Scaling Bayesian Network Parameter Learning with Expectation Maximization using MapReduce

Erik B. Reed
Carnegie Mellon University
Silicon Valley Campus
NASA Research Park
Moffett Field, CA 94035
erikreed@cmu.edu

Ole J. Mengshoel
Carnegie Mellon University
Silicon Valley Campus
NASA Research Park
Moffett Field, CA 94035
ole.mengshoel@sv.cmu.edu

Abstract

Bayesian network (BN) parameter learning from incomplete data can be a computationally expensive task for incomplete data. Applying the EM algorithm to learn BN parameters is unfortunately susceptible to local optima and prone to premature convergence. We develop and experiment with two methods for improving EM parameter learning by using MapReduce: Age-Layered Expectation Maximization (ALEM) and Multiple Expectation Maximization (MEM). Leveraging MapReduce for distributed machine learning, these algorithms (i) operate on a (potentially large) population of BNs and (ii) partition the data set as is traditionally done with MapReduce machine learning. For example, we achieved gains using the Hadoop implementation of MapReduce in both parameter quality (likelihood) and number of iterations (runtime) using distributed ALEM on for the BN *Asia* over 20,000 MEM and ALEM trials.

1 Introduction

Bayesian networks (BNs) are fundamental structures for probabilistic inference and modeling [14]. Parameters to BN networks, their conditional probability tables, may need to be learned. When the BN structure is known, but data is incomplete, several algorithms have been used: Expectation Maximization [4, 8, 9, 15], Markov Chain Monte Carlo methods such as Gibbs sampling [10], and gradient descent methods. In this paper, we investigate Expectation Maximization (EM). EM operates by first performing inference over a dataset, finding the states of all hidden nodes (E-step), followed by an MLE estimate over the now complete data to calculate an improved set of parameters (M-step). The EM algorithm iterates in this manner until it no longer earns any gains in likelihood, resulting in convergence. The likelihood of BN parameters increases monotonically as EM iterates, resulting in guaranteed convergence.

Unfortunately, depending on initial conditions, solution quality can vary greatly: EM is very susceptible to local optima. Methods of making EM less prone to these problems have been investigated, including: adjusting the initialization phase of the EM algorithm [4]; introducing stochastic variants of EM [3, 7]; and exploiting parameter constraints [13]. An age-layered approach for EM (ALEM) has also been developed [16], based on research on preventing premature convergence in genetic algorithms [5]. Scaling EM for large BNs and data sets has also been explored on single machines with parallel processing [2, 17]. MapReduce for BN learning from complete and incomplete data, using a single BN, has been developed as well [1].

This papers aims to address two of the main problems for BN parameter learning using EM: high computation time and convergence to local optima (i.e. low likelihoods and poor solution quality). For BN learning using the EM algorithm, computation time for the E-step is a linear function in the

amount of data (or evidence), and each E-step can take a substantial amount of time even when an efficient junction tree algorithm is used [1]. Since EM is iterative, an increase in data has a multiplicative effect on computation time. EM convergence is greatly influenced by the initial parameters, which can result in convergence to local optima with poor solution (likelihood) quality.

The above problems motivate our use of populations of BNs and distributed computing, in particular MapReduce (MR) and Hadoop with the EM algorithm. In this paper, we develop and perform experiments with Multiple Expectation Maximization (MEM) and Age-Layered Expectation Maximization (ALEM) for MapReduce. This work integrates two existing research directions: EM on MapReduce [1] and age-layered EM [16]. In particular, this paper discusses an approach to use MapReduce with a population of BNs. In other words, we are executing multiple EM runs (each run updates a separate BN) distributed according to the MapReduce framework, leading to substantial reductions in wall clock time for EM computation. Originally, ALEM did not use MapReduce [16] and previous work on EM with MapReduce did not operate on a population of BNs [1].

2 MapReduce for EM: Multiple BNs and Data Partitioning

2.1 MEM on MapReduce

To adapt both MEM and ALEM to MR, we fully distribute the E-step. The MR input is evidence for the BN, a list of n observations. The expectation values collected by the inference are serialized and transferred, in the M-step, to a single reducer to be summed. The reducer collects all the inference data and computes the MLE of the parameter values to be used in the next iteration of EM.

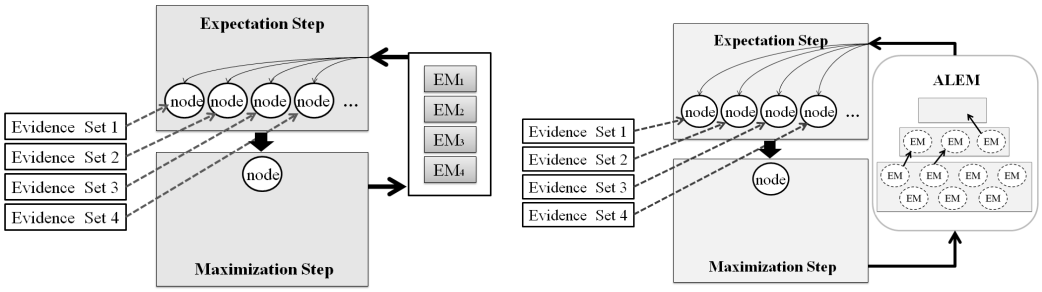
For a *single BN*, this procedure allows MEM and ALEM to be parallelized in a manner similar to what has been done for traditional EM [1]. However, since MR enables massive computation, each mapper can potentially compute the E-steps for *multiple BNs* (Figure 1a), and that is what we do in this paper. Each mapper operates on multiple BNs, while BN evidence (the data set) is partitioned and split among the mappers. BN inference is performed on each piece of evidence for the population of BNs. MEM terminates when BNs in its population have all converged. This allows both ALEM and MEM to be performed with coarse-grained parallelism while enabling a large amount of evidence to be processed. We are unaware of other existing research where both multiple BNs and partial evidence is distributed to multiple mappers.

2.2 ALEM on MapReduce

This paper adopts the ALEM algorithm [16], which is based on the genetic algorithm concept of creating and computing with a population of randomly initialized individuals. Each individual has a fitness, which is to be optimized, as well as an age corresponding to the amount of time the individual has been in a population [5]. Individuals are separated in layers with other individuals of like ages. Lower layers have young individuals in the GA, while the higher layers has the oldest (and often the fittest) member of the population. As individuals age, they ascend to high layers. The maximum age of each layer is determined by the *age gap* parameter; once individuals reach this age, they ascend to the next layer. Additionally, there are limits to the maximum number of individuals per layer. The age-layered structure reduces the possibility of fit, old individuals, stuck in a local optima, overtaking the population due to their high fitness.

In ALEM, a population of EM runs is created and updated [16]. The age of each EM run corresponds to its number of iterations, and the fitness of each EM run is its likelihood. EM runs are randomly initialized in the 1st layer, iterate until an age where they ascend to the next layer, and may need to compete for a spot in the next layer. Competition occurs when a layer is full: if an ascending EM run has greater likelihood, the non-ascending EM run is discarded to make room. Otherwise, the ascending EM run is not competitive enough and is discarded. ALEM continues until a given number of EM runs successfully converge using a pre-defined convergence criterion ϵ. ALEM terminates when a specified number of EM runs converge.

For both MEM and ALEM, multiple BNs are being processed for each MR operation. However, they differ in their treatment of the population of EM runs (i.e., BN parameters). MEM has a fixed population. ALEM, on the other hand, both terminates and starts new BNs. ALEM also performs likelihood checks for BNs changing layers as illustrated in Figure 1b. Since ALEM operates on a



(a) MEM on MapReduce (b) ALEM on MapReduce

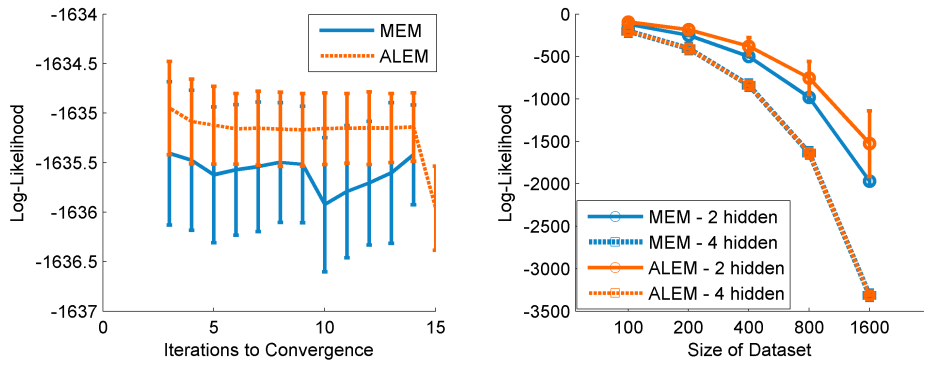
Figure 1: EM with populations of BNs on MapReduce. (a) shows MEM being applied on MR, each mapper performs expectation computation on a single evidence set and multiple BN instances. A single reducer performs MLE on the expectations. In (b), the reducer now performs MLE and creates/terminates new EM runs according to the ALEM layers.

dynamic population structure, the number of EM runs performed for each MR operation varies. In a way, we transition from random restart, which has been shown to be effective against premature convergence [6, 11], to ALEM using MapReduce.

3 Experimental Results

The C++ software library libDAI [12] is used extensively in our experiments.[1] The libDAI library has many inference algorithms built-in, including Junction Tree, Variable Elimination, and Gibbs sampling. We extend libDAI with Hadoop Streaming MapReduce to perform ALEM and MEM in both local and cluster environments. For experimentation, we rely on the Junction Tree algorithm for BN inference; to generate a dataset of evidence for the BNs, we use forward sampling.

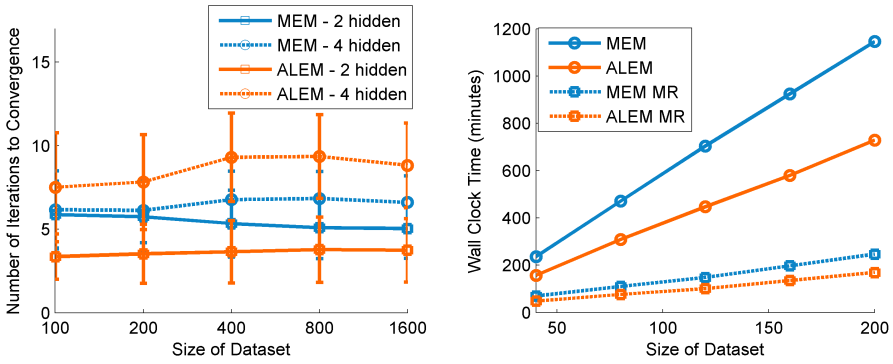
For MR experiments, 16 small instance (single core, 1GB RAM) machines on Amazon EC2 perform computation. For ALEM, we use the following parameters: number of layers = 5; age gap = 5; and minimum runs in lowest layer = 5. For both ALEM and MEM, the convergence tolerance was set to $\epsilon = 10^{-4}$, the maximum number of iterations was set to 100, and the population size was set to 15 (this means that ALEM and MEM halted when 15 EM runs converged).



(a) (b)

Figure 2: Experimental results for 20,000 trials of ALEM and MEM on the BN Asia. (a) shows the mean MEM and ALEM likelihoods versus the number of iterations to convergence for a dataset of size 800 and four hidden nodes; (b) shows the mean of converged likelihoods of ALEM and MEM for five dataset sizes (100, 200, 400, 800, and 1600 samples) and two sets of hidden nodes.

[1] http://libdai.org



(a) (b)

Figure 3: Comparing the performance of MEM and ALEM for varying data set sizes. (a) shows the impact on the mean number of iterations for ALEM and MEM over 20,000 trials of the BN Asia of varying dataset size and number of hidden nodes. (b) shows a linear reduction in wall clock time from running both MEM and ALEM on a single machine versus 16 machines on the BN Water for a fixed 15 iterations.

First we investigate the performance of ALEM versus MEM. A set of 20,000 ALEM and MEM trials were performed on the BNs Asia and Alarm to measure differences in likelihood and iterations until population convergence. Trials were run using MR due to the amount of computation. Each trial used a data set sampled from an existing BN using forward sampling. After forward sampling, a random set of nodes was hidden. We observe in Figure 2a the converged likelihoods and number of iterations for trials with a dataset of size 800 with four hidden nodes on the BN Asia. For this configuration, ALEM performs as well as MEM for finding a high likelihood, attaining a log-likelihood mean/standard-deviation of $-1.635/0.308$, compared to MEM's $-1.635/0.171$. Figure 2b shows the means and standard deviations of the likelihood over several dataset sizes, as well as for two and four hidden nodes. For four nodes, there is negligible difference in log-likelihoods, while two nodes demonstrates an increase in mean and standard deviation using ALEM. Figure 3a shows the number of iterations required for convergence of MEM and ALEM for the same trials. ALEM requires fewer iterations when two nodes are hidden, whereas MEM requires fewer iterations when four nodes are hidden. The reason for this is a topic for future research.

Next, we test ALEM and MEM on the BN Water, which takes 19.5 hours on a single machine to perform 15 iterations of MEM. Figure 3b shows the wall clock time of non-distributed ALEM and MEM versus MR ALEM and MEM. With and without MR, ALEM shows a speedup versus MEM. For a dataset size of 200, the wall clock time of MEM reduces from 1146 to 247 minutes, a 78.5% decrease, while the wall clock time of ALEM reduces from 729 to 169 minutes, a 76.8% decrease. The dataset size has a linear effect on wall clock time in all cases, with the proportional speedup of MR increasing as the dataset size increases. MR ALEM shows a further 30% reduction in wall clock time versus MR MEM (see Figure 3b).

4 Conclusion and Future Work

In this paper we adapted Multiple Expectation Maximization and Age-Layered Expectation Maximization to the MapReduce framework; we partitioned the dataset across multiple machines, each running a population of BNs. For both ALEM and MEM, scalability is achievable when inference computation time is high, when there are large amounts of evidence, or when there is a large population of BNs. Further work will look at testing more BNs, better understanding the differences between MEM and ALEM, increasing the amount of evidence available, and investigating how well ALEM and MEM on MapReduce scale when the population size (and potentially the number of mappers) grows to hundreds or thousands.

Acknowledgements: This material is based, in part, upon work supported by NSF awards CCF0937044 and ECCS0931978.

References

[1] A. Basak, I. Brinster, X. Ma, and O.J. Mengshoel. Accelerating Bayesian network parameter learning using Hadoop and MapReduce. In *Proceedings of the 1st International Workshop on Big Data, Streams and Heterogeneous Source Mining: Algorithms, Systems, Programming Models and Applications*, BigMine '12, pages 101–108, Beijing, China, 2012. ACM.

[2] P.S. Bradley, U. Fayyad, and C. Reina. Scaling EM (Expectation-Maximization) clustering to large databases. *Microsoft Research Report, MSR-TR-98-35*, 1998.

[3] B. Delyon, M. Lavielle, and E. Moulines. Convergence of a stochastic approximation version of the EM algorithm. *Annals of Statistics*, (27):94–128, 1999.

[4] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal Of The Royal Statistical Society, Series B*, 39(1):1–38, 1977.

[5] G.S. Hornby. ALPS: The age-layered population structure for reducing the problem of premature convergence. In *Proceedings of the 8th annual conference on Genetic and evolutionary computation*, pages 815–822. ACM, 2006.

[6] S.H. Jacobson and E. Yucesan. Global optimization performance measures for generalized hill climbing algorithms. *Journal of Global Optimization*, 29(2):173–190, 2004.

[7] W. Jank. The EM algorithm, its randomized implementation and global optimization: Some challenges and opportunities for operations research. In F. B. Alt, M. C. Fu, and B. L. Golden, editors, *Perspectives in Operations Research: Papers in Honor of Saul Gass 80th Birthday*. Springer, 2006.

[8] H. Langseth and O. Bangsø. Parameter learning in object-oriented Bayesian networks. *Annals of Mathematics and Artificial Intelligence*, 32(1):221–243, 2001.

[9] S.L. Lauritzen. The EM algorithm for graphical association models with missing data. *Computational Statistics & Data Analysis*, 19(2):191–201, 1995.

[10] W. Liao and Q. Ji. Learning Bayesian network parameters under incomplete data with domain knowledge. *Pattern Recognition*, 42(11):3046–3056, 2009.

[11] O.J. Mengshoel, D.C. Wilkins, and D. Roth. Initialization and restart in stochastic local search: Computing a most probable explanation in Bayesian networks. *IEEE Transactions on Knowledge and Data Engineering*, 23(2):235–247, 2011.

[12] J.M. Mooij. libDAI: A free and open source C++ library for discrete approximate inference in graphical models. *Journal of Machine Learning Research*, 11:2169–2173, August 2010.

[13] R.S. Niculescu, T.M. Mitchell, and R.B. Rao. Bayesian network learning with parameter constraints. *The Journal of Machine Learning Research*, 7:1357–1383, 2006.

[14] J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann, San Mateo, CA, 1988.

[15] M. Ramoni and P. Sebastiani. Robust learning with missing data. *Machine Learning*, 45(2):147–170, 2001.

[16] A. Saluja, P.K. Sundararajan, and O.J. Mengshoel. Age-Layered Expectation Maximization for parameter learning in Bayesian Networks. In *Proceedings of Artificial Intelligence and Statistics(AIStats)*, La Palma, Canary Islands.

[17] B. Thiesson, C. Meek, and D. Heckerman. Accelerating EM for large databases. *Machine Learning*, 45(3):279–299, 2001.