ConvNet for MNIST Classification

Homework 2 for Introduction to Deep Learning, Fall 2019

Deadline: 2019.11.03 23:59:59

1 Introduction

MNIST digits dataset is a widely used dataset for image classification in machine learning field. It contains 60,000 training examples and 10,000 testing examples. The digits have been size-normalized and centered in a fixed-size image. Each example is a 784×1 matrix, which is transformed from an original 28×28 grayscale image. Digits in MNIST range from 0 to 9. Some examples are shown below. **Note**: During training, information about testing examples should never be used in any form.

In this homework, you need to use **Convolutional Neural Network (ConvNet)** to perform digits classification. Since ConvNet operates on two-dimensional input, we need to convert the raw data of 784×1 vector into 28×28 matrix. The data preprocessing and conversion have been done in the jupyter notebook.

2 ConvNet for MNIST Classification

2.1 Files Description

There are several files included in the source package. Some files are identical to homework-1, you can implement it by yourself, or waiting for our solution.

The new file's description is listed below.

- **homework_2.ipynb** is similar to homework-1, and detailed homework requirements is listed in this file. **Please read this file carefully**.
- **./layers/conv_layer.py**, the convolutional layer perform convolution with input data. It consists of two trainable weightWand bias b. W is stored in 4 dimensional matrix with

dimensions $n_{out} \times n_{in} \times k \times k$, where k specifies the height and width of each filter (also called kernel size), n_{in} denotes the channels number of input which each filter will convolve with, and n_out denotes the number of the filters. For simplicity, we only need to implement the standard convolution with stride equal to 1. There is another important parameter **pad** which specifies the number of zeros to add to each side of the input. Therefore the expected height dimension of output should be equal to $H + 2 \times pad - kernel\_size + 1$ and width likewise.

- **./layers/pooling_layer.py**, **MaxPoolingLayer** only need to be implemented in non-overlapping style (stride=2). Therefore the expected height dimension of output should be equal to $(H + 2 \times pad)/kernel\_size$ and width likewise.
- **./layers/reshape_layer.py**, will be used to reshape feature maps and δ.

Note: You can implement the ConvLayer by either the convolve function introduced in slides, or element-wise multiplication. We recommend the latter.

Note: The training process of ConvNet will take several minutes to several hours. To accelerate convolution and pooling, you should avoid complex nested for loops and use matrix multiplication with NumPy built-in functions as much as possible. If your computer is too slow to complete the training process, you can modify the network by yourself, such as reducing the convolution kernels.

2.2 Requirements

You are required to complete the '# TODO' parts in above files. If implemented correctly, just by running the IPython notebook step by step, you can obtain lines of output and reach a relatively high test accuracy. **You need to submit all codes and a report** with the following requirements:

- Record the training and testing accuracy, plot the training loss curve and training accuracy curve in the report.
- Compare the difference of results when using **ConvNet** and **MLP**(done in homework-1) (you can discuss the difference from the aspects of training time, convergence and accuracy).
- The given hyerparameters maybe performed not very well. You can modify the hyerparameters by your own, and observe how does these hyerparameters affect the classification performance. Write down your observation and record these new results in the report.
- **(Optional)** Try to visualize the filters and the feature maps of the first layer in ConvNet. Refer to visualization tutorial[1] for more details.

3 Attention

- You need to submit all codes and a report (at least two pages **in PDF format**). Delete the MNIST dataset before submit.
- Pay attention to the efficiency of your implementation. Try to replace the use of **for-loops** with matrix multiplication.
- Do not paste a lot of codes in your report (only some essential lines could be included). Any extra modifications of above homework files or adding extra Python files should be explained and documented.
- Any open source neural network toolkits, such as TensorFlow, Caffe, PyTorch, are **NOT** permitted in finishing homework-2.
- **Plagiarism is not permitted**.

[1] http://nbviewer.jupyter.org/github/BVLC/caffe/blob/master/examples/00-classification.ipynb