Homework 4

AMATH 482/582, Winter 2024

Assigned Feb 25, 2024. Checkpoint due on Mar 4, 2024 at midnight.
Full version due on Mar 11, 2024 at midnight.
LATE REPORTS WILL NOT BE ACCEPTED.

Directions, Reminders and Policies

Read these instructions carefully:

You are required to upload a PDF report to Canvas along with a zip of your code.

The report should be a maximum of 6 pages long with references included. Minimum font size 10pts and margins of at least 1inch on A4 or standard letter size paper.

Do not include your code in the report. Simply create a zip file of your main scripts and functions, without figures or data sets included, and upload the zip file to Canvas.

Your report should be formatted as follows:

- Title/author/abstract: Title, author/address lines, and short (100 words or less) abstract. This is not meant to be a separate title page.
- Sec. 1. Introduction and Overview
- Sec. 2. Theoretical Background
- Sec. 3. Algorithm Implementation and Development
- Sec. 4. Computational Results
- Sec. 5. Summary and Conclusions
- Acknowledgments (no more than four or five lines, also see the point below on collaborations)
- References

LaTeX(Overleaf is a great option) is recommended to prepare your reports. A template is provided on Canvas in Homework/Files. You are also welcome to use Microsoft Word or any other software that properly typesets mathematical equations and properly allows you to include figures.

Collaborations are encouraged, however, everything that is handed in (both your report and your code) should be your work. You are welcome to discuss your assignments with your peers and seek their advice but these should be clearly stated in the acknowledgments section of your reports. This also includes any significant help or suggestions from the TAs or any other faculty in the university. You don't need to give all the details of the help you received, just a sentence or two.

Your homework will be graded based on how completely you solved it as well as neatness and little things like: did you label your graphs and include figure captions. **The homework is worth 20 points. 10 points will be given for the overall layout, correctness and neatness of the report, and 10 additional points will be for specific technical things that the TAs will look for in the report itself.**

LATE REPORTS WILL NOT BE ACCEPTED.

Problem Description: Image Classification with Deep Neural Networks

Your goal in this assignment is to train Fully Connected Deep Neural Networks (FCNs / DNNs) to classify images in FashionMNIST data set. FashionMNIST is similar to MNIST that you worked with in assignment 3. As in MNIST, each sample is a grayscale 28×28 image and the training set consists of 60K images and the testing set consists of 10K images. Images depict articles of clothing that belong to a class from one of 10 classes. It is a fundamental benchmark dataset in machine learning as well. In comparison to MNIST, FashionMNIST classification is more challenging. This warrants more powerful and nonlinear models. Here you will implement FCNs toward this aim.

Figure 1 First 64 Images in FashionMNIST Dataset.

Some comments and hints

Here are some useful comments and facts to guide you along the way.

1. You are provided a code template (HW4_HelperTemplate.ipynb) for FCN for FashionMNIST. Your goal will be to complete the template into a working model and decide on the architecture of your model (i.e. number of layers and neurons in each layer). Keep in mind that the more layers you will be adding the more weights you will have to train which will result into a higher computational complexity which is not necessarily beneficial for classification accuracy and for your experiments. Once you have completed this you will perform hyperparameter tuning and analyze the classification for this dataset.
2. In the provided template, the dataset is loaded with *torchvision*, which contains multiple datasets such as MNIST, FashionMNIST, and has useful transformations (e.g. normalization) and loads data as Pytorch tensors. The training data is split into Training and Validation datasets and loaded into DataLoader as *batches*. DataLoader allows for easy iteration over the batches and samples within it. Batches are useful for setting when weights updates will occur and also if you are using GPU to speed up computation. You will need to define the batch size you want for your training and test sets.
3. The input data (features) are saved as 28×28 images. You will need to reshape images into a vector of 784 which will be input into your network.
4. For implementation of different optimizers, regularization, initialization and normalization, refer to example codes and pytorch library tutorials (https://pytorch.org/tutorials/) for examples.

Tasks

1. Design your FCN model to have an adjustable number of hidden layers and neurons in each layer with *relu* activation. The first (input) layer should be of dimension 784 and the last (output) layer of 10 corresponding to 10 classes. Use Cross Entropy loss to perform the classification and SGD optimizer with learning rate as an adjustable parameter. Set the number of epochs as an adjustable parameter and train the network. Inspect the training loss curve and include validation of accuracy, on the validation set, at each epoch. Perform testing at the end of training. In summary, your parameters should allow you to vary the number of training batches, number of layers, number of neurons in each layer, learning rate, number of training epochs. You can add more parameters as you see fit.

2. Find a configuration of the FCN (by varying the adjustable parameters) that trains in reasonable time (several minutes is reasonable) and results in reasonable training loss curve and testing accuracy (at least above 85% testing accuracy). These parameters will be your baseline configuration/parameters.

3. Perform hyperparameter tuning from the baseline, trying the following configurations. Report your results in terms of tables, plots of loss curves and accuracy curves.

   (a) Consider different optimizers including *RMSProp, Adam and SGD* with different learning rates. You can find the corresponding functions in the torch.optim library. Log your training loss, validation and test accuracy. Compare these optimizers, and observe which optimizer is most suitable for your FCN. Try to explain why.

   (b) Analyze the overfitting/underfitting situation of your model. Include *Dropout* regularization and discuss whether this improves performance.

   (c) Consider different *Initializations*, such Random Normal, Xavier Normal, Kaiming (He) Uniform. Discuss how these initialization affect the training process and the accuracy.

   (d) Include normalization such as *Batch Normalization*. Discuss the effect of the normalization on the training process or testing accuracy.

4. **Bonus (+2 points)**: Perform hyperparameter tuning of FCN to achieve testing accuracy of $> 90\%$ for FashionMNIST and $> 98\%$ for MNIST. Report your configurations and try to explain them. Apply the most successful classifier from assignment 3 to FashionMNIST (on full dimension of 784 features) and compare with results above.