Homework 2

Due: 2020-10-11

Introduction

In this assignment you will practice putting together a simple image classification pipeline, based on the k-Nearest Neighbor or the SVM/Softmax classifier. The goals of this assignment are as follows:

- understand the basic **Image Classification pipeline** and the data-driven approach (train/predict stages)
- understand the train/val/test **splits** and the use of validation data for **hyperparameter tuning**.
- develop proficiency in writing efficient **vectorized** code with numpy
- implement and apply a k-Nearest Neighbor (**kNN**) classifier
- implement and apply a Multiclass Support Vector Machine (**SVM**) classifier
- implement and apply a **Softmax** classifier
- implement and apply a **Three layer neural network** classifier
- understand the differences and tradeoffs between these classifiers
- get a basic understanding of performance improvements from using **higher-level representations** than raw pixels (e.g. color histograms, Histogram of Gradient (HOG) features)

Setup

- Start IPython
  After you have the CIFAR-10 data, you should start the IPython notebook from the `homework_2` directory, with the jupyter notebook command.

- Download data
  Given the code, you need to run `/sducs2020/datasets/download_datasets.py` to download the CIFAR-10 dataset

Experiments

There are `### START CODE HERE` / `### END CODE HERE` tags denoting the start and end of code sections you should fill out. Take care to not delete or modify these tags, or your assignment may not be properly graded.

- **Q1: k-Nearest Neighbor classifier (20 points)**
  The IPython Notebook knn.ipynb will walk you through implementing the kNN classifier.

- **Q2: Training a Support Vector Machine (25 points)**
  The IPython Notebook svm.ipynb will walk you through implementing the SVM classifier.

- **Q3: Implement a Softmax classifier (20 points)**
  The IPython Notebook softmax.ipynb will walk you through implementing the Softmax

classifier.

- **Q4: Three-Layer Neural Network (25 points)**
  The IPython Notebook three_layer_net.ipynb will walk you through the implementation of a two-layer neural network classifier.

- **Q5: Higher Level Representations: Image Features (10 points)**
  The IPython Notebook features.ipynb will walk you through this exercise, in which you will examine the improvements gained by using higher-level representations as opposed to using raw pixel values.

- See the code file for details.

Submission

- You need to accomplish the following files:
  1) knn.ipynb
  2) svm.ipynb
  3) softmax.ipynb
  4) three_layer_net.ipynb
  5) features.ipynb
  6) sducs2020/classifiers/k_nearest_neighbor.py
  7) sducs2020/classifiers/linear_classifier.py
  8) sducs2020/classifiers/linear_svm.py
  9) sducs2020/classifiers/softmax.py
  10) sducs2020/classifiers/neural_net.py

- You just need to upload all your code and report and do not upload datasets.
- Please convert your experiment report to PDF format.

● Three_layer_net.ipynb

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.