Assignment 3
Backpropagation

Jia He Lim
Po-Chih Kuo

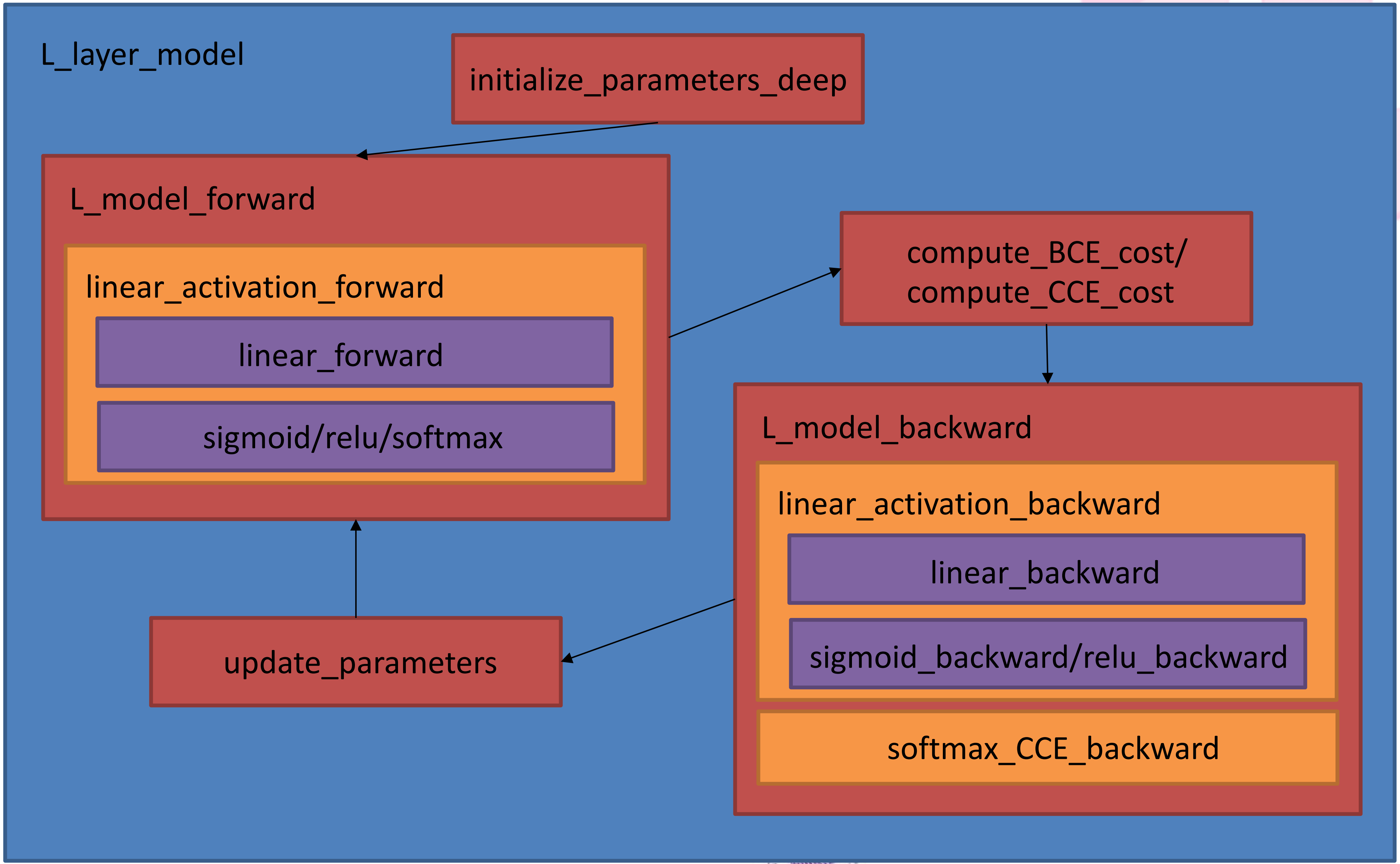
Goal

- Build your own deep neural network step by step
- Implement all the functions required to build a deep neural network
- Understanding forward propagation, backward propagation and update
- Implement Binary Cross-Entropy loss (basic part) and Categorical Cross-Entropy loss (bonus part)
- Implement binary classifier (basic part) and multi-class classifier (bonus part)

Grading Policy

Item	Score
Basic Implementation	70%
Basic Report	30%
Bonus Implementation	25%
Bonus Report	5%

Overview


L_layer_model
initialize_parameters_deep
L_model_forward
linear_activation_forward
linear_forward
sigmoid/relu/softmax
compute_BCE_cost/
compute_CCE_cost
L_model_backward
linear_activation_backward
linear_backward
sigmoid_backward/relu_backward
softmax_CCE_backward
update_parameters

Basic Implementation (70%)

1. Create and initialize parameters of a linear layer with He initialization. (5%)
2. Create and initialize parameters for an L-layer neural network with He initialization. (5%)
3. Apply the linear transformation. (5%)
4. Implement activation function. (10%) (Sigmoid and ReLU)
5. Implement linear-activation layer. (5%) (Sigmoid and ReLU)
6. Compute the binary cross-entropy cost. (5%)
7. Implement linear backward. (5%)
8. Implement backward function. (10%) (Sigmoid and ReLU)
9. Implement the backpropagation for the linear-activation layer. (5%)
10. Implement gradient descent to update your parameters. (5%)
11. Implement a binary classifier and tune hyperparameter. (10%)

Basic Report (30%)

- Describe what problems you encountered when implementing the basic functions.
- Describe how you solve those problems.
- Briefly describe how you build the binary classifier.
- Describe if you pay extra effort to improve your model (e.g. hyperparameter finetuning).
- Summarize your work.
- Do not exceed 2 pages!
- Name your report file as “**[STUDENT_ID]_report.pdf**”.

Bonus Implementation (25%)

1. Implement activation function and linear-activation layer. (5%) (Softmax)
2. Compute the categorical cross-entropy cost. (5%)
3. Implement backward function. (5%) (Softmax+CCE_loss)
4. Implement a multi-class classifier and tune hyperparameter. (10%)

Bonus Report (5%)

- Describe what problems you encountered when implementing the bonus functions.
- Describe how you solve those problems.
- Briefly describe how you build the multi-class classifier.
- Describe if you pay extra effort to improve your model (e.g. hyperparameter finetuning).
- Summarize your work.
- Do not exceed 1 page!
- Write bonus part after basic part in the same report.

Data

Binary classification: Iris flower data set

The data set consists of 50 samples from each of three species of Iris (Iris setosa, Iris virginica and Iris versicolor). Four features were measured from each sample: the length and the width of the sepals and petals, in centimeters.

Only Iris setosa and Iris versicolor are used in our assignment, so there are 100 samples in total. The details of the training set and validation set are shown below:

- shape of X: (4, 100)
- shape of y: (1, 100)
- shape of X_train: (4, 90) shape of y_train: (1, 90)
- shape of X_val: (4, 10) shape of y_val: (1, 10)

Data

Multi-class classification: Classify gestures by reading muscle activity

The armband has 8 sensors placed on skin surface, each measures electrical activity produced by muscles beneath. Each dataset column has 8 consecutive readings of all 8 sensors, so 64 rows of electromyography (EMG) data.

A classifier given 64 numbers would predict a gesture class (0-3). Gesture classes were : rock - 0, scissors - 1, paper - 2, ok - 3. Rock, paper, scissors gestures are like in the game with the same name, and OK sign is index finger touching the thumb and the rest of the fingers spread.

Data

Multi-class classification: Classify gestures by reading muscle activity

Each column has the following structure:

[8sensors]	[8sensors]	[8sensors]	[8sensors]	[8sensors]	[8sensors]	[8sensors]	[8sensors]

The details of the training set and testing set:

- shape of X_train: (64, 9342)
- shape of y_train: (4, 9342)
- shape of X_test: (64, 2336)

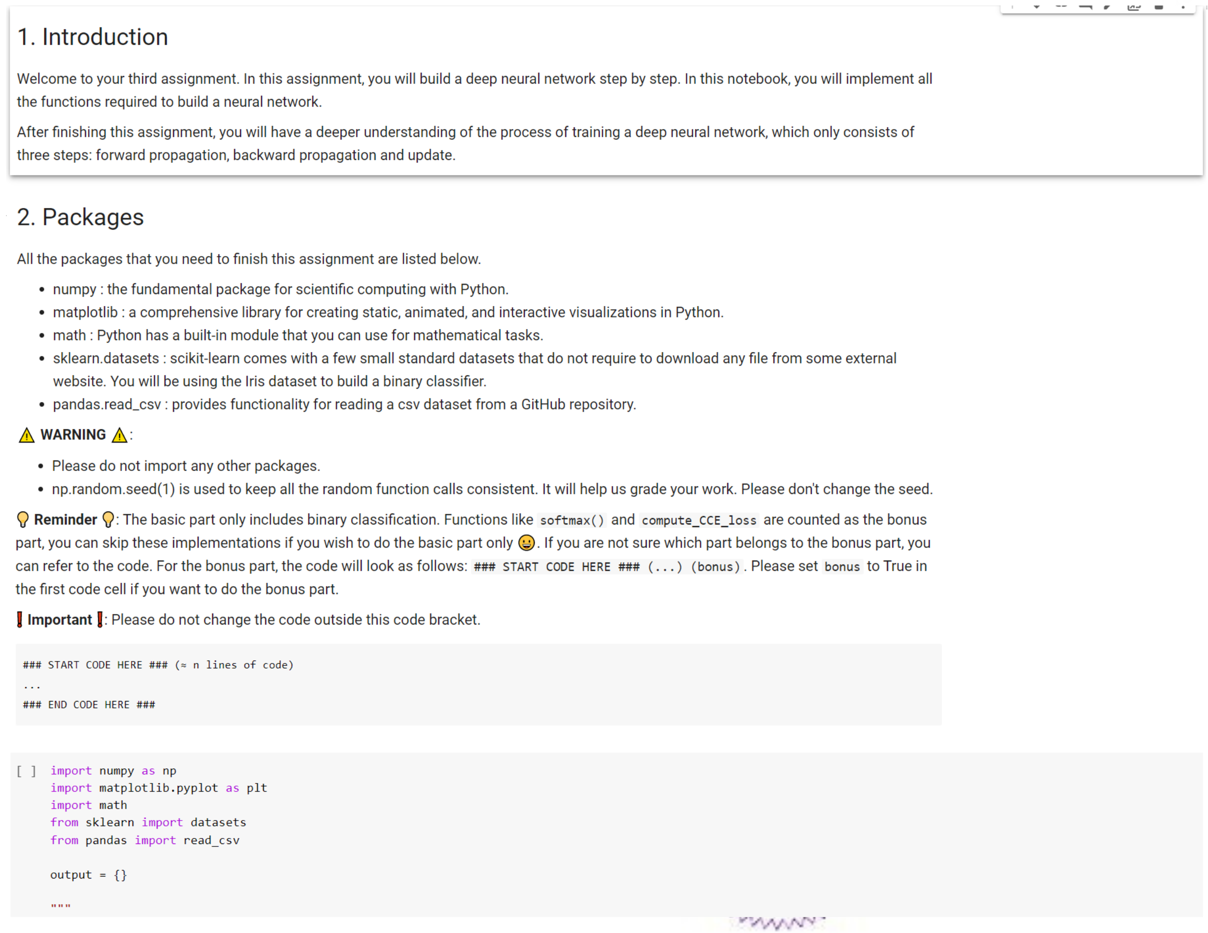
We have already done the data preprocessing for you. The training data and testing data are scaled by using a standard scaler so that the data will have a mean of zero and a standard deviation of one for each feature.

You will have the following items

- Template: HW3_Backpropagation.ipynb (input data inside) (You are encouraged to use Colab!)
- Sample output: sample_output.npy

Template

Except for the imported packages in the template, you cannot use any other packages.
Remember to save the code file to **[STUDENT_ID]_hw3.ipynb**



1. Introduction

Welcome to your third assignment. In this assignment, you will build a deep neural network step by step. In this notebook, you will implement all the functions required to build a neural network.

After finishing this assignment, you will have a deeper understanding of the process of training a deep neural network, which only consists of three steps: forward propagation, backward propagation and update.

2. Packages

All the packages that you need to finish this assignment are listed below.

- numpy : the fundamental package for scientific computing with Python.
- matplotlib : a comprehensive library for creating static, animated, and interactive visualizations in Python.
- math : Python has a built-in module that you can use for mathematical tasks.
- sklearn.datasets : scikit-learn comes with a few small standard datasets that do not require to download any file from some external website. You will be using the Iris dataset to build a binary classifier.
- pandas.read_csv : provides functionality for reading a csv dataset from a GitHub repository.

⚠ **WARNING** ⚠:

- Please do not import any other packages.
- np.random.seed(1) is used to keep all the random function calls consistent. It will help us grade your work. Please don't change the seed.

💡 **Reminder** 💡: The basic part only includes binary classification. Functions like `softmax()` and `compute_CCE_loss` are counted as the bonus part, you can skip these implementations if you wish to do the basic part only 😀. If you are not sure which part belongs to the bonus part, you can refer to the code. For the bonus part, the code will look as follows: `### START CODE HERE ### (...) (bonus)`. Please set `bonus` to True in the first code cell if you want to do the bonus part.

❗ **Important** ❗: Please do not change the code outside this code bracket.

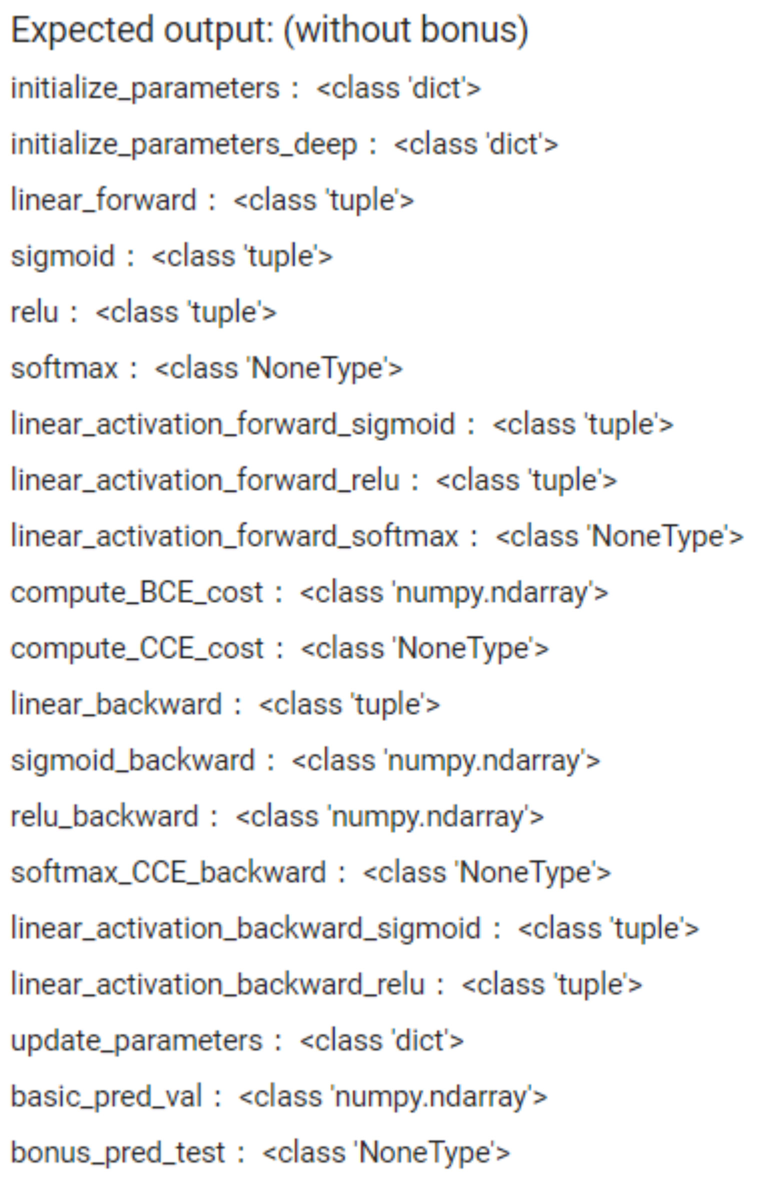
```
### START CODE HERE ### (≈ n lines of code)
...
### END CODE HERE ###
```

```
[ ]  import numpy as np
     import matplotlib.pyplot as plt
     import math
     from sklearn import datasets
     from pandas import read_csv

     output = {}

     """
```

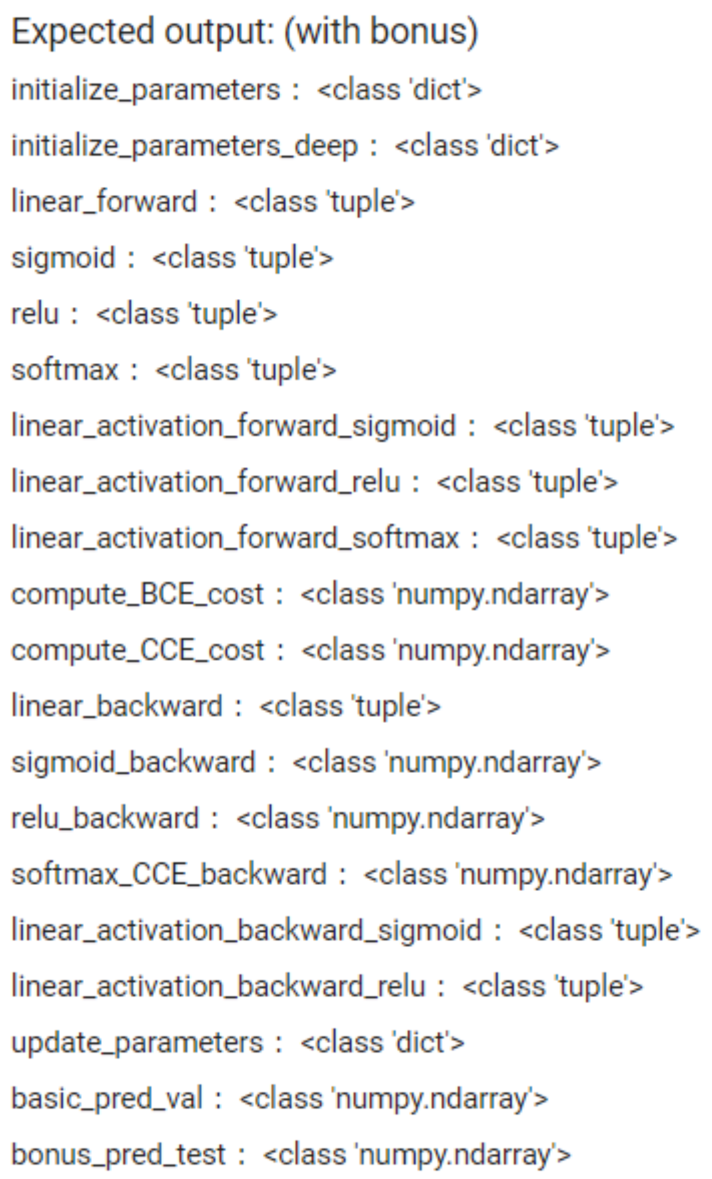
Output NPY File Format

- Named as “**[StudentID]_output.npy**”
- This file is a dictionary that stores your output for each function and the prediction for both binary classification and multi-class classification.
- The dictionary should have the following keys: 'initialize_parameters', 'initialize_parameters_deep', 'linear_forward', 'sigmoid', 'relu', 'softmax', 'linear_activation_forward_sigmoid', 'linear_activation_forward_relu', 'linear_activation_forward_softmax', 'compute_BCE_cost', 'compute_CCE_cost', 'linear_backward', 'sigmoid_backward', 'relu_backward', 'softmax_CCE_backward', 'linear_activation_backward_sigmoid', 'linear_activation_backward_relu', 'update_parameters', 'basic_pred_val', 'bonus_pred_test’
- If you do not complete the bonus part, the value for the bonus part key will be None.

Output NPY File Format



```
Expected output: (without bonus)
initialize_parameters :  <class 'dict'>
initialize_parameters_deep :  <class 'dict'>
linear_forward :  <class 'tuple'>
sigmoid :  <class 'tuple'>
relu :  <class 'tuple'>
softmax :  <class 'NoneType'>
linear_activation_forward_sigmoid :  <class 'tuple'>
linear_activation_forward_relu :  <class 'tuple'>
linear_activation_forward_softmax :  <class 'NoneType'>
compute_BCE_cost :  <class 'numpy.ndarray'>
compute_CCE_cost :  <class 'NoneType'>
linear_backward :  <class 'tuple'>
sigmoid_backward :  <class 'numpy.ndarray'>
relu_backward :  <class 'numpy.ndarray'>
softmax_CCE_backward :  <class 'NoneType'>
linear_activation_backward_sigmoid :  <class 'tuple'>
linear_activation_backward_relu :  <class 'tuple'>
update_parameters :  <class 'dict'>
basic_pred_val :  <class 'numpy.ndarray'>
bonus_pred_test :  <class 'NoneType'>
```



```
Expected output: (with bonus)
initialize_parameters :  <class 'dict'>
initialize_parameters_deep :  <class 'dict'>
linear_forward :  <class 'tuple'>
sigmoid :  <class 'tuple'>
relu :  <class 'tuple'>
softmax :  <class 'tuple'>
linear_activation_forward_sigmoid :  <class 'tuple'>
linear_activation_forward_relu :  <class 'tuple'>
linear_activation_forward_softmax :  <class 'tuple'>
compute_BCE_cost :  <class 'numpy.ndarray'>
compute_CCE_cost :  <class 'numpy.ndarray'>
linear_backward :  <class 'tuple'>
sigmoid_backward :  <class 'numpy.ndarray'>
relu_backward :  <class 'numpy.ndarray'>
softmax_CCE_backward :  <class 'numpy.ndarray'>
linear_activation_backward_sigmoid :  <class 'tuple'>
linear_activation_backward_relu :  <class 'tuple'>
update_parameters :  <class 'dict'>
basic_pred_val :  <class 'numpy.ndarray'>
bonus_pred_test :  <class 'numpy.ndarray'>
```

Assignment 3 Requirement

- Do it individually! Not as a team! (team is for final project)
- Announce date: 2021/11/18
- Deadline:
  - basic: 2021/11/30 10:00
  - bonus: 2021/12/2 23:59 (Late submission is not allowed!)
- Hand in your files in the following format
  - [StudentID]_hw3.ipynb (Please keep your code output when submitting your homework)
  - [StudentID]_output.npy
  - [StudentID]_report.pdf
  - Compress all files into [StudentID]_HW3.zip

The Evaluation Metric

For the basic function and bonus function implementations, you will get a full score if your output is exactly the same as the standard answer.

For the binary classifier and multi-class classifier, we will use accuracy to evaluate your classifier.

- Binary classification: You will get a full mark if your validation accuracy is higher than 80%.
- Multi-class classification: You will be compared with others who submit the bonus prediction.

Penalty

0 points if any of the following conditions:

- Plagiarism (randomly choose some of you to explain your implementation, problems in the next exam will be based on the basic part of hw3)
- Late submission
- Not using template or import any other packages
- Incorrect input/output format
- Incorrect submission format

Questions?

- TA: Jia He Lim (ivanljh123@gmail.com)