Scikit-Learn: Linear Regression (housing example)

Rao Vemuri

Statement

- In this exercise, you will investigate multivariate linear regression (using gradient descent).
- (You will also examine the relationship between the cost function, J(ϑ), the convergence of gradient descent, and the learning rate α .)

Data Files

- Download (and unzip) the data file
- This is a training set of housing prices in a city in the USA, where the outputs $y^{(i)}$ are the prices and the inputs are the living area and the number of bedrooms. There are m = 47 training examples.
- CAUTION: This is only an outline of how one can write a program. Details will differ and you have to make sure that your commands work!

Housing Data Sample

No	Sq ft	price
1	150	6450
2	200	7450
3	250	8450
4	300	9450
5	350	11450
6	400	15450
7	600	18450

There are 47 houses in the data file. The above is just a sample

Preprocess Data

- Load the data for the training examples into your program and add the x_0 =1 intercept term into your x matrix. Recall that the command in Matlab for adding a column of 1's is

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

- A similar command in Python would be

```
X = np.array([1,2,3],[2,3,4])
ones =np.ones((2,1), dtype=int64)
np.append(X,ones, axis=1)
```

Normalize Data

- Take a look at the values of the inputs and note that the living areas are about 1,000 times the number of bedrooms. This difference means that preprocessing the inputs will significantly increase gradient descent's efficiency.
- In your program, scale both types of inputs by their standard deviations and set their means to zero. In Matlab this can be executed with

```
sigma = std(x); mu = mean(x);
x(:,2) = (x(:,2) - mu(2))./ sigma(2);
x(:,3) = (x(:,3) - mu(3))./ sigma(3);
```

Input Data as CSV

```
# input_data.csv
#square_feet;price
#150;6459
#200;7450
#250;
#300;
.......
```

Hypothesis

- For linear regression

$$h_{\vartheta}(x) = \vartheta_o + \vartheta_1 x$$

Coding in Python: Step 1

- Open your favorite text editor or Jupyter notebook
- Name your program as predict_house_price.py
- We need the following packages. So type them into your predict_house_price.py
- (type them in, not copy and paste)

```
#Required packages
Import matplotlib.pyplot as plt
Import numpy as np
Import pandas as pd
from sklearn import datasets, liner_model
```

Run this code once to make sure you have all the packages

Step 2: Function to Get Data

```
def get_data (file_name):
   data   pd.read_csv(file_name, sep ";")
   X_parameter  [ ]
   Y_parameter [ ]
#replace the names of the fields 'square feet' ... and 'price' to
#suit your application
   for single_square_feet in data['square_feet']:
   X_parameter.append([float(single_square_feet)])
   for single_price_value in data['price']:
   Y_parameter.append([float(single_price_value)])
return    X_parameter , Y_parameter
```

Print X and Y

Print X_parameters and Y_parameters

1. X,Y = get_data ('input_data.csv')
2. print X
3. print Y

Script output

[[150.0], [200.0], [250.0], [300.0], [350.0], [400.0], [600.0]]

[6450.0, 7450.0, 8450.0, 9450.0, 11450.0, 15450.0, 18450.0,]

We converted the data to X_parameters and Y_parameters, let us find the parameters theta

Regression Code

```
# Function for fitting data to linear model
def linear_model_main(X_parameters, Y_parameters, predict_value):
# Create linear regression object
regr = linear_model.LinearRegression( )
regr.fit (X_parameters, Y_parameters)
predict.outcome = regr.predict(predict_value)
predictions { }
predictions [‘intercept’] = regr_intercept_
predictions [‘coefficient’] = regr_coef_
predictions [‘predicted_value’] = predict_outcome
return predictions
```

Explanation

- Lines 4, 5: we are creating a linear model and training it with X_parameters and Y_parameters
- Lines 7-11: we are creating a dictionary, entitled predictions, and storing the values of theta 0, theta 1 & predicted values and returning the predicted dictionary as an output

Test

#Test with a predict_value =700

```
X,Y = getdata (input_data.csv)
predict_value=700
result = linear_model_main(X,Y,predict_value)
print “intercept_value”, result [‘intercept’]
print “coefficient”, result [‘coefficient’]
print “predcited_value”, result [‘predicted_value’]
```

Intercept value 1771.80851064

Coefficient [28.77658574]

Predicted value = [21915.42553191]

How Good is the Fit?

1. def show_linear_line(X_parameters, Y_parameters):
2. regr = linear_model.LinearRegression( )
3. regr.fit (X_parameters, Y_parameters)
4. plt.scatter(X_parameters, Y_parameters, color = ‘blue’)
4. plt.plot(X_parameters, regr.predict(X_parameters), color = ‘red’, linewidth)
5. plt.xticks = (())
5. plt.yticks = (())
6. plt.show = ( )

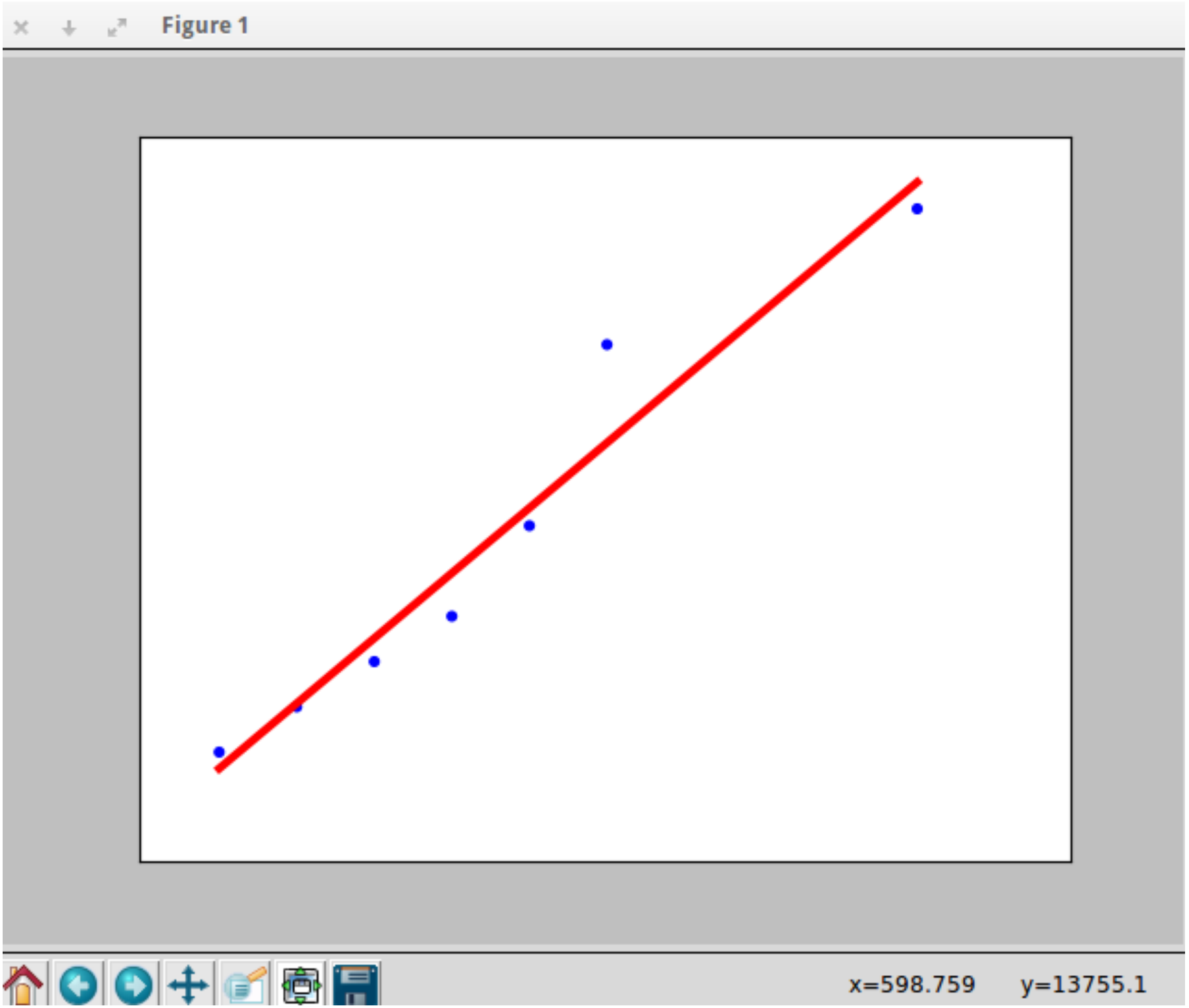
Figure 1
x=598.759
y=13755.1

Scatter Plot

```
from numpy import loadtxt,  zeros, ones, array, linspace,
logspace
from  pylab import scatter, show, title, xlabel, ylabel,
plot, contour

#load the dataset
data = loadtxt (ex1data1.txt, delimiter = ‘,’)

#plot the data
scatter (data[:,0], data [:,1], marker = ‘o’, c = ‘b’)
title (‘Profits Distribution’)
xlabel = (‘Population of City in 10,000s’)
ylabel = (‘Profit in $10,000s’)
Show ()
```