Assignment 3: Implementing convolutional neural networks in PyTorch

Small changes to this assignment are still possible. They will be listed here:

- Added bonus task

Due: November 25, 2020

Goal: Implement convolutional neural networks (CNNs) in PyTorch.

Submission: The assignment consists of two parts: implementation and an analysis. You are supposed to present results for both parts in the following manner:

1. Upload your code.
2. Prepare a report with an analysis of results obtained at home.

The code and the report must be uploaded due to the deadline to Canvas.

UPLOAD A **SINGLE FILE** (a zip file) containing your code and the report. Name your file as follows: [vunetid]_[assignment number].

Introduction

In this assignment we are going to learn convolutional neural networks (CNNs). The goal is to implement a CNN, which is a combination of convolutional layers, pooling layers, and fully-connected layers (aka linear layers) and analyze its behavior. The CNN models a conditional distribution:

$$p(y|x;w) = softmax(CNN_w(x)).$$

Here, we are interested in **minimizing** the following objective function (i.e., the negative logarithm of the likelihood function):

$$l(w) = -\sum_n \log p(y_n|x_n;w) .$$

Part 1: Implementation

Implement a convolutional neural network as a module class in PyTorch

Implement a CNN:

1. Write code in Python using PyTorch.
2. Please use the following architecture:

   Conv2d (1->16 channels, kernel: 3x3, stride: 1, padding: 1)

   ReLU()

   MaxPool (2x2)

Conv2d (16->4 channels, kernel: 3x3, stride: 1, padding: 1)

ReLU()

MaxPool (2x2)

Linear (196, 10)

Softmax()

3. Please calculate the number of weights and compare it to the number of weights in an MLP in the first assignment. Include your computation in your report.
4. Please use the ADAM optimizer.
5. Please use MNIST dataset.

BONUS:

In this task you will implement a Conv2d layer via other PyTorch functions. This task will give you an in-depth understanding of the convolution operation.

1. Implement a custom Conv2d layer in Pytorch **without** using convolutional operations.
2. Please include pseudo code in your report explaining the implementation of your custom Conv2d layer.
3. Please analyze the velocity of your implementation with regard to the original PyTorch Conv2d function. Run a single layer Conv2d (32->32 channels, kernel: 3x3, stride: 1, padding: 1) 1000 times for both implementations and report their mean duration. Analyze the results and include the analysis in your report .

TIPS:

- You can easily check your implementation by comparing the output of an original Conv2d and your custom Conv2d function for a particular random input signal. They should be equal.
- Some possibly interesting functions are: Unfold, matmul, einsum.

Part 2: Analysis

Learn the CNN using the ADAM optimizer and analyze the performance of the CNN:

1. Analyze learning curves during training. If you use a separate validation data (NOT test!), please analyze learning curves on it as well.
   At the end of training (e.g., after a fixed number of epochs), calculate the test classification error.
2. Run the ADAM optimizer multiple times (at least 3) and plot an average and a standard deviation of the objective value in each iteration (e.g., see : here).

3. Run the ADAM optimizer with different step sizes (e.g., 0.0001, 0.01, 0.05). Analyze how the step size value influences the final performance.