Tarea4_Regresion_Iterativa_Regularizada

December 14, 2017

1 Tarea 4. Regresión Dinámica.

L.E. Rojón
138442

```
In [1]: import pandas as pd
        import numpy as np

        from sklearn.model_selection import train_test_split
        from sklearn.metrics import confusion_matrix
        from scipy.stats import norm
        from sklearn import preprocessing
        from random import random

        import matplotlib.pyplot as plt
        from sklearn import datasets, linear_model
        from sklearn.metrics import mean_squared_error, r2_score
```

```
In [2]: data = pd.read_csv("regLin.csv")
        data.describe()
```

Out[2]:

	X	y
count	1029.000000	1029.000000
mean	50.008111	32.893917
std	29.055066	18.083566
min	0.030369	-4.586608
25%	25.201087	17.583093
50%	50.884987	33.564129
75%	75.636823	48.339811
max	99.959580	71.762748

División de datos.

```
In [3]: X_train, X_test, Y_train, Y_test = train_test_split(data[['X']], data[['y']], train_size
        X_train, X_test, Y_train, Y_test = np.array(X_train), np.array(X_test), np.array(Y_train
```

Estandarizamos los datos.

```
In [5]: scaler = preprocessing.StandardScaler().fit(X_train)
        X_train=scaler.transform(X_train)
        X_test=scaler.transform(X_test)
```

Añadimos una columna de unos para agregar el coeficiente de la intersección de la regresión.

```
In [6]: X = np.concatenate((np.ones((len(X_train), 1)), X_train), axis=1)
        w = np.ones((1,len(X[0])))
        eta = 0.1
        lam = 0
```

Hacemos una función para calcular la regresión lineal iterativa con parámetros eta y lambda=0 (por el momento sin regularizar).

```
In [9]: def RegresionLinealIterativa(X, y, w, eta, lam):
            for i in range(len(X)):
                v_i=np.dot(w,X[i])
                error_i = eta*(y[i]-v_i)
                for j in range(len(w)+1):
                    w[0][j] += error_i*X[i][j] - lam*np.mean(X[i])
            return w

In [10]: w = RegresionLinealIterativa(X, Y_train,w,eta,lam)
         w

Out[10]: array([[ 33.80051658,  16.47706183]])
```

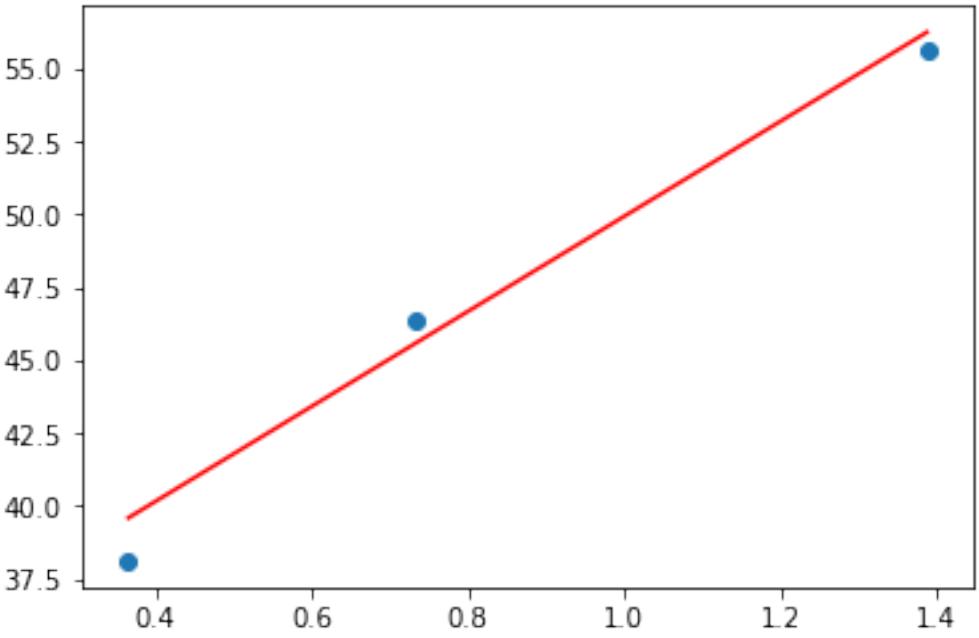
Ahora, comparamos la regresión iterativa con el procedimiento usual de regresión con mínimos cuadrados.

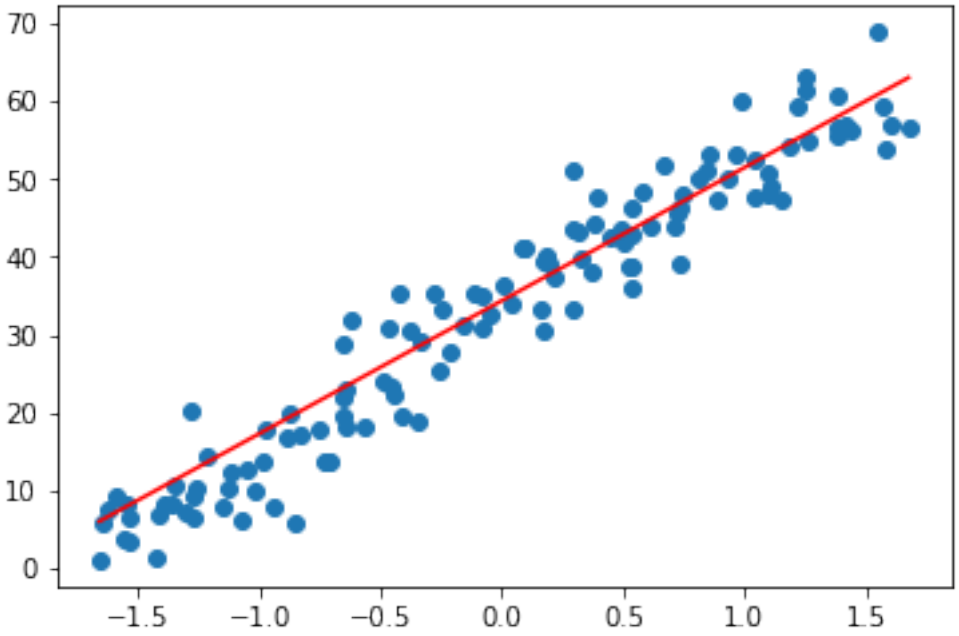
```
In [11]: reg = linear_model.LinearRegression()
         linreg = reg.fit(X_train, Y_train)
         b0, b1 = linreg.intercept_, linreg.coef_
         print b0, b1

[ 33.22119462] [[ 17.74869343]]
```

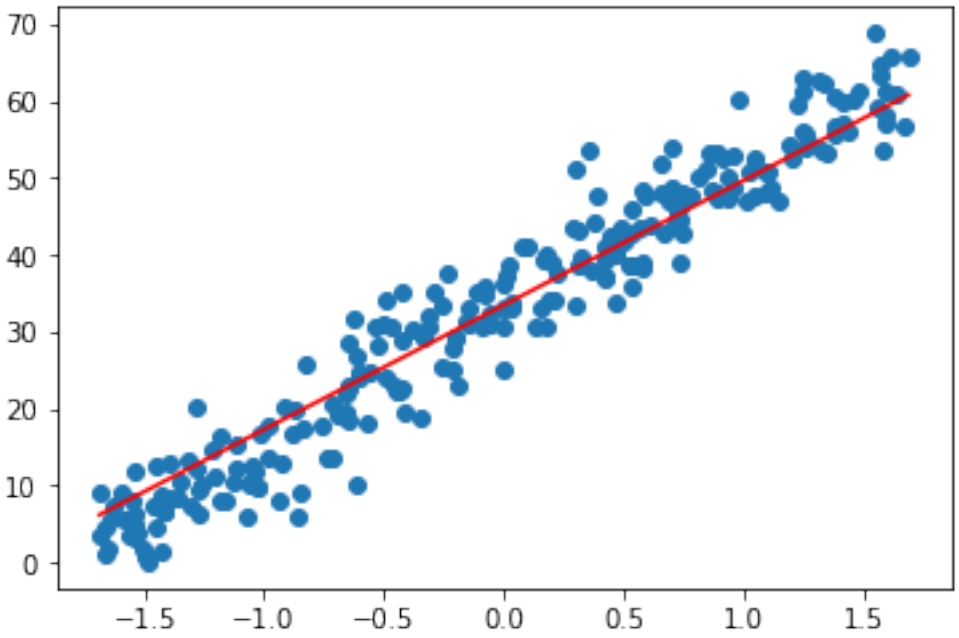
Finalmente, hacemos cuatro gráficas donde se muestra la progresión del ajuste al ir alimentando la función iterativa cada 120 datos.

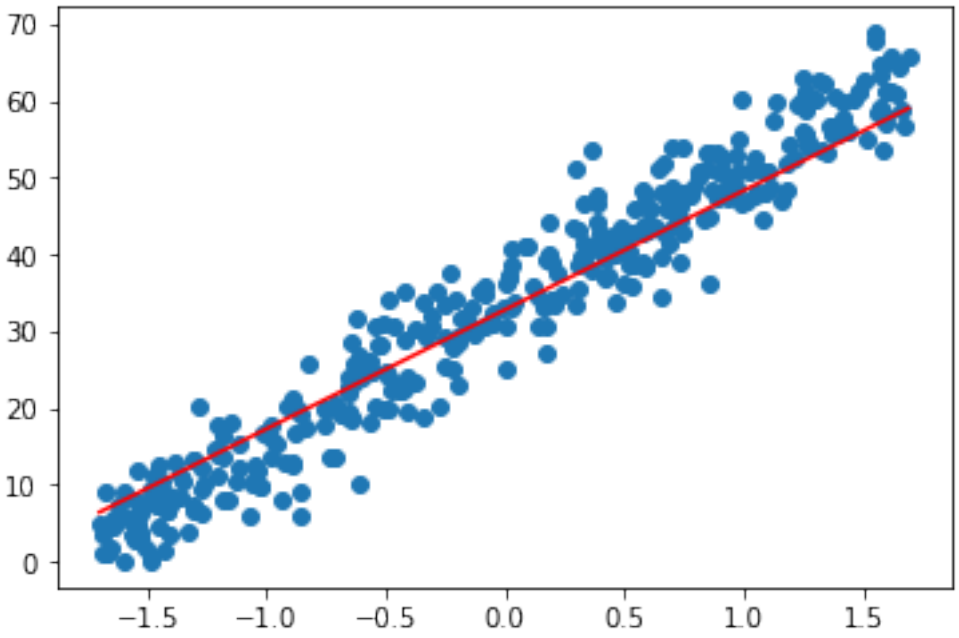
```
In [12]: for i in range(len(X)):
             V = np.dot(w, X[i])
             error = Y_train-V
             for j in range (len(w)):
                 w[j] += eta * (error[i]) * X[i][j]
             if i % 120 == 2:
                 plt.scatter(X_train[:i+1], Y_train[:i+1])
                 x0 = np.amin(X_train[:i+1])
                 x1 = np.amax(X_train[:i+1])
```

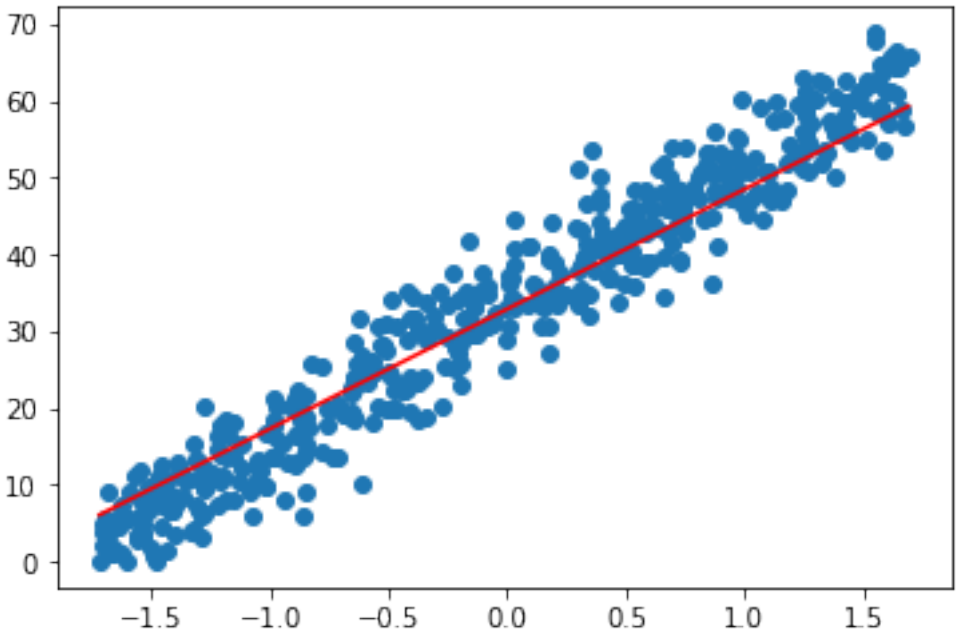
```
b = w[0][0]
m = w[0][1]
y0 = m*x0 + b
y1 = m*x1 + b
plt.plot([x0, x1], [y0, y1], c='r')
plt.show()
```

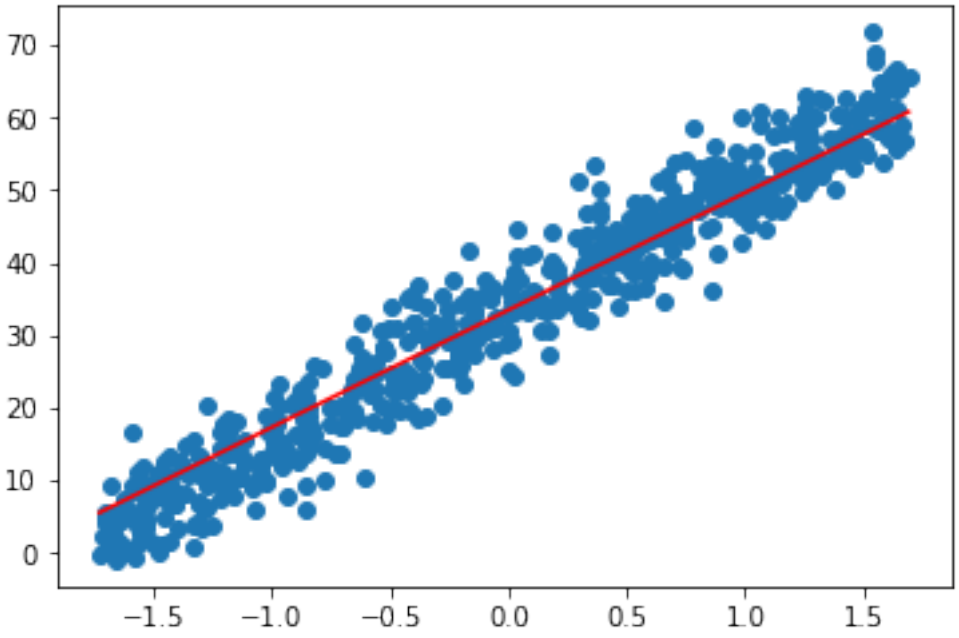


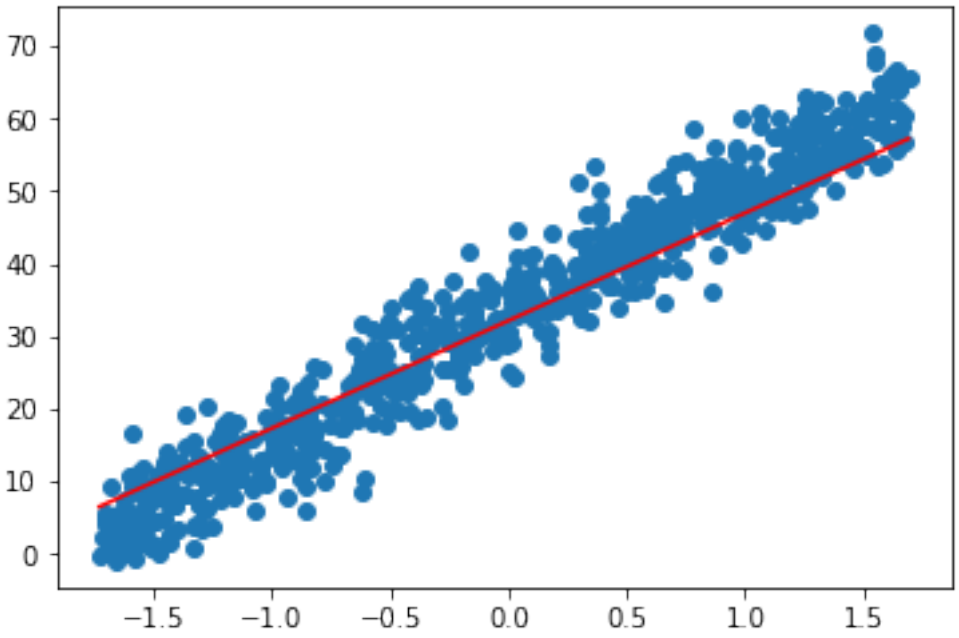



70
60
50
40
30
20
10
0
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5


70
60
50
40
30
20
10
0
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5


70
60
50
40
30
20
10
0
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5


70
60
50
40
30
20
10
0
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5


70
60
50
40
30
20
10
0
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5