CS231n Convolutional Neural Networks for Visual Recognition

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 **Two 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

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 1 snapshot can then be found here. If you're 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'll be ready to start on your assignment right away. We've 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're ready to go directly to working on the assignment.

[Option 2] Manual install, virtual environment: If you'd like to (instead of Anaconda) 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 assignment1
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 `assignment1` directory:

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

Start IPython: After you have the CIFAR-10 data, you should start the IPython notebook server from the `assignment1` 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 `assignment1` 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 `assignment1.zip`. Upload this file to your dropbox on the coursework page for the course.

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: Two-Layer Neural Network (25 points)

The IPython Notebook **two_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.

Q6: Cool Bonus: Do something extra! (+10 points)

Implement, investigate or analyze something extra surrounding the topics in this assignment, and using the code you developed. For example, is there some other interesting question we could have asked? Is there any insightful visualization you can plot? Or anything fun to look at? Or maybe you can experiment with a spin on the loss function? If you try out something cool we'll give you up to 10 extra points and may feature your results in the lecture.

---

cs231n
cs231n
karpathy@cs.stanford.edu