8C_LR_SVM

January 28, 2021

0.1 Task-C: Regression outlier effect.

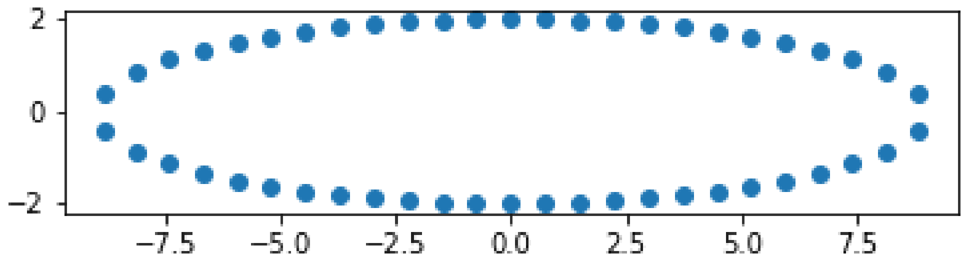
Objective:Visualization best fit linear regression line for different scenarios

[ ]:
```
# you should not import any other packages
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings("ignore")
import numpy as np
from sklearn.linear_model import SGDRegressor
```

[ ]:
```
import numpy as np
import scipy as sp
import scipy.optimize

def angles_in_ellipse(num,a,b):
    assert(num > 0)
    assert(a < b)
    angles = 2 * np.pi * np.arange(num) / num
    if a != b:
        e = (1.0 - a ** 2.0 / b ** 2.0) ** 0.5
        tot_size = sp.special.ellipeinc(2.0 * np.pi, e)
        arc_size = tot_size / num
        arcs = np.arange(num) * arc_size
        res = sp.optimize.root(
            lambda x: (sp.special.ellipeinc(x, e) - arcs), angles)
        angles = res.x
    return angles
```

[ ]:
```
a = 2
b = 9
n = 50

phi = angles_in_ellipse(n, a, b)
e = (1.0 - a ** 2.0 / b ** 2.0) ** 0.5
arcs = sp.special.ellipeinc(phi, e)

fig = plt.figure()
ax = fig.gca()
```

```
ax.axes.set_aspect('equal')
ax.scatter(b * np.sin(phi), a * np.cos(phi))
plt.show()
```



```
[ ]: X= b * np.sin(phi)
     Y= a * np.cos(phi)
```

```
[ ]:
```

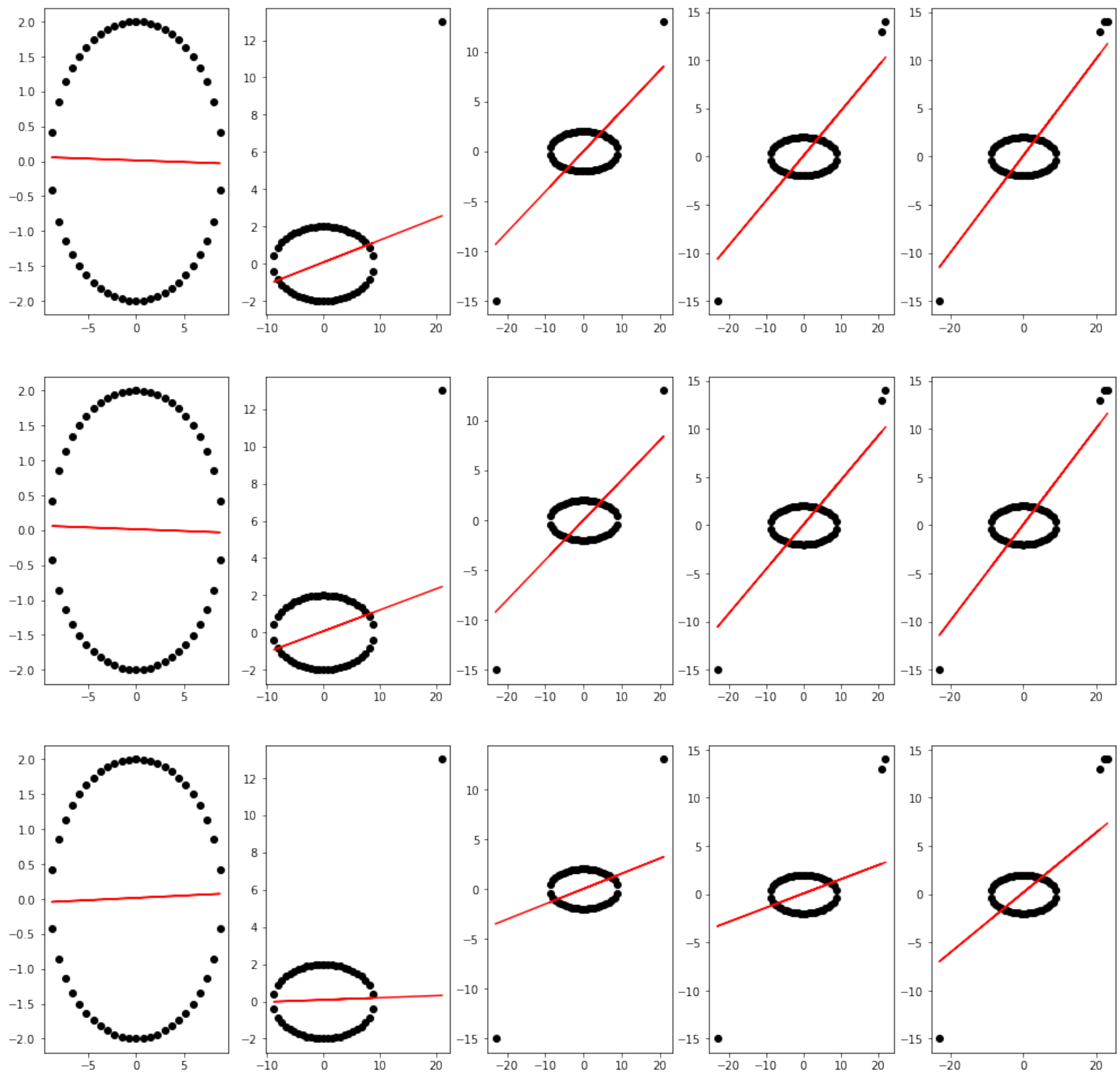
```
[ ]:
```

```
[ ]: regressor = SGDRegressor(eta0=0.
      ↪001,learning_rate='constant',random_state=0,alpha=0.001)
```

```
[ ]:
```

```
[ ]: #fig,ax = plt.subplots(len(alphas),len(outliers),figsize=(17,17))
```

```
[ ]: X= b * np.sin(phi)
     Y= a * np.cos(phi)
     alphas = [0.0001,1,100]
     outliers = [(0,2),(21,13),(-23,-15),(22,14),(23,14)]
     fig,ax = plt.subplots(len(alphas),len(outliers),figsize=(17,17))
     for row,reg in enumerate(alphas):
       model = SGDRegressor(eta0=0.
      ↪001,learning_rate='constant',random_state=0,alpha=reg)
       X= b * np.sin(phi)
       Y= a * np.cos(phi)
       #print('initial',X.shape,Y.shape)
       for col,out in enumerate(outliers):
         X,Y = np.append(X,out[0]),np.append(Y,out[1])
         model.fit(X.reshape(-1,1),Y)
         ax[row,col].scatter(X,Y,color='black')
         ax[row,col].plot(X,model.predict(X.reshape(-1,1)),color='red')
       #print('final',X.shape,Y.shape)
```



[ ]:

OBS

We know that to avoid overfitting the model to data, we add regularization term as a penality. we have L1 and L2 regularization terms where L1-creates sparsity in the weight vector.

[0.0001,1,100] when regularization parameter is = 0.0001, less weightage is given to regularization term implies giving model a chance to overfit to the data. so that's why the hyperplanes are almost passing through the outliers.

when regularization parameter is = 1: Here we notice the same plots i.e plots whose regularization parameter is = 0.0001.i think this is not the good parameter because it behaves similar when regularization parameter is = 0.0001.

when regularization parameter is = 100: Here we notice significant change in the position of the hyperplanes compared to those plots whose regularization parameter is = 0.0001 and 1.When the weightage is given more to the regularization term we try to decrease the overfitting and the model works better in getting rid of outliers.

[ ]:

```

```