FastTrack Machine Learning Curriculum

This document contains a fast-track guide to learning some basic machine learning tools (mostly neural network based). The emphasis will be on hands-on learning and here we'll mostly provide refrences to external resources rather than rehash what they already cover.

Online Courses

While the fastest route to getting a feel for things is probably to try implementing some code (as discussed in the tutorials below) it can also be helpful to supplement this with some more drawn out theoretical material in courses. There's a large number of high quality and free online courses about neural networks so here we'll just list a few:

- Geoffery Hinton has a very good and inspirational course here:
  `https://www.coursera.org/course/neuralnets`
  This course is light on details and implementation but is quite good for theory (you have to register and download the videos even if the course is over).
- A more basic and general machine learning course is the one by Andrew Ng:
  `https://www.coursera.org/learn/machine-learning`
  This course has been given many times so you should be able to download some version of it.
- Hugo Larouchelle has a set of youtube lectures and also course material for a very detailed neural network and machine learning course:
  `http://www.dmi.usherb.ca/~larocheh/index_en.html`
  `https://www.youtube.com/playlist?list=PL6Xpj9I5qXYEcOhn7TqghAJ6NAPrNmUBH`
- This stanford course has a nice intro tutorial on neural network and basic machine learning:
  `http://cs231n.github.io/`
  `http://cs231n.github.io/neural-networks-1/`
  Check out the earlier sections of their tutorial covering back-propagation and linear classifers as well. The course actually covers convolution neural networks for images which will not be our focus here (but is of course interesting).
- Another stanford couse with more of a natural language focus:
  `http://cs224d.stanford.edu/`
  `https://www.youtube.com/watch?v=sU_Yu_USrNc`

Neural Network Basics

Neural networks are very simple but powerful models. The best way to learn about them is to start with a basic tutorial, then implement a toy model on your own (from scratch) and then a more interesting model (perhaps using a real toolkit like Theano).

1. Some basic neural network tutorials:
   - `http://iamtrask.github.io/2015/07/12/basic-python-network/`
   - `http://www.wildml.com/2015/09/implementing-a-neural-network-from-scrat`
   - Appendix A of 1411.2738 contains detailed explanations of backpropagation, etc...

2. Using inputs from these tutorials implement *your own* neural network. You can/should use libraries like numpy but not higher level libraries like Theano or TensorFlow that implement backpropagation.

   - Implement a neural network that outputs a real number y given as input a real number x. Train it to output $y = sin(x)$.
   - Modify the output of the network so the input is two real numbers (x_0, x_1) and the final layer is a soft-max with two possible outputs. One output should fire if $x_0 > sin(x_1)$ and the other should fire for $x_0 < sin(x_1)$.
   - In both cases above try varying the range of the training data (i.e. the range over which x, x_0, x_1 span and see how the network performs both inside and outside that range).
   - Try varying the depth and the activation function of the network. Try making a deeper network and using a rectified linear activiation unit ($f(x) = x$ for $x > 0$ and $f(x) = 0$ for $x < 0$).

3. To do more advanced things with neural networks its useful to work with a more powerful library that not only implements matrix operations very efficiently on the CPU and GPU but also can automatically handle back-propagating derivatives. Theano and TensorFlow are two (python-based) examples of such libraries (there's also Torch in Lua and many others in different languages). Here we focus on Theano.

   - Work through some of the Theano tutorials starting here:
     `http://deeplearning.net/software/theano/tutorial/adding.html`
     `http://deeplearning.net/tutorial/logreg.html#logreg`

4. The second tutorial above implements a network to classify hand-written "digits" (numbers between 0 and 9). Try implementing this both in Theano but also in your own neural network implementation (from above). This task is easy enough that you should be able to acheive 97-98% accuracy or more even with your own implementation.

Less Basic Neural Network

Here are a few interesting themes from the world of neural networks and some associated literature:

Dropout is **regularizer**. Put another way its a way to stop neural networks from overfitting the training data. The original paper is 1207.0580.

Autoencoders ...

Word Embeddings

The first practical problem we'll address with neural networks is word embeddings. The idea of a word embedding is to assign, to each word, a vector in such a way that mathematical operations between vectors (addition, distances, etc...) have semantic meaning in language. Here are some useful references:

- A very nice explanation of the idea of word vectors can be found here:
  `http://colah.github.io/posts/2014-07-NLP-RNNs-Representations/`
- One of the recent papers that brought this idea into the spotlist is 1310.4546.

- A more expository explanation of the above paper (along with some info on back-prop) can be found in 1411.2738.

References