Homework 2 Bonus

Convolutions and Pooling
(in the 2nd Dimension)

11-785: Introduction to Deep Learning (Fall 2020)

Out: **Oct 23rd 2020 12:00:00am EST**
Due: **TBD**

(Last updated: 10/23/20 12:00AM EST)

Start Here

- **Collaboration policy:**
  - You are expected to comply with the University Policy on Academic Integrity and Plagiarism.
  - You are allowed to talk with / work with other students on homework assignments
  - You can share ideas but not code, you must submit your own code. All submitted code will be compared against all code submitted this semester and in previous semesters using MOSS.

- **Overview:**
  - **Conv2d**
  - **MaxPool2d**
  - **AvgPool2d**

- **Directions:**
  - You are required to do this assignment using Python3. Do not use any auto-differentiation toolboxes (PyTorch, TensorFlow, Keras, etc) - you are only permitted and recommended to vectorize your computation using the NumPy library.
  - If you haven't already been doing so, use `pdb` to debug your code and please PLEASE Google your error messages before posting on Piazza.
  - Note that Autolab uses numpy v1.18.1

Introduction

You know the drill.

Homework Structure

Below is a list of files that are **directly relevant** to hw2 bonus.

IMPORTANT: First, copy the highlighted files/folders from the HW2 Bonus handout over to the corresponding folders that you used in hw1.

NOTE: We recommend you make a backup of your hw1 files before copying everything over, just in case you break code or want to revert back to an earlier version.

```
handout
├── autograder
│   └── hw2_bonus_autograder ..... Copy over this entire folder
│       ├── runner.py
│       ├── test_conv2d.py
│       └── test_pooling.py
├── mytorch
│   └── nn
│       ├── conv.py ..... [Question 1] You'll write the Conv2d module here.
│       ├── functional.py
│       └── pooling.py ..... [Question 2  3] Needed for pooling modules.
├── sandbox.py
├── create_tarball.sh
├── grade.sh
└── hw2_bonus ..... Copy over this entire folder
    └── stubs.py ..... Contains new code for functional.py that you'll need to copy over
```

Next, copy and paste the following code stubs from hw2_bonus/stubs.py into the correct files.

1. Copy `MaxPool2d(Function)` into `nn/functional.py`.
2. Copy `AvgPool2d(Function)` into `nn/functional.py`.

0.1 Running/Submitting Code

This section covers how to test code locally and how to create the final submission.

0.1.1 Running Local Autograder

Run the command below to calculate scores and test your code locally.

```
./grade.sh 2b
```

If this doesn't work, converting line-endings may help:

```
sudo apt install dos2unix
dos2unix grade.sh
./grade.sh 2b
```

If all else fails, you can run the autograder manually with this:

```
python3 ./autograder/hw2_autograder/runner.py
```

0.1.2 Running the Sandbox

We've provided `sandbox.py`: a script to test and easily debug basic operations and autograd.

Note: We will not provide new sandbox methods for this homework. You are required to write your own from now onwards.

```
python3 sandbox.py
```

0.1.3 Submitting to Autolab

Note: You can submit to Autolab even if you're not finished yet. You should do this early and often, as it guarantees you a minimum grade and helps avoid last-minute problems with Autolab.

Run this script to gather the needed files into a `handin.tar` file:

```
./create_tarball.sh
```

If this crashes (with some message about a hw4 folder) use `dos2unix` on this file too.

You can now upload `handin.tar` to Autolab .

---

1 `Conv2d` [Total: 5 points]

You'll need to implement a 2d convolutional layer, this time from scratch.

Before we get to the autograd code, let's first create the user-facing `Conv2d(Module)` class in `conv.py`.

During initialization, it should accept four args: `in_channel, out_channel, kernel_size, stride=1`.

Use the following code for weight/bias initialization:

```
# Kaiming init (fan-in) (good init strategy)
bound = np.sqrt(1 / (in_channel * kernel_size * kernel_size))
weight = np.random.uniform(-bound, bound, size=(out_channel, in_channel,
                                                kernel_size, kernel_size))
self.weight = Tensor(weight, requires_grad=True, is_parameter=True)
bias = np.random.uniform(-bound, bound, size=(out_channel,))
self.bias = Tensor(bias, requires_grad=True, is_parameter=True)
```

Fill out the rest of this class as you see fit.

Second, create a `Conv2d(Function)` class in `functional.py`.

Complete the forward and backward passes as you see fit. Pseudocode in slides.

2 `MaxPool2d` [Total: 5 points]

In `functional.py`, complete the `MaxPool2d(Function)` stub.

Pseudocode in slides. Note that you should have copied over `nn/pooling.py` to get the autograder to work.

3 `AvgPool2d` [Total: 5 points]

In `functional.py`, complete the `AvgPool2d(Function)` stub.

Pseudocode in slides.

If something breaks, we encourage you to solve the problem by yourself! You're the expert now.

You got this! 💪