Assignment 2

Task 1

In this task you will practice writing some basic codes in neural networks. This part refers to Stanford cs231n assignment 2 [1]. The goal of this task is as follows:

- understand **Neural Networks** and how they are arranged in layered architectures
- understand and be able to implement (vectorized) backpropagation
- implement various update rules used to optimize Neural Networks
- implement batch normalization for training deep networks
- implement dropout to regularize networks
- effectively cross-validate and find the best hyperparameters for Neural Network architecture
- understand the architecture of Convolutional Neural Networks and train gain experience with training these models on data

Setup

Get the code as a zip file here.

You will need a unix shell for the command scripts. If you're on Windows, that means cygwin, a Linux VM such as VirtualBox or Docker. If you're not already a cygwin user, you will probably find it easier to set up Virtual Box or Docker. That will also give you a standard Python distribution. We also provide a hands-on about how to use Jupyter Notebook remotely in Azure VM (see jupyter_azure_setup.pdf first).

Dependencies

1. Installing Python 3.5+

2. Virtual environment: We recommend using virtual environment for the project. If you choose not to use a virtual environment, it is up to you to make sure that all dependencies for the code are installed globally on your machine. To set up a virtual environment, run the following:

```
cd assignment2
sudo pip install virtualenv      # This may already be installed
python3 -m venv .env       # Create a virtual environment (python3)
source .env/bin/activate       # Activate the virtual environment
pip install -r requirements.txt  # Install dependencies

# Work on the assignment for a while ...
# ... and when you're done:
deactivate                    # Exit the virtual environment
```

Note that every time you want to work on the assignment, you should run source .env/bin/activate (from within your assignment2 folder) to re-activate the virtual environment, and deactivate again whenever you are done.

Download data

Run the following from the assignment2 directory to download CIFAR-10 dataset.

```
cd cs231n/datasets
./get_datasets.sh
```

Compile the Cython extension

Convolutional Neural Networks require a very efficient implementation. We have implemented the functionality using Cython; you will need to compile the Cython extension before you can run the code. From the cs231n directory, run the following command:

```
python setup.py build_ext --inplace
```

Start IPython

After you have the CIFAR-10 data, you should start the IPython notebook server from the assignment2 directory, with the jupyter notebook command.

If you are unfamiliar with IPython, you can also refer to our IPython tutorial.

Some notes

NOTE 1: The assignment2 code has been tested to be compatible with python versions 3.5 and 3.6 (it may work with other versions of 3.x, but we won't be officially supporting them). For this assignment, we are NOT officially supporting python2. Use it at your own risk. You will need to make sure that during your virtualenv setup that the correct version of python is used. You can confirm your python version by (1) activating your virtualenv and (2) running python --version.

NOTE 2: If you are working in a virtual environment on OSX, you may potentially encounter errors with matplotlib due to the issues described here. In our testing, it seems that this issue is no longer present with the most recent version of matplotlib, but if you do end up running into this issue, you may have to use the start_ipython_osx.sh script from the assignment2 directory (instead of jupyter notebook above) to launch your IPython notebook server. Note that you may have to modify some variables within the script to match your version of python/installation directory. The script assumes that your virtual environment is named .env.

NOTE 3: In this task, you are not allowed to use any deep learning framework, such as Tensorflow, Pytorch, Keras and so on. Writing basic codes in this task will help you deepen the understanding of neural networks.

NOTE 4: You are NOT required to finish Q5 in Stanford cs231n. The subtasks you need to finish are listed below. Please note that Q1-Q4 are the same as cs231n but Q5 is different.

Q1: Fully-connected Neural Network (25 points)

The IPython notebook FullyConnectedNets.ipynb will introduce you to our modular layer design, and then use those layers to implement fully-connected networks of arbitrary depth. To optimize these models you will implement several popular update rules.

Q2: Batch Normalization (25 points)

In the IPython notebook BatchNormalization.ipynb you will implement batch normalization, and use it to train deep fully-connected networks.

Q3: Dropout (10 points)

The IPython notebook Dropout.ipynb will help you implement Dropout and explore its effects on model generalization.

Q4: Convolutional Networks (30 points)

In the IPython Notebook ConvolutionalNetworks.ipynb you will implement several new layers that are commonly used in convolutional networks.

Q5: Train the best model. (10 points)

Train the best model on full CIFAR-10 dataset using what you have learned in Q1-Q4. Organizing your codes in main.py in assignment2 directory. Full dataset (including training, validation and test data) can be obtained by function get_CIFAR10_data in data_utils.py at cs231n package. You should evaluate accuracy on validation set and test set.

Submitting your work

Once you are done working, run the collectSubmission.sh script; this will produce a file called assignment2.zip.

Please prepare a brief document for your best model in Q5. In this document, you should:

- Specify how to run your scripts(main.py).
- Introduce your model components.
- Make some comparisons regarding how to improve neural network model.

Please submit your codes and documents on Google Drive. You should first create a folder named <your_student_id>, such as 2017xxxxxx, and put your work in this directory. Also, you need fill in Google Sheet with your results in Q5.

You DO NOT need to submit this work at learn.tsinghua.edu.cn.

References

[1] https://cs231n.github.io/assignments2017/assignment2/
[2] https://bcourses.berkeley.edu/courses/1453965/pages/assignment-2-description

Useful resources

[1] Tutorials/slides in https://cs231n.github.io/
[2] Working on this task in a docker: http://deeplearningexperience.logdown.com/posts/252597-cs231n-assignment-1-in-a-docker-container

Task2

The most important feature/standard of a high-qualified research paper is about its “repeatability”. In this task, you should re-implement several network embedding algorithms. Network representation learning (NRL) is learning vector representation for nodes in networks. It is also called network embedding and graph embedding.

You should implement at least four network embedding algorithms. You are required to implement DeepWalk[1] and LINE[2]. They are two representative NRL algorithms and advance the research of

NRL successfully. You are free to choose the other two algorithms. You can refer to some useful resources such as [3], but your choices are not limited to them.

Dataset: Cora (citation dataset) and Tencent Weibo (following network)
- The two datasets and data description can be found here.
- Cora is used for semi-supervised learning on graphs. Specifically, you will do multi-class classification for research papers. We also provide a script (data_utils_cora.py) to specify how to split dataset to train, validate and test. You should use the same data partitioning method to evaluate your codes. For DeepWalk and LINE, you should use logistic regression as classifier.
- Dataset for Tencent Weibo is used for unsupervised learning on graphs. Specifically, you will do link prediction task for nodes in the network. That is to say, you should infer whether there exists an edge between two given nodes. We have already divided this dataset into several parts for you to train, validate and test. You should use the same divided datasets to evaluate your codes.

Evaluation:
- Cora: you should evaluate classification accuracy on test data.
- Tencent Weibo: you should obtain AUC score on test edges (including positive and negative edges).

NOTE1: Word2vec, Tensorflow, PyTorch and other deep learning frameworks can be used, but you can't call off-the-shelf packages of these algorithms directly.

NOTE2: Each dataset should be used at least in one algorithm. That is to say, some algorithms should support semi-supervised learning on Cora and some should support unsupervised learning on Tencent Weibo.

Submission:
- You should submit all your codes here. Compress your codes in a file in zip format and the file is named <your_student_id>_hw2_task2.zip, such as 2017xxxxxx_hw2_task2.zip.
- You should specify how to run codes of each algorithm.
- Please fill in Google sheet with your results. The links are Cora, Tencent Weibo.
- You DO NOT need to submit this task on learn.tsinghua.edu.cn.

References:
[1] DeepWalk: Online Learning of Social Representations. Bryan Perozzi, Rami Al-Rfou, Steven Skiena. KDD 2014.
[2] LINE: Large-scale Information Network Embedding. Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, Qiaozhu Me. WWW 2015.
[3] Must-read papers on network representation learning: https://github.com/thunlp/NRLPapers