CS324 Assignment 1 Report

How to run my code?

Just simple start a Jupyter server for `main.ipynb` in `./Part 1/`, and `train_mlp.ipynb` in `./Part 2/`, and click `Run All` button in the popping out Jupyter web page.

Part I

- **What I did?**
  - Gaussian distribution dataset generation and `train_test_split` function are implemented with the help of `numpy`, which is used to generate train and test datasets.
  - I implement a simple perceptron with forward and backward part from scratch, and the training process of this simple perceptron is also included. Several functions are added

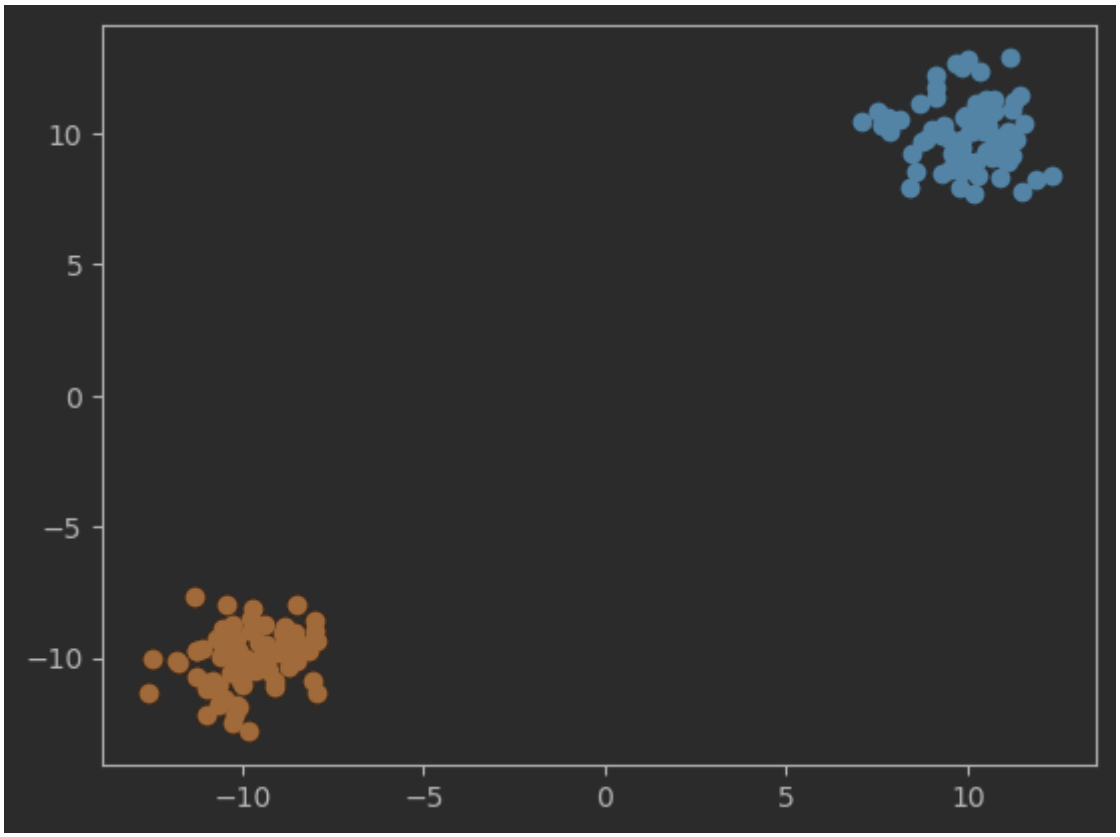
```
class Perceptron(object):
    def predict(self, inputs):
        pass

    def accuracy(self, test_data, test_labels):
        pass
```

    `Perceptron.predict()`: Used to predict the classes of all inputs.

    `Perceptron.accuracy()`: Used to calculate current perceptron prediction accuracy.

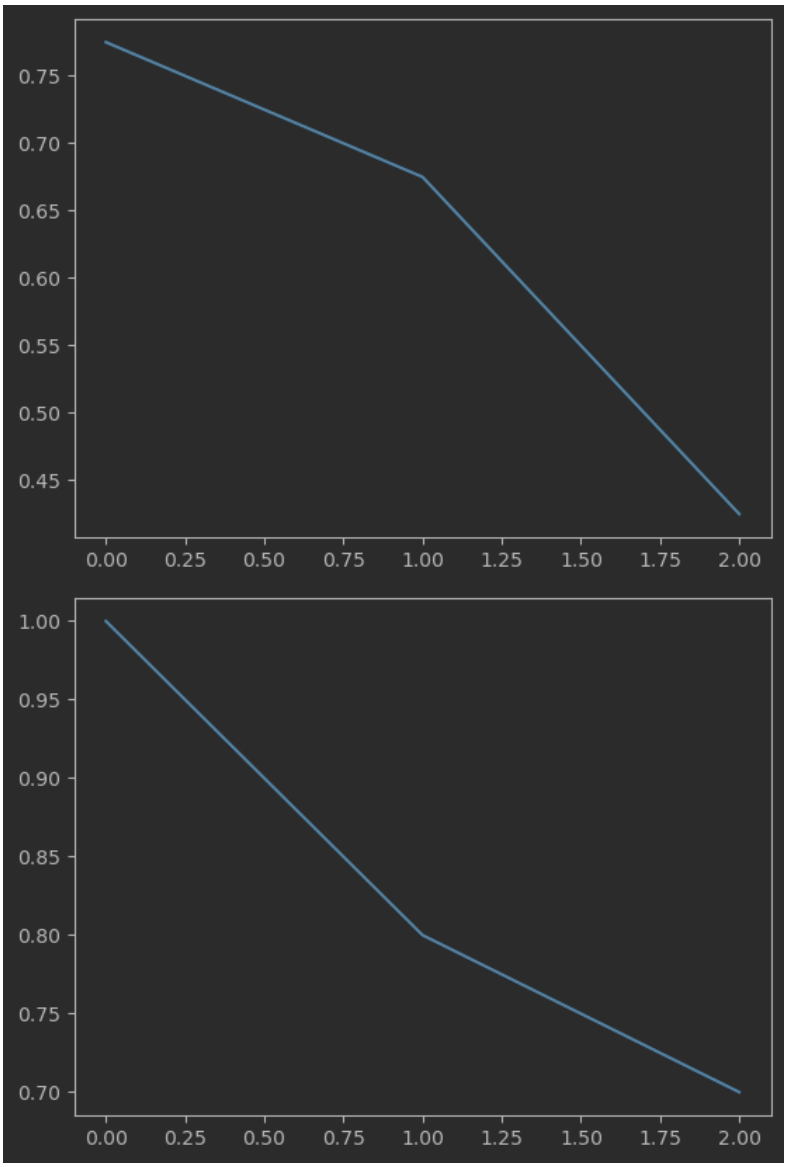
  - When $\mu_1 = (10, 10), \mu_2 = (-10, -10), \sigma_1 = 1, \sigma_2 = 1$, the output is as follows




```
Class 1: mean = [10, 10], cov = [[1.0, 0.0], [0.0, 1.0]]
Class 2: mean = [-10, -10], cov = [[1.0, 0.0], [0.0, 1.0]]
Accuracy: 1.0
```

- Some experiments on how the characteristics of the two distributions would affect the classification accuracy after training are also done. That is to find out

> What happens during the training if the means of the two Gaussians are too close and/or if their variance is too high?



- **Results**

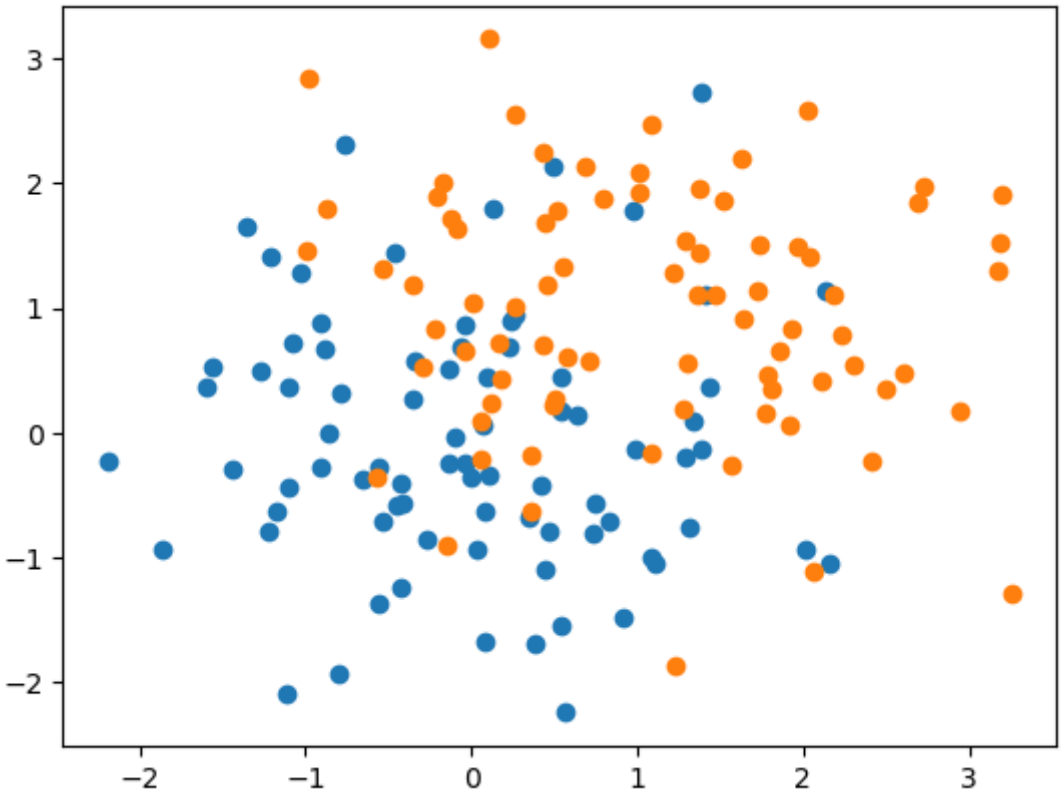
  The accuracy of the final experimental results is basically stable at 100% (when the distributions of the two samples are significantly different), implying that the implementation is able to reach the level of accurate classification.

  As the means of the two Gaussians decreases, the trend of classification accuracy tends to decrease gradually.

  As the variance of the two Gaussians increases, the trend of classification accuracy also tends to decrease gradually.

- **Analysis**

  When the variance is constant and the difference of the means is decreasing, the two datasets (points) will be distributed closer and closer together. As shown below,



The two sets of points are too close, which means it is hard to find a straight line, i.e. the border line to separate the two datasets perfectly, and this is why our accuracy will decrease.

When the difference of the means is constant and the variance is increasing, both of the two datasets are leaving away from their mean point further and further, so the two datasets will be get closer, thus accuracy decreases.

Part II

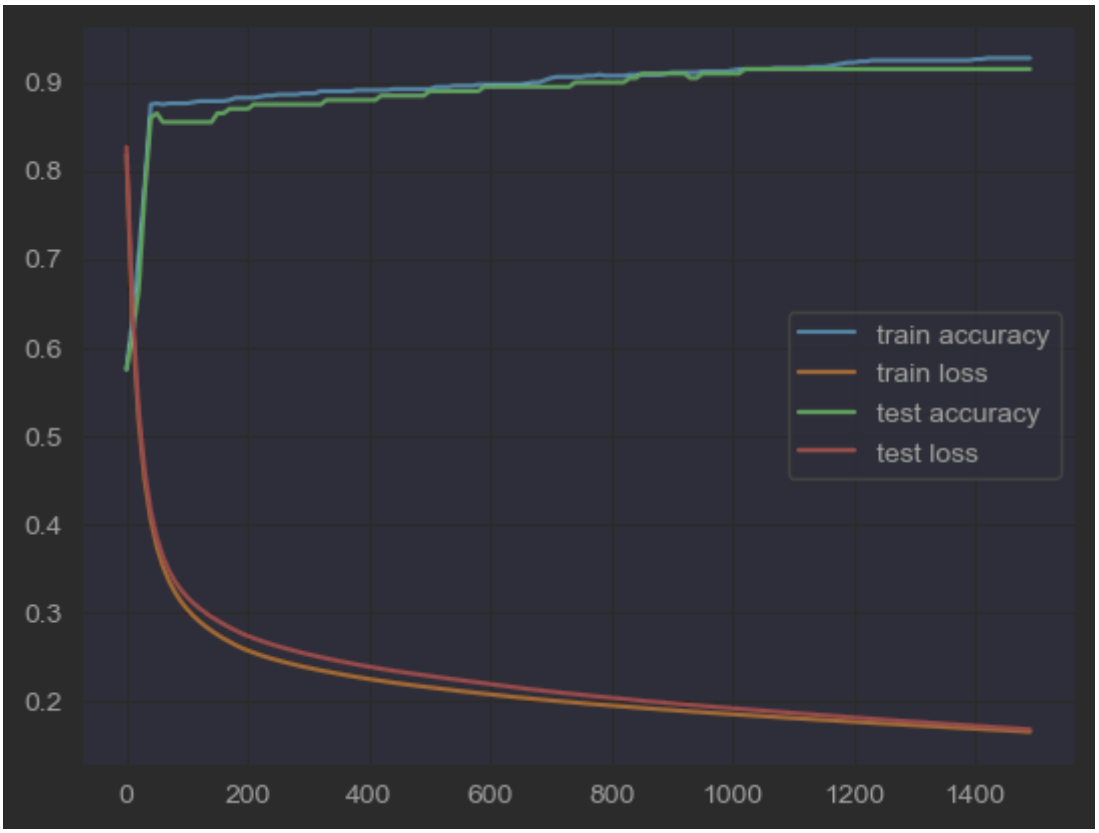
Task 1 & 2

All given functions in the code framework are implemented. Several functions are added to complete the tasks:

- `make_data(noise:, n, train_size, shuffle)`: Generate train and test datasets with given size of percentage.
- `accuracy_loss(mlp, data, label, criterion)`: Compute the accuracy and loss value of current MLP model, with given datasets and loss function.

Task 3

After training using the default parameters, my experiment result is really good. The final accuracies on both training and testing datasets are both above 95%. The accuracy and loss curve of train and test dataset during training process are listed below.



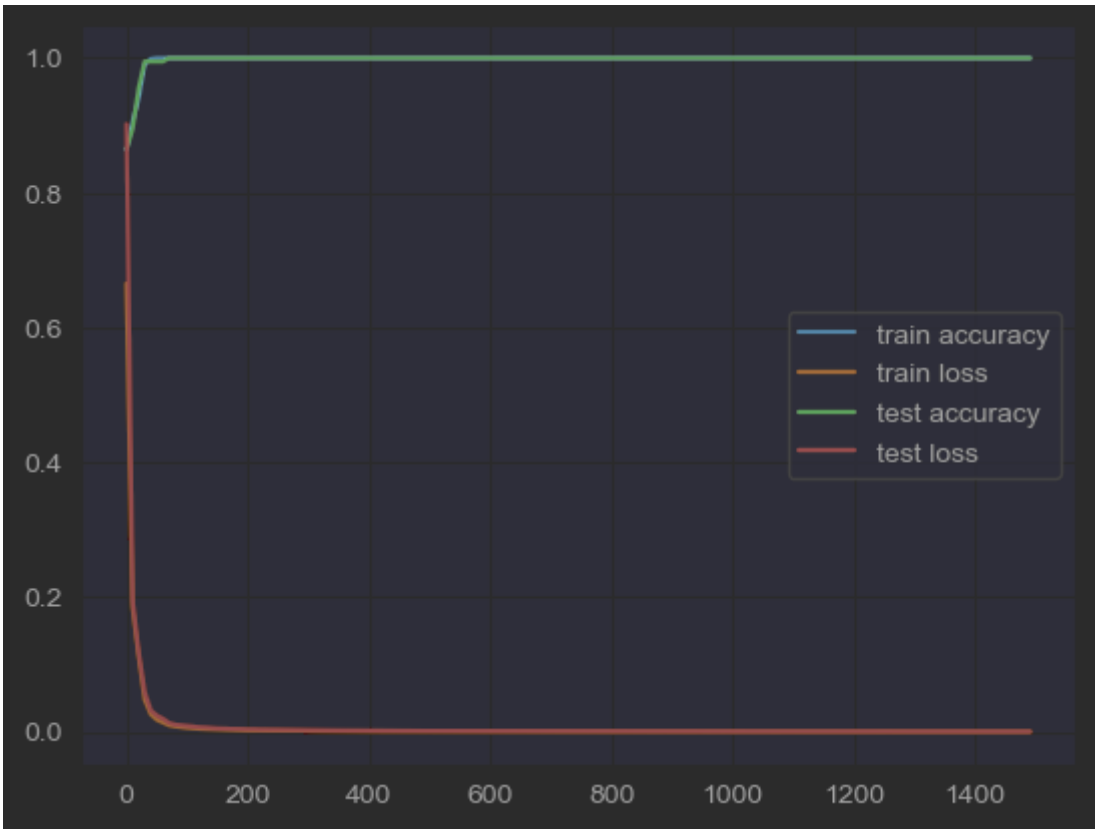
Part Ⅲ

Task 1

Set `FLAG.method` to `SGD`, then it will use **Stochastic Gradient Descent** method to train our model.

Task 2

After training using the default parameters, my experiment result is really good. The final accuracies on both training and testing datasets are both 100%. The accuracy and loss curve of train and test dataset during training process are listed below.



Compare **SGD** with **BGD**, **SGD** converges more quickly, that is because **SGD** does a `batch_size` times of gradient descent update for each batch in every epoch, while **BGD** only updates once in one batch. So **SGD** converges faster.