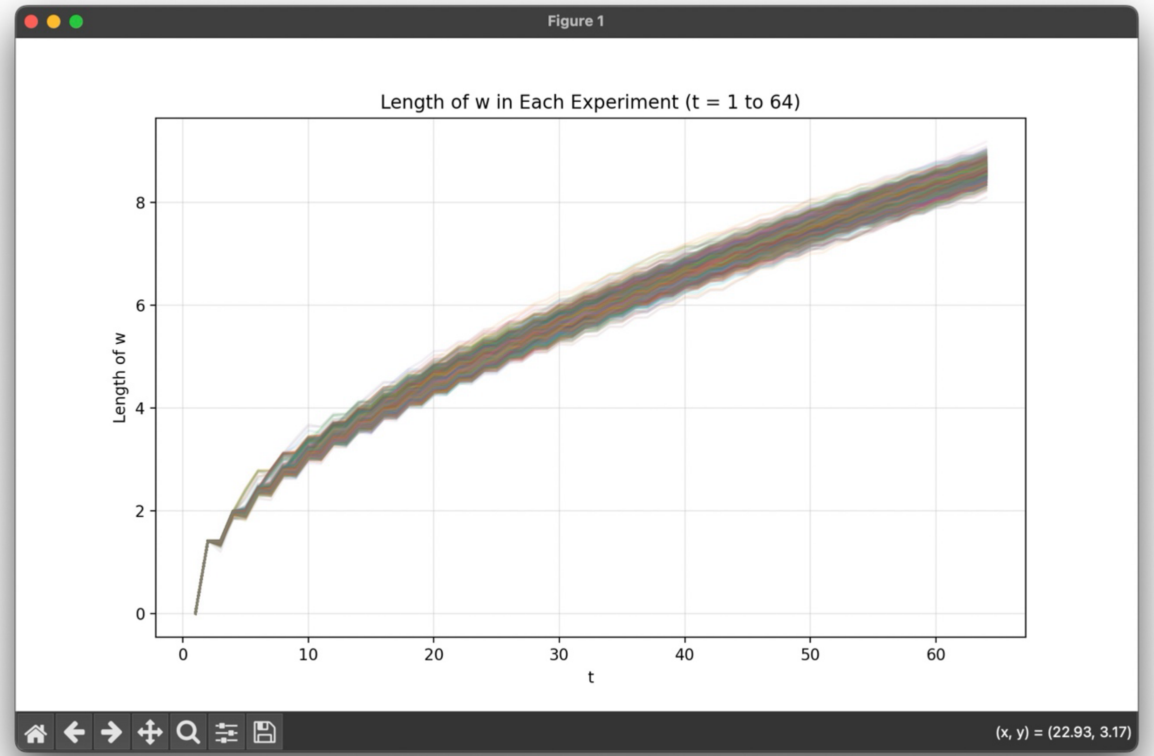
ML homework 1: question 11



Findings:

The minimum number of updates $T_{\min}$ (or min_T in my code) I got is 64.

From this graph, we can see that the growth of the length of the weight vector is slow, and the length until the 64-th update did not exceed 10. This might reflect what we learned in class: the square of the growing length in each update is bounded by R^2.

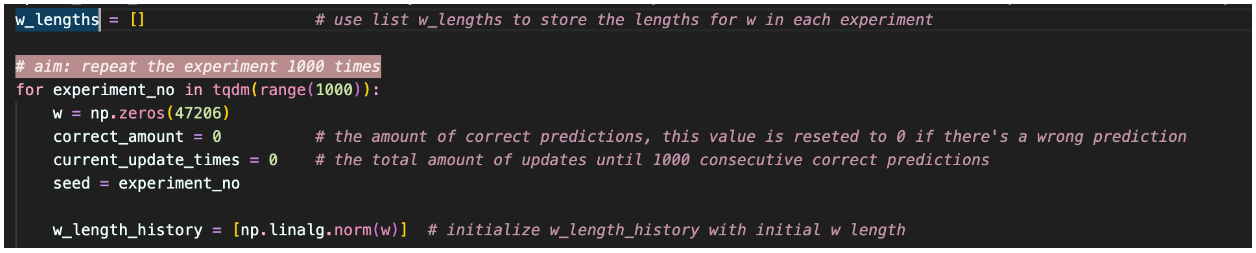
However, we can still see that compare to the later updates, the length grows faster in the first 10 updates. This may due to the fact that the initial value of the weight vector is a zero vector, so that it is far from the ultimate w_{PLA}, hence making bigger changes in the first few updates. The growth became more stable after some updates, it is probably because after the first few updates, we have constructed a hyperplane that can classify most of the examples correctly, so the amplitude of the adjustments became small.

Moreover, we can see that the variation of the 1000 experiments is larger in the first few updates, but as the number of update (t) increases, the variations became small, indicating that we get similar weight lengths after a certain number of updates. I think that the reason is because we got 99% accuracy after 1000 consecutive correct predictions, and the dataset we use is the same, so that if the data points are linear separable, then there exists a hyperplane that can correctly classify all the points, even if there is some noise, we still can get a hyperplane that can separate most of the points. Therefore, as the high accuracy we got in each experiment, the weight vector we got will represent similar hyperplanes, hence resulting a similar value of the weight vector length.

Part of code:

Most of the code in this question is similar to question 10, I just add some modifications to record the history length of $\overrightarrow{w}$, take the minimum value from the resulting value of Ts, and change the figure plotting part.

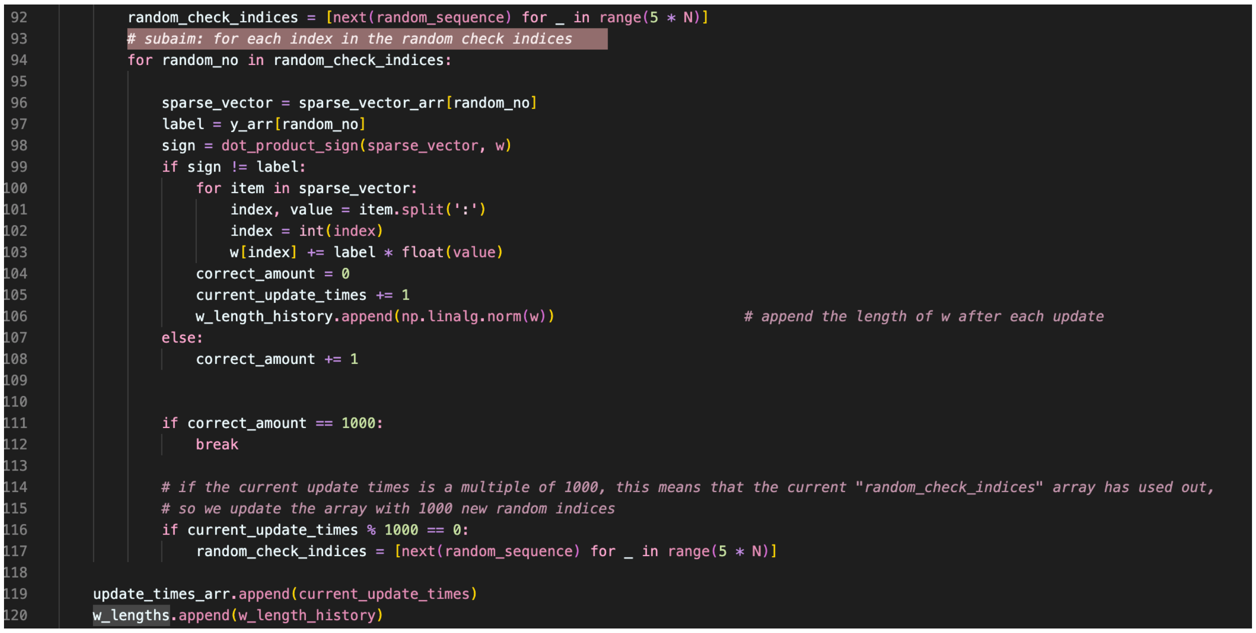
Due to the same problem as question 10, there's too many comments in my code, so I would provide the snapshots of the parts that include the modifications as below:



```
w_lengths = []                  # use list w_lengths to store the lengths for w in each experiment

# aim: repeat the experiment 1000 times
for experiment_no in tqdm(range(1000)):
    w = np.zeros(47206)
    correct_amount = 0          # the amount of correct predictions, this value is reseted to 0 if there's a wrong prediction
    current_update_times = 0    # the total amount of updates until 1000 consecutive correct predictions
    seed = experiment_no

    w_length_history = [np.linalg.norm(w)]  # initialize w_length_history with initial w length
```



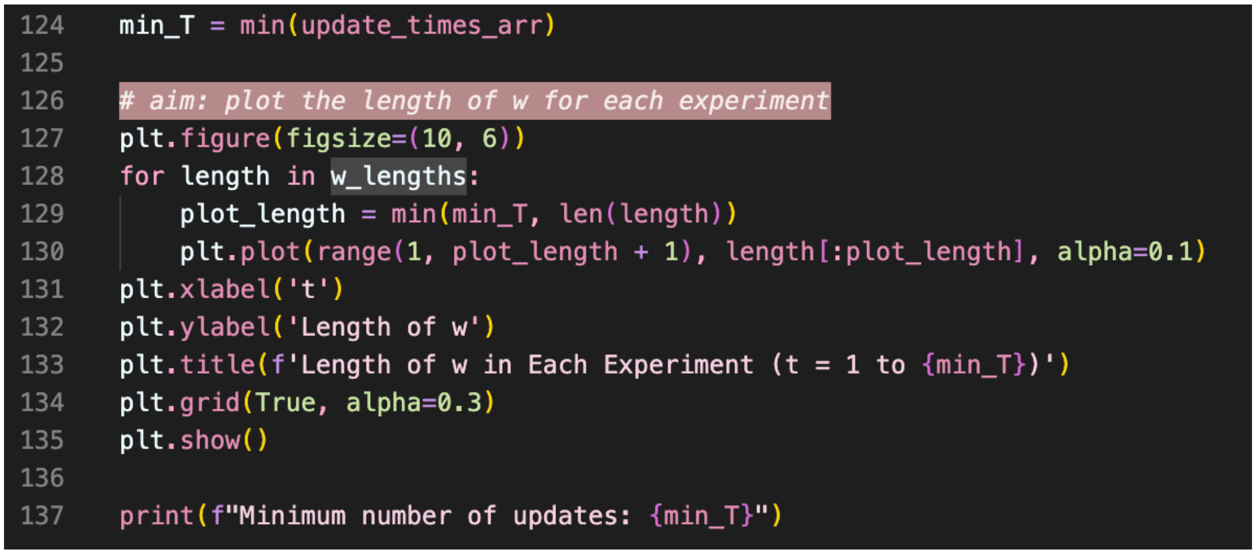
```python
            random_check_indices = [next(random_sequence) for _ in range(5 * N)]
            # subaim: for each index in the random check indices
            for random_no in random_check_indices:

                sparse_vector = sparse_vector_arr[random_no]
                label = y_arr[random_no]
                sign = dot_product_sign(sparse_vector, w)
                if sign != label:
                    for item in sparse_vector:
                        index, value = item.split(':')
                        index = int(index)
                        w[index] += label * float(value)
                    correct_amount = 0
                    current_update_times += 1
                    w_length_history.append(np.linalg.norm(w))                      # append the length of w after each update
                else:
                    correct_amount += 1


                if correct_amount == 1000:
                    break

                # if the current update times is a multiple of 1000, this means that the current "random_check_indices" array has used out,
                # so we update the array with 1000 new random indices
                if current_update_times % 1000 == 0:
                    random_check_indices = [next(random_sequence) for _ in range(5 * N)]

        update_times_arr.append(current_update_times)
        w_lengths.append(w_length_history)
```



```python
min_T = min(update_times_arr)

# aim: plot the length of w for each experiment
plt.figure(figsize=(10, 6))
for length in w_lengths:
    plot_length = min(min_T, len(length))
    plt.plot(range(1, plot_length + 1), length[:plot_length], alpha=0.1)
plt.xlabel('t')
plt.ylabel('Length of w')
plt.title(f'Length of w in Each Experiment (t = 1 to {min_T})')
plt.grid(True, alpha=0.3)
plt.show()

print(f"Minimum number of updates: {min_T}")
```