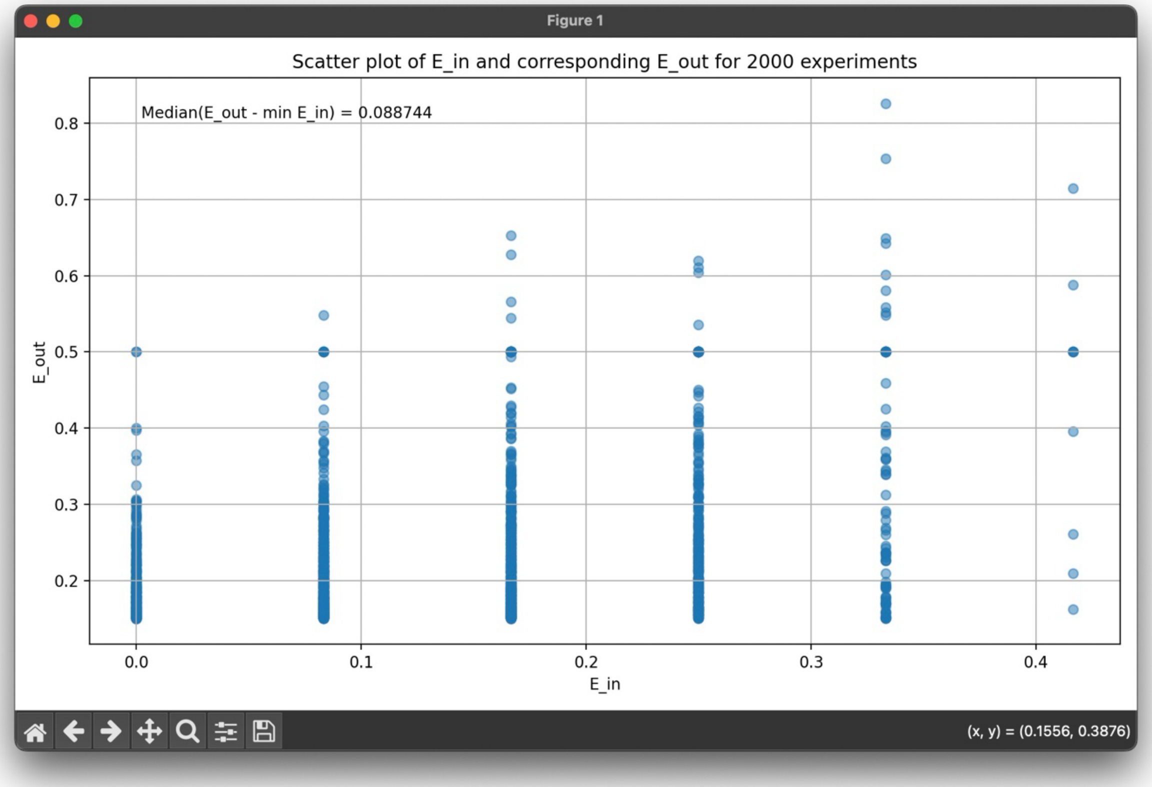
ML homework 2: question 11

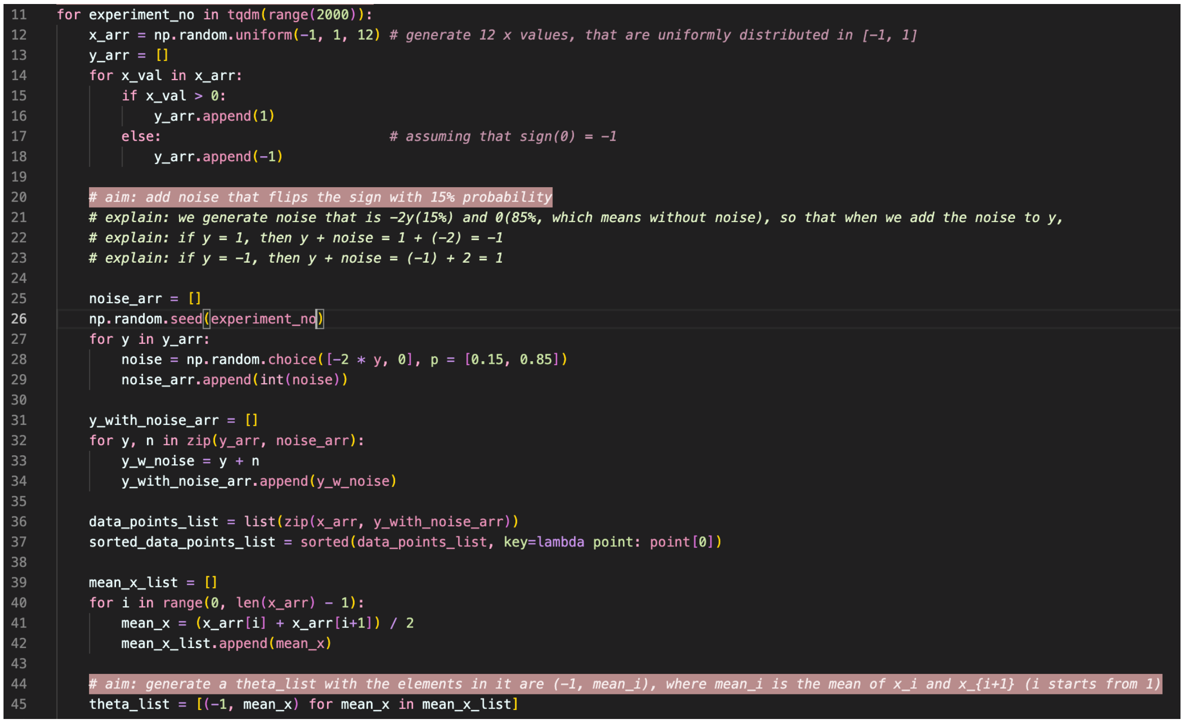
The scatter plot of $(\mathrm{E}_{\mathrm{out}}(\mathrm{g}),\ \mathrm{E}_{\mathrm{in}}(\mathrm{g}))$ is as the figure below:



The median of the difference $\mathrm{E}_{\mathrm{out}}(\mathrm{g})$ - $\mathrm{E}_{\mathrm{in}}(\mathrm{g})$ is about 0.088744.

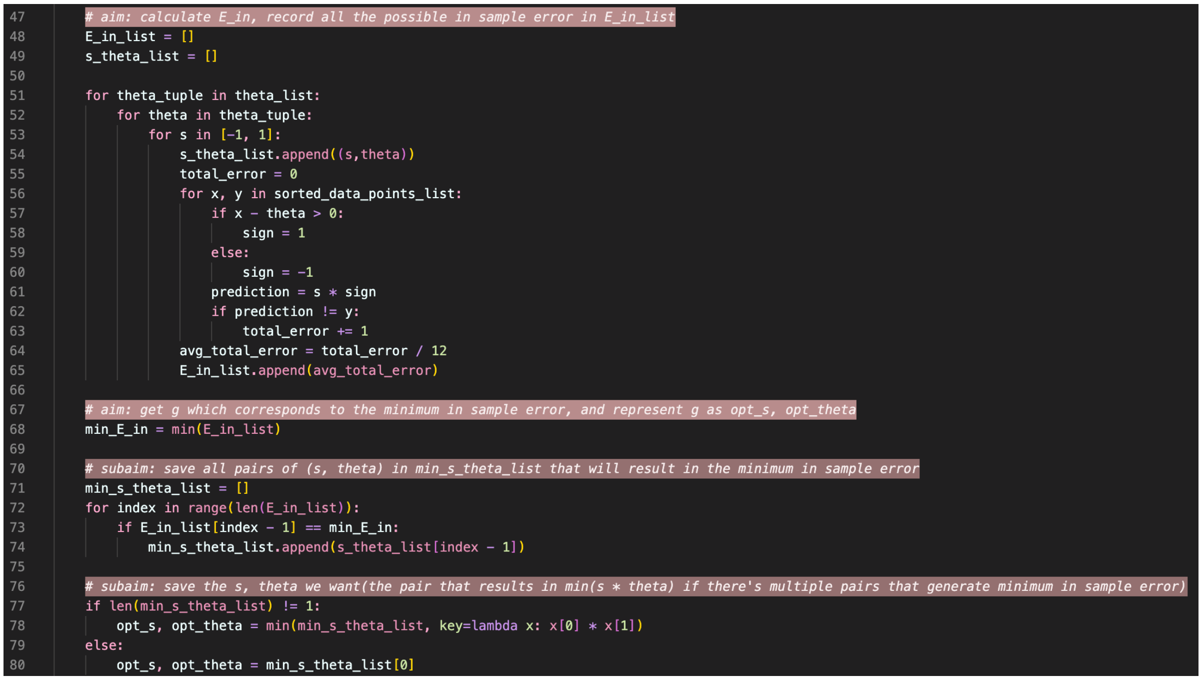
Code snapshot:

In the first part of my code, it's about the basic setups, like generating the x values and the y values with noise, combine them into tuples to present like data points, and sort the data points by the x value as required:



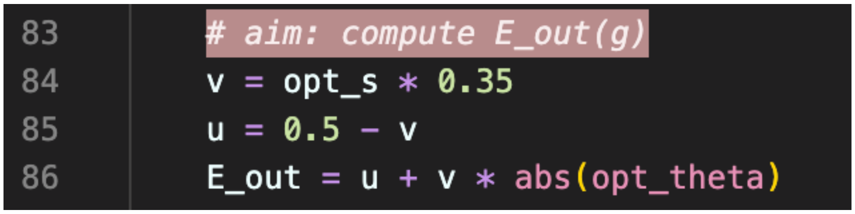
```
for experiment_no in tqdm(range(2000)):
    x_arr = np.random.uniform(-1, 1, 12) # generate 12 x values, that are uniformly distributed in [-1, 1]
    y_arr = []
    for x_val in x_arr:
        if x_val > 0:
            y_arr.append(1)
        else:                            # assuming that sign(0) = -1
            y_arr.append(-1)

    # aim: add noise that flips the sign with 15% probability
    # explain: we generate noise that is -2y(15%) and 0(85%, which means without noise), so that when we add the noise to y,
    # explain: if y = 1, then y + noise = 1 + (-2) = -1
    # explain: if y = -1, then y + noise = (-1) + 2 = 1

    noise_arr = []
    np.random.seed(experiment_no)
    for y in y_arr:
        noise = np.random.choice([-2 * y, 0], p = [0.15, 0.85])
        noise_arr.append(int(noise))

    y_with_noise_arr = []
    for y, n in zip(y_arr, noise_arr):
        y_w_noise = y + n
        y_with_noise_arr.append(y_w_noise)

    data_points_list = list(zip(x_arr, y_with_noise_arr))
    sorted_data_points_list = sorted(data_points_list, key=lambda point: point[0])

    mean_x_list = []
    for i in range(0, len(x_arr) - 1):
        mean_x = (x_arr[i] + x_arr[i+1]) / 2
        mean_x_list.append(mean_x)

    # aim: generate a theta_list with the elements in it are (-1, mean_i), where mean_i is the mean of x_i and x_{i+1} (i starts from 1)
    theta_list = [(-1, mean_x) for mean_x in mean_x_list]
```

The next part is to calculate the in sample error of all possible combinations of s and theta, then find the minimum in sample error, and record its corresponding s and theta, if multiple pairs of s and theta can result in the minimum, then choose the optimal pair as the one with the smallest product:



```
    # aim: calculate E_in, record all the possible in sample error in E_in_list
    E_in_list = []
    s_theta_list = []

    for theta_tuple in theta_list:
        for theta in theta_tuple:
            for s in [-1, 1]:
                s_theta_list.append((s,theta))
                total_error = 0
                for x, y in sorted_data_points_list:
                    if x - theta > 0:
                        sign = 1
                    else:
                        sign = -1
                    prediction = s * sign
                    if prediction != y:
                        total_error += 1
                avg_total_error = total_error / 12
                E_in_list.append(avg_total_error)

    # aim: get g which corresponds to the minimum in sample error, and represent g as opt_s, opt_theta
    min_E_in = min(E_in_list)

    # subaim: save all pairs of (s, theta) in min_s_theta_list that will result in the minimum in sample error
    min_s_theta_list = []
    for index in range(len(E_in_list)):
        if E_in_list[index - 1] == min_E_in:
            min_s_theta_list.append(s_theta_list[index - 1])

    # subaim: save the s, theta we want(the pair that results in min(s * theta) if there's multiple pairs that generate minimum in sample error)
    if len(min_s_theta_list) != 1:
        opt_s, opt_theta = min(min_s_theta_list, key=lambda x: x[0] * x[1])
    else:
        opt_s, opt_theta = min_s_theta_list[0]
```

After this, calculating the corresponding out of sample error is quite simple, we just plug in the optimal s and theta values:



```
    # aim: compute E_out(g)
    v = opt_s * 0.35
    u = 0.5 - v
    E_out = u + v * abs(opt_theta)
```

The last part is recording the results of each experiment and plot the scatter plot. This part is quite simple so I won’t put the code here, if other part of the code is needed, please let me know ☺