Coding Assignment 1

In this assignment, we will implement the softmax regression for multi-class classification.

Task and Dataset. The task is to classify handwritten digits in the MNIST dataset. This iconic dataset comprises 60,000 training images and 10,000 testing images, each a 28x28 pixel grayscale representation of digits ranging from 0 to 9. We will split 10,000 from the training samples for validation. You can download the dataset from here and the codebase here.

Approach. We will implement stochastic gradient descent (SGD) for cross-entropy loss of softmax as the learning algorithm. The measure of success will be the accuracy (i.e., the fraction of correct predictions).

Note: For softmax classification, you may encounter numerical overflow if you just follow the equation mentioned in the lecture.

$$z_i = w_i^T x + b_i$$

$$y_i = \frac{e^{z_i}}{\sum_j e^{z_j}} = \frac{e^{w_i^T x + b_i}}{\sum_j e^{w_j^T x + b_j}}$$

The observation is that the exponential function increases very fast with its input, and very soon, e^z will give NAN (not a number). A trick to overcome this is to subtract every z_i by the maximum value among $z_1, z_2, \ldots$. In other words, we compute $\hat{z}_i = z_i - z_{max}$, where $z_{max} = \max_i z_i$. Then, we have

$$y_i = \frac{e^{\hat{z}_i}}{\sum_j e^{\hat{z}_j}}$$

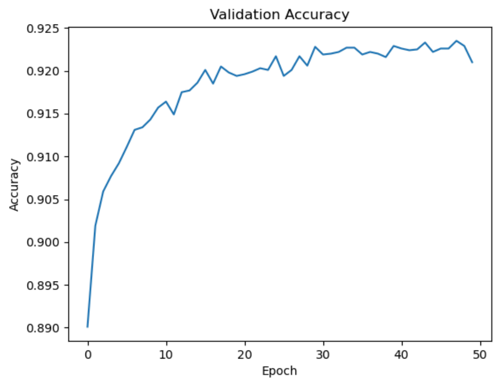
Deliverables.

1. **Reproduction [50 marks]**: Complete **`softmax_reg.py`** and run **`main.py`**. Without changing the default hyperparameters, report
    - The number of epoch that yields the best validation accuracy,
    - The validation performance (accuracy) in that epoch, and
    - The test performance (accuracy) in that epoch.

   and draw
    - The learning curve of the training cross-entropy loss, and
    - The learning curve of the validation accuracy.

   Note: A numerical error, i.e., -log(0), may occur when plotting the loss. In that case, you can truncate the loss to -log(1e-16).

A sample of the validation accuracy curve is provided to you as a reference to validate your replication:



Note: Your curve may not look exactly the same, because of the randomness in the algorithm (e.g., sample shuffling, initial weights).

2. **Analysis of the learning rate [30 marks]:** In this part, we would like to design and conduct experiments to compare three different learning rates: 1, 0.1 (default), and 0.01. Other hyperparameters remain the same as the default.

   Present two plots
   - The learning curves of training losses under different learning rates, and
   - The learning curves of validation accuracies under different learning rates.

   Note: If the training loss becomes NAN, it need not be shown in the plot. In other words, you may truncate the learning curve if the loss diverges.

   Summarize the results in the following table:

Learning rate	Best validation accuracy along training
1	0.9283
0.1	0.9237
0.01	0.9091

   Draw the conclusions:

   From the results, we see that
   - If the learning rate is too large, the learning curve is noisy and the training does not converge well.
   - If the learning rate is too small, the learning curve is smooth but the training is slow.
   - A moderate learning rate is desired.

   (Fill in the blanks with “large”, “small”, and “moderate”).

3. **You own analysis** [20 marks]: Ask one meaningful scientific question yourself, design your experimental protocol, present results in an appropriate way (slots or tables), and draw a conclusion.

   Note:
   A *scientific question* means that we can give a verifiable hypothesis that can be either confirmed or declined.

   Example of a scientific question: Is the learned classifier for this dataset better than the majority guess? Your hypothesis could be either yes or no, and it can be verified by experiments.
   Example of a non-scientific question: Does the learned classifier become better if I have super-power? My hypothesis could be either yes or no, but cannot be verified by any experiment. I don't know what superpower is, and I can say yes, or I can also say no. Neither is wrong, nor even correct.

   A *meaningful* scientific question means that you'll learn something from the experiment. Of course, what is meaningful itself is subjective. In terms of this coding assignment, the scientific question is considered meaningful as long as the student would learn something, or verify some results we mentioned in lectures.

   Example of a meaningful scientific question: How does linear regression perform for classification for this dataset? Doing this experiment will give us first-hand experience on why we should not use regression models to do classification. But you **cannot** ask this question as the solution. You need to ask your own scientific question that interests you and/or inspires others.

   Example of a not-so-meaningful scientific question: Is the learned classifier for this dataset better than the majority guess? Ok, this question is considered scientific, but too trivial for us, although it is not necessarily trivial for those who don't know machine learning at all. Again this shows the subjectivity of evaluating the significance of science.

Submission.

The submission should include a codebase and a report.

- The codebase must be a single .zip file, containing all .py files. Note: .ipynb is not allowed, because in research we typically code with .py.
  - For Problem 3, the student needs to create one or more new .py files.

- The report should be a .pdf file, containing solutions to all deliverables. The easiest way to create a report is to make a copy of this Google Doc and fill in the blanks. No code snippets or program logs are allowed in the pdf report. However, pseudo-code (if necessary) is allowed.
    - In Problem 3, a student needs to present the purpose of the analysis as well as the experimental protocols. The results must be shown in tables and/or plots as appropriate; they cannot be loose numbers (such as “this model achieves a performance of 90%”). The tables and/or plots should include baselines or alternative setups to draw your conclusions. Students may mimic the experimental design in Problem 2.