MNIST Digits Classification with MLP and CNN

Background

In this homework, we still focus on MNIST digits classification problem. After implementing the details about MLP and CNN, you might know more about them. In fact, similar neural networks with standard structures are often encapsulated as modules in deep learning frameworks. It's convenient and also important for us to master the skills to use these modules and construct our models with deep learning frameworks.

You will be permitted to use frameworks such as TensorFlow. We hope that you can understand the characteristics of them (e.g. *data flow graphs* in TensorFlow) and implement MLP and CNN to finish the task of MNIST digits classification.

In addition, you should implement another technique: *batch normalization*, which aims to deal with the *internal covariate shift* problem. Specifically, during the training process, the distribution of each layer's inputs will change when the parameters of the previous layers change. Researchers propose the batch normalization of the input to activation function of each neuron, so that the input of each mini-batch has a mean of 0 and a variance of 1. To normalize a value x_i across a mini-batch,

$$BN_{initial}(x_i) = \frac{x_i - \mu_B}{\sqrt{\sigma_B^2 + \epsilon}}$$

where μ_B and σ_B donate the mean and standard deviation of the mini-batch. ϵ is a small constant to avoid dividing by zero. The transform above might limit the representation ability of the layer, thus we extend it to the following form:

$$BN(x_i) = \gamma \cdot \frac{x_i - \mu_B}{\sqrt{\sigma_B^2 + \epsilon}} + \beta$$

where γ and β are learnable parameters. For instance, the output of the hidden layer in MLP is

$$y = \sigma(Wx + b)$$

After we normalize the input to activation function σ, the output can be represented as

$$y = \sigma(BN(Wx + b) \qquad (1)$$

Hint:

1. Batch normalization of CNN should obey the convolutional property – so that different units in the same feature map should be normalized in the same way. Refer to Sec 3.2 in [1].
2. When we test our samples one by one, the normalization may not work because we don't have *mini-batch* this time. We should maintain the population mean and variance during training process and use them to estimate the "μ_B" and "σ_B" when testing. Simply, you can try to calculate the average μ_B and σ_B when training and take them as the population mean and variance. Refer to Sec 3.1 in [1].

Requirements

Python version 2.7, TensorFlow >= 1.0.0

Dataset Description

Utilize load_data.py to read the training set and test set. During your training process, information about testing samples in any form should never be introduced. Note that the shapes of data are different in MLP and CNN.

MLP:

To load data, use `load_mnist_2d()` in load_data.py.

CNN:

To load data, use `load_mnist_4d()`.

Python Files Description

In this homework, we provide unfinished implementation of MLP and CNN in **Tensorflow** framework.
Both programs share the same code structure:
`main.py`: the main script for running the whole program.
`model.py`: main script for model implementation, including some utility functions.
`load_data.py`: functions for data loading.

MLP:
You are supposed to:
1) Implement "input -- Linear – BN – ReLU – Linear – loss" network in **`__init__()`** in `model.py`.
2) Implement **`batch_normalization_layer()`** function in `model.py`, and use it in 1).

CNN:
You are supposed to:
1) Implement "input – Conv – BN – ReLU – MaxPool – Conv – BN – ReLU – MaxPool – Linear – loss" network in **`__init__()`** in `model.py`.
2) Implement the **`batch_normalization_layer()`** function in `model.py` and use it in constructing conv layers and fully-connected layers in 1).

Attention: The implementation detail of BN is slightly different in MLP and CNN.

Report

In the experiment report, you need to answer the following basic questions:
1. Plot the loss value against to every iteration during training.
2. Construct the multi-layer perceptron and convolutional neural networks with batch normalization. Compare the differences between the results of MLP and CNN.
3. Construct MLP and CNN without batch normalization, discuss the effects of batch normalization.
4. Complete the one-by-one test and analyze the result.

Attention: On your final submission, you need to submit the **BN-version** MLP and CNN codes.

Submission Guideline

You need to submit both report and codes, which are:
- **report**: well formatted and 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 extra modifications or specific usage. Ensure that others can successfully reproduce your results following your instructions. **DO NOT include model weights/raw data/compiled objects/unrelated stuff over 50MB (due to the limit of XueTang)**

Deadline: Nov. 8th

TA contact info: Pei Ke (柯沛)，kepei1106@outlook.com
Zheng Zhang (张正)，zhangz.goal@qq.com

[1] Ioffe S, Szegedy C. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate

Shift, In Proceedings of the International Conference on Machine Learning, 2015: 448-456.