Machine Learning in Natural Language Processing

Bio

I completed my undergraduate education in computer science, math, and psychology at Purdue University. I then went to Berkeley Law School, where I focused on intellectual property law. After graduating, I worked as a patent attorney for three years. I am now a 2nd year PhD student in the Business and Public Policy program at the Haas School of Business.

I am interested in conducting empirical analysis of the U.S. patent system. I focus on concepts such as empirical measures of patent importance, the efficiency of patent markets and transfers, and patent-based evidence of trends in innovation. I also plan to work in large-scale statistical computing and computational graph theory. In this class, I would like to learn techniques and tools for use in this research.

Application Problem: Natural Language Processing

I am interested in natural language processing on large data sets. Natural language processing (NLP) may be divided into a learning phase and an analysis phase. In the analysis phase, NLP is typically embarrassingly parallel and may be solved via a MapReduce algorithm. Once the NLP engine is trained, sections of text may each be assigned to a different processor, and communication between processors is relatively limited. Processors evaluating nearby or related chunks of text may need to communicate, but in general each processor need only receive the chunk of text data and transmit back the analysis.

The learning phase is more complex. Typically, a set of pre-processed data is provided to the NLP system in both problem and solution form so that the NLP system can "learn" to map a problem to its solution. However, machine learning on large datasets presents significant difficulties in this context for two reasons. First, running time is somewhat limited since data grows much more rapidly than the computing power of a single processor. Second, many machine learning algorithms over large data sets can be difficult to parallelize. Accordingly, these machine learning algorithms are very difficult to scale.

Many machine learning algorithms for natural language processing often rely on graph traversal algorithms. Graph traversal algorithms are typically associated with apparently random memory access patterns. This makes parallelization difficult because it is unclear at the outset which data needs to be sent to which processor. Since the amount of data that needs to be sent between nodes scales faster than the number of nodes, the graph traversal algorithm does not scale very well. (See Hong et al., Efficient Parallel Graph Exploration on Multi-Core CPU and GPU (2011)).

However, some work has been done in carefully constructing machine learning algorithms to benefit from a MapReduce approach. If algorithms can be written in a summation form, then they can be easily parallelized. (See Chu et al., Map-Reduce for Machine Learning on Multicore). The availability of parallelizable algorithms for machine learning in this context depends at least on part on the complexity and characteristics of the particular problems being addressed. (See Gillick et al., MapReduce: Distributed Computing for Machine Learning (2006)).