John Allard
1/28/2016

# Project Proposal

**Project Name** : Crash Course on LSTMs
**Elevator Pitch** : I will implement and compare long-short-term-memory recurrent neural networks in 3 different languages; Python, Haskell, and C++. I will then train and use these LSTMs as character generation models, kind of like what is done here https://github.com/karpathy/char-rnn.
**Group Members** : Me

I have decided to use this opportunity to learn about both programming languages and machine learning by implementing one of my favorite ML models, LSTMs, in my 3 favorite languages. LSTMs are a relatively complicated beast to implement, but if done correctly they can be used to make accurate predictions over time-series data. What this means is that, unlike vanilla neural networks, RNNs (of which LSTMs are a specific type) feed the output of one layer at time $t$ as the input to other layers at $t+1, t+2, ..., t+n$ where $n$ is a parameter of the model. This means that RNNs can learn how different inputs are related to each other over time, which makes the model much more accurate than vanilla neural networks when trying to model systems whose output depends on previous outputs.

I will start by implementing these algorithms/models in Python and Haskell, and if enough time is left over I will move onto C++. The main reason I chose these languages is because it will give me an opportunity to do side-by-side comparisons between low-level-imperative (C++), high-level-scripting (Python), and high-level-functional (Haskell) languages. I believe that this will help me to better understand the benefits and drawbacks of the different types of languages while simultaneously improving my understanding of machine-learning.

I plan on implementing optional CUDA GPGPU bindings for each of the LSTMs. For C++, I'll be using the TensorFlow library released by Google. For python, I will use Theano to handle the tensor processing which I can build GPU support on top of. For Haskell, I will be using the Accelerate package which supports both CUDA and OpenGL programming. I plan on documenting the speed up seen by using a GPU compared to a CPU to do the training and prediction for each of the models.

As for what I plan to actually do with the models, I want to do a generative model that takes in large blocks of ASCII-text (e.g. the entirety of Shakespeare, or the King James Bible, etc.), and can learn to generate text that is extremely similar to that of which it is

trained on. It does this by learning the relations between characters (e.g. *t* comes after *e* 12.42% of time, g comes after *e* 0.46% of the time), and the structure of the LSTMs allow them to learn the relations between characters that are distant from each other, which allows the models to learn proper punctuation and sentence structure. So after I get the general LSTM libraries implemented in the given languages, I will train them on different types of data and showcase the results.

As for timing, I plan on knocking out the python one first because I am most fluent in python and it'll be easiest for me to learn the techniques and nuances of implementing LSTMs by starting with Python. From there I will move onto the implementation in Haskell. This will definitely be the trickiest part of my project, and I will allocate the most time (2-3 weeks) to this task. If time permits, at the end I will try and knock out a good implementation in C++, a language that I am very comfortable with. After a model is implemented I will train them at home on my desktop, which has a very powerful Nvidia GPU allowing me to test the GPGPU bindings for each of my models.

This is going to be a lot of work, but I think that it will have a great benefit on my understanding of all of the languages used. I am sure I can get done at least the Python and Haskell implementations but the C++ one might be a stretch.