Task-C: Regression outlier effect.

Objective:Visualization best fit linear regression line for different scenarios

In [1]:
```python
# 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
```

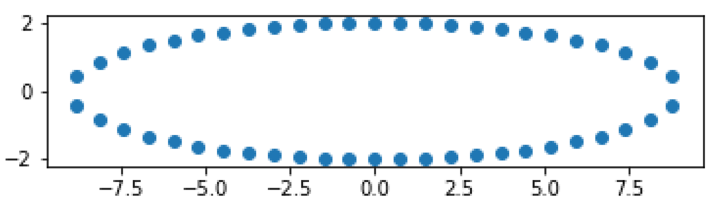
In [2]:
```python
import numpy as np
import scipy as sp
import scipy.optimize

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
```

In [3]:
```python
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()
```



In [4]:
```python
X= b * np.sin(phi)
Y= a * np.cos(phi)
print(len(X))
print(len(Y))
```

```
50
50
```

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
```

1. *MAKE SURE YOU WRITE THE DETAILED OBSERVATIONS, PLEASE CHECK THE LOSS FUNCTION IN THE SKLEARN DOCUMENTATION (please do search for it). </i> </pre>*

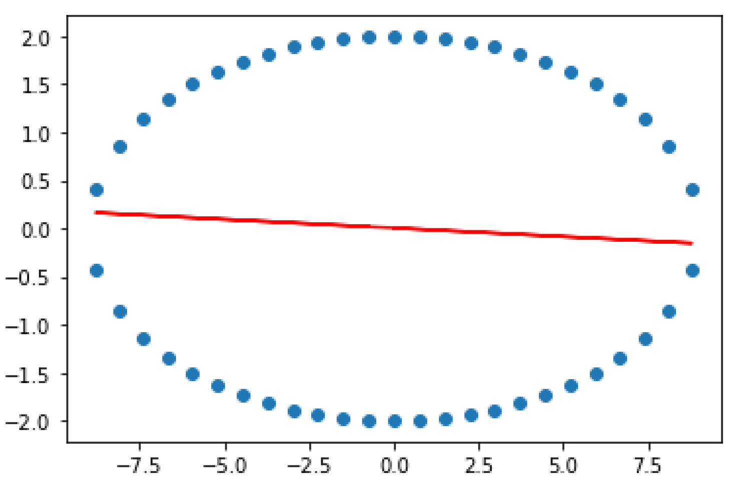
In [30]:
```python
reg = SGDRegressor()
X = X.reshape(-1,1)
Y = Y.reshape(-1,1)
reg.fit(X,Y)
reg.coef_,reg.intercept_
```

Out[30]:
```
(array([-0.0180877]), array([0.00389805]))
```

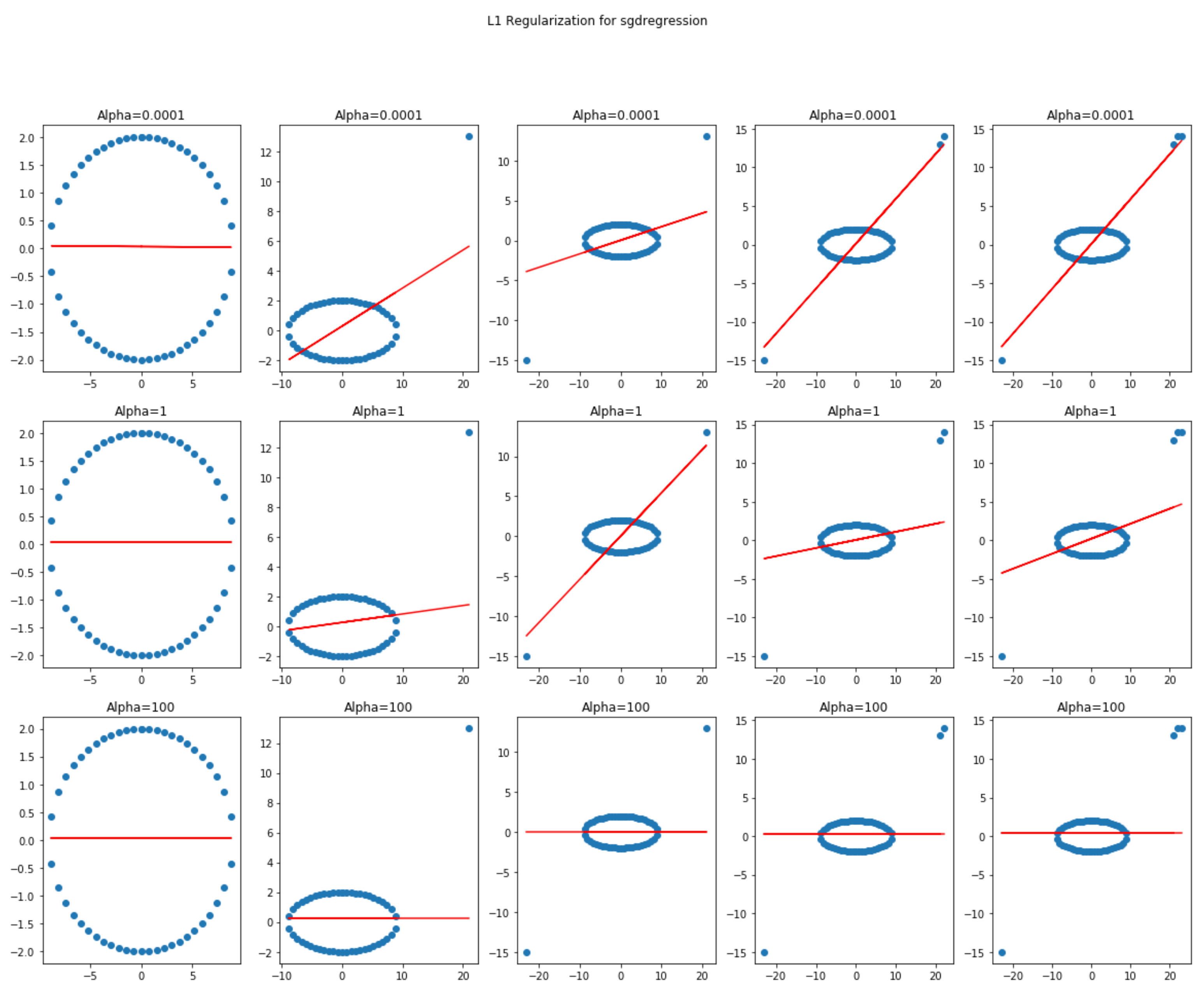
In [31]:
```python
pred = reg.predict(X)
pred
```

Out[31]:
```
array([ 0.00389805, -0.00957262, -0.02304094, -0.03650436, -0.04995988,
       -0.06340374, -0.07683082, -0.09023364, -0.10360028, -0.1169096 ,
       -0.13011772, -0.14310601, -0.15527435, -0.15527435, -0.14310601,
       -0.13011772, -0.1169096 , -0.10360028, -0.09023364, -0.07683082,
       -0.06340374, -0.04995988, -0.03650436, -0.02304094, -0.00957262,
        0.00389805,  0.01736873,  0.03083705,  0.04430046,  0.05775599,
        0.07119985,  0.08462693,  0.09802974,  0.11139639,  0.1247057 ,
        0.13791383,  0.15090211,  0.16307046,  0.16307046,  0.15090211,
        0.13791383,  0.1247057 ,  0.11139639,  0.09802974,  0.08462693,
        0.07119985,  0.05775599,  0.04430046,  0.03083705,  0.01736873])
```

In [33]:
```python
import matplotlib.pyplot as plt
plt.scatter(X,Y)
plt.plot(X,pred,color='red')
plt.show()
```



In [37]:
```python
alpha = [0.0001, 1, 100]
outliers = [(0,2),(21, 13), (-23, -15), (22,14), (23, 14)]
plt.figure(figsize=(20,15))
k = 0
for value in alpha:
    X= b * np.sin(phi)
    Y= a * np.cos(phi)
    reg = SGDRegressor(alpha=value,penalty='l1') #l1 works better than l2 in case of outlier
s as l2 blows up the error after squaring it
    for outlier in outliers:
        k+=1
        X=np.append(X,[outlier[0]])
        Y=np.append(Y,[outlier[1]])
        #print(X,Y)
        reg.fit(X.reshape(-1,1),Y.reshape(-1,1))
        pred = reg.predict(X.reshape(-1,1))
        plt.subplot(3,5,k)
        plt.scatter(X,Y)
        plt.plot(X,pred,color='red')
        plt.title('Alpha={}'.format(value))
plt.suptitle('L1 Regularization for sgdregression')
plt.show()
```



Observation

- *Here we have used l1 regularization since it works better with the outliers*
- *In the first case we have used alpha value as too low so it giving less importance to regularization strength so adding a one outlier changing decision surface much*
- *In the second case when alpha = 1 the decision boundary not changing much with the single outlier*
- *In the third case alpha = 100 the decision boundary not changing much with the multiple outliers*