Auto Scaling Online Learning

Marco Tulio Ribeiro, Shrainik Jain

Mar 17, 2014

We propose a framework and algorithms for scaling online machine learning up or down, according to demand, and the priorities of the system. Different systems have different needs in terms of the cost assigned to machines, the cost of a bad user experience, etc. In this project, we focus most on the part that is general to auto scaling any application (and not only machine learning algorithms), but propose a framework that takes some ML particularities into account.

1 Introduction

Online machine learning algorithms operate on a single instance at a time. They have become particularly popular in natural language processing and applications with streaming data, including classification, ranking, etc [1, 2, 3]. Online learning is particularly interesting in the scenarios where data keeps streaming in, such as a web search engine doing advertisement placement. It is also interesting for scenarios where the whole dataset is too large to fit in main memory, as online learning only operates on a single example at a time.

A lot of tasks that use online learning have a particular structure that can be broken down into 2 major components: 1 - Learning a model from data, and 2 - making predictions according to the model. Going back to the web search engine scenario as an example: the system needs to make predictions for every user doing a query - and must also learn from the feedback given by those users.

This structure comes with multiple challenges. First, the the amount of data is always growing, so archiving it comes at a cost, both because of storage constraints and computation constraints. A solution to this is to just keep the current model in memory, and archive the rest in the background. A second challenge is the variable speed at which data streams in. Imagine a learning problem where in the learning dataset is a live twitter stream for a hashtag. In this scenario the rate

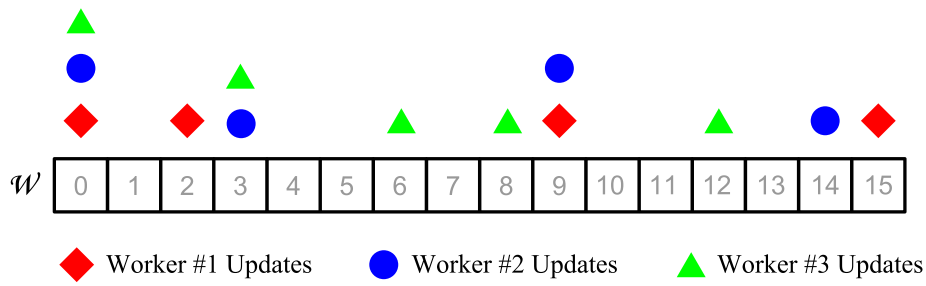

Figure 1: When multiple workers are simultaneously updating common parameters in the parameter server, write-write conflicts will arise. This diagram illustrates three sets of sparse updates have been submitted. How these updates are combined depends on the consistency algorithm used by the parameter server.

at which the data comes in is a function of the popularity of the hashtag. Finally, different applications have different costs for learning, and different requirements for the latency of predictions.

The problem we tackled in this project is the problem of automatically handling the resources needed for online learning. The ideal system would allocate the resources necessary to keep the prediction latency acceptable, while at the same time learning appropriately. Finally, the system would be able to handle bursts (such as increase in demand) and different learning requirements for different systems. Since online learning in a distributed system is a research problem on its [4, 5], we abstracted this part from our work, and focused on some of the systems challenges.

Current works in auto-scaling[6, 7] address many of the issues we address in this report (such as load prediction, framing the problem as a cost minimization problem, etc). However, one aspect that we thought was lacking in current work (at least from the papers we read) is dealing with node failures and uncertainty. We reformulate the cost function in terms of expected cost, in order to account for uncertainty pertaining future load predictions and node failures. Finally, we evaluate some baselines and our proposed approach with simulated loads.

References

[1] Antoine Bordes and Léon Bottou. The huller: A simple and efficient online svm. In *Proceedings of the 16th European Conference on Machine Learning*, ECML'05, pages 505–512, Berlin, Heidelberg, 2005. Springer-Verlag.

[2] Vitor R. Carvalho and William W. Cohen. Single-pass online learning: Performance, voting schemes and online feature selection. In *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '06, pages 548–553, New York, NY, USA, 2006. ACM.

[3] Mark Dredze, Koby Crammer, and Fernando Pereira. Confidence-weighted linear classification. In *Proceedings of the 25th International Conference on Machine Learning*, ICML '08, pages 264–271, New York, NY, USA, 2008. ACM.

[4] Qirong Ho, James Cipar, Henggang Cui, Seunghak Lee, Jin Kyu Kim, Phillip B. Gibbons, Garth A. Gibson, Gregory R. Ganger, and Eric P. Xing. More effective distributed ml via a stale synchronous parallel parameter server. In *NIPS workshop on BigLearning*, pages 1223–1231, 2013.

[5] Mu Li, Li Zhou, Z Yang, A Li, Fei Xia, D Andersen, and A Smola. Parameter server for distributed machine learning. In *NIPS workshop on BigLearning*, 2013.

[6] Ming Mao and Marty Humphrey. Auto-scaling to minimize cost and meet application deadlines in cloud workflows. In *Proceedings of 2011 International Conference for High Performance Computing, Networking, Storage and Analysis*, SC '11, pages 49:1–49:12, New York, NY, USA, 2011. ACM.

[7] N. Roy, A. Dubey, and A. Gokhale. Efficient autoscaling in the cloud using predictive models for workload forecasting. In *Cloud Computing (CLOUD), 2011 IEEE International Conference on*, pages 500–507, July 2011.