

# **Instituto Tecnológico de Costa Rica**

**Escuela de Ingeniería en Computación**

## **Tarea: Regresión Lineal**

**Curso:** Inteligencia Artificial – GR 40

**Docente:** Ing. José Rafael Castro Mora

**Tutor:** Esteban Villavicencio Soto

**Estudiante:** Joseph Santamaria Castro

**Carné:** 2021044250

**Fecha de entrega:** Miércoles 24 de Septiembre

**Lugar:** San Jose, Costa Rica

## Tabla de contenido

<b>1. Introducción .....</b>	<b>3</b>
<b>2. Regresión Lineal con una Variable.....</b>	<b>3</b>
2.1 Calentamiento.....	3
2.2 Graficando Datos .....	3
2.3 Función de Costo .....	4
2.4 Descenso por Gradiente .....	4
2.5 Visualización de $J(\theta)$ .....	5
<b>3. Regresión lineal con múltiples variables.....</b>	<b>5</b>
3.1 Normalización de características .....	6
3.2 Descenso por gradiente.....	6
3.3 Ecuación normal .....	7
<b>4. Experimentación con <math>\alpha</math> y número de iteraciones .....</b>	<b>8</b>
• $\alpha = 0.01$ : .....	8
• $\alpha = 0.03$ : .....	8
• $\alpha = 0.1$ : .....	9
<b>5. Conclusiones .....</b>	<b>10</b>
<b>6. Entregables .....</b>	<b>10</b>

# 1. Introducción

En esta tarea trabajé con la implementación de regresión lineal utilizando **MATLAB**, tanto en el caso de una variable como en el caso multivariable. El objetivo fue entender cómo funciona el modelo, cómo se implementa el **descenso por gradiente** y cómo se compara con la solución obtenida por la **ecuación normal**.

El profesor nos proporcionó plantillas de funciones, las cuales ya incluían la estructura básica del código, y mi trabajo consistió en completarlas o modificarlas para que funcionaran correctamente.

Los archivos de datos utilizados fueron:

- **ej1data1.txt**: población de una ciudad vs. ganancias de un negocio ambulante (una variable).
- **ej1data2.txt**: tamaño de casa, número de cuartos vs. precio de la casa (múltiples variables).

## 2. Regresión Lineal con una Variable

### 2.1 Calentamiento

Lo primero fue completar el archivo **calentamiento.m**, donde debía devolver una **matriz identidad de 5x5**.

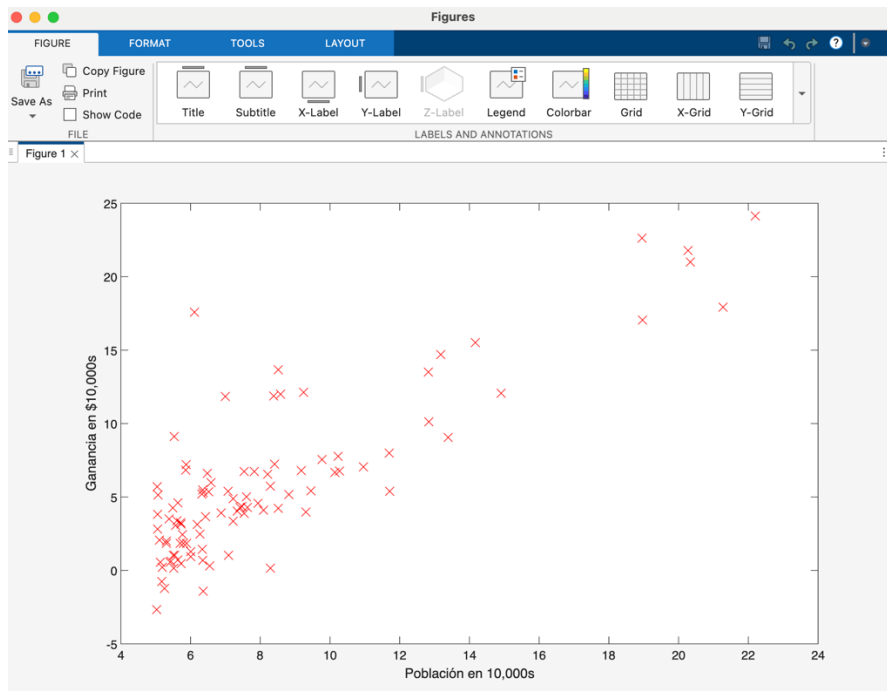
Al ejecutar ej1.m, obtuve en consola:

```
Corriendo ejercicio de calentamiento ...  
Matriz Identidad 5x5:
```

```
ans =  
|  
    1    0    0    0    0  
    0    1    0    0    0  
    0    0    1    0    0  
    0    0    0    1    0  
    0    0    0    0    1
```

### 2.2 Graficando Datos

Completé plotData.m para graficar el conjunto de datos. El scatter plot muestra en el eje **X** la población (en decenas de miles) y en el eje **Y** la ganancia (en decenas de miles de dólares).



## 2.3 Función de Costo

En `calculoCosto.m` implementé la función de costo. Con parámetros inicializados en ceros, el costo fue: **ans = 32.0727**  
**Lo cual coincide con lo esperado.**

## 2.4 Descenso por Gradiente

En `descensoXGradiente.m` implementé la regla de actualización. Al correr el script, obtuve:

**Corriendo Descenso por Gradiente ...**

**Theta encontrado por descenso por gradiente: -3.630291 1.166362**

**Para una población = 35,000, predecimos una ganancia de 4519.767868**

**Para una población = 70,000, predecimos una ganancia de 45342.450129**

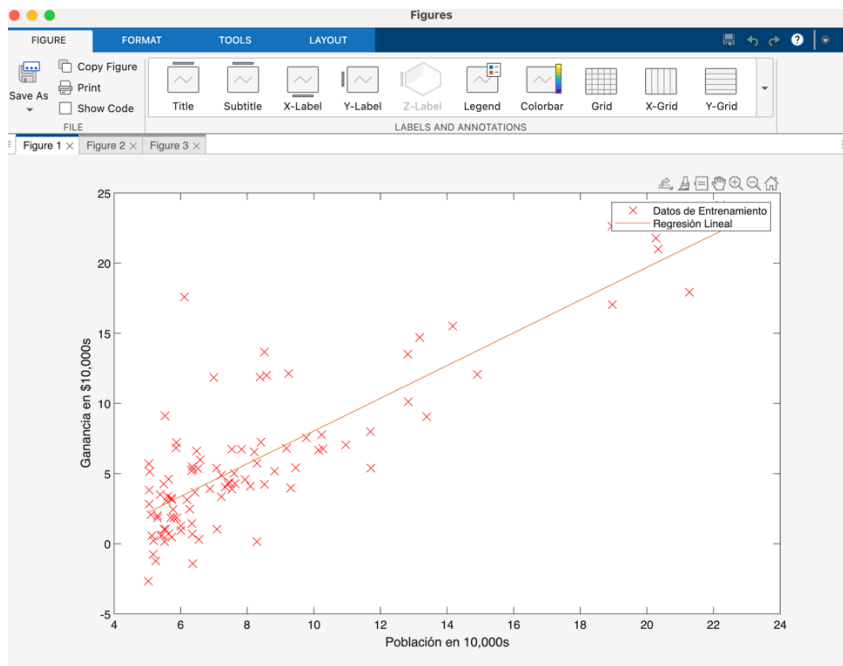
**El costo disminuyó hasta converger, confirmando que la implementación fue correcta.**

Programa en pausa. Oprima enter para continuar.  
 Corriendo Descenso por Gradiente ...

ans =

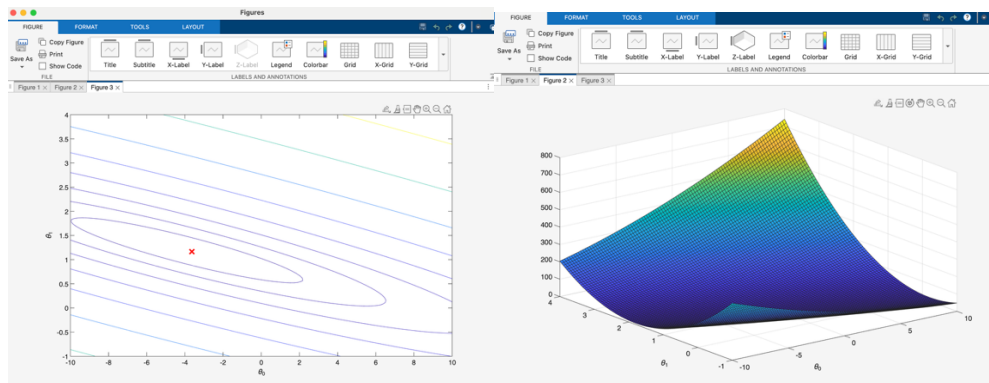
32.0727

Theta encontrado por descenso por gradiente: -3.630291 1.166362  
 Para una población = 35,000, predecimos una ganancia de 4519.767868  
 Para una población = 70,000, predecimos una ganancia de 45342.450129  
 Programa en pausa. Oprima enter para continuar.  
 Visualizando  $J(\theta_0, \theta_1)$  ...



## 2.5 Visualización de $J(\theta)$

Finalmente, visualicé la superficie y el diagrama de contorno de la función de costo. Ambas gráficas muestran un mínimo claro en el valor óptimo de  $\theta$ .



## 3. Regresión lineal con múltiples variables

Primero cargué los datos del archivo ej1data2.txt. En la consola se mostraron los **10 primeros ejemplos**:

$x = [2104 \ 3]$ ,  $y = 399900$

$x = [1600 \ 3]$ ,  $y = 329900$

$x = [2400 \ 3]$ ,  $y = 369000$

```
x = [1416 2], y = 232000
x = [3000 4], y = 539900
x = [1985 4], y = 299900
x = [1534 3], y = 314900
x = [1427 3], y = 198999
x = [1380 3], y = 212000
x = [1494 3], y = 242500
```

Programa en pausa. Oprima enter para continuar.

### 3.1 Normalización de características

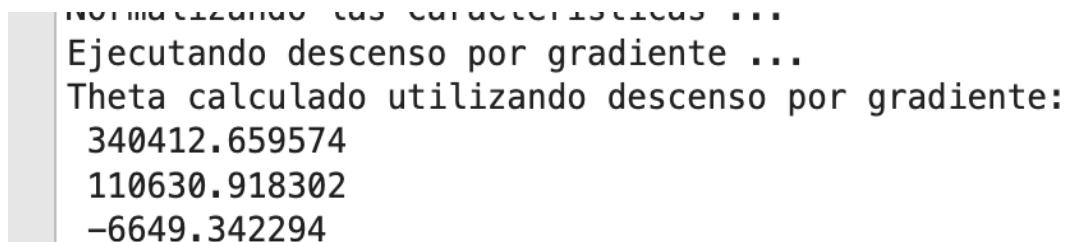
En esta parte implementé la función **normaliceCaracteristicas.m**. El objetivo fue llevar cada característica a una media de 0 y una desviación estándar de 1. Con esto, las variables quedaron en la misma escala, lo cual permitió que el descenso por gradiente convergiera más rápido.

### 3.2 Descenso por gradiente

Después de normalizar, ejecuté el descenso por gradiente con un número de iteraciones de 1000 y un valor de  $\alpha = 0.03$ . El resultado obtenido para los parámetros fue:

**Theta calculado utilizando descenso por gradiente:**

```
340412.659574
110630.918302
-6649.342294
```

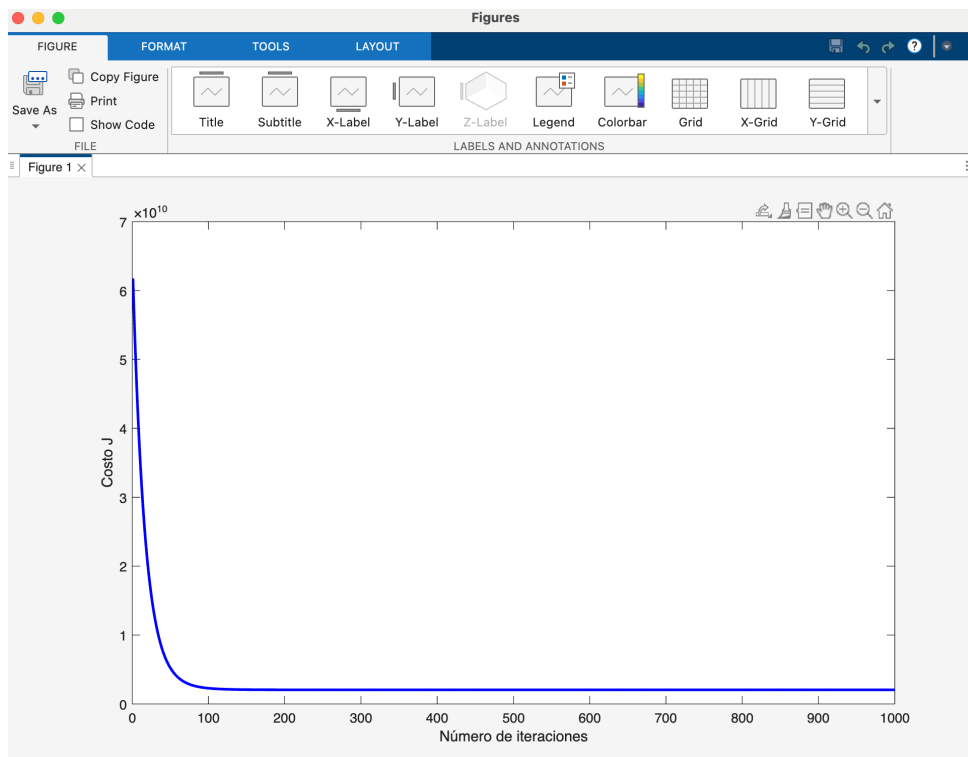


```
Normalizando las características ...
Ejecutando descenso por gradiente ...
Theta calculado utilizando descenso por gradiente:
340412.659574
110630.918302
-6649.342294
```

Con estos valores de  $\theta$ , la predicción para una casa de **1650 pies cuadrados y 3 cuartos** fue:

**\$293081.493053**

La siguiente imagen muestra la evolución del costo  $J(\theta)$  durante las iteraciones.



### 3.3 Ecuación normal

Para validar el resultado, resolví el problema utilizando la ecuación normal. Los valores obtenidos fueron:

**Theta calculado a partir de la ecuación normal:**

**89597.909545**

**139.210674**

**-8738.019113**

Resolver con la ecuación normal ...

Theta calculado a partir de la ecuación normal:

89597.909545

139.210674

-8738.019113

La predicción fue prácticamente la misma que con descenso por gradiente:

**\$293081.464335**

Esto confirma que ambas implementaciones son correctas y que efectivamente convergen al mismo resultado.

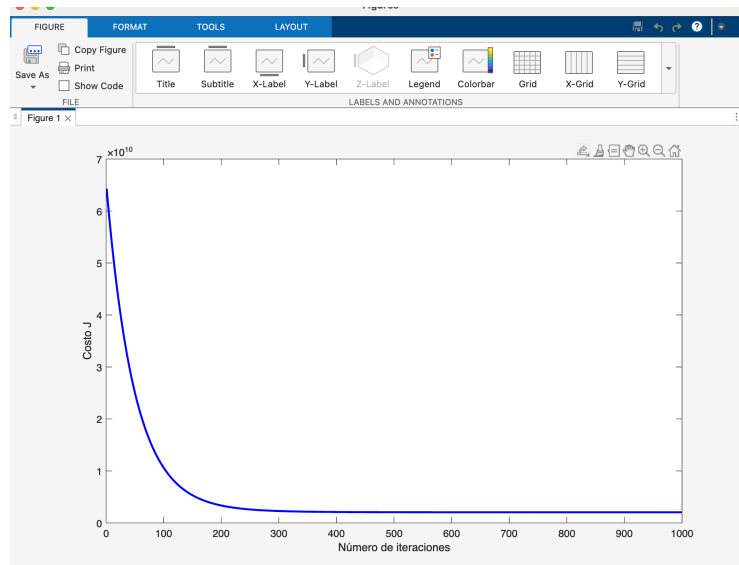
La predicción para el valor de una casa de 3 cuartos y 1650 pies cuadrados (utilizando ecuación normal):  
\$293081.464335

## 4. Experimentación con $\alpha$ y número de iteraciones

Durante la implementación de la regresión lineal multivariable realicé pruebas con diferentes valores de la tasa de aprendizaje  $\alpha$  y con distintos números de iteraciones para analizar cómo afectaban la convergencia del descenso por gradiente.

- **$\alpha = 0.01$ :**

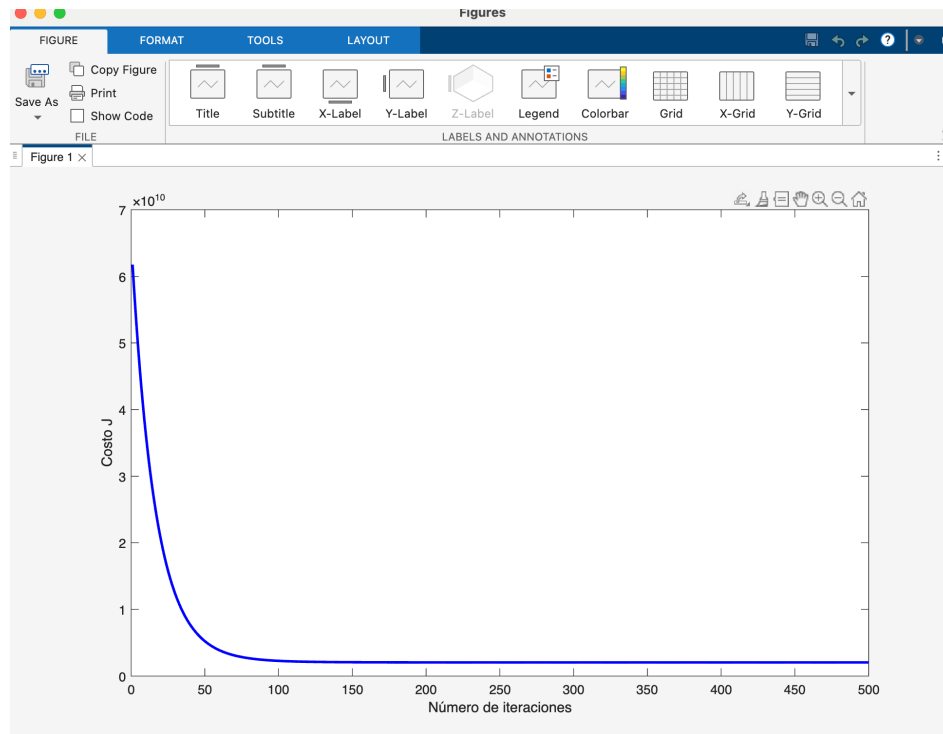
El costo decreció de manera estable, aunque la convergencia fue relativamente lenta. Después de  $\sim 1000$  iteraciones alcanzó un valor cercano al mínimo.



- **$\alpha = 0.03$ :**

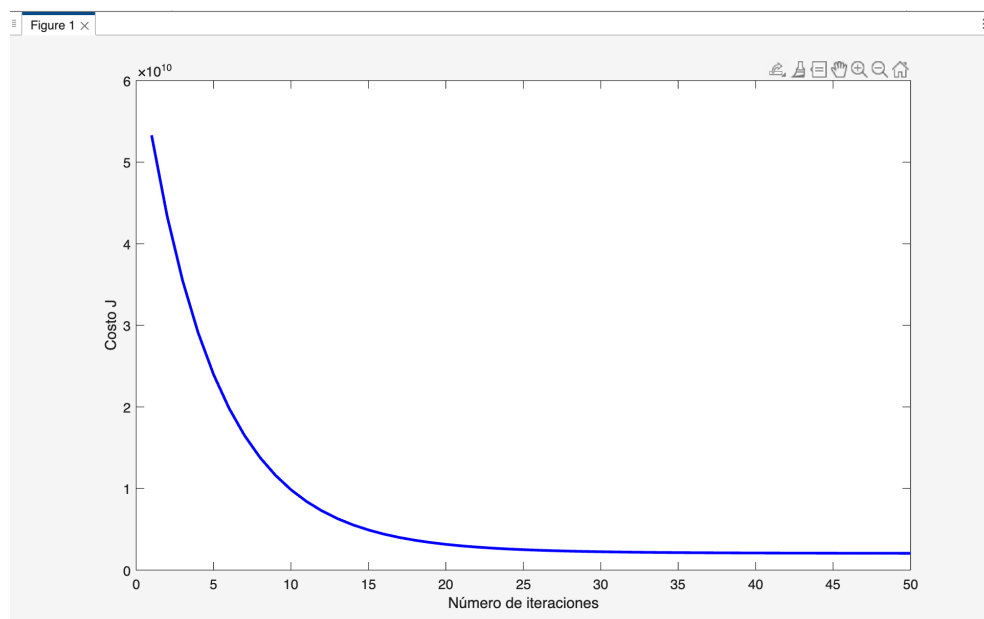
Se observó un buen balance entre velocidad de convergencia y estabilidad. El costo disminuyó rápidamente y alcanzó el mínimo en menos iteraciones en comparación con  $\alpha = 0.01$ .





- **$\alpha = 0.1$ :**

El algoritmo no logró converger. El costo comenzó a aumentar y en algunos casos explotó hacia valores muy grandes, indicando divergencia.



En cuanto al número de iteraciones, comprobé que:

- Con 400–500 iteraciones y  **$\alpha = 0.03$**  ya se alcanzaba un valor muy cercano al mínimo.

- Con 1000 iteraciones y  $\alpha = 0.01$  también se garantizó una convergencia estable, aunque más lenta.

👉 Por lo tanto, la mejor configuración que utilicé para esta tarea fue:

- $\alpha = 0.03$
- Iteraciones = 1000

Con estos parámetros obtuve los valores de  $\theta$  y la predicción final que coincidió con la obtenida por la ecuación normal, confirmando la validez del modelo.

## 5. Conclusiones

En esta tarea entendí mejor cómo funciona la regresión lineal y lo importante que es usar bien el descenso por gradiente. Me di cuenta de que normalizar las variables ayuda a que el algoritmo aprenda más rápido, que el valor de  $\alpha$  es clave para que el método sea estable, y que tanto el descenso por gradiente como la ecuación normal llegan al mismo resultado, lo cual confirma que todo se implementó bien.

- La normalización hizo que el modelo convergiera más rápido.
- El valor de  $\alpha$  fue decisivo: muy grande se descontrola, muy pequeño se hace lento.
- Ambos métodos dieron la misma predicción, lo que valida la implementación.

## 6. Entregables

1. Archivos MATLAB modificados
2. Documento PDF