Advanced Machine Learning - Classification Midterm Project

November 2023

1 Instructions

- The entire project will be graded and will contribute 30% of the final grade.
- It is required to submit notebooks only, you should not write a final report. Nevertheless, the notebooks you submit should contain comments and explanations of your procedures.
- I will not grade you on the final quality of the classification tasks, but I will grade i) correctness of the Python implementation, ii) quality of the dataset visualization, iii) quality of the comments and findings, iv) scientific correctness of the performed steps.
- Please submit only one zip file containing all the scripts you wrote.
- Submit the zip file on iCorsi. The deadline is **November 19, 23:59**.
- You can talk about the task to your fellow students, but please do not share your implementation/code. Plagiarism in the source code or the report will be penalized with a grade of 0 and might be reported to the faculty.
- You are absolutely welcome to ask me anything about things that are unclear to you in this assignment. Feel free to contact me for feedback or suggestions.

2 Tasks

- If not specified, for evaluation of your results use the metric you think is more appropriate among those we saw in the lectures.

2.1 Task 1

In the following Tasks you are going to work with the digits dataset in sklearn and the `Fashion MNIST` dataset from `tensorflow.keras`. Use the tool you prefer (e.g., LDA, t-SNE) to perform a preliminary visualization of the two datasets.

2.2 Task 2

Run a multilayer perceptron (a densely connected feed forward neural network) with two hidden layers and rectified linear nonlinearities (i.e., `RELU`) on the digits dataset from sklearn using the `tensorflow.keras Sequential` interface. Include code for selecting $l2$-norm regularization strength and number of hidden units using `GridSearchCV` and evaluation on an independent test-set.

Bonus points

Transfer learning - Reuse an existing architecture and pre-trained weights from keras (https://keras.io/api/applications/). Compare retraining only the densely connected layers with fine-tuning the whole network.

2.3 Task 3

Train a multilayer perceptron (fully connected) on the `Fashion MNIST` dataset using the traditional train/test split as given by `fashion_mnist.load_data` in `tensorflow.keras`. Use a separate 10000 samples (from the training set) for model selection and to compute learning curves (accuracy vs epochs, not accuracy vs n_samples). Compare a "vanilla" model with a model using drop-out and evaluate if using drop-out allows you to learn a bigger network. Then, compare to a model using batch normalization. Visualize learning curves for all models.

Bonus points

Augment the data using rotations, mirroring and possibly other transformations. How much can you improve your original model by data augmentation?

2.4 Task 4

Perform calibration on one of the previously implemented models of your choice. Plot the Reliability Diagram before and after the calibration procedure.

3 Suggestions

- You can use `StratifiedShuffleSplit` to create a single train-validation split for use with `GridSearchCV`. Use of `GridSearchCV` might not be the best option for any task but Task 2, though.
- Preprocess the images before visualization and/or training a model.
- Test your code on a small part of the data before training the model. You don't want your code to fail on a print statement after waiting for the network to train.