2.1.simple_linear_regression

September 15, 2020

1 Machine Learning Course

1.0.1 Part 2: Regression

Simple linear regression The most esay way that a dataset can be related is with a linear regression, mathematically:

$$y = b + ax$$

In this lecture we are going to learn how to do a simple linear regression with python.

Firstly (as always), import the basic libraries:

[1]:

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
```

And now import the dataset:

[2]:

```
dataset = pd.read_csv('Salary_Data.csv')
X = dataset.iloc[:,:-1].values
Y = dataset.iloc[:,-1].values
print(X)
print(Y)
```

```
[[ 1.1]
 [ 1.3]
 [ 1.5]
 [ 2. ]
 [ 2.2]
 [ 2.9]
 [ 3. ]
 [ 3.2]
 [ 3.2]
 [ 3.7]
 [ 3.9]
 [ 4. ]
 [ 4. ]
 [ 4.1]
 [ 4.5]
 [ 4.9]
 [ 5.1]
 [ 5.3]
 [ 5.9]
 [ 6. ]
 [ 6.8]
 [ 7.1]
 [ 7.9]
 [ 8.2]
 [ 8.7]
 [ 9. ]
 [ 9.5]
 [ 9.6]
 [10.3]
 [10.5]]
[ 39343.  46205.  37731.  43525.  39891.  56642.  60150.  54445.  64445.
  57189.  63218.  55794.  56957.  57081.  61111.  67938.  66029.  83088.
  81363.  93940.  91738.  98273. 101302. 113812. 109431. 105582. 116969.
 112635. 122391. 121872.]
```

Split the dataset in train set and test set:

[3]:

```python
from sklearn.model_selection import train_test_split
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=1/3,␣
 ↪random_state=0)
```

For regression we are going to use a funciton called *LinearRegression* from *sckit-learn* library:

[4]:

```python
from sklearn.linear_model import LinearRegression
regressor = LinearRegression()
regressor.fit(X_train, Y_train)
```

[4]:

```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=False)
```

Now, if we want to make prediction we only have to do (for example, with test data):

[5]:

```python
y_pred = regressor.predict(X_test)
```

Finally we want to visualize the regression. Red dots are the real data and the blue line is the prediction regression.

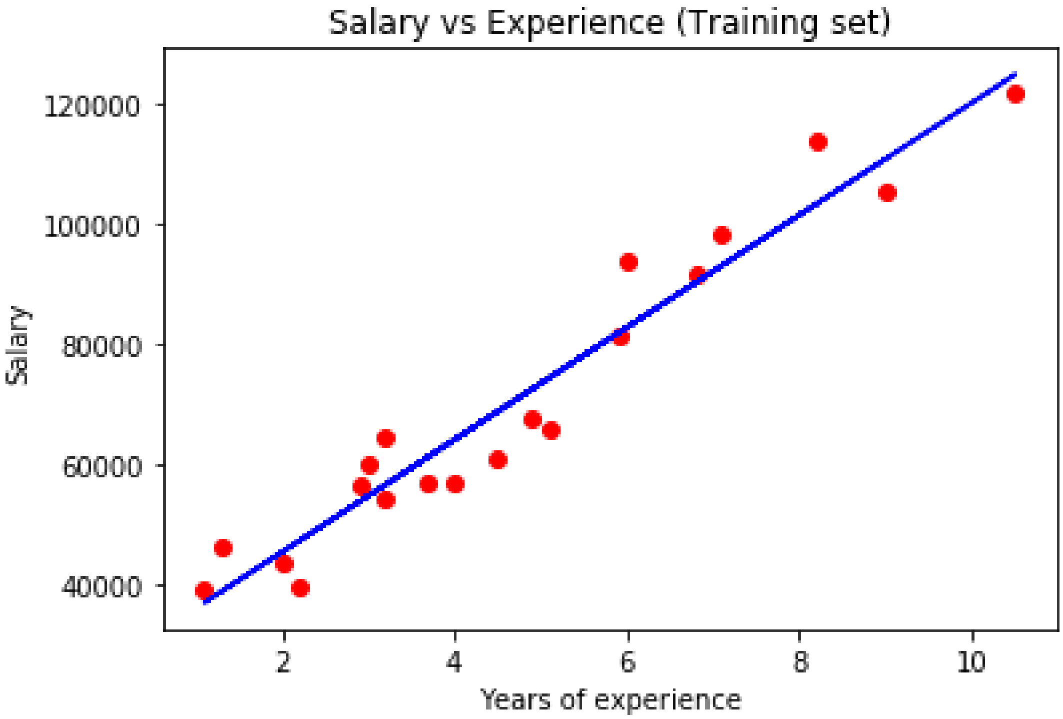
Use *matplotlib* library to get the figures.

First we generate the train data with train regression:

[6]:

```python
plt.scatter(X_train,Y_train,color='red')
plt.plot(X_train, regressor.predict(X_train), color='blue')
plt.title('Salary vs Experience (Training set)')
plt.xlabel('Years of experience')
plt.ylabel('Salary')
```

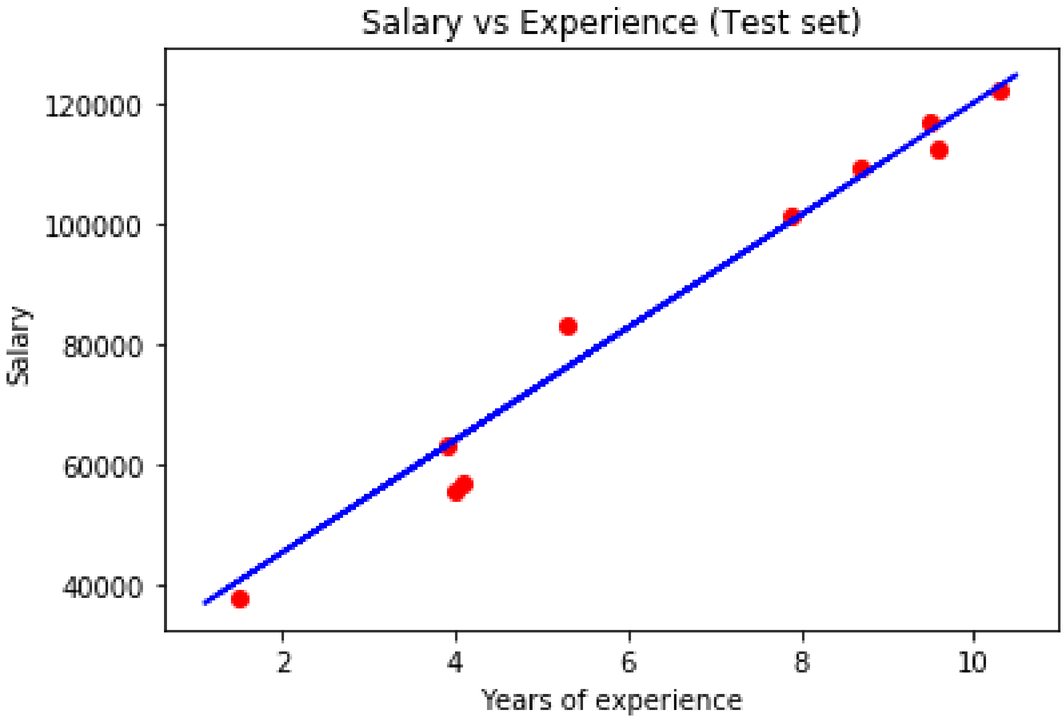
[6]: Text(0, 0.5, 'Salary')



And now the train regression with test data:

```
[7]: plt.scatter(X_test,Y_test,color='red')
     plt.plot(X_train, regressor.predict(X_train), color='blue')
     plt.title('Salary vs Experience (Test set)')
     plt.xlabel('Years of experience')
     plt.ylabel('Salary')
```

```
[7]: Text(0, 0.5, 'Salary')
```



And we can see that it is a valid regressin, because fits good with test data.

If we want to know the parameters **a** and **b** of our regression we can do:

```
[8]: a = regressor.coef_
     b = regressor.intercept_
     print(a)
     print(b)
```

```
[9345.94244312]
26816.19224403119
```