Learning with Less: Can Approximate Storage Systems Save Learning From Drowning in Data?

Nitin Agrawal
Samsung Research

Ashish Vulimiri
Samsung Research

Abstract

Data empowers learning. But soon, we may have too much of it to store, process, and analyze in a timely and cost-effective manner. We take the position that approximate storage systems have a role to play in alleviating this problem. The paper is intended to generate discussion on the merits and pitfalls of data approximation, its applicability, and lack thereof, to a variety of learning algorithms, and its broader appeal to AI. Tackling the challenges of large-scale data analysis requires not only expertise in systems, but also in machine learning, statistics, and algorithms. The paper borrows from the lessons the authors learnt in building SummaryStore [4], an approximate storage system capable of storing large streams of time–series data (~100 Terabyte on a single node), while preserving high degrees of accuracy and real-time querying at unprecedented cost savings.

1 Introduction

Data has become a societal utility. Applications previously unimaginable for computers to perform are becoming a reality through fundamental innovations in natural language processing, computer vision, voice agents, information retrieval, and search, among many others. Large-scale data processing lies at the core for many of these advances. It is also interesting to note that a significant fraction of the data is being generated *and* consumed exclusively by machines; a human reader is rarely involved. Generation is increasingly being dominated by sensors [5], wearables and personal computing devices [27], and data centers [26]; consumption is driven by algorithms for machine learning, analytics, and AI at large [9, 17, 18, 22, 25, 28].

From a systems perspective, storing and accessing large volumes of data has been a long-standing challenge but one that the community has a firm grasp on and has developed effective solutions for [8, 15]. Often, scaling-out storage without compromising on the required semantics comes at a cost.

From a learning perspective, for large-scale data processing, in addition to cost, the ability to perform computational tasks in a timely manner is particularly relevant [23].

AISys, 2017, Shanghai
2017. ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

A principle methodology to speed-up such data-centric computational tasks is to exploit their inherent parallelism but, in spite of significant advances [1, 12, 16, 30], this approach is not universally applicable; several learning processes are fundamentally hard to parallelize [21, 24]. Learning algorithms that operate on very large datasets often struggle to keep up with the growth in data [11, 12]. The paper raises the following motivating question: *can approximate storage systems help alleviate the challenges of large-scale data analysis for learning?*

An approximate storage system maintains a compact representation of the raw data that it is entrusted to store. A number of approximate stores have been built [2–4, 10, 19, 20] each offering a different trade-off between storage consumption, access latency, and query accuracy. Several of these stores, such as BlinkDB [3], employ sampling as the basis for approximation [2, 3, 19]; others, such as Aperture [10] employ windowed aggregation; SummaryStore [4] proposes time-decayed summaries as the primary abstraction for approximation; it maintains summaries that favor recent data over older data and continually *decay* over time. The majority of these approximate stores have been designed with big-data analytics, search, forecasting, outlier detection, and time-series analysis as target applications.

Co-designing the lower-layer storage system in light of the requirements of the higher-layer *learning* tasks offers the potential of cost-effective low-latency data processing. As an example, active disks [31] pioneered an early approach to allow storage drives to execute application-level functions directly in the device; applied in the context of large-scale data mining, active disks provided reduced data traffic and increased throughput by pushing some of the "intelligence" down to the storage system. In the current context, the intelligence pushed down to the storage can form the basis for the data approximation.

SummaryStore maintains compact summaries through aggregates, samples, sketches, and other probabilistic data structures in lieu of raw data; its time-decayed scheme favors recent data by allocating progressively fewer bytes for older data thereby increasing the extent of approximation with age. Initial results for analytics queries and machine-learning forecasting with SummaryStore are promising [4]. In particular, Facebook's Prophet forecasting engine [33] yielded nearly the same forecast accuracy using SummaryStore for

10x compaction on three real-world datasets from economics, climatology, and Internet traffic relative to a baseline with no approximation. Outlier detection using SummaryStore for Google's cluster management dataset [35] required about 6x less space for negligibly-low false positives. Under microbenchmarks, a 100 Terabyte synthetic stream was stored using only a Terabyte of disk space with 95%-ile error below 5% for a variety of queries.

2 Opportunities and Challenges

We present potential avenues for exploration and solicit feedback from systems and machine-learning practitioners; this list is by no means comprehensive.

Improve completion times of training tasks. Compacting the input dataset fundamentally "improves" the learning outcome by enabling the computations to complete faster and more tasks to be performed in a given time duration. Learning a new model is often an iterative process. It is not uncommon for a machine learning expert to train tens to hundreds of models before converging on an acceptable one. Each iteration takes time and computing resources. If each iteration, with the requisite tweaks to model parameters, can be performed on an approximate subset of the input dataset, the end-to-end process can be substantially sped up.

Reducing the I/O and memory footprint also helps in a more balanced architecture. By reducing the memory requirements, more of the learning can be accomplished on a high-performance single node before making the expensive transition to a distributed system. GPUs can further help algorithms saturate single-node performance and finish quicker. Of course, the scientific challenge is to ensure that the approximation is representative and the trained model does not deviate too much from the one learnt on the entire dataset.

Enable learning on the edge. By reducing the compute and I/O requirements, learning can be more efficiently and frequently performed on resource-constrained "edge" devices. While the cloud might continue to do the heavy lifting when it comes to data processing and learning, the edge has an increasingly greater role to play. Since data being generated at the edge is growing faster than the network connectivity to the very same edge devices, transferring all the data to the cloud can become cost-prohibitive. Privacy concerns make learning at the edge even more desirable; approximate data storage systems naturally align with techniques for privacy-preserving querying by using aggregates and summaries instead of individual values [29].

Build more robust models. Various learning techniques have explored algorithmic approximation as the source for increased model robustness. Dropout [32] randomly drops neural network units and their connections from an exponential number of "approximate" networks to prevent overfitting; this controlled compaction introduces positive attributes like the system learning more robust features. Stochastic gradient descent samples a subset of (summand) functions at every step to reduce the computational footprint of every iteration making it more effective for large-scale machine learning problems [7] compared to gradient descent. SummaryStore demonstrated benefits of its approximation scheme through Prophet's forecasting evaluation; for the Federal Reserve's economic dataset [13], it exhibited a net reduction in forecast error with approximate data. For the workload, the decay diminished the effect of older outliers in the stream and positively influenced the outcomes.

The selection of an approximation scheme, for different workloads and datasets, can be a challenge. Sampling is popular, and effective, but suffers from several limitations. Uniform sampling can unintentionally discard events of significant interest to the learning algorithm. More sophisticated sampling methods, *e.g.*, stratified [34], take greater care to include rarer subgroups relative to the uniform random sample. Beyond sampling, other approximation techniques can include algorithms for aggregation, statistical modeling, and temporal decay. Aggregation can narrow down the scope of the queries while statistical modeling can introduce additional errors in modeling; furthermore, rich statistical models, *e.g.*, ARMA, can be fairly expensive to maintain [14] contradicting the intent for approximation. Temporal decay can be effective but does not necessarily apply to all workloads.

The tolerance of different learning algorithms to incompleteness in the data will ultimately dictate the suitability of approximation schemes; building general-purpose solutions will be challenging both from systems and algorithmic viewpoints. Approximation will also introduce errors in data accesses and any scheme will need to provide robust quantification of errors; in some cases, this may be impossible. The learning algorithms must also deal with the increased uncertainty; perhaps error will need to be treated as a first-class programming construct [6].

The premise that an abundance of data is putting an undesirable strain on compute and I/O resources for learning algorithms is at odds with the desire to produce richer, more accurate, models. As an example, the resurgence in deep learning is in large measures driven by the ability to train sophisticated models over huge amounts of data. We need to understand if (data) approximation and (model) accuracy can go hand in hand and have outlined a few challenges and avenues for future exploration. Broader applicability of SummaryStore, and approximate storage systems in general, for a variety of machine-learning algorithms needs further investigation and discourse.

References

[1] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, et al. Tensorflow: A system for large-scale machine learning. In *OSDI*, volume 16, pages 265–283, 2016.

[2] R. Agarwal, A. Khandelwal, and I. Stoica. Succinct: Enabling queries on compressed data. In *12th USENIX Symposium on Networked Systems Design and Implementation (NSDI 15)*, pages 337–350, 2015.

[3] S. Agarwal, B. Mozafari, A. Panda, H. Milner, S. Madden, and I. Stoica. Blinkdb: queries with bounded errors and bounded response times on very large data. In *Proceedings of the 8th ACM European Conference on Computer Systems*, pages 29–42. ACM, 2013.

[4] N. Agrawal and A. Vulimiri. Low-Latency Analytics on Colossal Data Streams with SummaryStore. In *Proceedings of the 26th ACM Symposium on Operating Systems Principles (SOSP '17)*, Shanghai, China, October 2017. ACM.

[5] M. P. Andersen and D. E. Culler. Btrdb: optimizing storage system design for timeseries processing. In *14th USENIX Conference on File and Storage Technologies (FAST 16)*, pages 39–52, 2016.

[6] J. Bornholt, T. Mytkowicz, and K. S. McKinley. Uncertain< t>: A first-order type for uncertain data. *ACM SIGPLAN Notices*, 49(4):51–66, 2014.

[7] L. Bottou. Large-scale machine learning with stochastic gradient descent. In *Proceedings of COMPSTAT'2010*, pages 177–186. Springer, 2010.

[8] F. Chang, J. Dean, S. Ghemawat, W. C. Hsieh, D. A. Wallach, M. Burrows, T. Chandra, A. Fikes, and R. Gruber. Bigtable: A Distributed Storage System for Structured Data. In *OSDI '06*, pages 205–218, Seattle, WA, Nov. 2006.

[9] Cory Watson. Observability at Twitter. https://blog.twitter.com/2013/observability-at-twitter, 2013.

[10] H. Cui, K. Keeton, I. Roy, K. Viswanathan, and G. R. Ganger. Using data transformations for low-latency time series analysis. In *Proceedings of the Sixth ACM Symposium on Cloud Computing*, pages 395–407. ACM, 2015.

[11] J. Dean. Large scale deep learning. In *Keynote GPU Technical Conference*, volume 3, page 2015, 2015.

[12] J. Dean, G. Corrado, R. Monga, K. Chen, M. Devin, M. Mao, A. Senior, P. Tucker, K. Yang, Q. V. Le, et al. Large scale distributed deep networks. In *Advances in neural information processing systems*, pages 1223–1231, 2012.

[13] Federal Reserve Economic Data. https://en.wikipedia.org/wiki/Federal_Reserve_Economic_Data, 2017.

[14] D. Freedman, R. Pisani, and R. Purves. *Statistics*. W. W. Norton and Company Inc, New York, NY, first edition, 1978.

[15] S. Ghemawat, H. Gobioff, and S.-T. Leung. The Google File System. In *SOSP '03*, pages 29–43, Bolton Landing, NY, Oct. 2003.

[16] K. Hsieh, A. Harlap, N. Vijaykumar, D. Konomis, G. R. Ganger, P. B. Gibbons, and O. Mutlu. Gaia: Geo-distributed machine learning approaching lan speeds. In *NSDI '17*, 2017.

[17] K. J. Jacob and D. Shasha. Fintime – a financial benchmark.

[18] Jamie Wilkinson. Google Prometheus: A practical guide to alerting at scale. https://docs.google.com/presentation/d/1X1rKozAUuF2MVc1YXElFWq9wkcWv3Axdldl8LOH9Vik/edit#slide=id.g598ef96a6_0_341, 2016.

[19] S. Kandula, A. Shanbhag, A. Vitorovic, M. Olma, R. Grandl, S. Chaudhuri, and B. Ding. Quickr: Lazily approximating complex adhoc queries in bigdata clusters. 2016.

[20] A. Khandelwal, R. Agarwal, and I. Stoica. Blowfish: dynamic storage-performance tradeoff in data stores. In *13th USENIX Symposium on Networked Systems Design and Implementation (NSDI 16)*, pages 485–500, 2016.

[21] J. K. Kim, Q. Ho, S. Lee, X. Zheng, W. Dai, G. A. Gibson, and E. P. Xing. Strads: a distributed framework for scheduled model parallel machine learning. In *Proceedings of the Eleventh European Conference on Computer Systems*, page 5. ACM, 2016.

[22] Y. Koren. Collaborative filtering with temporal dynamics. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 447–456. ACM, 2009.

[23] Y. LeCun and M. Ranzato. Deep learning tutorial. In *Tutorials in International Conference on Machine Learning (ICML'13)*. Citeseer, 2013.

[24] H. Li, A. Kadav, E. Kruus, and C. Ungureanu. Malt: distributed data-parallelism for existing ml applications. In *Proceedings of the Tenth European Conference on Computer Systems*, page 3. ACM, 2015.

[25] L. Li, W. Chu, J. Langford, and R. E. Schapire. A contextual-bandit approach to personalized news article recommendation. In *Proceedings of the 19th international conference on World wide web*, pages 661–670. ACM, 2010.

[26] Mike Keane. 1.5 Million Log Lines per Second. http://www.bigdataeverywhere.com/files/chicago/BDE-15MillionLogLinesPerSecond-KEANE.pdf, 2014.

[27] The Rise of Consumer Health Wearables: Promises and Barriers. https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4737495/.

[28] T. Pelkonen, S. Franklin, J. Teller, P. Cavallaro, Q. Huang, J. Meza, and K. Veeraraghavan. Gorilla: a fast, scalable, in-memory time series database. *Proceedings of the VLDB Endowment*, 8(12):1816–1827, 2015.

[29] R. A. Popa, C. Redfield, N. Zeldovich, and H. Balakrishnan. Cryptdb: protecting confidentiality with encrypted query processing. In *Proceedings of the Twenty-Third ACM Symposium on Operating Systems Principles*, pages 85–100. ACM, 2011.

[30] R. Raina, A. Madhavan, and A. Y. Ng. Large-scale deep unsupervised learning using graphics processors. In *Proceedings of the 26th annual international conference on machine learning*, pages 873–880. ACM, 2009.

[31] E. Riedel, C. Faloutsos, G. A. Gibson, and D. Nagle. Active disks for large-scale data processing. *Computer*, 34(6):68–74, 2001.

[32] N. Srivastava, G. E. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *Journal of machine learning research*, 15(1):1929–1958, 2014.

[33] S. J. Taylor and B. Letham. Facebook open source project: Forecasting at scale. https://github.com/facebookincubator/prophet, 2017.

[34] Wikipedia. Stratified Sampling. https://en.wikipedia.org/wiki/Stratified_sampling.

[35] J. Wilkes. More Google cluster data. Google research blog, Nov. 2011. Posted at http://googleresearch.blogspot.com/2011/11/more-google-cluster-data.html.