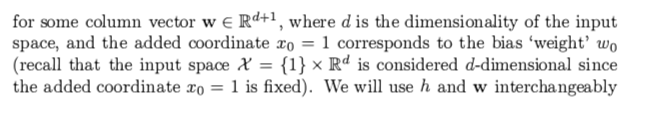
1. (a): Because we have to split the training data into two parts: 1) The new training dataset and 2) The validation dataset. The function is to allow user to input the original dataset (including data and labels) and the index where to cut the original dataset

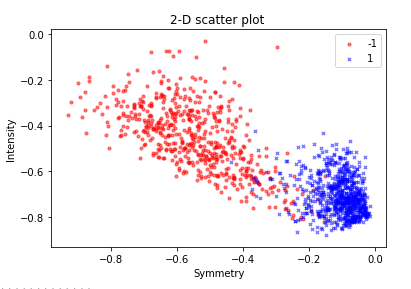
(b): Yes it is correct. We need to enhance the training model by cross validation checking

(c): N/A

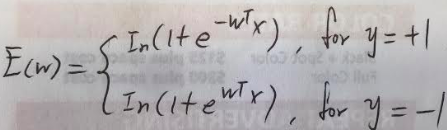
(d): Here we always have 1 because it is for the convenience of bias term (here is **w0**). We will have weight vector **W** which will contain the all coefficients for features. x0 = 1 corresponds to the bias “weight” w0.

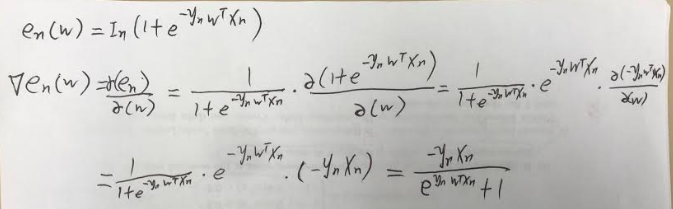


(f)



1. (a):



(b) 

(c) It is “decision boundary linear” because W^T \* X = w0 + w1\*x1+w2\*x2+ w3\*x2 …. Here we only have element **x** with power **1.** Here w1\*x1, or w2\*x2, or w3\*x3 are all linear. Thus, here W^T \* X is just a sum of linear regression. Since it is linear, it cannot classify classes - for example, when class A surrounds class B, we cannot use “decision boundary linear” to differentiate the class A and B. “decision boundary linear” is a monotonical.

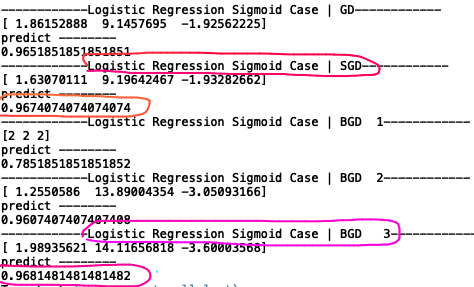
When will we be using sigmoid? 🡪 We use sigmoid when we cope with “Binary linear decision boundary” which mean only two classes in the whole sample.

(d) Yes it is still linear. Though we can change the prediction rule, it does not change the property of W^T \* x which is the combination of many linear regression lines

1. (d)  Test your code in “code/main.py” and visualize the results after training by using the function visualize results. Include the figure in your submission.

(e)

🡪 We have tested in below sequence.



As can be seen above, the test accuracy works the best for “﻿BGD” with batch size “10”. The second best is SGD. Both GD and SGD iterate for 400 times.

2. Basically we have to calculate the gradient like this:

