Jubatus: An Open Source Platform for Distributed Online Machine Learning

Shohei Hido **Seiya Tokui**
Preferred Infrastructure Inc.
Tokyo, Japan
{hido, tokui}@preferred.jp

Satoshi Oda
NTT Software Innovation Center
Tokyo, Japan
oda.satoshi@lab.ntt.co.jp

Abstract

Distributed computing is essential for handling very large datasets. Online learning is also promising for learning from rapid data streams. However, it is still an unresolved problem how to combine them for scalable learning and prediction on big data streams. We propose a general computational framework called loose model sharing for online and distributed machine learning. The key is to share only models rather than data between distributed servers. We also introduce Jubatus, an open source software platform based on the framework. Finally, we describe the details of implementing classifier and nearest neighbor algorithms, and discuss our experimental evaluations.

1 Introduction

The demand for learning from big data is increasing in many industries. However, traditional algorithms in machine learning generally do not scale to big data. The main difficulty lies with their memory constraint. Although algorithms typically assume that training data samples exist in main memory, big data does not fit into it. A common approach to learning from a large dataset is data distribution. By replacing batch training on the original training dataset with separated computations on the distributed subsets, one can train an alternative prediction model at a sacrifice of accuracy. For example, PSVM [3] uses approximated matrix factorization on each machine to efficiently parallelize SVM. Another way is to use online learning, in which memory usage does not depend on the size of the dataset. A number of studies have been done on online learning for large datasets. The traditional perceptron also belongs to this category of algorithm, and many variants exist.

However, neither online learning nor distributed learning is sufficient for learning from big data streams, in which thousands of samples arrive each second. There are two reasons for this. First is that the data size is too large to be relaxed by either online or distributed learning. Sequential online learning on big data requires too much time for training on a single machine. On the other hand, distributed learning with a large number of machines reduces the gained efficiency per machine and affects the overall performance. The second reason is that combining real-time training and prediction has not been studied, even though it will be the key in many applications of big data. Since big data is typically used after being stored in (distributed) storage, the learning process also tends to work in a batch manner. The necessity for online learning from big data streams, however, is increasing in applications such as IoT/M2M sensor analytics, network security intelligence, and real-time advertisement optimization. We therefore have to combine two approaches to achieve better performance, higher scalability, and shorter training time. Classifiers based on stochastic gradient descent are good examples of such algorithm, but the computational framework is not applicable to other tasks such as recommendation or anomaly detection.

Previous studies on big learning have mainly focused on reducing the training cost when given a fixed dataset. Once the training has finished or converged, predictions to new samples have been

assumed to be an easy process. However, prediction can be computationally intensive if thousands of new samples are coming each second. Concept drift can also occur during the training on big data streams so the prediction models have to be updated to reflect the changes. Therefore, we need to make prediction more scalable for big data streams in parallel with the continuous training process.

In this paper, we propose a general computational framework named loose model sharing for online distributed learning and prediction. The framework can support many machine learning tasks including classification and others. Then we introduce some implementations on an OSS that depends on this framework and discuss our experimental evaluation.

2 UPDATE-MIX-ANALYZE

We assume that data samples are continuously generated from one or more sources of data streams. There are two types of queries, one for training (updating) machine learning models, and one for making predictions using the models. A model can be a linear classifier, recommender, or outlier detection model, and it has a kind of computational structure and a set of parameter values. For simplicity, we do not distinguish between supervised and unsupervised learning. Because we cannot store all of the data samples on the streams, in-memory and data-size-independent processing are necessary. Our goal is to propose a computational framework as a baseline for implementing online and distributed algorithms for many machine learning tasks.

The most naive approach is simply to divide the data streams into a set of substreams for distributed servers and to build and use local models on each server. In this case, the performance of the local models improves slowly due to a lack of training samples. Another approach is to share training data between the servers to use more training samples. However, this leads to intensive communication among them, which degrades the scalability. To tackle these problems, we propose a framework called loose model sharing. The learning and prediction process can be divided into the following three operations (UPDATE-MIX-ANALYZE).

UPDATE. UPDATE corresponds to the training phase of machine learning algorithm. Each server is assigned a local model. A given training sample in the system will be sent to a randomly selected server or to specific servers based on consistent hashing. By using the sample, the corresponding server updates its model in an online learning manner.

MIX. MIX is the key of this framework. MIX refers to sharing not data but models between the servers. At the beginning of each MIX operation, a server is randomly selected as a parent. The parent server communicates with the rest of the servers to collect their models. Then the parent merges the models into one model, sends it back to the others, and terminates the MIX operation. Although right after this process, all of the servers have the same model, they soon start individually updating their models[1].

ANALYZE. The ANALYZE operation is for prediction. A given test sample will also be sent to one server. The server applies its current local model and outputs a prediction result for the sample. Although most distributed machine learning algorithms use an ensemble approach, which sends the test sample to and combines predictions from all of the servers, our ANALYZE works locally for scalable prediction.

There are two advantages in this simple loose model sharing. The first one is that models are typically much smaller than the size of the training samples used to train them, so the communication cost in model sharing is much lower than that in data sharing[2]. The second advantage is that training and prediction are linearly scalable with respect to the number of servers thanks to the completely local operations, UPDATE and ANALYZE. Though only MIX is a global operation between servers, in which the cost also increases linearly, we can control the trade-off between the cost and the performance improvement by MIX by changing the frequency of MIX operations. We do not guarantee the consistency of predictions, as it depends on the choice of server for ANALYZE since each server has a different local model. Note that we can also use alternative strategies for MIX based on preference. For example, a one-to-one MIX between randomly selected pairs of servers works with higher

[1]Note that the latest model updated during the MIX operation is discarded.

[2]By only sending the model difference from the last shared model instead of the current model itself, the communication tends to be more efficient.

scalability, in a manner similar to Gossip learning [21]. In this case, the cost of each MIX operation is independent from the number of servers, at a sacrifice of the consistency and fast convergence of the models.

3 Classification and nearest neighbor implementations

In this section, we briefly introduce Jubatus as an analytics platform based on the loose model sharing framework. Jubatus is open source software [12] developed by Preferred Infrastructure, Inc. and NTT SIC. Along with standard client-server architecture, clients make UPDATE and ANALYZE queries to servers. Server processes consist of two parts, feature vector preprocessing modules (fv_converter) and algorithm modules. The algorithm modules support most machine learning tasks, including classification, regression, recommendation, clusering anomaly detection, simple statistics, and graph analysis. Because there have only been a few online and distributed algorithms for these tasks, we basically modify existing online algorithms to make them work in a distributed environment using MIX operations. We briefly describe how a classification module (jubaclassifier) and a nearest neighbor module (jubanearest_neighbor) are implemented [3].

Jubaclassifier. Jubatus supports only linear models for classification. Currently, it supports perceptron [23], Passive-Aggressive (PA) [4], Confidence Weighted (CW) [5], AROW [6], and NHERD [7]. Although the details have been omitted due to space limitations, the differences between these models are only in their equations for updating the coefficients of the linear model. A Jubaclassifier model includes a list of coefficients, and their deviations if needed. On each jubaclassifier server, the ANALYZE operation is done simply to apply the linear model to the feature vector of the test sample. The UPDATE operation also works the same way as in the original online learning algorithm: given a training sample with a true label, it predicts its label with the current model, then modifies the model coefficients using the update equation of the used algorithm if the prediction is wrong. In the MIX operation, the locally updated models are collected in the parent server. The parent calculates the unweighted average of the coefficients and deviations as a single unified model. Because the unified model is represented as well as each local model, it can replace them after the MIX. By repeating MIX operations, the linear models on the servers are continuously updated to reflect the recent decision boundaries, with scalable update and ANALYZE. For multi-class classification, Jubaclassifier can also use one-vs-rest predictions based on multiple binary classifiers.

Jubaclassifier can be regarded as a variant of online linear classification algorithms based on Iterative Parameter Mixture (IPM) [19]. For conditional max-entropy models, Mann et al. showed that the parameter mixture approach is comparable in performance and much more network-efficient than the distributed gradient approach. They also proved that the parameter mixture has a good convergence bound. Hall et al. conducted large-scale experiments with click-through data on a cluster environment at Google [9]. The IPM-based perceptron worked better than distributed gradient and asynchronous stochastic gradient descent with much lower network usage. Although we do not yet have theoretical bounds for the IPM variants of PA, CW, AROW or NHERD, jubaclassifier worked better than the IPM-based perceptron in practice.

Jubanearest_neighbor. Jubatus supports a hash-based method for approximate nearest neighbor search (ANN) under UPDATE-MIX-ANALYZE. Locality-sensitive hashing (LSH) [8] is widely used for efficient NN, by using the hamming distance between two bit-arrays for approximating the cosine similarity between the original feature vectors. Similarly, b-bit minwise hashing [15] and Euclid LSH [17] can also be used to approximate the Jaccard coefficient and Euclidean distance, respectively. The point is to represent original high-dimensional feature vectors as a set of bit-arrays of a much smaller size. Given a data sample on a server, the UPDATE operation computes the hash values in the same way as the algorithm. The ANALYZE operation is also identical in computing the approximated similarity or distance and obtaining the nearest neighbors that are expected to have the highest similarity or smallest distance values in the original space. In this case, the model of jubanearest_neighbor is represented as a table where columns represent bit-arrays, and rows are samples. The purpose of the MIX operation is to enable servers to find nearest neighbors from all of the past data samples, some of which have not been shared yet. For efficiency, they send only the newly added samples by the UPDATE operations after the last MIX as the model difference to the parent server. The parent combines them into one difference table to be shared. With these

[3] Information on other tasks is on the official site [14]

operations, the addition of new samples and the nearest neighbor search are also linearly scalable with respect to the number of servers. Based on the nearest neighbor modules, recommendation (jubarecommender) and outlier detection (jubaanomaly) are implemented.

4 Scalability evaluations

Table 1 lists the query-per-second for jubaclassifier (PA) and jubanearest_neighbor (LSH) on the distributed environment when changing the number of servers from 1 to 8. The data set consists of 256-dimensional random samples. The PA classifier tries to predict random binary labels, and LSH is based on a 64-bit array.

Table 1: Throughput performance (queries per second) for artificial data set.

# Server	1	2	4	8
PA classifier	1243	1371	3759	6154
LSH nearest neighbor	174	294	415	678

This table indicates that the throughput for both tasks increases almost linearly with the number of servers. Note that we omit the results for ANALYZE since the prediction on server is already very fast (over 10,000 qps with a single server), network on the client becomes a bottleneck, and throughput does not increase much on our experimental environment with commodity machines. However, it is easily estimated that ANALYZE queries from thousands of clients should scale with a few servers, since ANALYZE is a completely local operation.

5 Activities and related work

The original concept of loose model sharing was first presented in conjunction with the first release of Jubatus 0.1.0 [20]. Since then, Jubatus community held some workshops and hands-on seminars in Japan and made some presentations at conferences around the world [10, 18, 11]. Hands-on tutorials and VM images are also available online [13] so that users can try it without having to do a complicated installation. The deployment of Jubatus in real-world applications is in progress; for example, Japanese largest online real-estate service uses Jubatus to estimate customer preferences [24].

Some software libraries and tools for machine learning on big data have been introduced over the last decade. Mahout is a set of algorithms for machine learning on Hadoop clusters [22]. Vowpal Wabbit supports many efficient algorithms for linear learning [1]. GraphLab is a general-purpose framework for distributed computation, of which the first target was graph-based machine learning algorithms [16]. Apache Spark [25] is also a popular cluster computing system, and MLlib supports classification, regression, clustering, and collaborative filtering [2]. Generally, they aim at learning from big data already stored in a large distributed storage. Although some of them are also capable of online training of the classification models, they are not fully integrated with distributed computation, and prediction is assumed to be a separated process. On the contrary, Jubatus focuses on real-time applications for big data streams, which naturally includes distributed online learning and prediction at the same time.

6 Conclusion

In this paper, we introduced loose model sharing as a general computation framework for distributed online machine learning. Jubatus, which is based on this framework, can achieve high throughput for both online training and prediction by separating the model mixture from local model operations. We are working on extending Jubatus capabilities including time-series analysis, learning with fixed memory size, and integration with computer vision. We would also like to work with academia to study the theoretical characteristics of distributed online algorithms implemented on Jubatus.

Acknowledgments

The authors would like to thank all of the developers, contributors, and supporters for the Jubatus project.

References

[1] A. Agarwal, O. Chapelle, M. Dudík, and J. Langford. A reliable effective terascale linear learning system. *CoRR*, abs/1110.4198, 2011.

[2] Apache Spark. Machine learning library (mllib). http://spark.incubator.apache.org/docs/latest/mllib-guide.html.

[3] E. Y. Chang, K. Zhu, H. Wang, H. Bai, J. Li, Z. Qiu, and H. Cui. Psvm: Parallelizing support vector machines on distributed computers. In *Advances in Neural Information Processing Systems*, 2007. Software available at http:/code.google.com/p/psvm/.

[4] K. Crammer, O. Dekel, J. Keshet, S. Shalev-Shwartz, and Y. Singer. Online passive-aggressive algorithms. *The Journal of Machine Learning Research*, 7:551–585, 2006.

[5] K. Crammer, M. Dredze, and F. Pereira. Confidence-weighted linear classification for text categorization. *Journal of Machine Learning Research*, 13:1891–1926, 2012.

[6] K. Crammer, A. Kulesza, and M. Dredze. Adaptive regularization of weight vectors. *Machine Learning*, 91(2):155–187, 2013.

[7] K. Crammer and D. D. Lee. Learning via gaussian herding. In *Advances in Neural Information Processing Systems*, pages 451–459, 2010.

[8] A. Gionis, P. Indyk, and R. Motwani. Similarity search in high dimensions via hashing. In *Proceedings of the 25th International Conference on Very Large Data Bases*, pages 518–529, 1999.

[9] K. Hall, S. Gilpin, and G. Mann. Mapreduce/bigtable for distributed optimization. In *NIPS 2010 Workshop on Learning on Cores, Clusters and Clouds*, 2010.

[10] S. Hido. Distributed online machine learning framework for big data. Presented at the 1st eXtremely Large Database Conference at Asia, Beijing, China, June 2012.

[11] S. Hido. Jubatus: Real-time and highly-scalable machine learning platform. Presented at Hadoop Summit 2013, San Jose, June 2013.

[12] Jubatus Development Team. Jubatus Git repository on Github. https:/github.com/jubatus/jubatus/.

[13] Jubatus Development Team. Jubatus hands-on tutorial. http://download.jubat.us/event/handson_01/en/.

[14] Jubatus Development Team. Jubatus official web site. http:/jubat.us/en/.

[15] P. Li and A. C. König. b-bit minwise hashing. In *Proceedings of the 19th International Conference on World Wide Web*, pages 671–680, 2010.

[16] Y. Low, J. Gonzalez, A. Kyrola, D. Bickson, C. Guestrin, and J. M. Hellerstein. Graphlab: A new framework for parallel machine learning. In *UAI*, pages 340–349. AUAI Press, 2010.

[17] Q. Lv, W. Josephson, Z. Wang, M. Charikar, and K. Li. Multi-probe lsh: Efficient indexing for high-dimensional similarity search . In *Proceedings of the 33th International Conference on Very Large Data Bases*, pages 950–961, 2007.

[18] H. Makino. Jubatus: Distributed online machine learning framework for realtime analysis of big data. Presented at the 6th eXtremely Large Databases Conference, Stanford, Sept. 2012.

[19] G. Mann, R. T. McDonald, M. Mohri, N. Silberman, and D. Walker. Efficient large-scale distributed training of conditional maximum entropy models. In *Advances in Neural Information Processing Systems*, pages 1231–1239, 2009.

[20] D. Okanohara, Y. Unno, U. Kota, S. Oda, S. Nakayama, and H. Tanaka. An overview and roadmap of Jubatus. Presented at Jubatus Workshop Tokyo (in Japanese), Nov. 2011. http://www.zusaar.com/event/165003.

[21] R. Ormándi, I. Hegedüs, and M. Jelasity. Gossip learning with linear models on fully distributed data. *Concurrency and Computation: Practice and Experience*, 25(4):556–571, 2013.

[22] S. Owen, R. Anil, T. Dunning, and E. Friedman. *Mahout in Action*. Manning Publications Co., 2011.

[23] F. Rosenblatt. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, 65, 1958.

[24] T. Shimogaki. A use case of online machine learning using Jubatus. Presented at GOTO International Conference on Software Development at Berlin, Oct. 2013.

[25] M. Zaharia. Spark: In-memory cluster computing for iterative and interactive applications. Invited talk at NIPS Workshop on Big Learning, Dec. 2011.