Regresión lineal

Código

Regresión lineal con una variable

```
import matplotlib.pyplot as plt
import numpy as np
from pandas.io.parsers import read_csv
from mpl_toolkits.mplot3d import Axes3D
from matplotlib import cm

data = read_csv('p1/ex1data1.csv', header=None).values.astype(float)
x = data[:,0]
y = data[:,1]
size = len(y)
plt.scatter(x, y)
plt.xlabel("Population")
plt.ylabel("Income")
plt.show()

X = np.ones((size,2))
X[:,1] = x

theta = np.zeros((2, 1))
iterations = 1500
alpha = 0.01

def cost_function(X, y, theta):
    size = y.size
    pred = X.dot(theta).flatten()
    sse = (pred - y) ** 2
    J = (1.0 / (2 * size)) * sse.sum()
    return J

def gradient_descent(X, y, theta, alpha, iterations):
    size = y.size
    Js = np.zeros((iterations, 1))
    thetas = np.zeros((iterations, 2))
    for i in range(iterations):
        pred = X.dot(theta).flatten()
        err_theta0 = (pred - y) * X[:, 0]
        err_theta1 = (pred - y) * X[:, 1]
        theta[0][0] = theta[0][0] - alpha * (1.0 / size) * err_theta0.sum()
        theta[1][0] = theta[1][0] - alpha * (1.0 / size) * err_theta1.sum()
        Js[i, 0] = cost_function(X, y, theta)
        thetas[i][0] = theta[0][0]
        thetas[i][1] = theta[1][0]
    return theta, Js, thetas

result = gradient_descent(X, y, theta, alpha, iterations)
plt.scatter(x, y)
plt.xlabel("Population")
plt.ylabel("Income")
plt.show()
linspace = np.linspace(5, 22.5, 1000)
plt.scatter(x, y)
```

```
plt.plot(linspace, result[0][0] + linspace*result[0][1], color='red')
plt.xlabel("Population")
plt.ylabel("Income")
plt.show()
plt.plot(range(iterations), result[1])
plt.xlabel("Epochs")
plt.ylabel("Cost function")
plt.show()

fig = plt.figure()
ax = fig.gca(projection='3d')
a = np.arange(-10, 10, 0.05)
b = np.arange(-1, 4, 0.05)
A, B = np.meshgrid(a, b)
zs = np.array([cost_function(X, y, np.vstack((t0, t1))) for t0,t1 in
zip(np.ravel(A), np.ravel(B))])
Z = zs.reshape(A.shape)

ax.plot_surface(A, B, Z)
ax.set_xlabel("Theta0")
ax.set_ylabel("Theta1")
ax.set_zlabel("Cost function")
plt.show()

theta0 = np.linspace(-10, 10, 100)
theta1 = np.linspace(-1, 4, 100)
J = np.zeros((theta0.size, theta1.size))

for num0, th0 in enumerate(theta0):
    for num1, th1 in enumerate(theta1):
        J[num0, num1] = cost_function(X, y, np.vstack((th0, th1)))

J = J.T
plt.contour(theta0, theta1, J, np.logspace(-2, 3, 20))
for i in range(iterations):
    if i%100 == 0:
        plt.scatter(result[2][i][0], result[2][i][1], color ='blue')
plt.scatter(result[0][0], result[0][1], color='red', marker='x')
plt.xlabel("Theta0")
plt.ylabel("Theta1")
plt.show()
```

Regresión lineal con varias variables

```
import matplotlib.pyplot as plt
import numpy as np
from pandas.io.parsers import read_csv

data = read_csv('p1/ex1data2.csv', header=None).values.astype(float)
x = data[:,:2]
y = data[:,2]
size = y.size
y.shape = (size, 1)

def normalize(X):
    mu = []
    sigma = []
    X_norm = X
    features = X.shape[1]
```

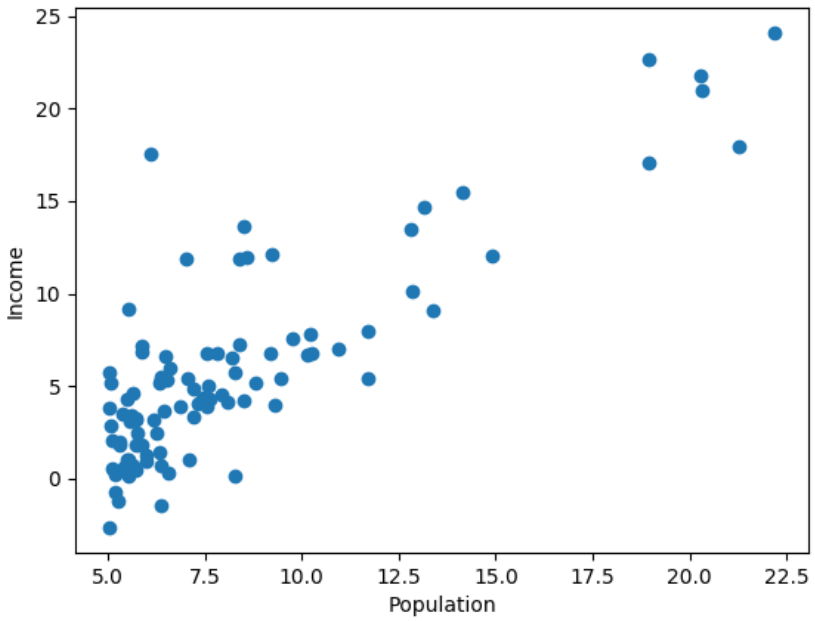
```
    for i in range(features):
        mean = np.mean(X[:, i])
        std_dev = np.std(X[:, i])
        mu.append(mean)
        sigma.append(std_dev)
        X_norm[:, i] = (X_norm[:, i] - mean) / std_dev
    return X_norm, mu, sigma

x, mu, sigma = normalize(x)

X = np.ones((size,3))
X[:,1:3] = x

theta = np.zeros((3, 1))
iterations = 1500
alpha = 0.01

def cost_function(X, y, theta):
    size = y.size
    pred = X.dot(theta)
    sse = (pred - y)
    J = (1.0 / (2 * size)) * sse.T.dot(sse)
    return J

def gradient_descent(X, y, theta, alpha, iterations):
    size = y.size
    Js = np.zeros((iterations+1, 1))
    Js[0, 0] = cost_function(X, y, theta)
    for i in range(iterations):
        pred = X.dot(theta)
        theta_size = theta.size
        for j in range(theta_size):
            xj = X[:, j]
            xj.shape = (size, 1)
            err = (pred - y) * xj
            theta[j][0] = theta[j][0] - alpha * (1.0 / size) * err.sum()
        Js[i+1, 0] = cost_function(X, y, theta)
    return theta, Js

result = gradient_descent(X, y, theta, alpha, iterations)
prediction = np.array([1.0, ((1650.0 - mu[0]) / sigma[0]), ((3 - mu[1]) /
sigma[1])]).dot(theta)
print("Prediction: %s" % prediction)

alphas = [round(0.1 * 0.1**i, i+2) for i in range(4)]
print("Gradient descent for different values of alpha")
for alpha in alphas:
    theta = np.zeros((3, 1))
    result1 = gradient_descent(X, y, theta, alpha, iterations)
    theta = np.zeros((3, 1))
    result2 = gradient_descent(X, y, theta, alpha*3, iterations)
    r1 = plt.plot(range(iterations+1), result1[1], label=alpha)
    r2 = plt.plot(range(iterations+1), result2[1], label=alpha*3)
    plt.xlabel("Epochs")
    plt.ylabel("Cost function")
    plt.legend()
    plt.show()
```

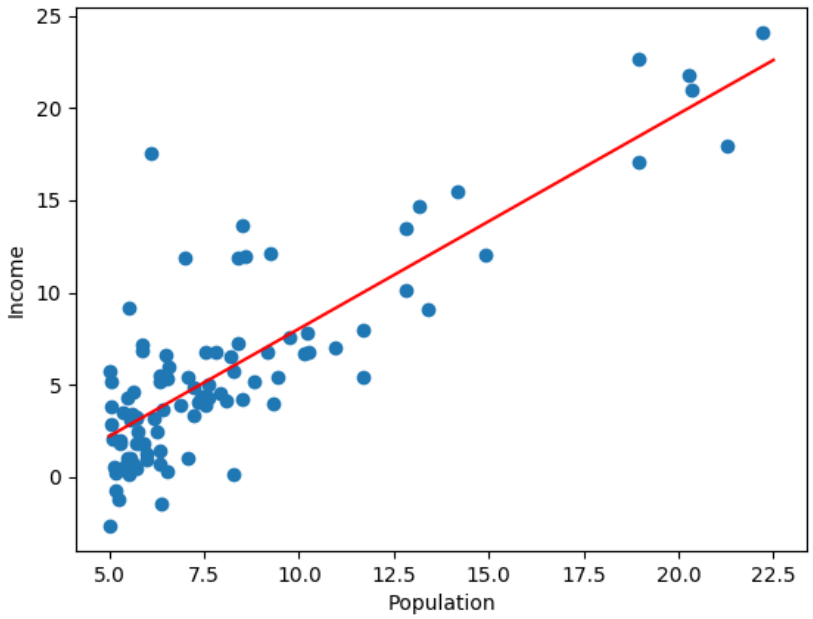
Ecuación normal

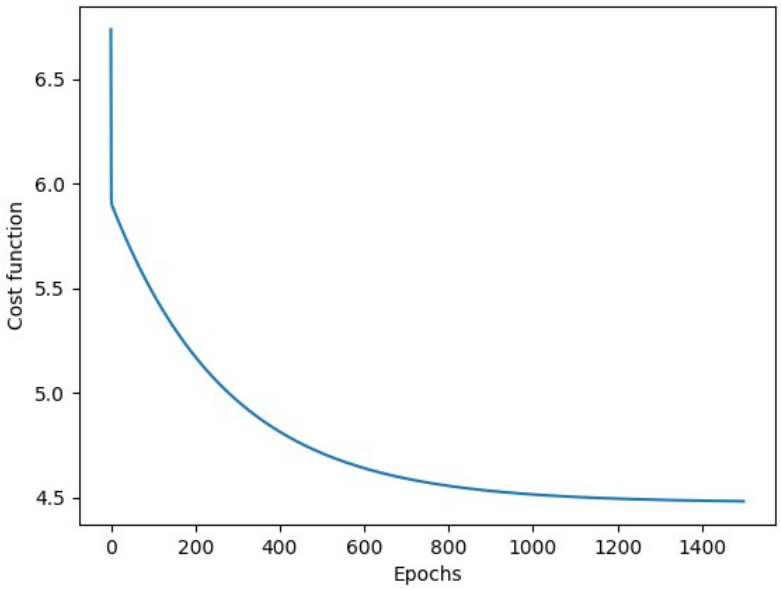
```
import matplotlib.pyplot as plt
import numpy as np
from pandas.io.parsers import read_csv

data = read_csv('p1/ex1data2.csv', header=None).values.astype(float)
x = data[:,:2]
y = data[:,2]
size = y.size
y.shape = (size, 1)

X = np.ones((size,3))
X[:,1:3] = x

def normalEquation(X, y):
    return np.matmul(np.linalg.pinv(np.matmul(np.transpose(X), X)),
np.matmul(np.transpose(X), y))

normalEquationTheta = normalEquation(X, y)
normalEquationPrediction = np.array([1.0, 1650.0, 3]).dot(normalEquationTheta)
print("Normal equation prediction: %s" % normalEquationPrediction)
```

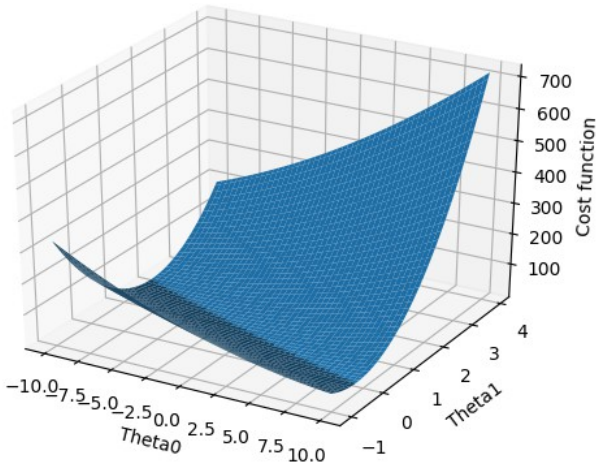
Resultados

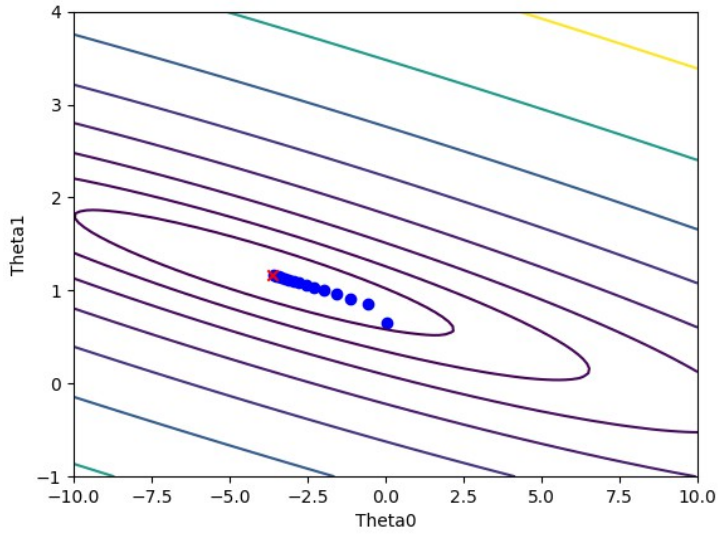
Regresión lineal con una variable




Income
Population
25
20
15
10
5
0
5.0
7.5
10.0
12.5
15.0
17.5
20.0
22.5

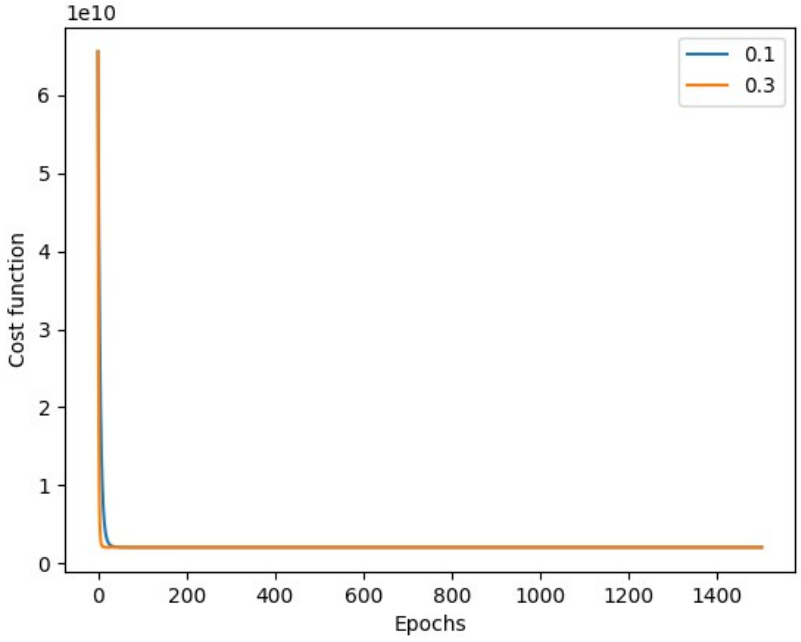

Cost function
Epochs
6.5
6.0
5.5
5.0
4.5
0
200
400
600
800
1000
1200
1400

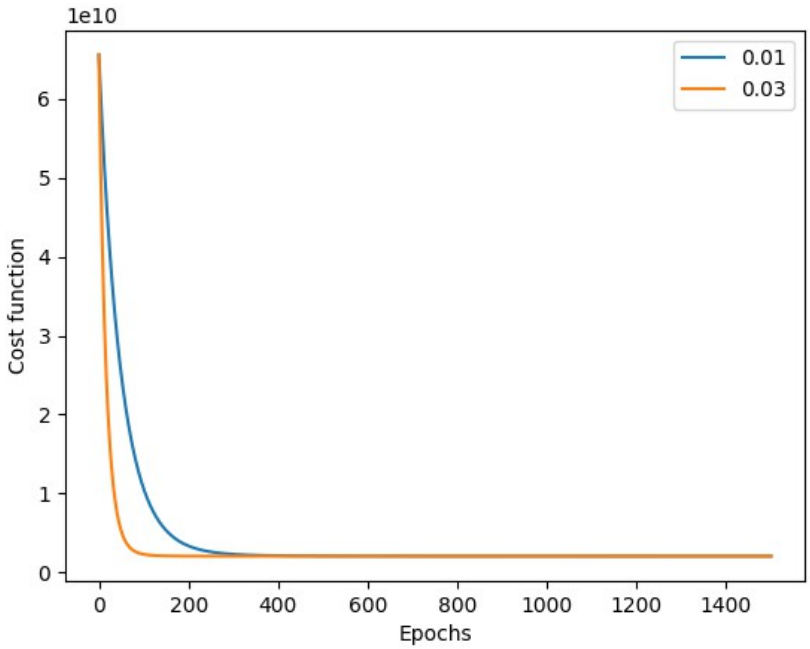

Cost function
Theta0
Theta1
700
600
500
400
300
200
100
−10.0
−7.5
−5.0
−2.5
0.0
2.5
5.0
7.5
10.0
−1
0
1
2
3
4

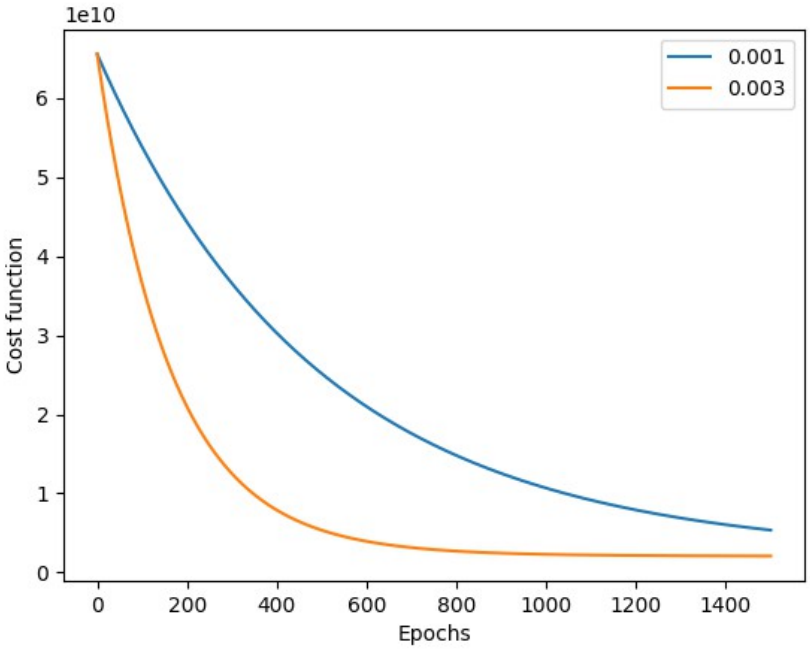


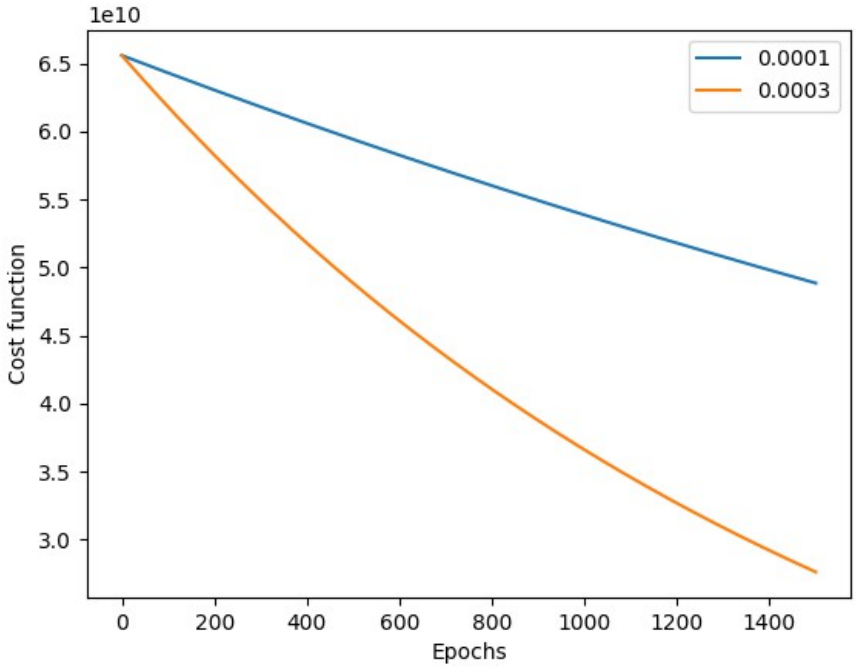
Regresión lineal con varias variables

Gradient descent for different values of alpha




1e10
0.01
0.03
Cost function
Epochs
0
1
2
3
4
5
6
0
200
400
600
800
1000
1200
1400


1e10
0.001
0.003
Cost function
Epochs
0
1
2
3
4
5
6
0
200
400
600
800
1000
1200
1400




```
Prediction: [293098.46667577]
```

Ecuación normal

```
Normal equation prediction: [293081.46433497]
```

Comentarios

Cuando la tasa de aprendizaje se aumenta, el valor de función de coste se disminuye más rápido. La predicción obtenida en regresión lineal con varias variables es conforme a la obtenida en ecuación normal.