Homework 5, Due 11:59 p.m., May 19

MPCS 53111 Machine Learning, University of Chicago

This homework set is worth 1.5 times the other homework sets.

Practice problem, do not submit

1. There are several activation functions other than the logistic function (also called the *sigmoid* function). Please research (e.g., see Stanford CS231 notes) the following functions: *tanh, ReLU,* and *leaky ReLU* and answer the following:

   (a) What is the advantage of tanh over the logistic function?

   (b) What is the advantage of the leaky ReLU over ReLU?

Graded problems, submit

1. The output image volume (for a single example) of a convolutional layer is $600 \times 500 \times 50$ (height $\times$ width $\times$ depth). Each filter is $35 \times 15 \times 20$ and is applied with a stride of 13. (i) How many filters were used? (ii) What was the input image volume, if a pad of 2 units was used on the top and bottom, and 3 units on the left and right sides of each image.

2. Let the image (of a single example) at some layer of a CNN be a $3 \times 3 \times 3$ matrix of numbers, corresponding to the dimensions height, width, and depth, in order. Let the numbers in the image be denoted by

   $$a_{h,w,d}, \text{where } h \in \{1,2,3\},\ w \in \{1,2,3\}, \text{ and } d \in \{1,2,3\}.$$

   Let a filter of size $2 \times 2 \times 3$ be applied on the image to produce an output image of size 2×2. Let the weights in the filter be denoted by

   $$f_{h,w,d}, \text{where } h \in \{1,2\},\ w \in \{1,2\}, \text{ and } d \in \{1,2,3\}.$$

   Let the gradient of the loss with respect to the output image be denoted by

   $$D_{h,w}, \text{where } h \in \{1,2\} \text{ and } w \in \{1,2\}.$$

   Derive explicitly the gradient of the loss with respect to $f_{1,1,2}$ and $a_{2,2,2}$.

3. Implement a Python class `CNNLayer` for a convolutional layer in the convolutional neural networks with the following methods:

   - `__init__()`: Initializes an convolutional layer object given number of filters, filter height, filter width, filter depth, and type of activation to use. You should implement layers with (i) no activation, and (ii) ReLU. (This is already provided for you.)
   - `forward_step()`: Computes a forward convolution step using the convolution layer using input values. You may assume that the stride is always 1. (This is already provided for you.)
   - `backward_step()`: Computes the $\partial Loss/\partial a_i$ values (using Russell-Norvig notation) given $\partial Loss/\partial a_j$ values.
   - `filter_gradient()`: Computes the $\partial Loss/\partial w_{i,j}$ values given $\partial Loss/\partial a_j$ values.

   The exact input parameters are specified in the accompanying `cnn-starter.ipynb`. It contains all the code for the first two functions above, and other helpful comments. It also contains four tests to test your code. You may alter the provided code as long as you don't change the function names and parameters. We'll run tests similar to the ones provided, with different parameter values. Our tests will have stride set to 1, so you may assume so in your code.

   We expect you to write code by understanding and applying the essential idea of backpropagation to the specific constraints of a CNN. You may, however, refer to Sujit Rai's blog post to confirm your understanding.

4. **FashionMNIST, CIFAR-100 using ML libraries**. In this problem, you will work on image classification with two datasets, FashionMNIST[1] and CIFAR 100[2]. FashionMNIST has the same image format as the MNIST dataset, but contains fashion related images. Each image is of shape 28×28 in gray scale. CIFAR 100 is a dataset of tiny images in 100 classes. The classes vary; e.g., they include "orchid", "castle", and "camel". The image format is $3 \times 32 \times 32$—it has 3 RGB color channels and images of size 32×32.

[1] https://github.com/zalandoresearch/fashion-mnist
[2] https://www.cs.toronto.edu/~kriz/cifar.html

The goal is to classify the images to the correct categories. You are free to choose or design whatever CNN architectures you like. You may use either the PyTorch or TensorFlow libraries. (We recommend using TensorFlow only if you are already familiar with the library.)

To start with, you may try simple architectures such as the following sequence of layers: a 2D conv layer with kernel size 5 (i.e., a filter with 5×5 height and width), a 2D max pool layer, a ReLU layer, a 2D conv layer with kernel size 5, a 2D max pool layer, a linear layer, a ReLU layer, a linear layer, and finally a softmax layer. (See the CS231n reading for description of layers such as max pool.) Train this model with cross entropy loss. You should be able to get over 85% of accuracy on the Fashion MNIST dataset.

The CIFAR 100 dataset may require more exploration of model architectures and parameter tuning, it may also take more epochs of training.

Common CNN architectures can be found on the CS231n website[3]; its lecture nodes have some more complex CNN architectures as well[4].

Researchers use various tricks to improve model accuracy on vision tasks, such as *data normalization, batch normalization,* etc. Please search for such tricks online and apply the ones you think might help your accuracy.

Submit your code, plots and discussions in `hw5.py` and `hw5.pdf`. But also submit your predictions on the test set to our Kaggle competition pages at

- https://www.kaggle.com/c/2020-mpcs53111-hw5-cifar100/,
- https://www.kaggle.com/c/2020-mpcs53111-hw5-fashionmnist/.

The datasets can be downloaded from the Kaggle competition page.

You may submit up to two times every day. Your grade will not be affected by your ranking at the Kaggle site, but the students with the highest scores in each of the two competitions will receive special prizes.

[3] http://cs231n.github.io/convolutional-networks/
[4] http://cs231n.stanford.edu/slides/2017/cs231n_2017_lecture9.pdf

In your discussion you should describe your model architecture, and your test results from Kaggle. Also describe how you found your optimal architecture, and which parts of the model and hyperparameters you found more relevant to test accuracy.

Acknowledgement: This problem has been designed and written by Zewei Chu.

Optional problem, do not submit

OPT-1 Use your above `CNNLayer` to implement a Python class CNN for a convolution neural network, with methods for training and predicting. For the loss function use multi-class hinge loss. Use it for classifying the MNIST dataset of handwritten digits. The network should contain two convolutional layers, each with 32 filters of size 5×5, followed by a fully-connected layer of 10 nodes for ten-class classification. Report your results in a discussion section using appropriate plots.