AISTATS 2014

Seventeenth International Conference on Artificial Intelligence and Statistics

April 22 - April 24, 2014, Reykjavik, Iceland

Reviews For Paper

Track	Proceedings
Paper ID	252
Title	FlexiLearn: Slow-Worker-Agnostic Distributed Learning for Big Models on Big Data

Masked Reviewer ID: Assigned_Reviewer_1

Review:

Question	
Overall Rating	Accept
Detailed Comments	The authors present a distributed stochastic gradient framework focused on matrix decomposition type algorithms. The key components are to use a blocking scheme which ensures correctness of the updates and to allow fast workers to do additional updates within their currently assigned blocks. Some theoretical analysis of the correctness and convergence speed up is given, along with empirical comparisons to Parallel SGD, an implementation on top of GraphLab, and a version of the proposed framework without allowing fast workers to do additional updates. The paper is well written and clear. While I am not an expert in this type of algorithm analysis (hopefully one of the other reviewers is!) the big picture was at least easy to ascertain. The experimental results are pretty convincing, and I appreciated that the scale of the experiments was realistic. It is a little concerning that the convergence time gets so much worse going from 16 to 32 cores, since that is still a relatively small number for truely "large scale" data analysis. It would have been nice to see a comparison to a state-of-the-art single processor algorithm such as sampling online VB for LDA (Mimno et al. 2012), even if they can "only" scale to 1 million documents. While I felt the basic ideas of the paper (sensible blocking and allowing fast workers to do additional updates) were somewhat obvious/incremental, the combination with some theoretical analysis and strong empirical results make this into a nice paper. Minor comment: In the explanation for the eq in theorem 2 "the first line" should be "the first two lines".

Masked Reviewer ID: Assigned_Reviewer_2

Review:

Question	
Overall Rating	Neutral
	This paper proposes to partition both the variables and the data among multiple machines to improve computational efficiency. To resolve the straggler problem it allows the fast worker to keep working by making multiple passes as it waits for the slow workers. The key idea is to identify

Detailed Comments	independent (or somewhat independent) block of data as well as variables and schedule them to different machines. While this is indeed an interesting idea and the authors use a variant of stochastic optimization to solve the proposed problem, my only gripe is that this paper seems more like a systems paper than a core machine learning paper. I do like the paper and find that it to be interesting and also firmly believe that the technique would be useful and the results are quite impressive. Among other things, a few points require special mention. For example, after reading the paper I get the impression that the analysis holds for SGD only -- if so then the paper should mention so. Also, how is their insight based on SGD different from what is proposed in [8]? Overall I like the paper. Modulo the reservations that I have in regards to this paper being inappropriate for this conference venue, I would go with a weak accept for the paper.

Masked Reviewer ID: Assigned_Reviewer_3
Review:

Question	
Overall Rating	Accept
Detailed Comments	This paper develops an approach for parallel and distributed projected stochastic gradient descent for a class of problems whose structure admits natural block partitioning schemes for both data and model parameters where a subset of blocks can be "touched" in parallel in each "sub-epoch". To some extent, the paper builds heavily on Gemulla et al KDD 2011. who considered this setting for distributed matrix factorization problems - here a slightly broader set of problems are taken up. The main algorithmic contribution is the idea that a faster processor can continue performing local SGD updates while a slow worker finishes, before all processors synchronize and the next subepoch starts. A theoretical analysis is provided to show convergence, and to bound variance within and between sub-epochs, and an empirical analysis compares the approach against a few other natural alternatives. In Algorithm 1, the synchronization steps and their cost is unclear. Does beta start off being column partitioned, and then grouped by rows for projection step? Is every sub-epoch a map reduce job to redistribute current state variables (betas in Figure 1) across the processors? Is there any potential benefit to having all workers continue SGD updates despite being otherwise ready to synchronize? In a sense, the fact that the algorithm "works" is not surprising. It may be worth clarifying how the analyses and proof techniques differ from, or build on the work of Gemulla et al. What causes BarriedFlexiLearn to be particularly worse for MMND in Figure 3? Relative to GraphLab, FlexiLearn and B. FlexiLearn appear to be rather disappointing with respect to increasing number of cores in Figure 2. A comparison against Hogwild may also have been interesting.

	Some comments on other general classes of machine learning problems where these ideas might apply would be useful to include. Typos: section 3.1 first line.