Introduction to machine learning, WS 2021/22

Problem 1 – part B

Description: Use the standard `Fashion-MNIST` dataset from the `torchvision` library to train a model able to classify the images by their type.

Your task: Use the available train data to train a softmax regression model to predict the classes. Test the model on the test part of the data and calculate the average accuracy over the test set (accuracy = # correct / total instances).

Details of your task: You shall create a jupyter notebook in which you describe your steps (in plain English in markdown cells) and put your code (in the code cells) so that that your results can be reproduced when I (or anybody else) re-rerun the complete jupyter notebook.

Implementation rules: All work shall be done in Python using PyTorch and its API. ***For this problem, you should use the pytorch API as much as possible (in particular `torch.nn`, `torch.optim`, `torch.utils`, `torchvision` modules)***.

Teams: The same as for part A.

Submission: Submit the complete jupyter notebook via e-learning (only once for the team of two). ***The deadline is Sunday 14.11. 23:59***.

Questions: If you have any, please ask me right away (via mattermost). Don't wait till we see each other in the course, it may be too late for you then.

Detailed list of steps to follow:

1. Get the `Fashion-MNIST` train and test dataset from `torchvision`
2. Instantiate the DataLoaders to be able to iterate over batches of data.
3. Define the linear model which outputs the logits for each class $\mathbf{o}^{\mathbf{T}} = \mathbf{x}^{\mathbf{T}}\mathbf{W} + \mathbf{b}^{\mathbf{T}}$. Use the `Linear` layer of the `nn` module in torch. Remember this layer expects each input instance is a vector so multiple instances are organized in a N, d matrix. (Hints: check the `Flatten` layer and `Sequential` container.)
4. Define your loss function as the cross-entropy loss using the `nn` module. Remember that this actually performs two operations – softmax over the logits $\mathbf{o}$ to get the probabilities $p(\mathbf{y}|\mathbf{x})$ as well as the evaluation of the average cross-entropy loss. Check the documentation for the format (representation) of the targets (and the outputs of your model) it expects.
5. Define the optimization algorithm as the `SGD` from `optim` module of torch.
6. Write the model training procedure. Remember that this follows always the same logic for all supervised models (looping through batches in multiple epochs, evaluating the loss, updating the parameters, etc.) so you may want to write it as a reusable function (or even class). What changes is the data, the model, the loss function, and the optimization procedure. So you may want to make these into flexible arguments. Remember to monitor the training loss evolution and plot it to see it behaves reasonably.
7. The very last step is to use the trained model for predictions and evaluating its prediction accuracy. Write your own `accuracy` function which takes as arguments the true targets and your predictions (not the vector of probabilities but the single class with the highest probability) and calculates the

average accuracy over the dataset. Use this function to calculate the accuracy over the train data as well as the test data. Write a short comment in which you compare these + answer which one you think is more important. (Hint: The softmax transforms the logits $\mathbf{o}$ into the probabilities. But it actually does not change the ordering of the dimensions – the highest o_j has the highest value softmax(o_j) and vice versa.)

8. Play with the values of the hyper-parameters, try different combinations and see how good/bad your model training is. Try to find a good combination of these to get good accuracy. What is the best you got?