Deep Learning Homework 1
Backpropagation

due on March 10, 2021

Instruction: submit your report in English as a single pdf file and your code as a single file named *module.py* to web learning.

1 Backpropagation

1.1 Derivation

You have already learned how to do back propagation for a linear layer in the lecture. Now you can try to derive the back propagation for some other layers by yourself: convolution, max-pooling, and tanh.

Let's denote the loss function as L, the input to each layer as z and the output of each layer as y. We additionally assume that the gradient of the loss function with respect to the output of the layer, i.e., $\frac{dL}{dy}$, is given.

Your goal is to derive:

1. The gradient of the loss function with respect to the input of the layer, i.e., $\frac{dL}{dz}$;

2. (For convolutional layer only.) The gradient of the loss function with respect to all trainable parameters, i.e., $\frac{dL}{d\text{weight}}$ and $\frac{dL}{d\text{bias}}$.

The forward pass of each layer is defined below.

Convolutional layer. Suppose the input tensor z is of size (C_{in}, H_{in}, W_{in}) and the output y is of size$(C_{out}, H_{out}, W_{out})$, where C denotes the number of channels and H, W denote the height and width of the images. A convolutional layer computes

$$y(j) = bias(j) + \sum_{k=0}^{C_{in}-1} weight(j,k) \star z(k), \text{for } 0 \leq j \leq C_{out} - 1,$$

where "*weight*" is the convolutional kernel with size $(C_{out}, C_{in}, k_H, k_W)$, "*bias*" is of size $(C_{out},)$, and "$\star$" denotes the 2D convolution operator.

[Hint: you can start by working on the inputs with only one channel. Or consider the simplest case where the input is a $1 \times 3 \times 3$ tensor and the kernel size $(k_H, k_W) = (2, 2)$.]

Max-pooling layer. Assume the input size is (C, H_{in}, W_{in}) and the output size is (C, H_{out}, W_{out}). Suppose the kernel size is (k_H, k_W). The forward pass is defined as

$$y(j,h,w) = \max_{0 \le m \le k_H - 1,\ } \max_{0 \le n \le k_W - 1} z(j, stride[0] \times h + m, stride[1] \times w + n),$$
$$\text{for } 0 \le j \le C-1, 0 \le h \le H_{out}-1, 0 \le w \le W_{out}-1.$$

Tanh. This is an element-wise operator defined as

$$y = \frac{\exp(z) - \exp(-z)}{\exp(z) + \exp(-z)}.$$

1.2 Programming

See *module.py* in the attachment. We have provided the *_forward* functions of these modules for you and you need to implement their *_backward* functions following your derivation. You need to return the gradient of inputs, and save the gradient of weight into self.grads[“weight”], the gradient of bias into self.grads[“bias”].

You are **NOT** allowed to use any autograd framework, e.g. Tensorflow, PyTorch, etc. **DO NOT** change the input and output formats! We will use a specific data format following our *_forward* implementation to check your code. You **ONLY** need to submit the *module.py*.

[Hint: you can test your implementation by running “*python module.py*”. Note: passing the local test does not necessarily mean that your implementation is correct.]

2 Get Your Hand Dirty

In this problem, you need to train a neural network with different hyper-parameters to solve the spiral classification problem: `https://playground.tensorflow.org/#dataset=spiral`, and answer the following questions. Please include necessary visualizations for all your attempts to solve this problem.

1. List your best set of hyper-parameters and show us your best result, how do you find this configuration?

2. List your findings that how the learning rate, the number of hidden sizes, and the regularization influence the performance and convergence rate.