אוניברסיטת בן-גוריון בנגב
Ben-Gurion University of the Negev

Applied Deep Learning
O. Azencot
Spring 2025

Assignment 3: **Feedforward Neural Networks using PyTorch**
Deadline: May 15, 5 pm, 2025

In this home assignment, you will implement a few basic feedforward classification neural networks and their associated training and evaluation procedures using the PyTorch framework. Submission is in pairs. Your submission should include two files: 1) A PDF file named `report.pdf` containing the answers to the tasks below, and 2) A compressed container named `code.zip` containing all the code you used. Please submit your work via Moodle. Submission is in pairs. For questions, use the Slack channel, or contact me via email.

1 Background

In the fifth lecture, we learned about feedforward neural networks, how to design them properly, and various other useful tips. During the coding session, we demonstrated an example of a simple classification model on the MNIST dataset, which consists of 28×28 images of handwritten digits. In what follows you will extend the example we showed in class in terms of hyperparameter choice, features analysis, model performance analysis, and more.

Here, we define a few concepts that you will need for implementing the tasks below.

Dataset error. Let $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^N$ denote a dataset, and let $\mathcal{L}(\cdot, \cdot)$ be a loss function (e.g., the cross entropy). The *dataset error* is defined as the mean loss error over the entire dataset, i.e.,

$$e_{\text{data}} = \frac{1}{N} \mathcal{L}(y_i, \tilde{y}_i) \ ,$$

where $\tilde{y}_i := f(x_i; \theta)$ is the prediction of the trained model $f(\cdot\,; \theta)$ for x_i. We will use $e_{\text{tr}}, e_{\text{te}}, e_{\text{va}}$ to denote the train, test and validation errors, respectively.

2 Tasks

1. **MNIST classification.** Reproduce the code we showed in class: load the MNIST train and test sets, implement a basic two-layer fully connected neural network, define the cross entropy loss function and optimizer, and implement train and evaluation procedures. In your report, add a plot of the train and test error graphs during training. You should compute the $e_{\text{tr}}, e_{\text{te}}$ after every epoch. In addition, you should report the test error you obtain when training has finished. In addition, attach to your report a plot of some of the misclassified images.

2. **Mitigate pseudorandomness.** Adapt your code from Task 1 so that it produces the same results at each run (see e.g., link). Now, run your non-random code using 5 different seed numbers. Attach a plot of the test errors during training from the separate runs. Report the mean and standard deviation of the final test errors. Based on the variance you compute, try to determine whether your model can be considered robust to the choice of a seed number.

3. **Validation dataset.** Split the MNIST dataset to three sets (instead of two): train, test, and validation. This is typically done by randomly selecting 10,000 images from the train set to be used as a validation set. Repeat the training required in Task 2 of five different models. However, instead of reporting the final test error, you should report the test error that corresponds to the *minimum validation error*, e_{va}, that was obtained during training. Namely, compute $(e_{\text{te}}, e_{\text{va}})$ after every epoch. Report the pair $(e^*_{\text{te}}, e^*_{\text{va}})$ such that e^*_{va} is the minimum throughout training.

4. **Grid search.** The classification task we consider in this assignment has several hyperparameters. In particular, the user needs to choose values for the hidden size of the intermediate layer, the batch size, and the learning rate. The standard approach to choosing these parameters uses a grid search over the space of parameters. To do so, you should choose the possible values for each parameter we mentioned above, and train models for each combination of parameters. In your report, include a table with e_{te} that corresponds to the best e_{va} for every combination. Highlight the combination that yields the best performance.

5. **Feature analysis.** Let $z_i = \sigma(W^{(1)T} x_i + b^{(1)})$ represent the hidden features of the image x_i, obtained from applying the first layer on the input. Attach to your report a plot with the 2D embedding of z_i using `tSNE`, for all z_i in the train set. Each 2D point should be colored based on its label. Use different colors for the digits, so it will be easy to distinguish between the points. Generate a similar plot, but for x_i. What can you say about the differences between these plots? What can you say about the learned model?

3 Additional Comments

1. Do not attach code to the report. Please follow the guidelines in terms of how to attach code to your submission.

2. A significant portion of the grade is dedicated to the neatness and descriptiveness of the report. You should make all the figures to be as clear as possible.

3. At the same time, the report should not be too long. Please aim for an (at most) 8 page document.

4. If you struggle with computational resources, you are allowed to shrink the MNIST dataset by a factor of 10, i.e., use only 6000 instead of the total 60,000.

5. If you use `torchvision datasets`, you can follow this post to extract a validation set.

6. You should limit your grid search to $10 - 20$ models.

7. The package `sklearn` includes an implementation of `tSNE`, check the following documentation.