Deep Boltzmann Machines: Edgar Chen, Taichi Akiyama

Summary: We are going to implement Deep Boltzmann Machines with Parallel Tampering on a GPU.

Background: Deep Boltzmann Machines http://www.utstat.toronto.edu/~rsalakhu/papers/dbm.pdf are powerful generative models which are capable of doing unsupervised learning on large datasets. However, due to their expressivity, the training process is very much compute-bound. Our goal is to implement a Deep Boltzmann Machine training and inference program using modern-day GPUs. Boltzmann Machines can do very interesting things, and they don't require labels on the data to learn about the data, making them useful in settings where people have collected a lot of data but don't have the resources to label all of the data. They can learn structure about the data, which can then be used to classify similar datapoints or even generate new or similar data. They can learn, for instance, what a certain alphabet system's characters "look like" and generate plausible examples that would fit into the alphabet, or complete part of a character given the other half, which can be useful in filling in missing parts of data.

Challenge: http://15418.courses.cs.cmu.edu/fall2016/lecture/dnn/slide_048 is actually a pretty good example, since DBMs operate somewhat like neural networks in general (but use a lot more probabilistic sampling). There is a lot of memory movement required, and the parameters need to synchronize at each step. It's possible that this won't be challenging enough and we might want to do something like distribute the computation on latedays, but we think this is plenty challenging, given that the base code for a good DBM implementation is much more complicated than a neural network.

Resources: We will probably use a linear algebra library like Eigen, and maybe borrow some of CUDNN, a CUDA neural networks computation library, but probably not fully supplied neural network library like TensorFlow. http://www.utstat.toronto.edu/~rsalakhu/papers/dbm.pdf is the paper we will be using. We could benefit from access to a modern-day GPU, but we can probably just borrow one of the GHC machines for that.

Goals and Deliverables: Create a working DBM implementation with significant speedup over a naïve CPU version

Show off the different things that unsupervised learning can do: we'll probably be doing it on characters so we can do things like generate new characters that "look like" characters in the dataset, and generate closest representation in the dataset for a specified input.

Platform Choice: Obviously a CUDA program with a 1080

Schedule: Week of 11/7: Do research and create the sequential version

Week of 11/14: Create working parallel version of program

Week of 11/21: Do tuning on the parallel section to make it as fast as possible

Week of 11/28: Put together the deliverables (takes a nontrivial amount of time and coding), poster, and website