Getting Started with f15ece6504

In this course, we will be using python considerably. Most of the assignments will need a good amount of python. Although many distributions of python are available, we recommend that you use the Anconda Python distribution. The advantage of using Anaconda is that: - It installs with most of the important python-packages - It does not need root access to install new packages - Supported by Linux, OS X and Windows - It is completely free!

We suggest that you use either Linux (prerferably Ubuntu) or OS X! Follow the instructions here to install Anaconda python. Remember to make Anaconda python the default python on your computer. Common issues are addressed here in the FAQ.

Python If you are comfortable with python, you can skip this section!

If you are new to python and have sufficient programming experience in using languages like C/C++, MATLAB, etc., you should be able to grasp the basic workings of python necessary for this course easily. We will be using the Numpy package extensively as it is the fundamental package for scientific computing providing support for array operations, linear algebra,etc. A good tutorial to get you started is here. For those comfortable with the operations of MATLAB, this might prove useful.

For some assignments, we will be using the ipython notebook. Ipython is a command shell for interactive computing developed primarily for python. The notebook is a useful environment where text can be embedded with code enabling us to set a flow while you do the assignments! If you have installed Anaconda and made it your default python, you should be able to start the ipython notebook envirnoment with:

```
$ ipython notebook
```

The ipython notebook files have .ipynb extension which you should be able to open now by navigating to the right directory.

Starting homework 0: This homework is a warm up for the rest of the course. As part of this homework you will be coding two classifiers: Support Vector Machine (SVM) and logistic regression. You will train these to classify images in the CIFAR-10 dataset. The CIFAR-10 is a toy dataset with 60000 images of size 32 X 32, belonging to 10 classes. You need to start with svm.ipynb first to implement the SVM and then go ahead with Softmax.ipynb to implement logistic regression.

Getting the dataset Make sure you are on the internet and navigate to hw0/f15ece6504/data in the hw0 folder. Run:

```
./get_datasets.sh
```

This script will download the python version of the database for you and put it in hw0/f15ece6504/data/cifar-10-batches.py folder.

Getting Spearmint As part of this homework, you will be using spearmint to tune the hyper-parameters like learning rate, regularization strength, etc. Spearmint is a software package to perform Bayesian optimization. It is designed to automatically run experiments in a manner that iteratively adjusts a number of parameters so as to minimize some objective in as few runs as possible.

If you have anaconda installed, setting up spearmint should be pretty straightforward. You can find installation and usage instructions here. You need to use the command line interface to work with spearmint. You can look up the ../examples/branin/ to get an idea.

TLDR: Install spearmint! The rest is there on the ipython notebook.

SVM and Logistic Regression As you might already know, SVM and logistic regression clasisfiers are very similar except that they calculate the loss differently. Here is a brief summary of the classifiers and if you need a detailed tutorial to brush up your knowledge, this is a nice place!

Let x_i, y_i denote the instances and their labels respectively. The process of training is to learn a function f that maps x_i to y_i. A simple linear classifier can be expressed as

$$f(x_i, W, b) = Wx_i + b$$

To evaluate the performance of our mapping, we need a loss function. The loss function needs to be optimized w.r.t the weights to obtain the best performance. The SVM loss funtion L_i for each instance is given by

$$L_i = \sum_{j \neq y_i} \Big[max\big(0, f(W, b, x_i) - f(W, b, x_i) + \Delta\big) \Big] + \lambda R(W)$$

Here, Δ is the margin between the score for the correct class and for all other classes. $R(W)$ is the regularization loss which is the squared some of all the weights learned and λ is the regularization strength. For Logistic Regression, you can use the Softmax Loss function given by:

$$L_i = -log\left(\frac{e^{p_{y_i}}}{\sum_j e^{p_j}}\right) + R(W)$$

where p_j is $w_j^T x_i + b$ and j is over all classes.

The loss is optimized by using Stochastic Gradient Descent (SGD). It is inefficient to compute the loss on the entire training set before updating the weights everytime. So, we form batches of smaller sizes and obtain the loss and its gradients using these instances. Then the weights are updates as:

$$W_{iter} = W_{iter-1} - \eta \nabla L_{batch}$$

Here, η is the learning rate (which is a hyper parameter) and ∇L_{batch} is the gradient of the loss computed using the mini-batch.