2017 16th IEEE International Conference on Machine Learning and Applications

Novel Trends in Scaling Up Machine Learning Algorithms

Noel Lopes
UDI, Polytechnic of Guarda, Portugal
CISUC, University of Coimbra, Portugal
Email: noel@ipg.pt

Bernardete Ribeiro
CISUC, Department of Informatics Engineering,
University of Coimbra, Portugal
Email: bribeiro@dei.uc.pt

***Abstract*—Big Data has been a catalyst force for the Machine Learning (ML) area, forcing us to rethink existing strategies in order to create innovative solutions that will push forward the field. This paper presents an overview of the strategies for using machine learning in Big Data with emphasis on the high-performance parallel implementations on many-core hardware. The rationale is to increase the practical applicability of ML implementations to large-scale data problems. The common underlying thread has been the recent progress in usability, cost effectiveness and diversity of parallel computing platforms, specifically, the Graphics Processing Units (GPUs), tailored for a broad set of data analysis and Machine Learning tasks. In this context, we provide the main outcomes of a GPU Machine Learning Library (GPUMLib) framework, which empowers researchers with the capacity to tackle larger and more complex problems, by using high-performance implementations of well-known ML algorithms. Moreover, we attempt to give insights on the future trends of Big Data Analytics and the challenges lying ahead.**

I. Introduction

Big Data realization has sparked numerous challenges in a wide range of areas. At present, Machine Learning (ML) is the best way to exploit the opportunities encompassed in Big Data. From a more general point of view, ML field embodies the potential to extract context-aware valuable information from the ubiquitous volumes of data, leveraging strategic and competitive advantages to organizations and offer a solution to improve most human life activities. However, the rise of Big Data also exposed the limitations of traditional CPU-based Machine Learning algorithms' implementations, since most tools fail to process large amounts of data in a reasonable time frame. Moreover, ML algorithms are typically designed with emphasis on effectiveness (*e.g.* classification performance) rather than on efficiency (*e.g.* time required to produce a classifier) [1].

Rationally, as ML problems become more complex and computationally demanding, the pressure to shift to high throughput parallel architectures is increased. Subsequently, the Graphics Processing Unit (GPU) represents a compelling and feasible solution to address the increasing needs of computational performance, due to its inherent high-parallelism. We have taken on this challenge and developed a high performance open-source GPU Machine Learning Library (GPUMLib), available at http://gpumlib.sourceforge.net/, which aims to empower the ML community with the tools needed to explore larger datasets, by tapping into the GPU enormous computational power. Since its release, GPUMLib has attracted the interest of numerous people, using a wide-range of platforms and benefited researchers worldwide.

In this paper, Section II presents the strategies for dealing with Big Data in the context of ML problems addressing data analysis (feature selection and instance selection), predictive analytics (incremental and batch learning) and distributed and high-throughput parallel implementations. Moreover, in Section III we address the GPU computing challenges and how they can be approached with scalability in mind and are one way of modern computing. We show in Section IV the advantages of using GPUMLib (an open source GPU Machine Learning Library) in practical scenarios as well as the distinct ways it can be successfully used in a varied number of problems. In Section V we describe novel trends in advanced analytics and unify them in a common structure. Finally, in Section VI we present final remarks and emphasize how important are GPU computing tools for the scientific community in general.

II. Strategies for Machine Learning Big Data

To cope with the increasingly challenging and computationally demanding problems, several strategies are being employed, as depicted in Figure 1. Since the volume of data to process has a significant impact (sometimes exponential) in the computational requirements of ML algorithms, an obvious approach consists of reducing the size and complexity of the dataset (*i.e.* the number of samples and/or features).

In order to reduce the number of samples, instance selection methods, which aim at obtaining a representative subset of the original training data, can be used. The challenge is to identify which samples (instances) are more relevant (useful) for the learning process. López et al. [2] presented a review of instance selection methods dividing them into wrapper and filter methods, depending respectively on whether the selection criterion is based on the accuracy of a classifier or not. Although instance selection methods can effectively reduce the volume of data, their application may be time consuming (in particular for wrapper methods) and in some situations we may be simply transferring the complexity from the learning methods to the instance selection methods.

0-7695-6321-X/17/31.00 ©2017 IEEE
DOI 10.1109/ICMLA.2017.00-90

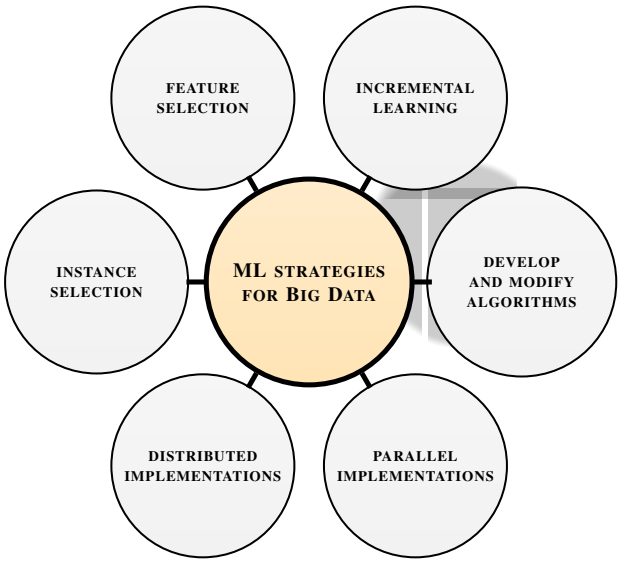

Fig. 1. Strategies for addressing large-scale datasets Machine Learning problems.

In addition to lowering computational complexity of algorithms, reducing the number of features has also the potential benefit of improving the generalization performance of ML models. Reducing the feature space can be accomplished by using dimensionality reduction techniques, which include both feature extraction and feature selection approaches [3]. Feature extraction techniques (*e.g.* Principal Component Analysis (PCA), Linear Discriminant Analysis (LDA)) project the original features into a new lower dimensional space, whose features are usually generated by combinations of the original ones. A review of those can be found in van der Maaten et al. [4]. Unfortunately, these methods are computationally demanding and therefore its application to large datasets may not be feasible when scaling up ML algorithms. Feature selection techniques (*e.g.* Information Gain, Fisher Score) have a different approach: they attempt to identify highly-discriminant features that are expected to be more useful to the learning process while minimizing redundancy. Likewise instance selection methods, feature selection methods can also be divided into wrapper and filter methods. As in the case of instance selection, wrapper methods (based on a classifier accuracy) should be avoided when tackling large-scale problems, since they are prohibitively expensive to run on sizable datasets [3].

A different strategy for dealing with Big Data consists of using incremental learning algorithms. These are designed to rapidly update their models, in order to incorporate new information on a sample-by-sample basis and therefore have the potential to handle also data streams and concept drifts. Moreover, incremental algorithms can be used for stream learning, since they typically, run in resource-aware environments, constructing decision models that are continuously evolving and tracking changes in the environment generating the data [5]. An example of such algorithm is the Incremental Hypersphere Classifier (IHC) [6], which yields good classification performance results and can also be used as an instance selection method with proven results [7], [8].

Naturally, developing new batch and incremental algorithms and modifying existing ones, in order to expedite the learning process, also plays an important role in scaling up ML algorithms. For existing algorithms, the driven motivation can be at least one of the following: to speed up the algorithm's convergence; to allow more parallelizable implementations; to make the algorithm easier to distribute through the available computational resources. One algorithm that combines all these three facets, while reducing the risk of creating unfitted/inadequate models is the Semi-Supervised NMF (SSNMF) [9], which is based on the Non-Negative Matrix Factorization (NMF) [10].

Finally, high-throughput parallel and distributed ML implementations will be crucial for scaling up algorithms and facing the more challenging and demanding problems that are yet to come. In this context, novel computational platforms both in terms of hardware and software will hold the key to lift up the potential of ML and Big Data Analytics, among other areas. In this scenario, a promising architecture consists of using the GPU for general purpose programming.

Ultimately, no single strategy will solve on its own this problematic as Big Data keeps on getting bigger and successful applications will most likely require a combination of these approaches.

III. Many-Core GPU Computing

Over the last decade the performance and capabilities of the GPUs have been significantly augmented and today's GPUs, included in mainstream computing systems, are powerful, massively parallel and general-purpose programmable devices [11], [12]. The peak performance of GPUs is over one order of magnitude larger than the corresponding performance of modern Central Processing Units (CPUs) and trend is for this gap to continue to increase in the future [12]. This aspect is depicted in Figure 2, updated from Owens et al. [13].[1]

In addition, GPUs are widely available and relatively inexpensive, since they are mass produced and regularly replaced by newer generations with additional levels of programmability and increasing computational power [14], [15], [16].

Applications and algorithms that present a high-degree of parallelism and large computational requirements can benefit the most from GPU implementations. It is not uncommon for these to obtain speedups of one or even two-orders magnitude, as compared with the corresponding CPU implementations. For example, weeks of processing on the CPU may be transformed into hours on the GPU [17]. Recognizing the potential of General-Purpose computing on Graphics Processing Units (GPGPU), hardware manufacturers saw the opportunity for a new market and developed novel platforms for simplifying the development process in this devices. Subsequently, the

[1]Figure 2 is a courtesy of Professor John Owens, from the University of California, Davis, USA.

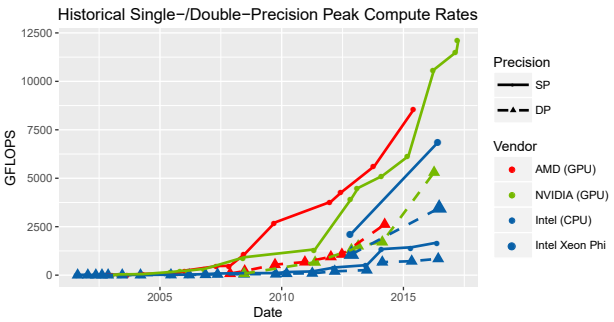

Fig. 2. Peak floating point performance disparity between the CPU and the GPU, over the years, in billions (10^9) of floating-point operations per second (GFLOPS).

NVIDIA Compute Unified Device Architecture (CUDA), rapidly gain wide adoption and become the elected platform of the scientific community for solving numerous problems that could not otherwise be solved in a realistic time frame, using the CPU. With CUDA, instead of using graphics Application Programming Interfaces (APIs), we can take advantage of industry-standard languages, such as C++, coupled with specific extensions to express GPU parallel computations. The CUDA architecture exposes the GPU as a massive-parallel device that operates as a co-processor to the host (CPU) as depicted in Figure 3. The rationale is to offload data parallel workloads to the GPU, where analogous operations are executed over large quantities of data. Hence, the challenge is to break down the original workload tasks into independent processing blocks that can be carried out in parallel in the thousands of cores available in a GPU, with occupancy and scalability in mind.

The fundamental principles to identify the algorithms that can benefit the most from a CUDA implementation are highlighted in Ryoo et al. [18]. Among other things, a very large number of threads (thousands or even millions, depending on the problem) are required to hide the global memory latency. Moreover, they must be grouped appropriately to avoid memory conflicts and non-sequential memory accesses, which may downgrade significantly the overall performance. Additionally, the adequate use of on-chip (shared memory) to reduce bandwidth usage and redundant execution is extremely important to speedup computations [12].

IV. Scaling Up Machine Learning using GPUMLib

Although most ML algorithms match the requirements for obtaining significant speedups from a GPU parallel implementation, mapping algorithms to the GPU is far from being an easy task. Moreover, being an excellent ML researcher does not necessary imply being an excellent programmer [19]. Therefore we have taken on the challenge of developing GPU parallel implementations of ML algorithms by developing an open-source GPU Machine Learning Library (GPUMLib), available at http://gpumlib.sourceforge.net/. GPUMLib aims at empowering the ML community with a high-performance

TABLE I
Machine Learning (ML) GPU implementations' speedups (GPUMLib).

Algorithm	CPU	GPU	Speedups (≈)
BP	Intel Core 2 Quad (2.50GHz)	GTX 280 (240 cores)	28× to 175×
MBP	Intel Core 2 Quad (2.50GHz)	GTX 280 (240 cores)	33× to 179×
NMF and SSNMF	Intel Core 2 Quad (2.50GHz)	GTX 280 (240 cores)	56× to 707×
SVM	Intel Quad Core i5 (3.33GHz)	GTX 570 (480 cores)	14× to 165×
RBMs and DBNs	Intel Dual Core i5 (2.70GHz)	GTX 460 (336 cores)	22× to 465×

library that they can use to build real-world applications. Moreover, by making the library open-source we aim at promoting cooperation among researchers since they can build, share and improve on top of existing resources rather than having to re-implementing them. This assumes particular relevance, because the speed at which a given scientific field advances depends on how well researchers collaborate with one another [20].

Currently, GPUMLib implements the following ML algorithms: Back Propagation (BP); Multiple Back Propagation (MBP); Neural Selective Input Model (NSIM), which allows Neural Networks (NNs) to handle missing data directly without any pre-processing (*e.g.* imputation); Autonomous Training System (ATS) for Neural Networks; Support Vector Machine (SVM); Radial Basis Function (RBF); NMF; SSNMF; Self-Organizing Maps (SOM) networks; Restricted Boltzmann Machines (RBMs); and Deep Belief Networks (DBNs). Many of these provide speedups of one or even two-orders magnitude. The speedup measures how many times faster is the GPU implementation in comparison with the corresponding CPU implementation (*e.g.* a speedup of 60× means the GPU is 60 times faster than the CPU and therefore is able to do in one minute the same amount of work that is performed by the CPU in one hour).

Table I presents the GPUMLib speedups, collected from Lopes and Ribeiro [12], that were obtained by some of the above mentioned algorithms. These results cover several distinct problem types, including benchmarks obtained from the UCI Machine Learning Repository [21], face recognition (*e.g.* ORL (http://www.cl.cam.ac.uk/research/dtg/attarchive/facedatabase.html), Yale (http://cvc.yale.edu/projects/yalefaces/yalefaces.html)) and hand-written digits/symbols (*e.g.* MNIST (http://yann.lecun.com/exdb/mnist/), HHreco (http://embedded.eecs.berkeley.edu/research/hhreco/)) domains, as well as real-world problems (*e.g.* ventricular arrhythmias detection, financial distress prediction).

Naturally, the speedups depend not only on the hardware used but also on the characteristics of the actual problem being handled. Nevertheless, as Figure 4 demonstrates, more complex problems (in this case with a larger number of samples, features and neurons, *i.e.* with a larger number of threads) will typically yield bigger GPU speedups, making GPU ML implementations more scalable.

Since its release, GPUMLib has attracted the interest of numerous people in the ML community, benefiting researchers

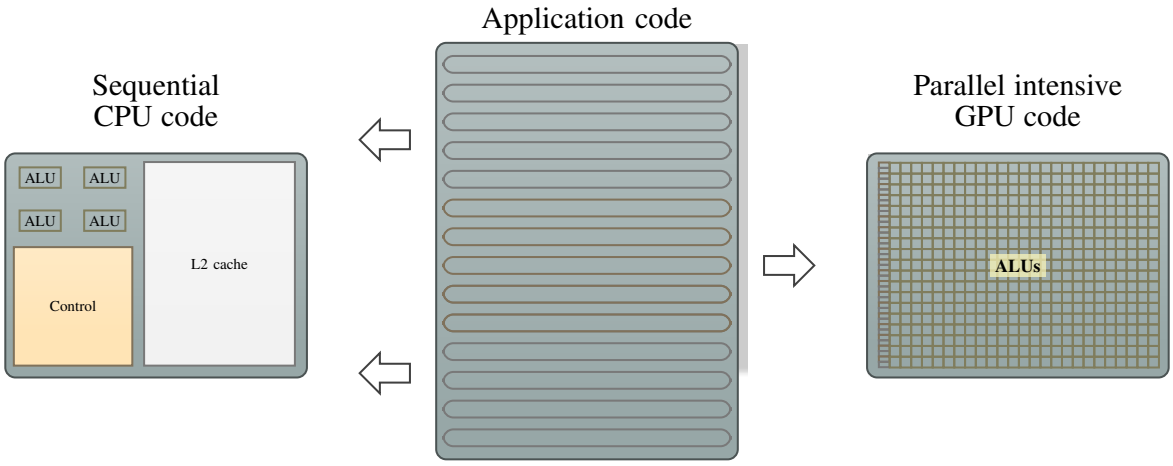

Fig. 3. Offloading data computations to the GPU.

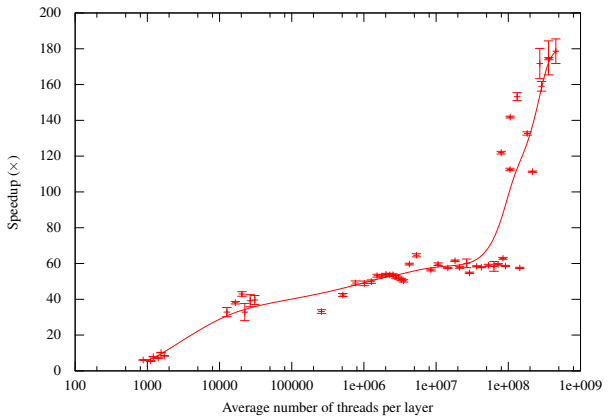

Fig. 4. BP and MBP GPU implementation speedups (×) versus NNs complexity.

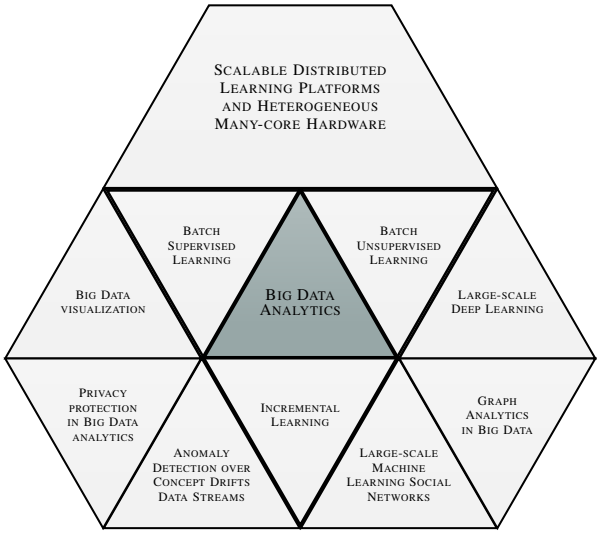

Fig. 5. Novel Trends in Big Data Analytics.

worldwide and contributing to the development of real-world applications, by extending the applicability of Machine Learning methods to much larger datasets than it is possible with traditional CPU implementations.

V. Novel Trends in Big Data Analytics

In an effort to identify trends in this data-driven world we present a unifying structure in Figure 5. The aim is to detect patterns and glean valuable findings in the growing sizes of modern datasets, which play an important role due to many interesting challenges posed so far.

As an attempt to underline the trends of Big Data Analytics in terms of large-scale learning it presents at its core triangle, from center to side, three main large-scale learning approaches: batch supervised and unsupervised, and incremental. On the top of the larger triangle side, scalable distributed learning platforms and heterogeneous many-core hardware need to be developed to ensure the sought implementation. From another perspective, six smaller triangles focus on novel problems to handle big data (e.g. big data visualization, large-scale deep learning, anomaly detection over concept drifts in data streams, large-scale social analysis, graph mining in Big Data, etc.) [12]. More specifically, on the left bottom main triangle side, three trends on data shape and form (visualization, privacy protection and anomaly detection) and on the right bottom triangle side trends more related to structure (deep layer networks, graph data and social networks).

Above areas will provide promising avenues of research. Nevertheless, the future Big Data Learning real-word applications will most likely require advanced computing hardware and software capabilities (*e.g.* using heterogeneous scalable parallel and distributed platforms).

VI. Final Remarks

This paper pointed out the strategies in machine learning and pattern recognition needed to handle the novel challenges posed by the large-scale data we have available nowadays. It also emphasizes the performance peak rates achieved in a number of problems by many-core GPU computing. Parallelizing ML algorithms is crucial for the development of successful real-world applications, in particular for a class of problems lying on the crossroads of several research topics including data sensing, data mining and data visualization [12]. In this scenario, GPUMLib is valuable for the scientific community, presenting several relevant aspects in machine learning for adaptive many-core machines. Altogether these challenges and opportunities launch new trends in machine learning computing that need to be tackled in a near future.

As a consequence of some of the above aspects, extension work in GPUMLib will include both the development of multi-GPU parallel and distributed ML implementations, to further enhance the scalability and performance of algorithms and extend their applicability to larger datasets.

Acknowledgment

This work was partly funded by FCT – Fundação para a Ciência e Tecnologia (UID/CEC/00326/2013).

References

[1] Z.-H. Zhou, "Three perspectives of data mining," *Artificial Intelligence*, vol. 143, no. 1, pp. 139–146, 2003.

[2] J. A. Olvera-López, J. A. Carrasco-Ochoa, J. F. Martínez-Trinidad, and J. Kittler, "A review of instance selection methods," *Artificial Intelligence Review*, vol. 34, no. 2, pp. 133–143, 2010.

[3] J. Tang, S. Alelyani, and H. Liu, "Feature selection for classification: A review," in *Data Classification: Algorithms and Applications*, C. Aggarwal, Ed. CRC Press, 2014.

[4] L. van der Maaten, E. Postma, and J. van den Herik, "Dimensionality reduction: A comparative review," Tilburg University, Tech. Rep. TiCC TR 2009–005, 2009.

[5] J. Gama, R. Sebastião, and P. Rodrigues, "Issues in evaluation of stream learning algorithms," in *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD 2009)*, 2009, pp. 329–338.

[6] N. Lopes and B. Ribeiro, "An incremental class boundary preserving hypersphere classifier," in *International Conference on Neural Information Processing (ICONIP 2011), Part II, LNCS 7063*, 2011, pp. 690–699.

[7] N. Lopes, D. Correia, C. Pereira, B. Ribeiro, and A. Dourado, "An incremental hypersphere learning framework for protein membership prediction," in *7th International Conference on Hybrid Artificial Intelligent Systems (HAIS 2012), LNCS 7208*, 2012, pp. 429–439.

[8] N. Lopes and B. Ribeiro, "Trading off distance metrics vs accuracy in incremental learning algorithms," in *Proceedings of the 21st Iberoamerican Congress on Pattern Recognition, Image Analysis, Computer Vision, and Applications (CIARP 2016)*. Springer International Publishing, 2017, pp. 530–538.

[9] N. Lopes and B. Ribeiro, "A fast optimized semi-supervised non-negative matrix factorization algorithm," in *IEEE International Joint Conference on Neural Networks (IJCNN 2011)*, 2011, pp. 2495–2500.

[10] D. D. Lee and H. S. Seung, "Learning the parts of objects by non-negative matrix factorization," *Nature*, vol. 401, pp. 788–791, 1999.

[11] J. D. Owens, M. Houston, D. Luebke, S. Green, J. E. Stone, and J. C. Phillips, "GPU computing," *Proceedings of the IEEE*, vol. 96, no. 5, pp. 879–899, 2008.

[12] N. Lopes and B. Ribeiro, *Machine Learning for Adaptive Many-Core Machines – A Practical Approach*, ser. Studies in Big Data. Springer, 2014, vol. 7.

[13] J. D. Owens, D. Luebke, N. Govindaraju, M. Harris, J. Krüger, A. Lefohn, and T. J. Purcell, "A survey of general-purpose computation on graphics hardware," *Computer Graphics Forum*, vol. 26, no. 1, pp. 80–113, 2007.

[14] M. Garland and D. B. Kirk, "Understanding throughput-oriented architectures," *Communications of the ACM*, vol. 53, no. 11, pp. 58–66, 2010.

[15] D. Steinkraus, I. Buck, and P. Y. Simard, "Using GPUs for machine learning algorithms," in *Proceedings of the 2005 Eight International Conference on Document Analysis and Recognition (ICDAR 2005)*, vol. 2, 2005, pp. 1115–1120.

[16] B. Catanzaro, N. Sundaram, and K. Keutzer, "Fast support vector machine training and classification on graphics processors," in *Proceedings of the 25th International Conference on Machine Learning (ICML 2008)*, vol. 307. ACM, 2008, pp. 104–111.

[17] N. Lopes and B. Ribeiro, "Fast pattern classification of ventricular arrhythmias using graphics processing units," in *Proceedings of the 14th Iberoamerican Congress on Pattern Recognition (CIARP 2009), LNCS 5856*. Springer, 2009, pp. 603–610.

[18] S. Ryoo, C. I. Rodrigues, S. S. Baghsorkhi, S. S. Stone, D. B. Kirk, and W. W. Hwu, "Optimization principles and application performance evaluation of a multithreaded GPU using CUDA," in *Proceedings of the 13th ACM Symposium on Principles and practice of parallel programming (PPoPP 2008)*, 2008, pp. 73–82.

[19] S. Sonnenburg, M. L. Braun, C. S. Ong, S. Bengio, L. Bottou, G. Holmes, Y. LeCun, K.-R. Müller, F. Pereira, C. E. Rasmussen, G. Rätsch, B. Schölkopf, A. Smola, P. Vincent, J. Weston, and R. C. Williamson, "The need for open source software in machine learning," *Journal of Machine Learning Research*, vol. 8, pp. 2443–2466, 2007.

[20] T. Hey, S. Tansley, and K. Tolle, Eds., *The Fourth Paradigm: Data-Intensive Scientific Discovery*. Microsoft Research, 2009.

[21] K. Bache and M. Lichman, "UCI machine learning repository," http://archive.ics.uci.edu/ml, 2013.