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
```

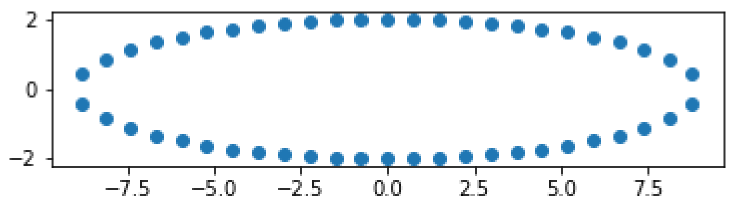
```
#Mounting Google drive folder
from google.colab import drive
drive.mount('/content/drive')
%cd /content/drive/My Drive/Appliedai colab/Assignment 8 - Linear models/
```

```
Go to this URL in a browser: https://accounts.google.com/o/oauth2/auth?client_id=947318989803-6bn6qk8qdgf4n4g3pfee6491hc0brc4i.a

Enter your authorization code:
··········
Mounted at /content/drive
```

```
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)
```

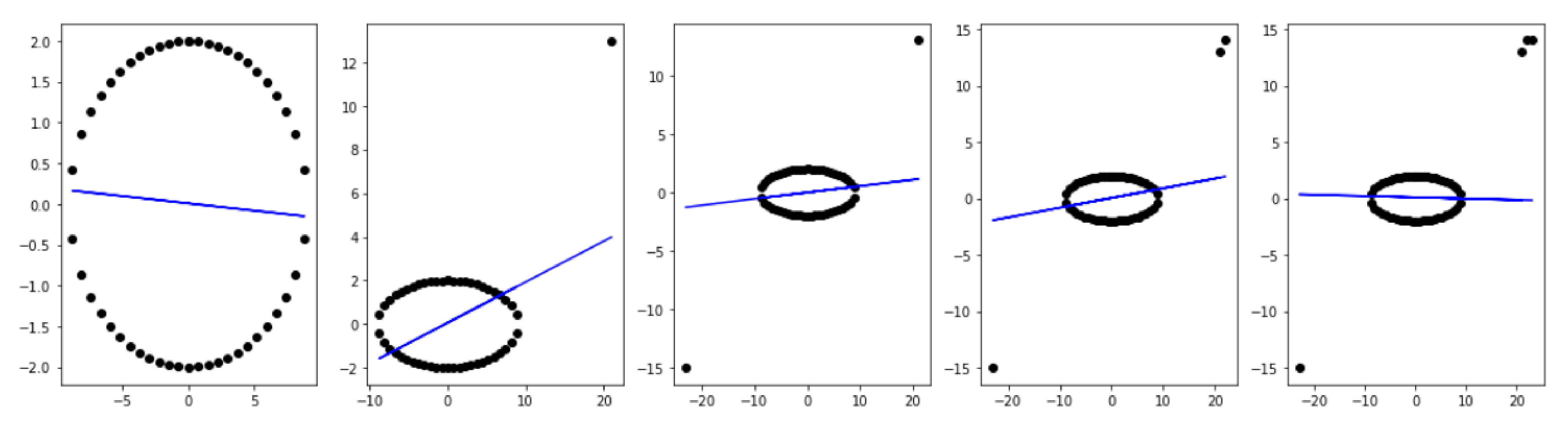
1. As a part of this assignment you will be working the regression problem and how regularization helps to get rid of outliers

2. Use the above created X, Y for this experiment.

3. to do this task you can either implement your own SGDRegression(prefered excatly similar to "SGD assignment" with mean sequared error or you can use the SGDRegression of sklearn, for example "SGDRegressor(alpha=0.001, eta0=0.001, learning_rate='constant',random_state=0)" note that you have to use the constant learning rate and learning rate ***eta0*** *initialized.*

4. as a part of this experiment you will train your linear regression on the data (X, Y) with different regularizations alpha=[0.0001, 1, 100] and observe how prediction hyper plan moves with respect to the outliers

5. This the results of one of the experiment we did (title of the plot was not metioned intentionally)



in each iteration we were adding single outlier and observed the movement of the hyper plane.

6. please consider this list of outliers: [(0,2),(21, 13), (-23, -15), (22,14), (23, 14)] in each of tuple the first elemet is the input feature(X) and the second element is the output(Y)

7. for each regularizer, you need to add these outliers one at time to data and then train your model again on the updated data.

*8. you should plot a 3*5 grid of subplots, where each row corresponds to results of model with a single regularizer.*

9. Algorithm:

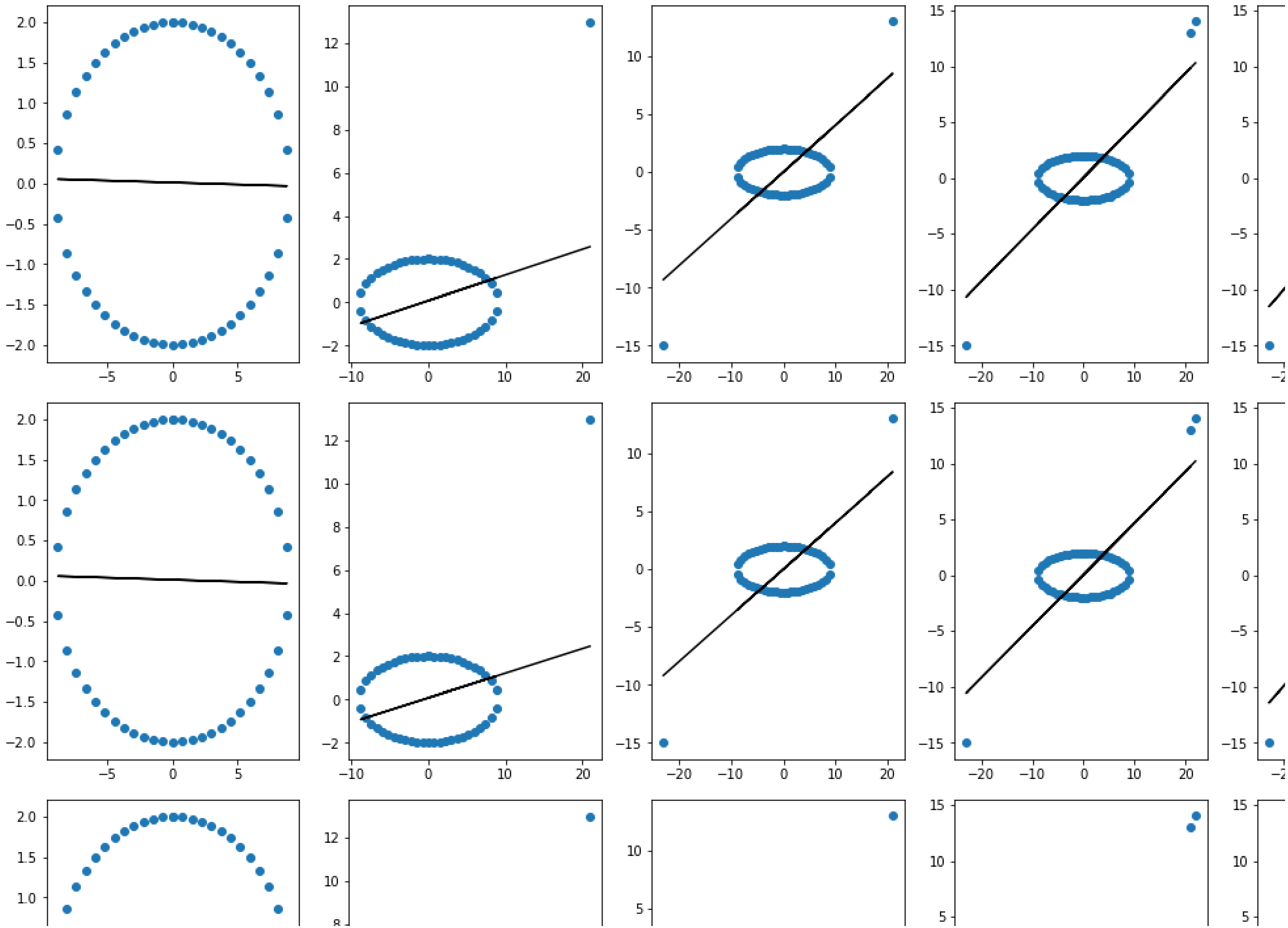
```
for each regularizer:
  for each outlier:
      #add the outlier to the data
      #fit the linear regression to the updated data
      #get the hyper plane
      #plot the hyperplane along with the data points
```

10. MAKE SURE YOU WRITE THE DETAILED OBSERVATIONS, PLEASE CHECK THE LOSS FUNCTION IN THE SKLEARN DOCUMENTATION (please do search for it).

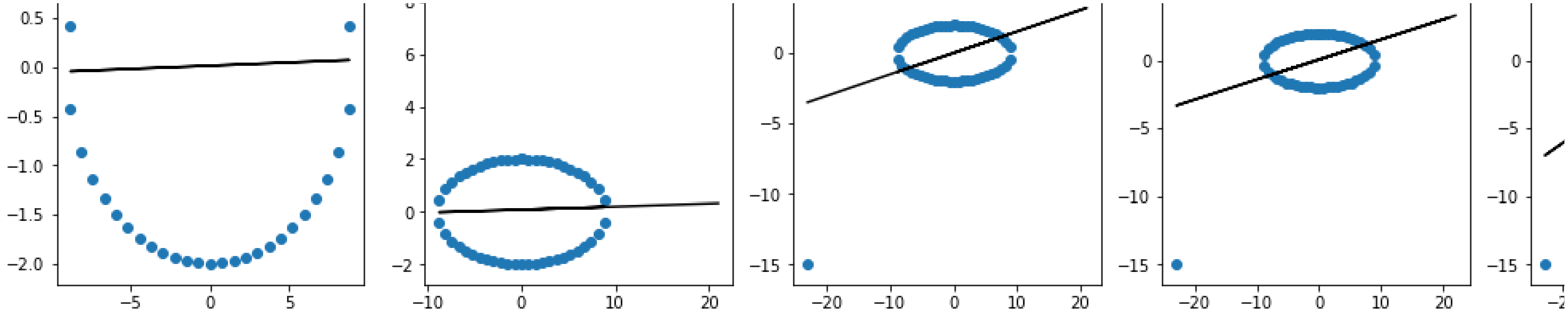
```
def plot_decision_boundary(coef, intercept, x):
  y = coef * x + intercept
  plt.plot(x, y, color = 'black')
```

```
reg_alphas = [0.0001, 1, 100]
outliers = [(0,2), (21, 13), (-23, -15), (22,14), (23, 14)]


for alpha in reg_alphas:  #for each regularizer
  c = 0
  plt.figure(figsize=(20,5))
  X = b * np.sin(phi)
  Y = a * np.cos(phi)
  for o in outliers:  #for each outlier
    plt.subplot(1, 5, c+1)
    c += 1
```

```
    c += 1

    #adding outlier to the data
    X = np.append(X,o[0]).reshape(-1, 1)
    Y = np.append(Y,o[1]).reshape(-1, 1)

    plt.scatter(X, Y)

    clf = SGDRegressor(alpha = alpha, eta0=0.001, learning_rate='constant', random_state = 0,\
                       max_iter = 1000, tol = 1e-3)
    clf.fit(X, Y)
    coef = clf.coef_
    intercept = clf.intercept_
    plot_decision_boundary(coef, intercept, X)
plt.show();
```



2.0
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−2.0
−5
0
5
12
10
8
6
4
2
0
−2
−10
0
10
20
10
5
0
−5
−10
−15
−20
−10
0
10
20
15
10
5
0
−5
−10
−15
−20
−10
0
10
20



In this assignment we increased the alpha for SGDRegressor to see how sensitive are the parameters to outliers in data.

Here Alpha is the constant that multiplies the regularization term.

The regularization term(penalty term) defaults to l2 (squared euclidean norm L2) for linear SVM models.

Observations

- When outliers makes the absolute value of parameters to be very large, the L2 norm regularization will increase the loss function more and thus it will detect unusual large parameters earlier, which means it is more useful for reducing the sensitivity of parameters to outliers.
- L-2 norm error usually produces much great output value (due to square exponent) and therefore can be logically considered as much more sensitive to outlier data.
- For an even better understanding I read this blog - http://www.chioka.in/differences-between-l1-and-l2-as-loss-function-and-regularization/ and learned how L2 regularization acts as a more stable function as compared to L1.