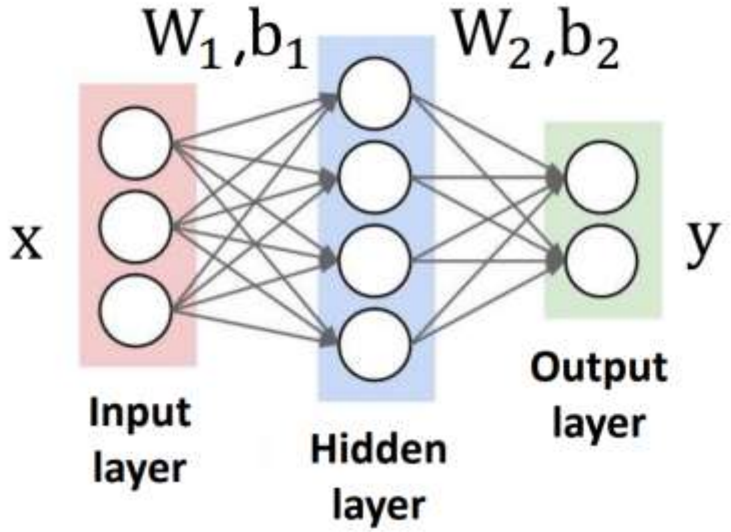
Assignment #1: MLP Implementation

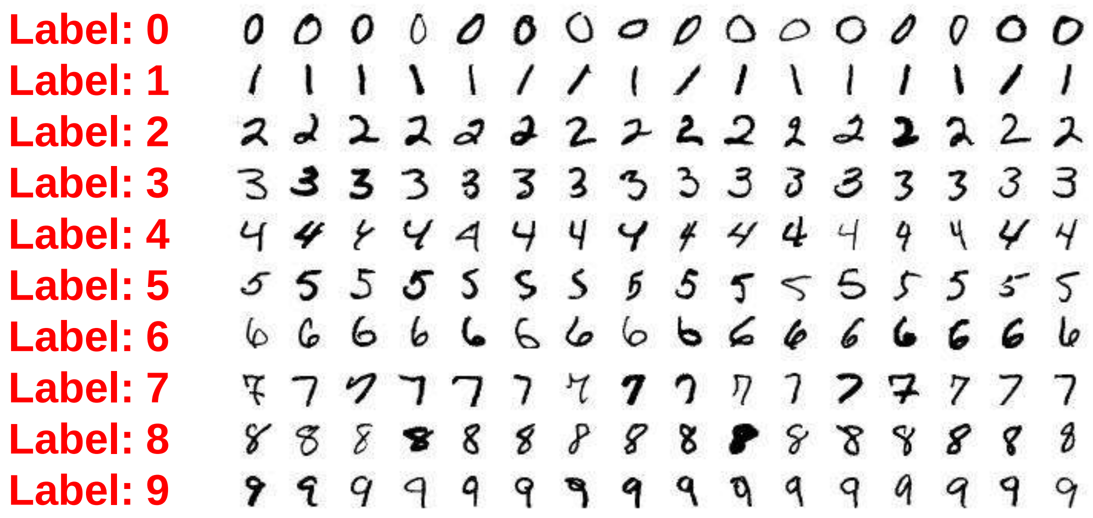
Paul Hongsuck Seo

Korea University

MLP Implementation

Implement 2-Layer Neural Net with Sofmax Classifier



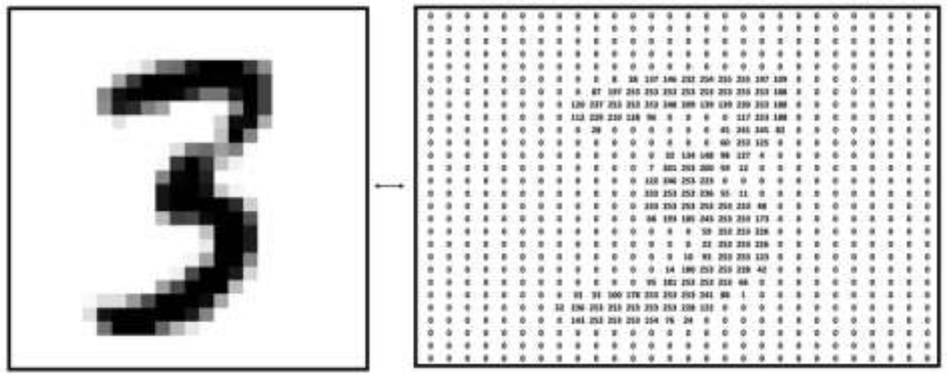


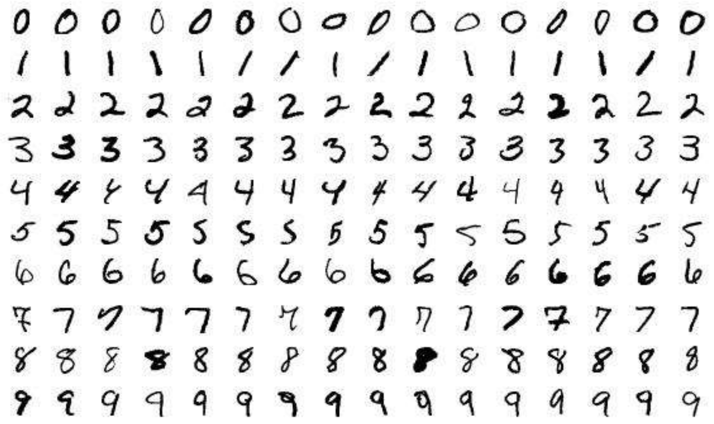
- Perform the image classification using “MNIST” dataset.
- Two weight matrices W_1, W_2 with biases b_1, b_2.
- Predicted output $\mathrm{y}' = \mathrm{W}_2\big(\mathrm{relu}(\mathrm{W}_1\mathrm{x} + \mathrm{b1})\big) + \mathrm{b}_2$.
- Total loss = data loss (softmax + log likelihood loss) + L-2 regularization loss (to W_1, W_2, not b_1, b_2).
- The Ipython Notebook “two_layer_net.ipynb” will walk you through the implementation of a two layer neural network classifier.

KOREA UNIVERSITY MIIL Multimodal Interactive Intelligence Laboratory

MINST Dataset

- What is MNIST?
  - MNIST stands for Modified National Institute of Standards and Technology database.
  - The dataset consists of 70,000 images of handwritten digits (0-9).
  - https://www.tensorflow.org/datasets/catalog/mnist
- Dataset Composition
  - Training Set: 60,000 images.
  - Test Set: 10,000 images.
  - Each image is a 28x28 grayscale pixel image, resulting in 784 features per image.

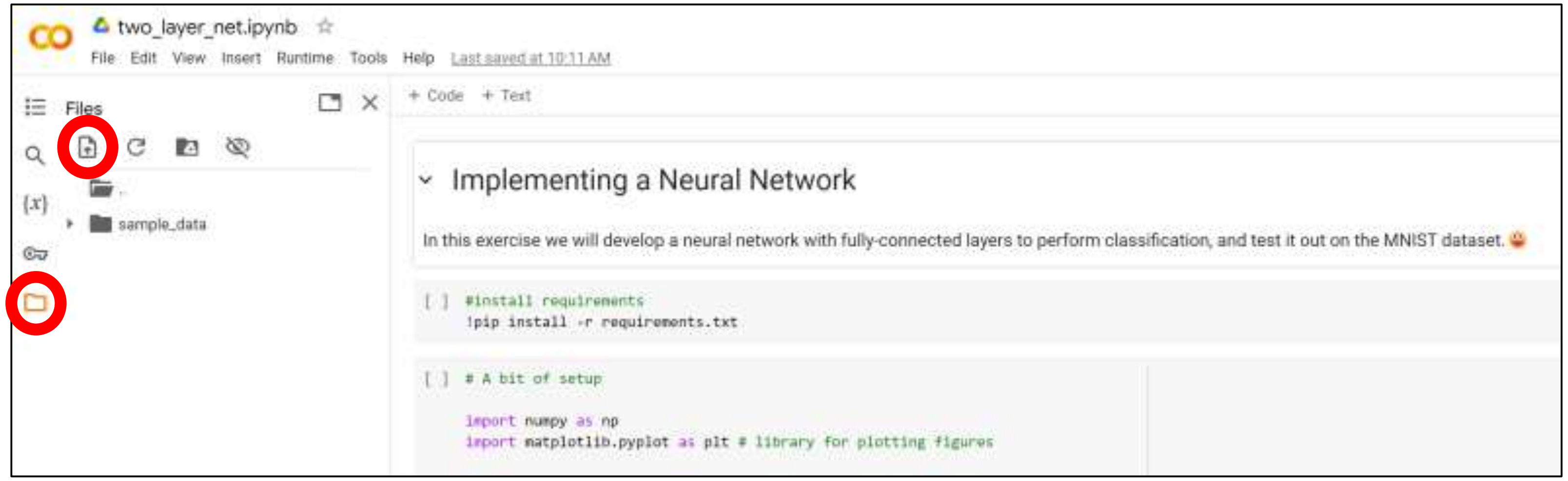




MLP Implementation

Requirements

- Download the attached zip file
- Open the **two_layer_net.ipynb** with colab notebook
- Upload the other .py files to session storage

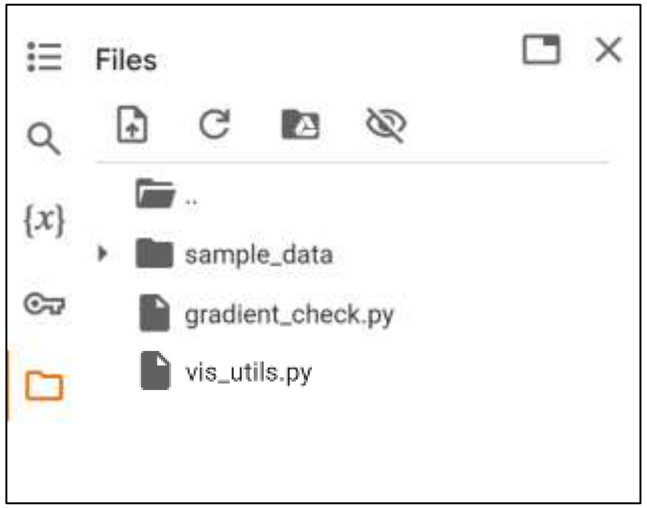

two_layer_net.ipynb
File Edit View Insert Runtime Tools Help Last saved at 10:11 AM
Files
+ Code + Text
sample_data

Implementing a Neural Network

In this exercise we will develop a neural network with fully-connected layers to perform classification, and test it out on the MNIST dataset.

```
[ ]  #install requirements
     !pip install -r requirements.txt
```

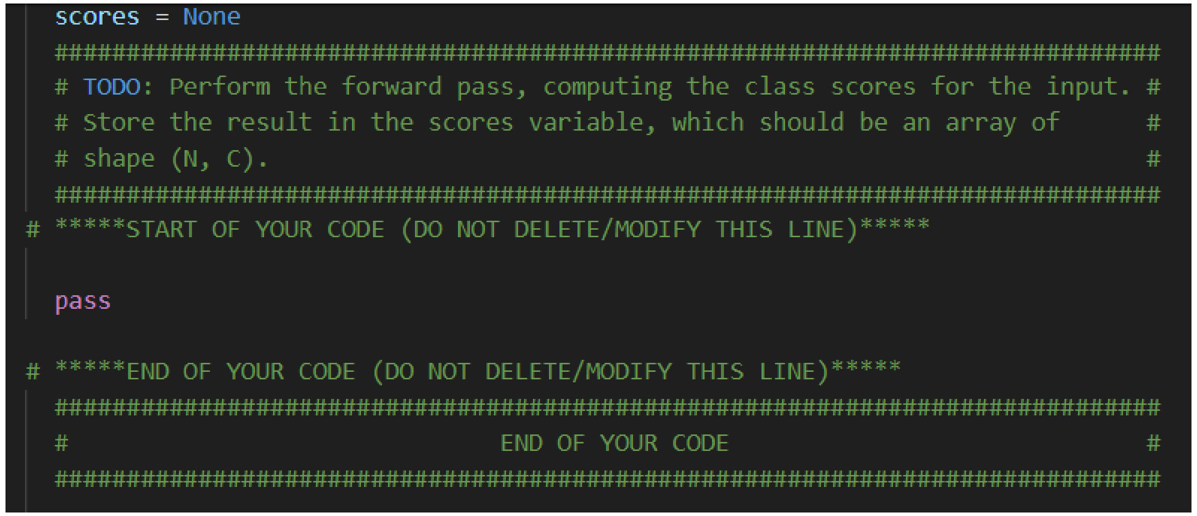
```
[ ]  # A bit of setup

     import numpy as np
     import matplotlib.pyplot as plt # library for plotting figures
```



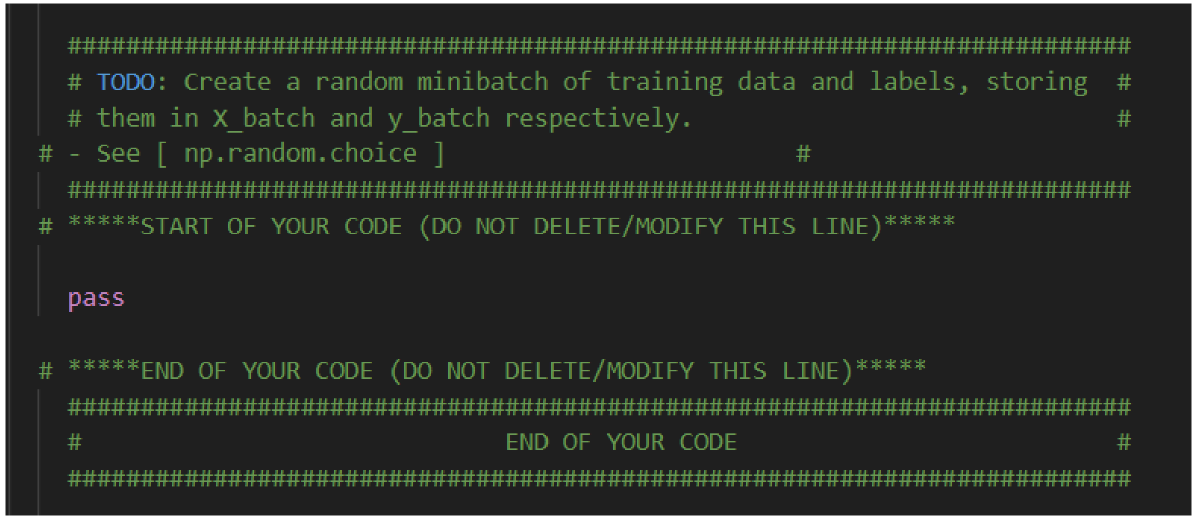
MLP Implementation

Do the following!

- Fill the code by following the instructions provided in the TODO comments
  - neural_net.py
- There are “#START OF YOUR CODE” / “#END OF YOUR CODE” tags denoting the start and end of code sections you should fill out.
- No need to modify any code other than **neural_net.py** and **two_layer_net.ipynb**



```
    scores = None
    #############################################################################
    # TODO: Perform the forward pass, computing the class scores for the input. #
    # Store the result in the scores variable, which should be an array of      #
    # shape (N, C).                                                             #
    #############################################################################
  # *****START OF YOUR CODE (DO NOT DELETE/MODIFY THIS LINE)*****

    pass

  # *****END OF YOUR CODE (DO NOT DELETE/MODIFY THIS LINE)*****
    #############################################################################
    #                              END OF YOUR CODE                             #
    #############################################################################
```



```
    #########################################################################
    # TODO: Create a random minibatch of training data and labels, storing  #
    # them in X_batch and y_batch respectively.                             #
  # - See [ np.random.choice ]                         #
    #########################################################################
  # *****START OF YOUR CODE (DO NOT DELETE/MODIFY THIS LINE)*****

    pass

  # *****END OF YOUR CODE (DO NOT DELETE/MODIFY THIS LINE)*****
    #########################################################################
    #                             END OF YOUR CODE                          #
    #########################################################################
```

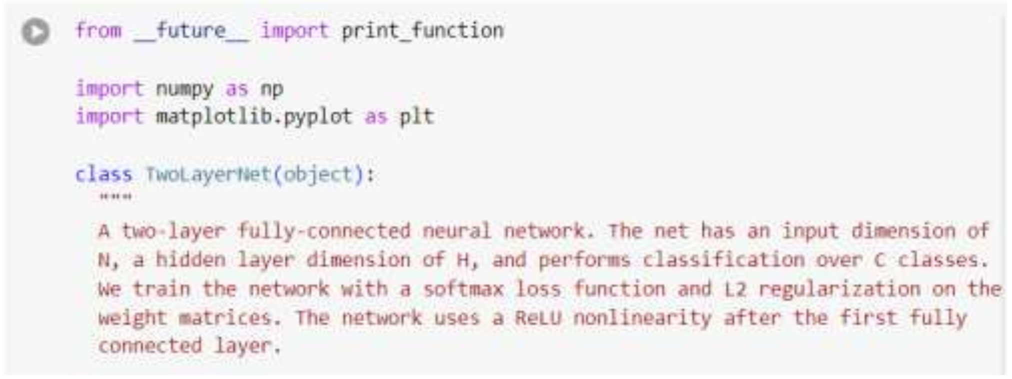
KOREA UNIVERSITY MIIL Multimodal Interactive Intelligence Laboratory

MLP Implementation

Do the following!

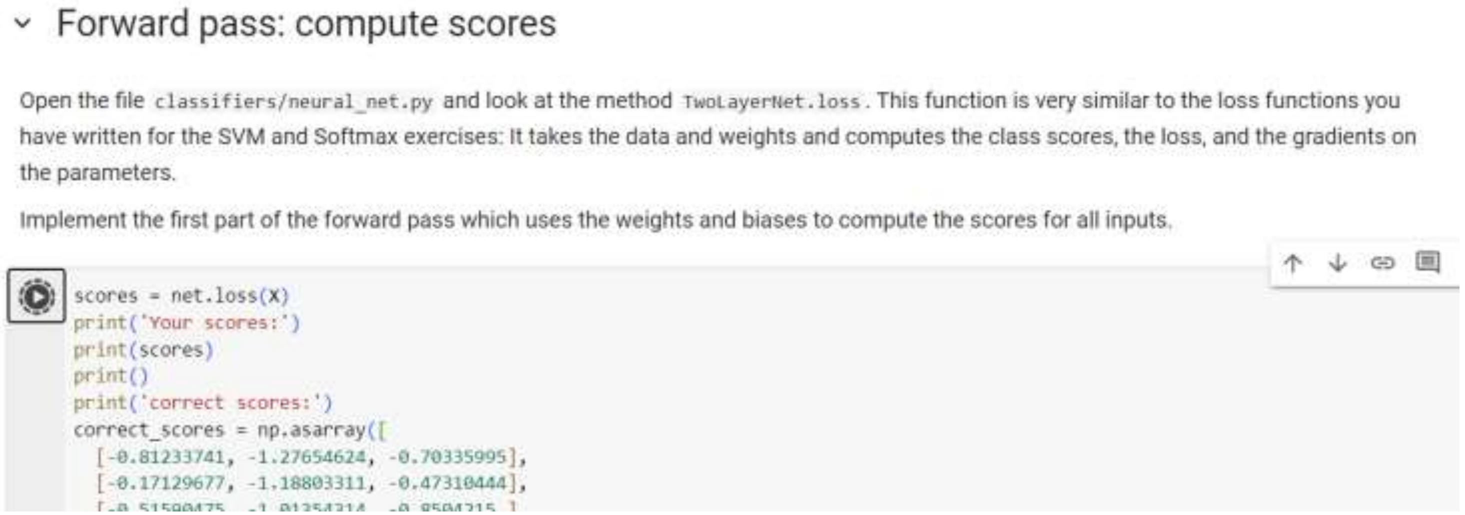
- After filling the code in neural_net.py, update the corresponding cell in the **two_layer_net.ipynb** with the modified code.
- Debug, training and tune your hyperparameters using the other cells in the **two_layer_net.ipynb**

Update TwoLayerNet class in two_layer_net.ipynb



```
from __future__ import print_function

import numpy as np
import matplotlib.pyplot as plt

class TwoLayerNet(object):
  """
  A two-layer fully-connected neural network. The net has an input dimension of
  N, a hidden layer dimension of H, and performs classification over C classes.
  We train the network with a softmax loss function and L2 regularization on the
  weight matrices. The network uses a ReLU nonlinearity after the first fully
  connected layer.
```

Complete the implementation using the other cells in two_layer_net.ipynb



Forward pass: compute scores

Open the file `classifiers/neural_net.py` and look at the method `TwoLayerNet.loss`. This function is very similar to the loss functions you have written for the SVM and Softmax exercises: It takes the data and weights and computes the class scores, the loss, and the gradients on the parameters.

Implement the first part of the forward pass which uses the weights and biases to compute the scores for all inputs.

```
scores = net.loss(X)
print('Your scores:')
print(scores)
print()
print('correct scores:')
correct_scores = np.asarray([
  [-0.81233741, -1.27654624, -0.70335995],
  [-0.17129677, -1.18803311, -0.47310444],
```

MLP Implementation

Due on Mar. 31 (Mon.), 11:59 pm (in Blackboard)

- You must submit **“neural_net.py”** along with the **report**.
  Submitting a zip file may cause error, try uploading the individual files.
- Include a **1 page** report in **CVPR** format that describes your code, results, and discussions.
- The report should be written in **English**.

CVPR format : https://cvpr.thecvf.com/Conferences/2025/AuthorGuidelines
→ Download CVPR 2024 Author Kit

MLP Implementation

Please do NOT copy your friends' and internet sources.

Please start your assignment EARLY.

"Late submissions will not be accepted"