1 Brief Bio

I am a first year Ph.D. student in computer science and my research interests lies in the interdisciplinary field of learning science and computer science, particularly in intelligent tutoring system. Through the systematic training CS267, I wish to get a substantial improvement on both the understanding of parallel computing and actual ability to implement a complicated parallel program.

2 Training RNNs in parallel

RNNs are a rich class of neural networks that extends traditional feedforward neural networks with directed cycles. RNNs have been proved to be good at tasks involving learning sequences and has achieved best known result in handwriting recognition recently. By adding recurrent connections, RNNs are able to 'remember' previous inputs in network's internal state, which can then be used to influence the network output. Thus, RNNs can model sequential data in a much more natural way than by only using a fixed number of previous inputs, e.g. n-gram language model. The most widely used training method for RNNs is backpropagation through time (BPTT), a simple generalization of backpropagation which unfolds a RNN over time to convert it into a feedforward neural network that always using same weights. However, standard RNN architecture suffers from the vanishing gradient problem which prevent RNN from storing information that is not used until a long time later. In 1997, Hochreiter and Schmidhuber the proposed the Long Short-Term Memory (LSTM) architecture to tackle this problem, which is the most effective solution so far.

Stimulated by the success of RNNs and the great amount of remaining challenges, more and more algorithmic innovations are taking place in this field, deriving models that sometimes differs significantly from a common or standard network topology. The new models are typically developed in productivity-oriented tools such as Matlab. However, in order to become relevant, these innovations must be ultimately tested on real-world data set. The key problem of doing so in such tools is their incompetence to handle large volume of data and efficiently exploit advanced hardware to accelerate computation.

To solve the dilemma described above, I’m interested in building a toolkit for RNN training that achieves productivity and performance at the same time. My toolkit will provide a Matlab-like matrix based API which enable users to easily implement RNNs with arbitrary network topology. Then the system will automatically translate user’s RNN model into a directed acyclic graph representation and then efficiently executed it using multiple threads. Specifically, the project can be divided into 2 phases. The first phase is to be able to train one sequence at a time using multiple threads and the second phase is training multiple sequences at a time using multiple threads. In the second phase, I plan to apply several merging and scheduling technique to gain further performance improvement.