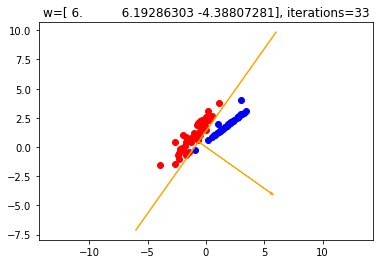
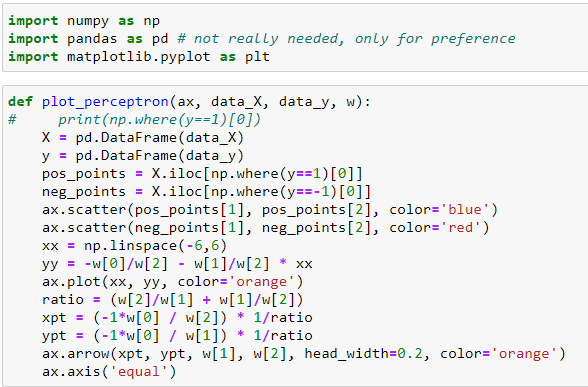
Z5305320

Dong AO

(a):

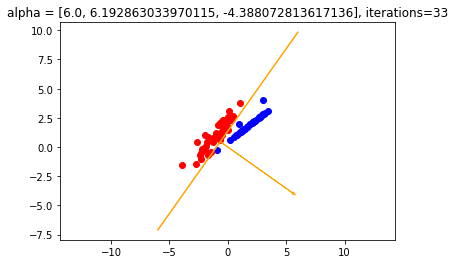


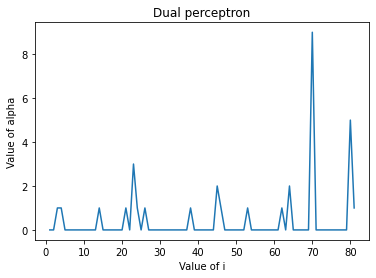


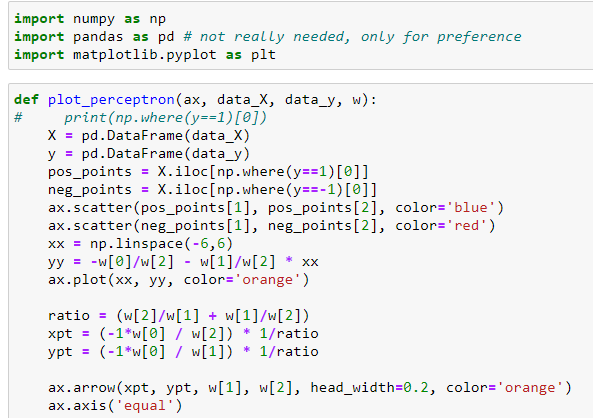
**(b):**

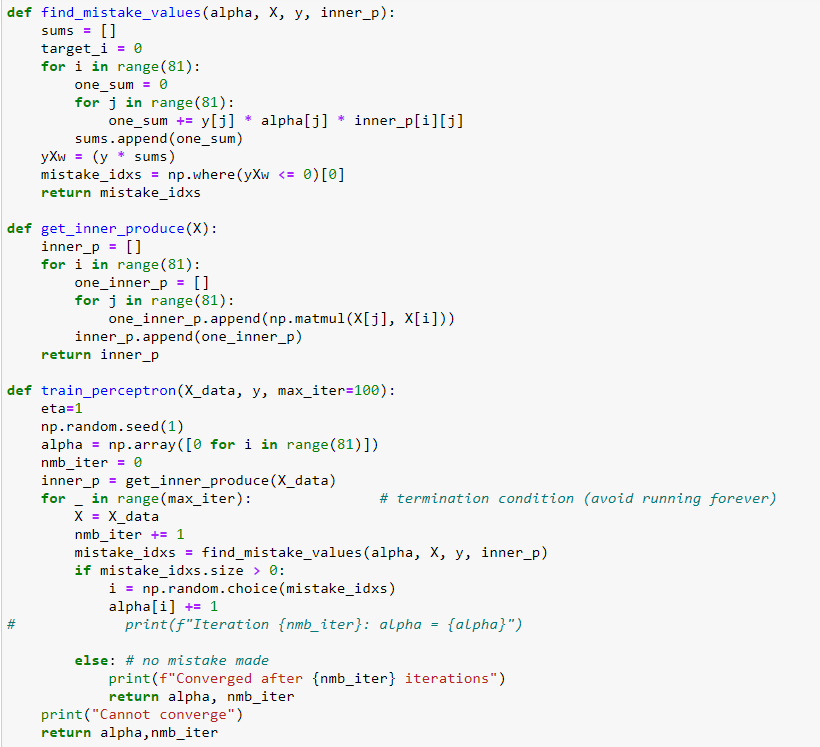
**Comment:**

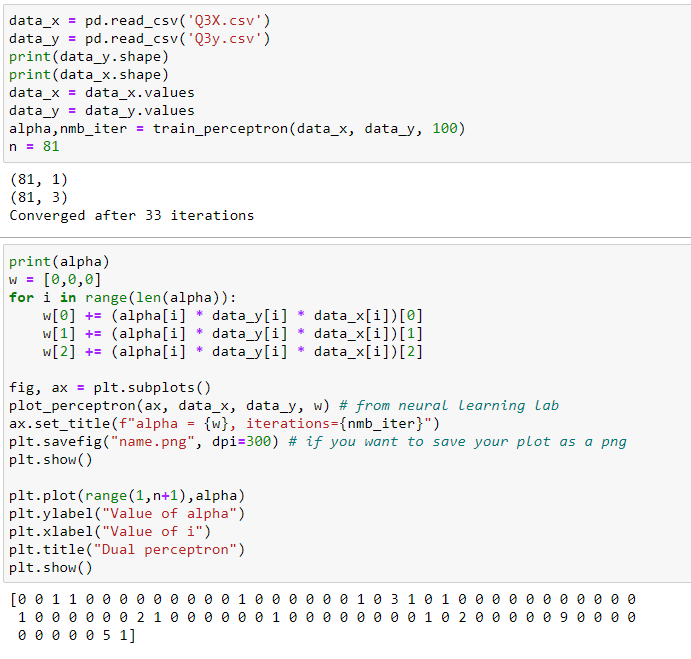
The outcomes of the two methods including number of iterations and the perceptron are the same, which makes sense. There is no essential difference between the two methods. The dual method can reduce the calculation time by calculating the inner product ahead of time.







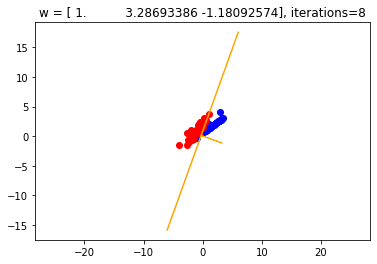


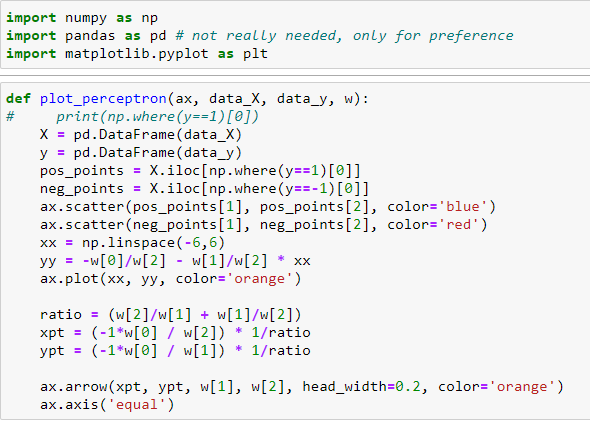


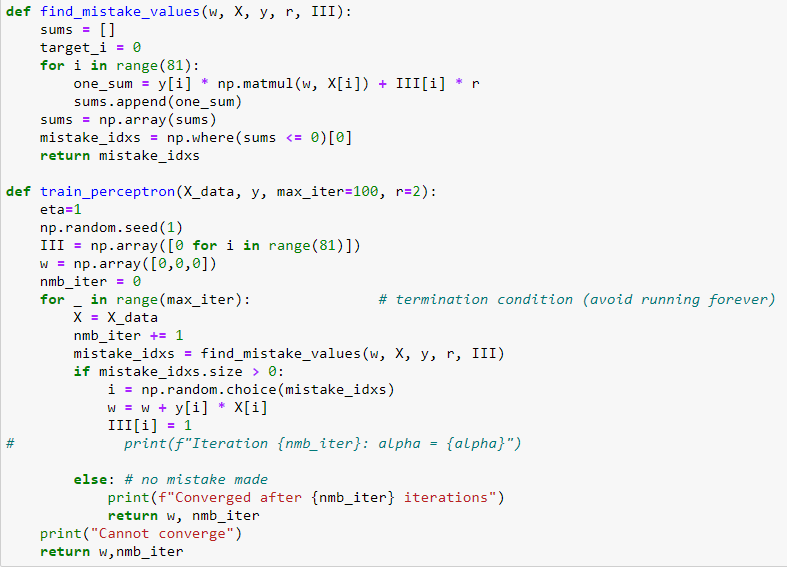
(c):

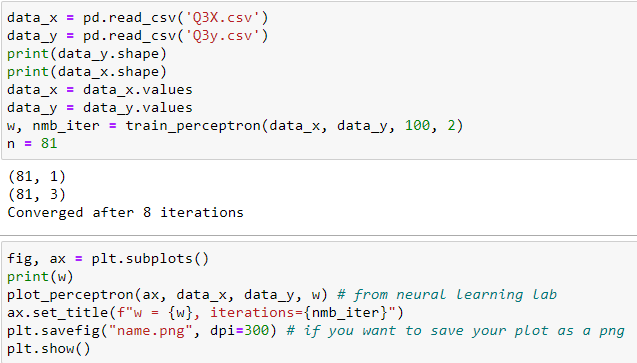
Weight:











(d):

Pseudo Code:

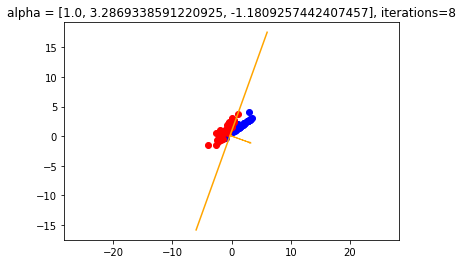
Input: .

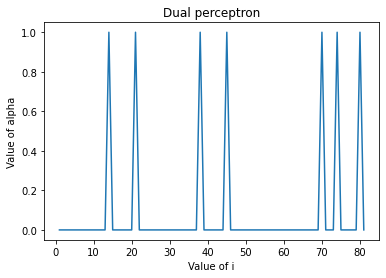
Initialize: .

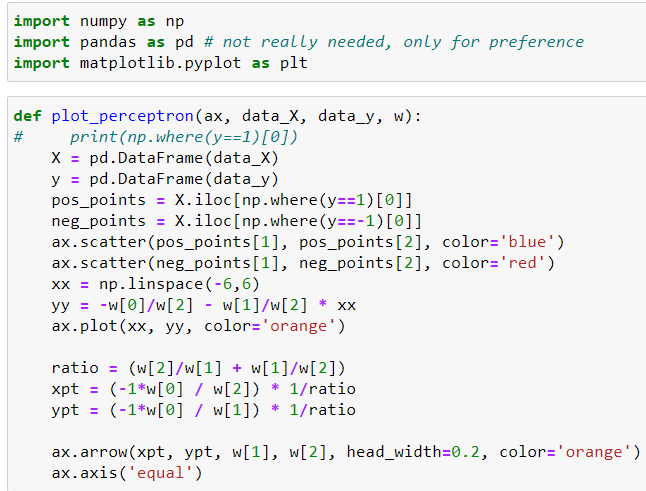
For t = 1…max\_iter:

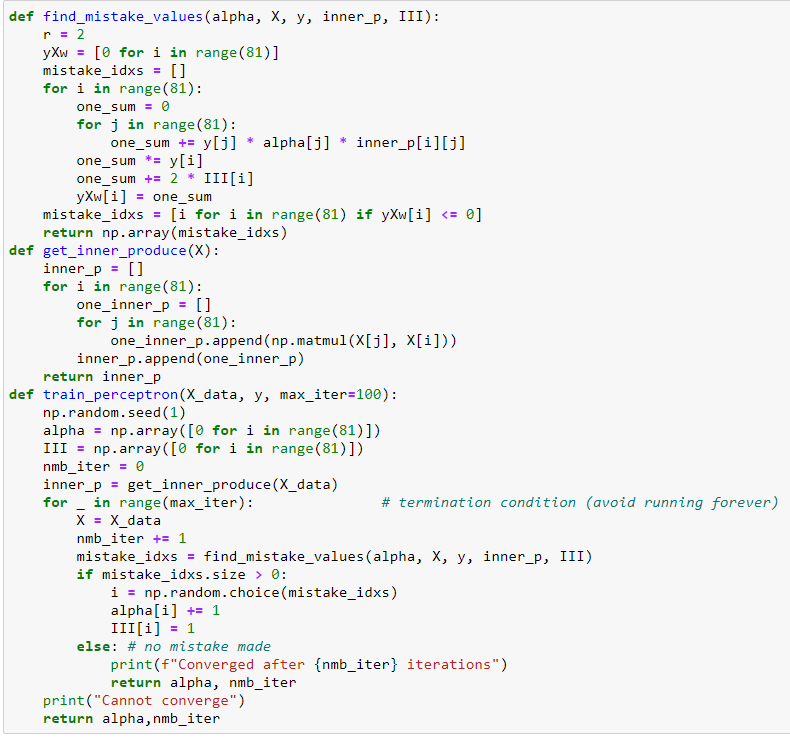
if there is an index i such that :

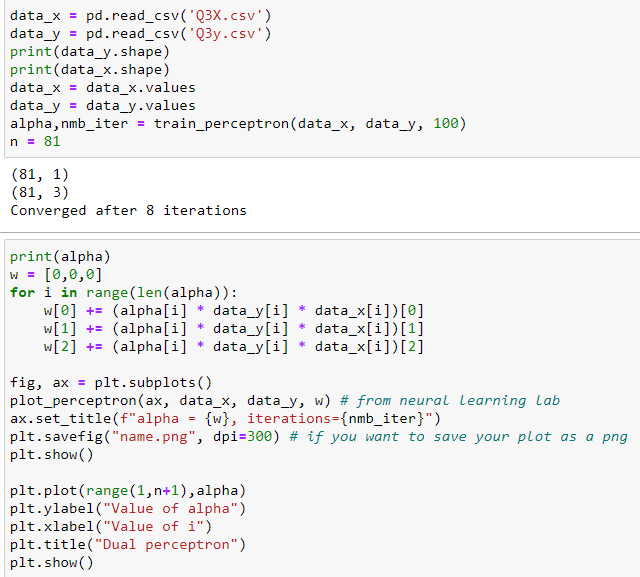
Else:











(e):

The effect: Once one index has been updated, it will become hard for it to get updated again. It must have a value smaller than -2 since I plus 1\*2 to it in next iteration. Through this, I can avoid updating the same index in different iterations.

When the dataset contains some noise data, this algorithm performs better since it can prevent one data been updated in the perceptron multiple times. By printing out the alpha list from (b) and (d).

(b): [0 0 1 1 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 1 0 3 1 0 1 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 2 1 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 1 0 2 0 0 0 0 0 9 0 0 0 0 0 0 0 0 0 5 1]

(d): [0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 1 0 0 0 0 0 1 0]

We can see that in (b) there are data points being updated 9 and 5 times. But instead in (b), they are only been updated once.

The disadvantage is when the values in the dataset is small enough which means we indeed need to update one data multiple times but we do not actually do that. The algorithm may stop us from doing that. It may stop us from getting a convergent solution or we have not gotten a good model but the algorithm stops.