MLP 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 **Multilayer Perceptron(MLP)** to perform digits classification.

Note: During your training process, information about testing samples in any form should never be introduced.

2 Files Description

There are several files included in the source package. Each file's description is listed below.

1) **network.py** describes network class, which can be utilized when defining network architecture and performing training.
2) ***optimizer.py** describes SGD class, which can be used to perform forward propagation and backward propagation.
3) **solver.py** describes train function, which designates the training routine, and test function, which designates the testing routine.
4) **plot.py** describes plot_loss_and_acc function which can be used to plot curves of loss and accuracy.

Our neural network implementation is guided by *modularity* idea. Each layer class has three methods: *__init__*, *forward* and *backward*. For some trainable layers with weights and biases, constructor methods function as parameter initialization, forward methods represents the data processing performed by the layer and backward methods perform backpropagation operations. Each layer's definition is listed below.

1) ***FCLayer** treats each input sample as a simple column vector (need to reshape input if necessary) and produces an output vector by doing matrix multiplication with weights and then adding biases $\mathbf{u} = \mathbf{W}\mathbf{x} + \mathbf{b}$.
2) ***SigmoidLayer** is a sigmoid activation unit, computing the output as $f(\mathbf{u}) = \frac{1}{1+\exp(-\mathbf{u})}$.
3) ***ReLULayer** is a linear rectified unit, computing the output as $f(\mathbf{u}) = max(\mathbf{0}, \mathbf{u})$.
4) ***EuclideanLossLayer** computes the sum of squares of differences between inputs and labels $\frac{1}{2}\sum_n \|logits(n) - gt(n)\|_2^2$.
5) ***SoftmaxCrossEntropyLossLayer** can be viewed as a mapping from input to a probability distribution in the following form:

$$P(t_k = 1|\mathbf{x}) = \frac{\exp(x_k)}{\sum_{j=1}^{K} \exp(x_j)} \tag{1}$$

where x_k is the k-th component in the input vector $\mathbf{x}$ and $P(t_k = 1|\mathbf{x})$ indicates the probability of being classified to class k given the input. Since the output of softmax layer can be interpreted as a probability distribution, we can compute the delta likelihood and its logarithm form is also called cross entropy error function,

$$E = -\ln p(t^{(1)}, ..., t^{(N)}) = \sum_{n=1}^{N} E^{(n)} \tag{2}$$

where

$$E^{(n)} = -\sum_{k=1}^{K} t_k^{(n)} \ln h_k^{(n)} \tag{3}$$

$$h_k^{(n)} = P(t_k^{(n)} = 1|\mathbf{x}^{(n)}) = \frac{\exp(x_k^{(n)})}{\sum_{j=1}^{K} \exp x_j^{(n)}} \tag{4}$$

.

***homework_2.ipynb** is similar to homework-1, and detailed homework requirements is listed in this file. **Please read this file carefully**.

Files and layers containing * before their name is **what you need to complete**. Please write down your code at '# TODO' in these files.

If you implement the above layers correctly, just by running homework_2 IPython notebook step by step, you

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

can obtain lines of logging and reach a relatively high test accuracy. Please check your implementation if you encounter any error.

Note: The definition of softmax loss layer is a little different from homework-1, since we don't include trainable parameters θ in the layer. However this parameter can be explicitly extracted out and can be equivalent to **FCLayer**.

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 **Sigmoid** and **ReLU** as activation function (you can discuss the difference from the aspects of training time, convergence and accuracy).
3) Compare the difference of results when using **EuclideanLoss** and **SoftmaxCrossEntropyLoss** as loss function.
4) Construct a MLP with **two hidden layers** (choose the number of hidden units by your own), using any activation function and loss function. Also, compare the difference of results between one layer structure and two layers structure.
5) 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.

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: 2018.11.04 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**.