MNIST Digits Classification with MLP

Background

MNIST digits dataset is a widely used database for image classification in machine learning field. It contains 60,000 training samples and 10,000 testing samples. Each sample is a 784×1 column vector, which is transformed from an original 28×28 pixels grayscale image. Some typical digits image (transformed from column vector) are shown below.

In this homework, you need to use multilayer perceptron (MLP) to perform digits classification.

Dataset Description

To get data, first run `get_mnist.m` script, then all the data are stored in `data/mnist.mat`. It includes four variables `train_data, train_label, test_data` and `test_label`. Digits range from 0 to 9, but corresponding labels are from 1 to 10 for MATLAB index style consideration.

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

MATLAB Files Description

In neural network, almost every data processing can be viewed as a functional layer. And a neural network can be constructed by stacking multiple layers to define a certain data processing pipeline. So our neural network implementation is guided by *modularity* idea. Each layer class has four main methods: constructor, forward, backprop and update. For some trainable layers with weights and biases, constructor functions as parameter initialization and update function will update parameters by stochastic gradient descent. Forward represents the data processing performed by the layer and backprop performs backpropagation operations. Each layer's definition is listed below.

- `Layer`: base class for all layers
- `Linear`: treat each input sample as a simple column vector (need to reshape input if necessary) and produce an output vector by doing matrix multiplication with weights and then adding biases $u = Wx + b$
- `Relu`: linear rectifier activation unit, compute the output as $f(u) = \max(0, u)$
- `Sigmoid` sigmoid activation unit, compute the output as $f(u) = \frac{1}{1+\exp(-u)}$
- `EuclideanLoss` compute the sum of squares of differences between inputs and labels $\frac{1}{2} \sum_n \|T(n) - y(n)\|_2^2$

When running backpropagation algorithm, δ is an essential component to compute gradient in each layer. We define δ to be the derivative of loss function with respect to layer's input.

Attention: Since layer definition here is a little different from lecture slides in that we explicitly extract activation unit out, you should calculate $delta$ in activation unit separately. Hope you recognize this subtlety :-).

There are also other three files included in the source package.

- `Network.m` describe network class, which can be utilized when defining network architecture and performing training
- `solve_mlp.m` function which designates the training and testing routine. It can be considered as an optimization scheme
- `run_mlp.m` the main script for running whole program. It demonstrates how to simply define a neural network by sequentially adding layers.

If you implement layers correctly, just by running `run_mlp.m`, you can obtain lines of logging information and reach a relatively good test accuracy. All three files are encouraged to be modified to adapt to personal needs.

Attention: any modifications of these three files or adding extra MATLAB files should be explained and documented in README.

Report

In the experiment report, you need to answer the following basic questions:

1. plot the loss function value against to every iteration during training
2. construct a neural network with one hidden layer, and 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. conducting same experiments above, but with two hidden layers. Also, compare the difference of results between one layer structure and two layers structure

Attention: The current hyperparameter settings may not be optimal for good classification performance. Try to adjust them to get 99+% accuracy.

Attention: MATLAB neural network toolbox, theano/caffe framework and any other open source codes are **NOT** permitted in this homework. Once discovered, it shall be regarded as plagiarism.

Submission Guideline:

- **report**: well formatted readable summary including your results, discussions and ideas. Source codes should *not* be included in report writing. Only some essential lines of codes are permitted for explaining complicated thoughts.

- **codes**: organized source code files with README for any extra installation or other procedures. Ensure one can successfully *reproduce* your results following your instructions. **DO NOT** include model weights/raw data/compiled objects/unrelated stuff over 100MB...

Deadline: Oct. 9th

TA contact info: Yulong Wang (王宇龙）, wang-yl15@mails.tsinghua.edu.cn