ConvNet for MNIST Classification

1 Introduction

MNIST[1] 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. It is a subset of a larger set available from NIST. The digits have been size-normalized and centered in a fixed-size image. Each sample is a 784×1 matrix, which is transformed from an original 28×28 pixels grayscale image. Digits in MNIST range from 0 to 9. Some typical digits images are shown below.

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.

Note: However, after the conversion, you may notice that a typical batch of image data passing through the network would be a 4-dimensional matrix with dimensions $N \times C \times H \times W$, where N denotes the batch size, H and W denote the height and width of an image, and C denotes the number of image channels (1 for gray scale and 3 for RGB scale).

2 Files Description

There are several files included in the source package. Some files are identical to homework-2, you can implement it by yourself, or watting for my solution (will be released on about Friday).

The new file's description is listed below.

1) **./layers/conv_layer.py**, the convolutional layer perform convolution with input data. It consists of two trainable weight $\mathbf{W}$ and bias $\mathbf{b}$. $\mathbf{W}$ 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 $H + 2 \times \mathbf{pad} - \mathbf{kernel\_size} + 1$ and width likewise.
2) **./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 $(H + 2 \times \mathbf{pad})/\mathbf{kernel\_size}$ and width likewise.
3) **./layers/reshape_layer.py**, will be used to reshape feature maps and δ.

homework_3.ipynb is similar to homework-1, and detailed homework requirements is listed in this file. **Please read this file carefully**.

If you implement the above layers correctly, just by running homework_3 IPython notebook step by step, you can obtain lines of logging and reach a relatively high test accuracy. Please check your implementation if you encounter any error.

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.

3 Report

In the experiment report, you are required to complete the following requirements:

1) Record the training and test accuracy, plot the training loss curve and training accuracy curve in the report.
2) Compare the difference of results when using **ConvNet** and **MLP**(done in homework-2) (you can discuss the difference from the aspects of training time, convergence and accuracy).

1. Here is the mnist page on Prof. LeCun's website.

3) 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.
4) (Optional) Try to visualize the filters and the feature maps of the first layer in ConvNet. Refer to visualization tutorial[2] for more details.

Note: Please convert your report to pdf format.

Note: Only some essential lines of codes are permitted to be included for explaining complicated thoughts.

4 ATTENTION

1) You need to submit the experiment report and all files mentioned in section 2. Please package and submit it at WebLearning.
2) **Deadline: 2018.11.18 23:59:59**.
3) Any open source neural network toolkits, such as TensorFlow, Caffe, PyTorch, are **NOT** permitted in finishing this homework.
4) **Plagiarism is not permitted**.

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