Auto Scaling Online Learning

CSE 550 Project Proposal

Marco Tulio Correia Ribeiro, Shrainik Jain
1323300, 1323338

1 Background and Prior work

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 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.

2 Proposed work

The problem we will tackle 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.

Most of the current auto-scaling solutions handle services that are easily replicated.

Some examples include Amazon Web Services[1], which has an option for auto-scaling, but that entails just spawning a new server with the same app or the Google AppEngine, which scales appropriately depending on the number of user requests. The assumption of easy replication is not true for machine learning scenarios. Learning in a distributed scenario is a problem on its own [4, 5]. Also, the system is two-tiered: there may be a case where more learners are needed, and there may be a case where more predictors are needed.

Our goal is to implement intelligent auto scaling while maintaining some prediction service SOA, and some measurement of learning, with the minimum cost possible. We plan on implementing a smart load-balancer that also handles the spawning of new nodes, but we also don't want this load-balancer to be a bottleneck, or a liability. Our current plan is to make it be a set of nodes. We won't focus on the machine learning side of it, so we plan on using an out-of-the-box fast machine learning solver: Vowpal Wabbit [2]. The challenges we see at the moment are:

- Determining when to start or shutdown more machines.
- Determining the role the new machines should play, and making sure that new predictors have a current model that is not too stale.
- Ensuring that the system is fault taulerant.

3 Roadmap

Project Milestone:

- This is a fairly new problem and we would devote some initial time towards looking at existing similar solutions and formalizing the problem we are trying to solve.
- Propose an initial design for the auto scaling service.

Short Presentation:

- Setting up (possibly simulated) environment and implement a first version of our auto scaling system.

Final Report:

- Detailed design specifications.
- Learning and results from the implementation (possibly simulated)

[1] http://aws.amazon.com/
[2] http://hunch.net/ vw/

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.