Homework V

You can submit in groups of 2!

All assignments need to be submitted via github classroom:
https://classroom.github.com/assignment-invitations/7bbd9202912cc1ddab7b773bb0a51f68
(the repository is empty for this assignment)

And on courseworks:
https://courseworks2.columbia.edu/courses/34678/assignments/70668

The homework is due 04/29/17 at 4pm.

All the tasks are to be completed using the keras Sequential interface. You can but are not required to use the scikit-learn wrappers included in keras. We recommend to run all but Task 1 on the habanero cluster which provides GPU support (though you could also run task 2 on a CPU). Running on your own machine without a GPU will take significantly longer. We have limited access to GPU resources on the cluster, so make sure to start working on the homework well in advance. Feel free to use any other compute resources at your disposal.

You can find instructions on accessing the cluster below.

Task 1 [10 Points]

Run a multilayer perceptron (feed forward neural network) with two hidden layers and rectified linear nonlinearities on the iris dataset using the keras Sequential interface. Include code for model selection and evaluation on an independent test-set.
[4pts for running model, 3pts for correct architecture, 3pts for evaluation]

Task 2 [30 Points]

Train a multilayer perceptron on the MNIST dataset. Compare a “vanilla” model with a model Qusing drop-out. Visualize the learning curves.
[Running model 10pts, model selection 7.5pts, dropout 7.5pts, learning curve 5pts]

Task 3 [30 Points]

Train a convolutional neural network on the SVHN dataset in format 2 (single digit classification). You should achieve at least 85% test-set accuracy with a base model. Also build a model using batch normalization. You can compare against other approaches reported here if you're curious. You are not required to use the unlabeled data. [10pts for working model, 15pts for 85%, 5pts for batch normalization, BONUS 10pts for acc >=90%]

Task 4 [30 points]

Load the weights of a pre-trained convolutional neural network, for example AlexNet or VGG, and use it as feature extraction method to train a linear model or MLP on the pets dataset. You should achieve at least 70% accuracy. It's recommended you store the features on disk so you don't have to recompute them for model selection.
The pets dataset can be found here: http://www.robots.ox.ac.uk/~vgg/data/pets/
We will be working with the 37 class classification task.
[10pts for loading model, 10pts for retraining, 10pts for 70% accuracy]

Note: We have compiled a list of Keras tips to help you with problems that you may run into here.

Files to submit:

The directory should contain 4 folders, one for each task

1. README with the performance that you get on each task.
2. Code and the plots in the respective folders.

If you are using a personal server for running experiments, include your conda environment file in the repository. Use this command:

```
$ conda env export > environment.yml
```

More details here: https://conda.io/docs/using/envs.html#export-the-environment-file
Make sure you use the same tensorflow version though (since they might use different CUDA versions and we cannot build CUDA again on the server)

Please refer to the Habanero HPC Cluster User Documentation for information about how to get started using the cluster.
Users log in to the cluster's submit node, located at habanero.rcs.columbia.edu. If logging in from a command line, type:

$ ssh <UNI>@habanero.rcs.columbia.edu
where <UNI> is your Columbia UNI. Your password will be your usual Columbia password. After logging in to Habanero you will be in your home directory (rigel/home/<UNI>). This storage space (10 GB) is appropriate for smaller files, such as documents, source code, and scripts but will fill up quickly if used for data sets or other large files.

There is also a scratch directory for the whole course, where you can store larger files:
`/rigel/edu/coms4995`

You can find sample scripts for submitting jobs to the cluster at
`/rigel/edu/coms4995/scripts`
There is a sub-folder of the scratch folder for each user, and you should use that as your working folder:
`/rigel/edu/coms4995/users/<UNI>`

Interactive sessions

If a GPU is currently available, you may launch an interactive session with the following commands:
STEP 1) First request a GPU node on Habanero:

```
$ srun --pty -t 0-02:00:00 --gres=gpu:1 -A edu /bin/bash
```

STEP 2) Load the following modules two modules:

```
$ module load cuda80/toolkit cuda80/blas cudnn/5.1
$ module load anaconda/2-4.2.0
```

STEP 3) Install tensorflow-gpu as user (this can take few minutes, please wait):

```
$ pip install tensorflow-gpu --user
$ pip install keras --user
```

STEP 4) Start python and test tensorflow

```
$ python
>>> import tensorflow as tf
>>> hello = tf.constant('Hello, TensorFlow!')
>>> sess = tf.Session()
>>> print(sess.run(hello))
Hello, TensorFlow!
```

Scheduled job (batch / non-interactive)

The following command will submit a batch job to the scheduler. This particular example will request a gpu and run a hello world program for tensorflow.

```
$ sbatch /rigel/edu/coms4995/scripts/gpu-submit-script.sh
Submitted batch job 404842
```

View information about the submitted job.

```
$ squeue -j 404843
JOBID PARTITION     NAME     USER ST       TIME  NODES
NODELIST(REASON)
404843 short gpu-subm  <UNI>  R       0:02      1 node025
```

View the status of your submitted jobs

```
$ squeue -u <UNI>
       JOBID PARTITION     NAME     USER ST       TIME  NODES
NODELIST(REASON)
       404841     short     bash  <UNI>  R      16:16      1 node025
```

After running, the output of your job will by default be saved into a file of the format slurm-<JOBID>.out