Problems & Solutions

- **Basic part:**

  I encountered some problems when implementing the softmax function. My implementation of softmax function is `A = np.exp(Z - np.max(Z)) / np.sum(np.exp(Z - np.max(Z)))`. Though I implemented it by the formula which TA provided, I still get the answer that is different from the expected result. So I checked the document of `np.sum()` and found out there's a parameter for choosing axis. After adding `axis = 0` in the `np.sum()`, I can get the correct answer from `A = np.exp(Z - np.max(Z)) / np.sum(np.exp(Z - np.max(Z)), axis = 0)`. Though I didn't apply the softmax function to my basic part, but I stuck at this problem for two days when proceeding to the basic part.

- **Advanced part:**

  In the advanced part, I didn't do one-hot encoding at the beginning, so I get the wrong answer and my accuracy is 0.0. After checking the problem on the eeclass forum, I found a classmate who has the same problem. So I send an email to her and discussed for the solutions and we solve the problem by doing one-hot encoding and checking the dimension of the data. In the part of the validation, I forgot to shuffle the data when pre-processing the data, so the accuracy of my validation set was frustrating at first since the data is arranged in sequence. Thanks for the notice from TA, my accuracy became better after doing shuffle.

Structures

- **Basic:**

  There are two layers in my basic part structure, and below are the hyperparameters and the dimension of my layer.

```
layers_dims = [X_train.shape[0], 25, 15, 10, 5, 2]
activation_fn = ['relu', 'relu', 'relu', 'relu', 'sigmoid']
learning_rate = 0.1, num_iterations = 13000, classes = 2
```

- **Advanced:**

  There are three layers in my advanced part structure, and below are the hyperparameters and the dimension of my layer.

```
layers_dims = [784, 100, 10], activation_fn = ['relu', 'softmax']
learning_rate = 0.1, num_iterations = 800
batch_size = 64, classes = 10
```

Efforts for improving the model

- Since the training process takes a liitle bit long on the Colab and the neural network is implement from scratch instead of using some packages or other frameworks, it's hard to apply some optimizer or analyzing tool such as `Weight&Bias`. Therefore, I tried serveral groups of the hyperparameters and record their performance and accuracy to decide which group I should adopt in the end.