Assignment 2

Due date: Wed, Nov 29, 11:59pm

Assignment 3 consists of one part – programming. The programming part can be submitted either as IPython Notebooks (recommended) or as stand-alone scripts. Python interpreter and imported libraries should be compatible with the latest Anaconda distribution (hCps://www.anaconda.com/).

Programming part (100 points total)

In this assignment, we will be working with the MNIST dataset. You are required to use the dataset available via the scikit-learn library: https://scikit-learn.org/stable/auto_examples/classification/plot_digits_classification.html. It is a lightweight version of the MNIST that contains 1,797 samples total and consists of 8x8 pixel images of the handwritten digits. The original dataset contains 60,000 samples with a size of 28x28. You are requested to implement three machine learning models – perceptron, fully connected neural network, and convolutional neural network. You have to implement them ***from scratch***, i.e., without using deep learning libraries with ready-to-use layers. For all models, use cross-entropy as the loss function for training and F1 score and confusion matrix for tracking the performance. The components can be reused between the models, so try to identify common tasks you will be performing on each step for this assignment to leverage it and reduce your workload.

(10 points total) Data preprocessing. Make sure that MNIST digits are represented in a way you can work with. One of the suggestions is the usage of 1-d arrays. Add padding with the size 1. Split the dataset into train (60%), validation (20%), and test (20%) sets with stratification:
https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html

(20 points total) Implement perceptron and evaluate its performance. Your program should be to perform the following:

1. Take MNIST images as an input.
2. Conduct optimization step using Stochastic Gradient Descent (SGD)
3. Use one of the following activation functions:
    a. Linear (Identity function)
    b. Sigmoid
    c. ReLU

4. Construct learning curves, i.e., record performance on test and validation sets for each training epoch. Construct a plot that displays training loss and validation loss (learning curves) simultaneously. Use cross-entropy loss for this task.
5. For each of the activation functions in 3, run training for 30 epochs. Record training curves. Evaluate performance on the test set. How well does the model perform for each of them? What is the state of the model based on the learning curves (e.g., underfitting, overfitting, converging/not converging)?

(30 points total) Implement neural network with 3 hidden fully-connected layers. In other words, your neural network should have input layer, 3 hidden layers, and an output layer. Like for the perceptron, you should implement it from scratch.

1. Take MNIST images as an input
2. Implement backpropagation for the usage with SGD. Do not forget to add bias.
3. Repeat steps 3-5 from perceptron problem in application to the neural network. Report your results

(40 points total) Implement convolutional neural network that consists of input layer, 2 convolutional layers, 1 fully connected layer, and an output layer.

1. Take MNIST images as an input
2. Implement a convolutional layer. You need to have the following functionality:
   a. Taking in multiple channel input
   b. Producing multiple channel output
   c. Specifying kernel size and stride for the layer
3. First convolutional layer needs to have a kernel size 4 and output 3 channels.
Second convolutional layer needs to have a kernel size 3 and output 9 channels (3 new output channels per one input channel)
4. Repeat steps 1-3 from perceptron exercise