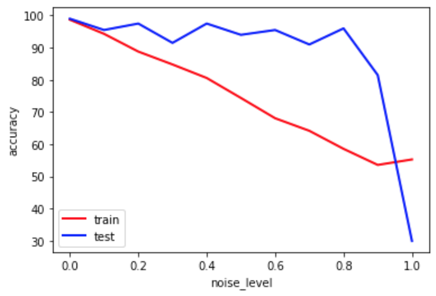
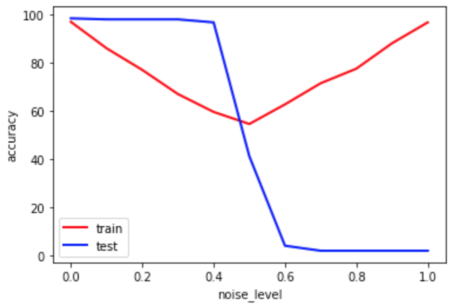
Some conclusions and issues

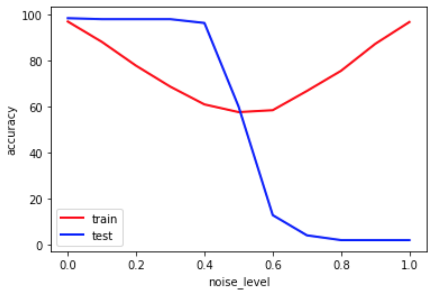
1. To make sure the data distribution of train dataset and test dataset, I created two datasets of 1000 sample data with make classification function. One is noisy while the other is clean. Then I used train_test_split to get 200 samples from the clean dataset as the test dataset. Obviously the noisy dataset of 1000 samples is the train dataset. The hidden neurons are 64 and 32. The learning rate is 0.0001. The epoch number is 200. The result is as follow:



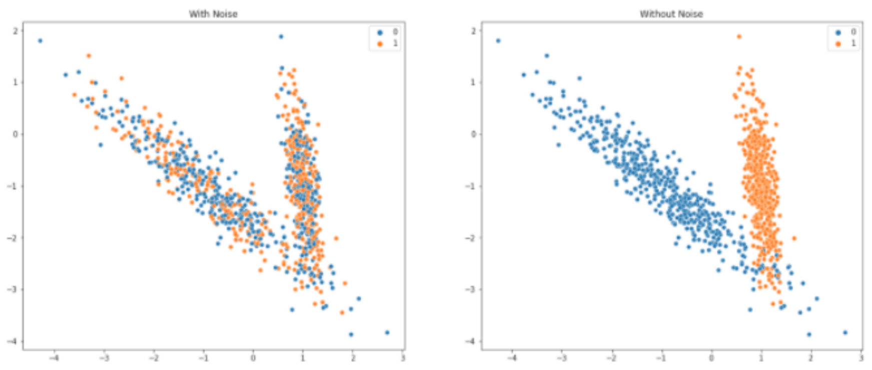
2. Then I used the additional flip function (circular noise) instead of just entering the value for flip_y in the make classification function. The network architecture and hyperparameters remain the same.



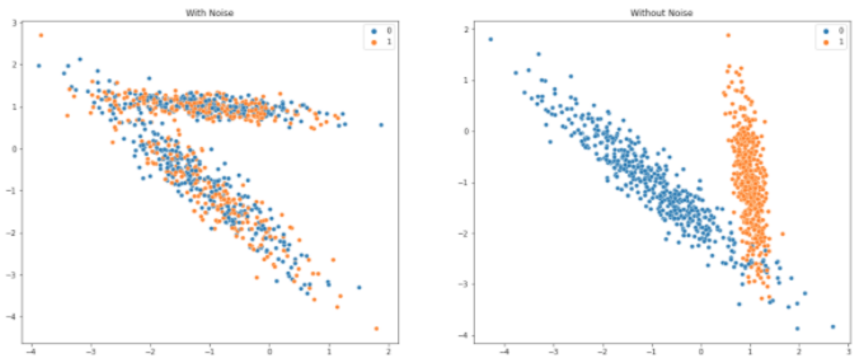
3. Additional flip function (randomly labeled noise):



4. Then I thought about the reason why they don't look like the same. I found that there are some problems regarding feature order when we only use the make classification function to realize label flipping.
   When flip_y<=0.9 (here, the figure shows 0.8):

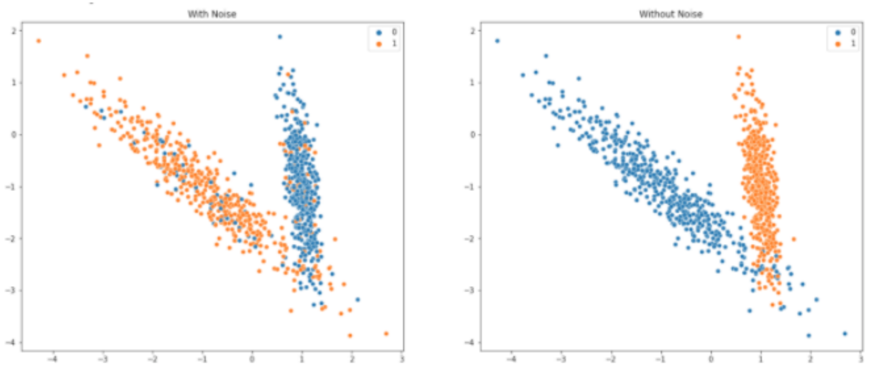


When flip_y>=0.9 (here, the figure shows 0.9):

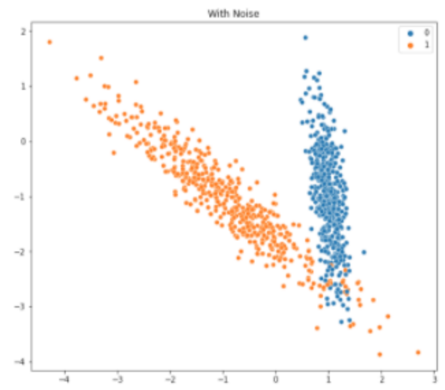


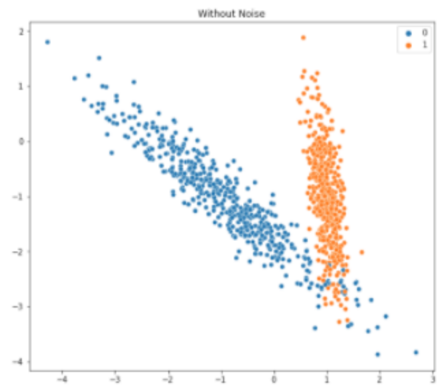
I didn't change the feature order for plotting. So when flip_y>=0.9, the feature order changes itself. When I changed the feature for plotting, the figure with noise looks like the one without noise again. Therefore, it has nothing to do with plot function. This is a very weird phenomenon. I checked the source code of make classification function, even though I think the order is not supposed to change according to the source code, it happens.

5. Now that there's a phenomenon like this, I also checked the data distribution for the additional flip function.
   When 0.9:



When 1:





It works well. Hence, I would say we just don't use flip_y in the make classification function.