Keras + Tensorflow Guide

Recitation 3
Slides by Suraj Nair & Sid Murching

Installation

Install and/or Upgrade Pip

- Installing pip
  - Already installed if you're using Python 2 >=2.7.9 or Python 3 >=3.4 binaries from python.org
  - Otherwise, download get-pip.py
    - Link: https://bootstrap.pypa.io/get-pip.py
  - Then, run `python get-pip.py` from the command line
  - Installation guide: https://pip.pypa.io/en/stable/installing/#installing-with-get-pip-py
- Upgrading pip
  - On Linux or macOS:
    - `$ pip install -U pip`
  - On Windows [5]:
    - `> python -m pip install -U pip`

Installing Tensorflow (Pip)

Pip Installation

- Link: https://www.tensorflow.org/versions/r0.11/get_started/os_setup
- Select the CPU-only binary corresponding to your operating system
  - Be sure to check if your system is 32 or 64 bit
- Set the TF_BINARY_URL environment variable
- Then, run: `$sudo pip install --upgrade $TF_BINARY_URL`

Installing Tensorflow (Anaconda)

- **Installing with Anaconda**
  - Link: https://www.tensorflow.org/versions/r0.11/get_started/os_setup#anaconda_installation
  - Create a new Anaconda environment for Tensorflow and its dependencies
    - `$ conda create -n tensorflow python=2.7`
  - Activate the conda environment: `$source activate tensorflow`
  - Now, install Tensorflow as described in the pip instructions
    - Export $TF_BINARY_URL, run `pip install --upgrade $TF_BINARY_URL`
  - Or use conda:
    - conda install -c conda-forge tensorflow

Installing Keras

- Keras
  - Deep learning library
  - Provides an high-level interface over Theano & Tensorflow for building/fitting neural nets
- For CS 155, please use the Tensorflow backend when using Keras
- OSX/Windows/Linux: `%> pip install keras`
- Install guide: https://keras.io/#installation

Creating a Deep Model with Keras

- Process
  - Define your model
  - Compile your model
  - Fit your model
  - Evaluate model
- Can see an example in HW4 sample code

Defining Your Model

- Use the Sequential class
  - keras.models.Sequential
    - Ex: model = Sequential()
  - You can then add layers with model.add
    - Ex: model.add(Dense(N))
    - Ex: model.add(Convolution3D())
  - Adds layers in order
  - Once you are done use model.summary()
    - Gives an overview of layers, parameters, input and output shape of each

Compiling your model

- To compile use model.compile()
- Takes following arguments:
  - Loss
    - ‘mse’ – mean squared error
    - ‘categorical_crossentropy’ – categorical cross entropy
  - Optimizer
    - ‘sgd’- Stochastic gradient Descent
    - ‘rmsprop’ – RMS Prop
  - Metrics
    - ’accuracy’ is you want it to maintain accuracy as well as loss, etc.
- If your model has problems (layers/dimensions that don’t match) an error will be raised during compiling

Training your model

- Use model.fit()
- Takes training X, training Y
- Also takes batch_size, nb_epoch
- If you input / output sizes don’t match what model expects will raise error

Evaluating your model

- Model.evaluate()
- Pass in in input and outputs you want to evaluate on.
- Can pass in training or testing sets
- Will return loss and other metrics included in model.compile()
- Can also use model.predict() to just get predictions

Other Notes

- There will be an OH specifically for installation problems
- There will be also be general OH for the set

HW4 Sample Code Walkthrough

- I will now walk you through the HW4 sample code.
  - Will explore variations to sample code
- Will also demo ConvnetJS, which is used on the HW
- http://cs.stanford.edu/people/karpathy/convnetjs/demo/mnist.html

More about ConvNetJS

- Has many different datasets that you can play with
- On the homework you will be using the MNIST dataset
- Lets you create a deep model with a javascript-like syntax
- Can set learning rate, optimizer, and all other criteria much like you can with Keras
- We will walk through an example convnetJS and try making modifications