Homework 1

Due: 2019-03-21 (Thur.) 10pm

1 Introduction

In this assignment you will practice putting together a simple image classification pipeline, based on the softmax classifier. The goals of this assignment are as follows:

- understand the basic Image Classification pipeline and the data-driven approach;
- implement and apply a softmax classifier;
- implement and apply a three-layer neural network classifier.

You will be given a set of *.py files which include tools to check your code.

You just need to upload all your code and report and do not upload datasets.

2 Submission

- You need to submit the following files:
  1) neural_net.py
  2) softmax.py
  3) experiment report.pdf
  4) three_layer_net.py
- Please convert your experiment report to PDF format

3 Softmax classifier

In this problem you will be given snippets of code. The snippets will be functions that you will be introduced to throughout the course and famous functions you might use in basic deep learning algorithms.

Your task is to complete and hand in this completed softmax worksheet (including its outputs and any supporting code outside of the worksheet) with your assignment submission. **You will:**

- Implement softmax.py in folder “classifier”
  - ➢ implement a fully-vectorized loss function for the softmax classifier.
  - ➢ implement the fully-vectorized expression for its analytic gradient.
  - ➢ optimize the loss function with SGD.
- Train your model in softmax_train.py
  - ➢ check your implementation with numerical gradient.
  - ➢ visualize the final learned weights.
- See the code file for details.

4 Three_layer_net.py

In this problem you will be given snippets of code. The snippets will be functions that you will be introduced to throughout the course and famous functions you might use in basic deep learning algorithms.

Your task is to complete and hand in this completed worksheet (including its outputs and any supporting code outside of the worksheet) with your assignment submission. **You will:**

- implement a neural network with three layers of fully-connected layers
  - ➢ perform the forward pass, computing the class scores for the input
  - ➢ finish the forward pass, and compute the loss
  - ➢ compute the backward pass, computing the derivatives of the weights and biases
  - ➢ create a random mini-batch of training data and labels, storing them in x_batch and y_batch respectively
  - ➢ use the gradients in the grads dictionary to update the parameters of the network
  - ➢ implement the predict function
  - ➢ tune hyperparameters using the validation set. Store your best trained model in best_net
- use the model to perform classification, and test it out on the CIFAR-10 dataset.

Notice:

1) Modify the package path as your need;
2) Modify the datasets path in data_utils.py;
3) You can download the dataset using download.py, or you can download the dataset "ciarf-10" yourself. Make sure you have torchvision installed when you using download.py.