# Inference and Representation: Programming Paradigms in Machine Learning

Rahul G. Krishnan

New York University

Lab 9, Nov 2, 2016

# Outline

1. Programming Models
2. Wrappers on wrappers
3. Specific kinds of models
4. Structuring your Projects

## Languages

There are quite a few choices. . . .

- **Python**: Widely used. With packages like numpy, scipy, and scikit-learn, you have most of the functionality of MATLAB.
- **Lua**: Very fast and easy to integrate with C code. The language is very simple but has no predefined scoping rules.
- **MATLAB/Octave**: Familiar environment, easy semantics, great plotting and visualization library
- **C/C++** : Rarely used for active research but used for building high-performance implementations of popular algorithms.

# Deep Learning Libraries

- Wrappers on languages that construct tools to allow you to easily build complex discriminative and generative models.
- **Torch:** Tensor library in Lua, gives you support for running code on the GPU
- **Theano:** Autodifferentiation library in python. Allows you to run code on the CPU or the GPU.
- **Tensorflow:** Google's autodifferentiation library. Much more recent though has a rapidly growing user-base
- **Caffe:** High level neural network library. Has a fairly elaborate model zoo. If you want to specify and run a model from the literature and play around with the features learned, this is a good approach.

- **Vowpal Wabbit** `https://github.com/JohnLangford/vowpal_wabbit`: Microsoft Research Library. Optimized for speed.
- **MALLET** `http://mallet.cs.umass.edu/`: Java Library implementing many tools for document classification.
- **Scikit-Learn** `http://scikit-learn.org/stable/`: Machine learning library in python. Implements *many* common discriminative, generative models and algorithms for dimensionality reduction and visualization. Should be your first stop if you work in python!

# Torch

- Torch Cheatsheet `https://github.com/torch/torch7/wiki/Cheatsheet`
- Torch ecosystem has many auxillary libraries that support plotting, multithreading, saving/loading to HDF5 etc.

# Keras

- If you like the torch interface for building neural networks but would like to have the python ecosystem, this is a good bet
- Allows you to use a tensorflow or theano as the backend

Lasagne `https://github.com/Lasagne/Lasagne` is similarly another popular library

# Blocks & Fuel

- Blocks `https://github.com/mila-udem/blocks` & Fuel `https://github.com/mila-udem/fuel`
- Blocks is a wrapper around theano written with functionality to help you save, reload and visualize your model easily.
- Fuel contains many datasets commonly used for machine learning tasks

# Edward

Available at: `http://edwardlib.org/tutorials/`

- Built on top of tensorflow
- Flexible language for Bayesian deep learning. i.e blending probablistic inference with techniques in deep learning.
- Contains many algorithms for inference and learning (both variational methods and MCMC methods)

# Stan

- `http://mc-stan.org/`
- Probablistic programming.
- Allows you to easily build hierarchical graphical models and perform inference within them.

## PyMC3

- `https://github.com/pymc-devs/pymc3`
- Probablistic programming using theano as a backend

# Generative Adversarial Networks (GANs)

If you're implementing GANs, I'd recommend starting from one of these

- DCGAN `https://github.com/Newmu/dcgan_code`
- DCGAN-Torch `https://github.com/soumith/dcgan.torch`
- LAPGAN `https://github.com/facebook/eyescream`

# Modeling Sequential Data

- Machine Translation Code `https://github.com/nyu-dl/dl4mt-tutorial`
- Torch RNN Lib`https://github.com/facebookresearch/torch-rnnlib`: Library that provides an implementation for an RNN from Facebook AI Research

# Choosing a language

- Pick *one* of the above frameworks.
- Try and anticipate the kinds of experiments you will do and pick based on which one will simplify your life the most
- If you're stuck at an implementation detail, ask a question in the Google group for the corresponding language.

# Starting out

- Use github or some form of version control
- Start simple. A good way to start a project is to reproduce someones experimental results.
- Baselines and benchmark datasets are important. They help keep a check on where you are with respect to what other methods are doing.

# Tracking Progress

- Remember the basics of doing machine learning: whitening data, feature normalization, the small things matter
- Draft and layout your plan
- For the project and specific experiments you will conduct to check off boxes