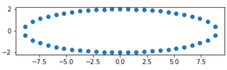
Task-C: Regression outlier effect.

Objective:Visualization best fit linear regression line for different scenarios

In [1]:
```
# 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 pandas as pd
```

In [2]:
```
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]:
```
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]:
```
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

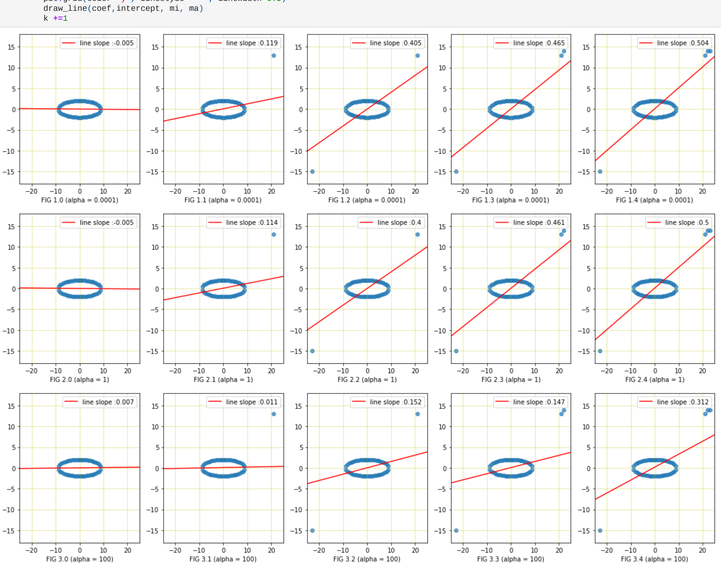
10. MAKE SURE YOU WRITE THE DETAILED OBSERVATIONS, PLEASE CHECK THE LOSS FUNCTION IN THE SKLEARN DOCUMENTATION (please do search for it).

In [5]:
```
def draw_line(coef,intercept, mi, ma):
    # for the separating hyper plane ax+by+c=0, the weights are [a, b] and the intercept is c
    # to draw the hyper plane we are creating two points
    # 1. ((b*min-c)/a, min) i.e ax+by+c=0 ==> ax = (-by-c) ==> x = (-by-c)/a here in place of y we are keeping the minimum value of y
    # 2. ((b*max-c)/a, max) i.e ax+by+c=0 ==> ax = (-by-c) ==> x = (-by-c)/a here in place of y we are keeping the maximum value of y
    points=np.array([[(mi - intercept)/coef, mi],[(ma - intercept)/coef, ma]])
    plt.plot(points[:,0], points[:,1],  color='red', label= "line slope :"+ str(np.round(coef,3)))
    plt.xlim(-25,25)
    plt.ylim(-18,18)
    plt.legend()
```

In [6]:
```
outliers = [(0,2),(21, 13), (-23, -15), (22,14), (23, 14)]

#creating 5 different datasets with above ratios and store them in a list.
datasets = []
datasets_labels = []

for j,i in enumerate(outliers):

    X = np.append(X, i[0])
    Y = np.append(Y, i[1])
    #### L2 Regularization: Huber Loss

    datasets.append(X)
    datasets_labels.append(Y)
```

L2 Regularization: 'squared_loss'

In [7]:
```
alpha_ = [0.0001, 1, 100]
#c = [1, 10 , 100]
k=1
plt.figure(figsize=(20,15))

#for i in range(len(datasets)):
for j in range(len(alpha_)):

    #for j in range(len(alpha_)):
    for i in range(len(datasets)):

        plt.subplot(len(alpha_), len(datasets), k)
        clf = SGDRegressor(alpha=alpha_[j], eta0=0.001, learning_rate='constant',random_state=0)
        clf.fit(datasets[i].reshape(-1, 1), datasets_labels[i].reshape(-1, 1))
        w = clf.coef_[0]
        intercept=clf.intercept_
        coef=list(clf.coef_)[0]


        mi = min(datasets[i])
        ma = max(datasets[i])


        #Plotting
        #plt.scatter(pos_pts[0], pos_pts[1], color = 'b')
        #plt.scatter(neg_pts[0], neg_pts[1], color='r')
        plt.scatter(datasets[i], datasets_labels[i], alpha=0.7)
        plt.xlabel("FIG {0}.{1} (alpha = {2})".format(j+1,i,alpha_[j]))
        #plt.ylabel("pos_neg ratio = {}".format(ratios[i]))
        plt.grid(color='y', linestyle='--', linewidth=0.5)
        draw_line(coef,intercept, mi, ma)
        k +=1
```



OBSERVATIONS:

- It can observed that SGD classifier is not a Robust model as it is significantly impacted by the outliers. Refering to the first row in the above graphs as we started increasing population of outliers, model starts overfitting towards outliers. As a result, hyperparameter starts shifting towards outliers.
- As we introduced Regulation parameter, It can be seen that regularization tries to minimise the impact of the outliers as it prevent the model from overfitting and allows misclassification to certain extent.
- Also the amount of misclassification allowed in the regression is depends on the regularization factor.This can be evident from the slope of the line (hyperplane).
- When regularization factor is extremely small (0.0001), even the single outlier can have create impact on the position of the hyperplane. Compare slopes of the line in figures FIG 1.0 and FIG 1.1, however this is not the case when we have a larger value of regularization factor. (FIG 3.0 & FIG 3.1).
- When we use higher regularization factor (in case of L2 reg), it reduces the impact of outliers to a greater extend by preventing the model from overfitting.