, X, Y);

endwhile

display("---------------");
alpha
tetaVector
(tetaVector' * [1;(1600 - mu(1))/sigma(1);(3-mu(2))/sigma(2)])
display("---------------");

endfor



s = load("ex1data2.txt");

m = length(s(1,:))-1;
tam = length(s(:,1));

X = [ones(tam,1), s(:,1:m)];
Y = s(:,m+1);

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

(tetaVector' * [1;1600;3])
```

FUNCIONES EMPLEADAS:

- Función para normalizar los atributos: a partir de su media y desviación estándar

```
function [X_norm, mu, sigma] = normalizaAtributo(x)
warning("off", "Octave:broadcast");
mu = mean(x);
sigma = std(x);

X_norm = ((x - mu) .* (1 ./ sigma));
warning("on", "Octave:broadcast");

endfunction
```

RESULTADOS

RESULTADOS DE DESCENSO DE GRADIENTE

Se observa que para valores grandes de alpha no se obtienen resultados adecuados, pero a partir de 0.03 los resultados prácticamente no varían.

```
alpha =  0.30000
tetaVector =

-6.6891e+139
-1.3735e+155
-1.3735e+155

ans =  9.9970e+154

---------------

---------------

alpha =  0.10000
tetaVector =

3.2924e+138
6.1867e+153
6.1867e+153

ans = -4.5031e+153

---------------

---------------

alpha =  0.030000
tetaVector =

3.4041e+05
1.1063e+05
-6.6495e+03
```

ans = 2.8612e+05

---

alpha = 0.010000
tetaVector =

3.4041e+05
1.1063e+05
-6.6495e+03

ans = 2.8612e+05

---

alpha = 0.0030000
tetaVector =

3.4041e+05
1.1063e+05
-6.6495e+03

ans = 2.8612e+05

---

alpha = 0.0010000
tetaVector =

3.4041e+05
1.1063e+05
-6.6494e+03

ans = 2.8612e+05

---

alpha = 3.0000e-04
tetaVector =

3.4041e+05
1.1063e+05
-6.6494e+03

ans = 2.8612e+05

---

alpha = 1.0000e-04
tetaVector =

3.4041e+05
1.1063e+05
-6.6493e+03

ans = 2.8612e+05

RESULTADOS CON LA NORMAL

tetaVector =

8.9598e+04
1.3921e+02
-8.7380e+03

ans = 2.8612e+05

COMPARACIÓN DE AMBAS VERSIONES

Se observa que el resultado para el caso probado [1;1600;3] (1600 pies, 3 habitaciones) es el mismo, aunque los valores de theta difieran.