In this assignment you will practice writing backpropagation code, and training Neural Networks and Convolutional Neural Networks. The goals of this assignment are 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

You can work on the assignment in one of two ways: locally on your own machine, or on a virtual machine through Terminal.com.

Working in the cloud on Terminal

Terminal has created a separate subdomain to serve our class, www.stanfordterminalcloud.com. Register your account there. The Assignment 2 snapshot can then be found HERE. If you are registered in the class you can contact the TA (see Piazza for more information) to request Terminal credits for use on the assignment. Once you boot up the snapshot everything will be installed for you, and you will be ready to start on your assignment right away. We have written a small tutorial on Terminal here.

Working locally

Get the code as a zip file here. As for the dependencies:

[Option 1] Use Anaconda: The preferred approach for installing all the assignment dependencies is to use Anaconda, which is a Python distribution that includes many of the most popular Python packages for science, math, engineering and data analysis. Once you install it you can skip all mentions of requirements and you are ready to go directly to working on the assignment.

[Option 2] Manual install, virtual environment: If you do not want to use Anaconda and want to go with a more manual and risky installation route you will likely want to create a 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
virtualenv .env                  # Create a virtual environment
source .env/bin/activate         # Activate the virtual environment
pip install -r requirements.txt  # Install dependencies
# Work on the assignment for a while ...
deactivate                       # Exit the virtual environment
```

Download data: Once you have the starter code, you will need to download the CIFAR-10 dataset. Run the following from the `assignment2` directory:

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

Compile the Cython extension: Convolutional Neural Networks require a very efficient implementation. We have implemented of 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. If you are unfamiliar with IPython, you should read our IPython tutorial.

NOTE: If you are working in a virtual environment on OSX, you may encounter errors with matplotlib due to the issues described here. You can work around this issue by starting the IPython server using the `start_ipython_osx.sh` script from the `assignment2` directory; the script assumes that your virtual environment is named `.env`.

Submitting your work:

Whether you work on the assignment locally or using Terminal, once you are done working run the

`collectSubmission.sh` script; this will produce a file called `assignment2.zip`. Upload this file under the Assignments tab on the coursework page for the course.

Q1: Fully-connected Neural Network (30 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 (30 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: ConvNet on CIFAR-10 (30 points)

In the IPython Notebook `ConvolutionalNetworks.ipynb` you will implement several new layers that are commonly used in convolutional networks. You will train a (shallow) convolutional network on CIFAR-10, and it will then be up to you to train the best network that you can.

Q5: Do something extra! (up to +10 points)

In the process of training your network, you should feel free to implement anything that you want to get better performance. You can modify the solver, implement additional layers, use different types of regularization, use an ensemble of models, or anything else that comes to mind. If you implement these or other ideas not covered in the assignment then you will be awarded some bonus points.

cs231n
cs231n
karpathy@cs.stanford.edu