Homework 4
600.482/682 Deep Learning
Spring 2021

February 26, 2021

Due 11:59pm on March 12, 2021
Please submit 1) a single zip file containing your Jupyter Notebook and PDF of your Jupyter Notebook to "Homework 4 - Notebook" and 2) your written report (LaTeX generated PDF) to "Homework 4 - Report" on Gradescope (Entry Code: D5PDXD)

Note 1) This is the first assignment that it is important to be cognizant of model training times. The MLP models you design in Problem 1 likely won't take longer than a few minutes to train at maximum. However, in Problem 2, you will implement two convolutional neural networks *and* experiment with hyperparameters/architectures. For context, our proposed designs required 40-60 seconds per epoch, and we ask you to train your models for at least 50 epochs. We recommend you design and test your model at a small scale (e.g. 5-10 epochs) before scaling up for the final run.

Note 2) You will be using PyTorch for the majority of this assignment. We **strongly recommend** using Google Colaboratory as well, though you are welcome to work locally if desired.

1. We learnt that a two-layer MLP can model polygonal decision boundaries. You are given data sampled from a "two-pentagons" decision boundary. Data points within the pentagons are considered "true" (labeled as 1) while data points outside are considered "false" (labeled as 0). The data can be downloaded here: `https://piazza.com/class_profile/get_resource/kjsqo5dq9wb7nh/klmp8ma5f196q5`.

   Please load the data via `np.load()` function and check that you obtain a numpy array with a shape of $(60000, 3)$. The first two colomns are x, y coordinates and the third one represent the labels.

   The vertices of the 2 polygons are (500, 1000), (300, 800), (400, 600), (600, 600), (700, 800) and (500, 600), (100, 400), (300, 200), (700, 200), (900, 400). We have provided some visualization functions in the Jupyter notebook. The decision boundary can be modeled with a total of $2 \times 5 + 2 + 1$ neurons with threshold activation functions.

   (a) We have set up an MLP with threshold activation by analytically obtaining weights that can model the above decision boundary.

      i. Implement the "AND gate" and "OR gate" to predict all points within the 2 polygons as the positive class. Use the `predict_output_v1()` to test your implementations. Attach the visualization to your report.

      ii. Modify the code in `predict_output_v2()` to predict only the points in polygon 1 as the positive class. Attach the visualization to your report.

      Note: We reverse-engineered the weights from the known decision boundaries but in the real world, we do not know the decision boundaries. In a more realistic scenario, we would thus need to discover (learn) the decision boundaries from training data which we will explore next.

   (b) Build an MLP using sigmoid activation functions to replace the "AND/OR" gates. You are required to use PyTorch for this question and all remaining parts of the assignment. We **strongly recommend using Google Colaboratory** for ease of debugging and modeling flexibility. Whenever we provide code in subsequent assignments, it will be optimized for this setup.[1]

Model hyperparameters:

i. Use 2 hidden layers, and 1 output layer with binary-cross-entropy loss [2]
ii. Use 10 nodes in the first hidden layer, 2 nodes in the second hidden layer, and 1 output node (binary classification), which mirrors the capacity of the model in 1a)
iii. Initialize the weights using Xavier Initialization
iv. You may use any combination of batch size, training epochs, and learning rate. (Note that a larger batch size typically corresponds to a larger learning rate)
v. Do not use any regularization for this problem

Use gradient descent to optimize the parameters. You should expect an accuracy of above 90%. **Please use the first 50000 points as your training set and the remaining 10000 as your test set. You may also want to normalize the data before training.**

TODO:

i. Train your model using 5 random seeds and report the mean and standard deviation of the train and test accuracy at the end of 500 epochs. Attach the plots of training loss and testing accuracy change over epochs.
ii. In your report, attach the visualization of the test set prediction (i.e. polygon figure).
iii. Brief discussion. Are you able to find a solution that performs almost as good as the “manual" solution in (a)?

(c) Build an MLP with a larger capacity (increase the depth and width) and see whether you can achieve higher classification accuracy.

i. Like before, train your model using 5 random seeds and report the mean and standard deviation of the train and test accuracy at the end of 500 epochs.
ii. Report what depth and width you used. Briefly discuss what you notice about your model performance as you change these architecture decisions. *Hint: Think about our discussions from lecture regarding model capacity.*
iii. Attach the plots of training loss and testing accuracy change over epochs.
iv. Attach the visualization of the prediction for the test set (i.e. polygon figure).

(d) In part (b) we had fixed the hyperparameter choices, but in part (c) you adjusted those choices for the MLP in terms of depth and width of the network. Would you consider your test results to be valid and generalizable? Please briefly discuss.

2. In this problem, we will explore convolutional neural networks (CNNs). We will be working with the FashionMNIST dataset.[3] This dataset only has a train-test split, therefore we will be using the last 10000 training instances as the validation set. Again, we please use PyTorch[4] for this assignment. Because training times can be quite long, you *do not* need to train your model with multiple random seeds — 1 is enough, but of course feel free to expand if you have time and are confident in your model. You should try to train your models for at least 50 epochs.

(a) Design a small CNN (e.g. 3-4 hidden layers) using the tools we learnt in class such as

i. convolution and pooling layers
ii. activation functions (other than sigmoid)

[1] https://pytorch.org/. If you have not programmed with PyTorch before, give yourself some time to go through some online tutorials. There are many on the internet so its hard to recommend a specific one, it would really depend on your level of comfort. You may also find it helpful to review the PyTorch resources on Piazza (https://piazza.com/jhu/spring2021/cs482682/resources).

[2] See discussion here https://stackoverflow.com/questions/53628622/loss-function-its-inputs-for-binary-classification-pytorch

[3] The full FashionMNIST dataset can be downloaded from the official website here: https://github.com/zalandoresearch/fashion-mnist. We provide starter code for loading and splitting the dataset in the notebook.

[4] https://pytorch.org/get-started/locally/

TODO:

i. **Briefly** describe your architecture in your report and explain your design choices (e.g. I used ReLU activation because...) **Please design the network architecture by yourself. This is not about performance.**
ii. Report the train, validation, and test accuracy of your best model (e.g. if the best model was obtained at epoch 50, report the train, validation, and test accuracy at epoch 50).
iii. Attach the plots of training loss and validation accuracy change over epochs.

Note: If you achieve more than 85% accuracy on the test set, move on to the next subproblem.

(b) Now, try to improve your model's performance by including additional architecture elements. You should implement *at least two* of the following:

i. dropout
ii. batch normalization
iii. data augmentation
iv. different optimizers (other than vanilla stochastic gradient descent)

Try to improve your accuracy over the model in 2(a). Your new model should perform at least as well as your previous model.
TODO:

i. Report the train, validation, test accuracy of your best model.
ii. Attach the plots of training loss and validation accuracy change over epochs.

(c) Search the internet for models that others have built on FashionMNIST. This can be from research papers or even something like Kaggle. Briefly describe one technique or architecture decision which you found interesting (remember to cite your sources). It can be an extension of an existing method, and you don't need to cover the paper's results for that method. As a rough guide, anything from approximately 50-100 words of prose is sufficient.