Chapter 7
Distributed Platforms and Cloud Services: Enabling Machine Learning for Big Data

Daniel Pop, Gabriel Iuhasz, and Dana Petcu

Abstract Applying popular machine learning algorithms to large amounts of data has raised new challenges for machine learning practitioners. Traditional libraries do not support properly the processing of huge data sets, so the new approaches are needed. Using modern distributed computing paradigms, such as MapReduce or in-memory processing, novel machine learning libraries have been developed. At the same time, the advance of cloud computing in the past 10 years could not be ignored by the machine learning community. Thus, a rise of cloud-based platforms has been of significance. This chapter aims at presenting an overview of novel platforms, libraries, and cloud services that can be used by data scientists to extract knowledge from unstructured and semi-structured, large data sets. The overview covers several popular packages to enable distributed computing in popular machine learning environments, distributed platforms for machine learning, and cloud services for machine learning, known as machine-learning-as-a-service approach. We also provide a number of recommendations for data scientists when considering machine learning approach for their problem.

Keywords Machine learning • Data mining • Cloud computing • Big data • Data scientist • Distributed computing • Distributed platforms

7.1 Introduction

Analyzing large amounts of data collected by companies, industries, and scientific domains is becoming increasingly important for all impacted domains. The data to be analyzed is no longer restricted to sensor data and classical databases, but it often includes textual documents and Web pages (text mining, Web mining), spatial data, multimedia data, or graph-like data (e.g., molecule configuration and social networks).

D. Pop (✉) • G. Iuhasz • D. Petcu
Institute e-Austria Timisoara, West University of Timisoara, Blvd. Vasile Parvan, nr. 4, 300223 Timişoara, Romania
e-mail: daniel.pop@e-uvt.ro

Z. Mahmood (ed.), *Data Science and Big Data Computing*,
DOI 10.1007/978-3-319-31861-5_7

Although, for more than two decades, parallel database products such as Teradata, Oracle, and Netezza have provided means to realize a parallel implementation of machine learning algorithms, expressing these algorithms in SQL code is a complex and difficult-to-maintain task. On the other side, large-scale installations of these products are expensive. Another reason for moving away from relational databases is the exponential growth of the unstructured data (e.g., audio and video) and semi-structured data (e.g., Web traffic data, social media content, sensor-generated data) in recent years. The needs of data science practitioners with respect to data analysis tools vary greatly across different domains, from medical statistics and bioinformatics to social network analysis or even in physics. This diversity is equally important for the advancement of machine learning tools and platforms. Consequently, in the past decade, researchers moved from the parallelization of machine learning algorithms and support in relational databases toward the design and implementation on top of novel distributed storage (e.g., NoSQL data stores, distributed file systems) and processing paradigms (e.g., MapReduce). From the business perspective, Software-as-a-Service (SaaS) model opened up new opportunities for machine learning providers, who moved the stand-alone tools toward cloud-based machine learning services.

In this chapter, we survey how distributed storage and processing platforms help data scientists to process large, heterogeneous sets of data. The tools, frameworks, and services included in this chapter share a common characteristic: all run on top of distributed platforms. Thus, parallelization of machine learning algorithms, either using multiple-core CPU or GPU, was not included here. The reader is referred to [42], a recent comprehensive study covering that topic. We also avoided commercial solution providers, small or big players, since their offerings are either based on distributed open-source packages or they do not disclose the implementation details.

In the first section, we briefly introduce the reader to the machine learning field, describing and classifying the types of problems and overviewing the challenges of applying traditional algorithms to large, unstructured data sets. The first category of tools considered in this survey covers tools, packages, and libraries that enable data scientists to use traditional environments for data analysis such as R Systems, Python, or statistics applications, in order to deal with large data sets. We survey, next, the distributed platforms for big data processing, either based on Apache Hadoop or Spark, as well as platforms specifically designed for distributed machine learning. We also include a section on scalable machine learning services delivered using Software-as-a-Service business model since they offer easy-to-use, user-friendly graphical interfaces supporting users in quickly getting and deploying models. The last section of the chapter summarizes our findings and provides readers with a collection of best practices in applying machine learning algorithms.

7.2 Machine Learning for Data Science

The broadest and simplest definition of machine learning is that *it is a collection of computational methods that use experience*, i.e., information available to the system to improve performance or to make predictions [28]. This information usually takes the form of electronic records collected and made available for analytical purposes. These records can take the form of pre-labeled training sets (usually by a human operator although this is not always the case). Another important source of data is that resulting from direct interactions with a given environment, either virtual, such as software interactions, network data, etc., or relying on real-world natural scenarios, such as weather phenomena, water level, etc. Data quality and quantity are extremely important in order to obtain an acceptable learned model. Machine learning relies on data-driven methods that combine fundamental concepts in the field of computer science with optimization, probability, and statistics [28].

There is a wide array of applications to which machine learning can and is being applied, such as taming (text mining and document classification), spam detection, keyword extraction, emotion extraction, natural language processing (NPL), unstructured text understanding, morphological analysis, speech synthesis and recognition, optical character recognition (OCR), computational biology, face detection, image segmentation, image recognition, fraud detection, network intrusion detection, board and video games, navigation in self-driving vehicles, planning, medical diagnosis, recommendation systems, or search engines. In all these applications, we can identify several types of learning-related issues, which are:

- *Classification* – to assign each item from a data set to a specific category, e.g., given a document, assign a domain (history, biology, mathematics) to which it belongs.
- *Regression and time series analysis* – to predict a real value for each item, e.g., future stock market values, rainfall runoff, etc.
- *Ranking* – to return an ordered set of features based on some user-defined criterion (e.g., Web search).
- *Dimensionality reduction (feature selection)* – to use for transforming initial large feature spaces into a lower-dimensional representation so that it preserves the properties of the initial representation.
- *Clustering* – to group items based on some predefined distance measure. It is usually used on very large data sets. In sociology, it can be used to group individuals into communities [30].
- *Anomaly detection* – to conduct observation or series of observations which do not resemble any pattern or data item in a data set [6, 37].

The most common classification techniques are called *linear classifiers*. In this case, classification is expressed in the form of a linear function. This function assigns scores to each possible category. Among the linear classifiers, we have linear regression, perceptron, and support vector machines (SVM) [28]. Another form of classification is based on kernel estimation in the form of the k-nearest neighbor (k-NN) algorithms. Decision trees such as C4.5 [28] are also used for this type of problem and are based on information theory (difference in entropy), which is used as the splitting criterion. Ensemble meta-algorithm-based techniques such as AdaBoost [28] are also used although they have questionable performance on noisy data sets. Some methods such as Classification and Regression Tree (CART) algorithm can be used for both regression and classification problems. Naïve Bayes, for example, can also be used for both types of problems.

Clustering algorithms are largely split according to their particular definition of what cluster model they use. Connectivity models (hierarchical clustering) are based on distance connectivity. Centroid models, such as k-means (k-M), represent each cluster with a single mean vector. Density models consider clusters as connected dense regions from the data space. DBSCAN and OPTICS [28] are two algorithms using this model. Statistical distribution-based models are also used.

Anomaly detection is a special case of either classification or clustering; thus, it uses mostly the same algorithms and methodologies. Feature selections' main goal is the reduction of the amount of recourses required to analyze big data set. They are extremely useful when no domain expert is available that could help in the reduction of the dimensionality of the available data. There are a number of general dimensionality reduction techniques such as principal component analysis (PCA), kernel PCA, multilinear PCA, and wrapping methods [25].

In machine learning, there are different types of training scenarios [28]. Arguably the most widely used type of training is called *supervised* learning. In this scenario, the learner receives a set of labeled data for training and validation. The learned prediction model can be then applied to a larger data set and identify all unseen data points. This type of learning is used for classification and regression (time series analysis). Supervised methods rely on the availability and accuracy of labeled data sets. In *unsupervised* learning, the learner receives unlabeled data that it has to group based on a distance measurement. In some scenarios, labeled data is extremely hard to come by; thus, training a classification model is often unfeasible. This type of learning is used for clustering, anomaly detections (a type of clustering), and dimensionality reduction.

In some cases, labeled data is only a small fraction of the overall training data set. This is called *semi-supervised* learning. The idea is that the distribution of unlabeled data can help the learner achieve a much better performance [11].

In *reinforcement* learning, the training is done using an evaluation function. This means that training and testing are much more interlaced than in other learning scenarios. The performance of an algorithm in a problem environment is continuously evaluated through the monitoring and evaluation of its performance. Favorable outcomes are rewarded, while unfavorable ones are punished. Reinforcement learning is used in genetic algorithm, neural networks, etc. *Online* learning is used when data is available in a sequential way. This means that the mapping between data sets and labels is established each time a new data point is received.

Due to the popularity of data analytics, machine learning techniques are being pursued by teams with complementary skills across very different businesses (finance, telecommunications, life sciences, etc.). This section aims to classify the diversity of groups of interests with respect to machine for big data. We must state upfront that there is no clear line between these perspectives, as competencies and expectations blur the edges and multidisciplinary teams are put in place to tackle complex scenarios. Some of the groups are:

- *Data scientists and machine learning practitioners*: One way of approaching the problem is from the data scientist's perspective. Statisticians and data scientists are now facing data set size explosion; thus, coping with large-size data sets is a must. These are users with strong mathematical background, proficient in statistics and mathematical software applications, such as R, Octave, MATLAB, Mathematica, Python, SAS Studio, or IBM's SPSS, but less experienced in coping with data sets of large dimensions, distributed computing, or software development. Their expectation is to easily reuse the algorithms already available in their preferred language and be able to run them against large data sets on distributed architectures (on-premise or cloud based). A later section in this chapter entitled "Distributed and Cloud-Based Execution Support in Popular Machine Learning Tools" overviews packages and tools available for this purpose.
- *Software engineers and developers*: Teams of software engineers often face client requirements asking for the transition from available (large) data warehouse to actionable knowledge. These are users with a vast experience in software development, skilled programmers in general-purpose programming languages, and they "speak" parallel and distributed computing. Deep mathematics and statistics are not necessarily their preferred playground, as they expect tools and libraries to enable them to integrate advanced ML algorithms in their systems and thus quickly get actionable results. They need fast, easy-to-customize (less number of parameters), and easy-to-integrate algorithms that run on distributed architectures and are able to fetch data from large data repositories. Tools addressing these requirements are discussed in a later section in this chapter entitled "Distributed Machine Learning Platforms."
- *Domain experts*: Domain experts (financial, telecommunications, physics, astronomy, biotechnologies, etc.) know their data best, but they are less experienced in ML algorithms and software tools. Ideally, they need off-the-shelf software applications, easy to install and use, or cloud-based Software-as-a-Service solutions allowing them to get insights on their data and produce reports and executable models for further usage. A later section on "Machine Learning as a Service" presents several machine learning services providers.

The dynamic of natural, social, and economic systems raises new challenges for data scientists, such as:

- *Massive data sets*. Data sets are growing faster, being common now to reach numbers of 100 TB or more. The Sloan Digital Sky Survey occupies 5 TB of storage, the Common Crawl Web corpus is 81 TB in size, and the 1000 Genomes Project requires 200 TB of space, just to name a few.
- *Large models*. Massive data sets need large models to be learned. Some deep neural networks are comprised of more than ten layers with more than a billion parameters [24, 25], collaborative filtering for video recommendation on Netflix comprises 1–10 billion parameters, and multitask regression model for simplest whole-genome analysis may reach 1 billion parameters as well.
- *Inadequate ML tools and libraries*. Traditional ML algorithms used for decades (k-means, logistic regression, decision trees, Naïve Bayes) were not designed for handling large data sets and huge models; they were not developed for parallel/ distributed environments.
- *"Operationalization" of predictive models*. "Operationalize" refers to integrate predictive models into automated decision-making systems and processes on a large scale in order to deliver predictions to end users, who will ultimately benefit from them. Integrating these models into multiple platforms (Web, stand-alone, mobile) across different business units requires a high degree of customization, which slows deployment, drives up costs, and limits scalability.
- *Lack of clear contracts*. More recently, terms such as Analytics as a Service (AaaS) and Big Data as a Service (BDaaS) are becoming popular. They comprise services for data analysis similarly as IaaS offers computing resources. Unfortunately, the analytics services still lack well-defined service-level agreements available for IaaS because it is difficult to measure quality and reliability of results and input data, to provide promises on execution times and guarantees on methods for analyzing the data. Therefore, there are fundamental gaps on tools to assist service providers and clients to perform these tasks and facilitate the definition of contracts for both parties [2].
- *Inadequate staffing*. Market research shows that inadequate staffing and skills, lack of business support, and problems with analytics software are some of the barriers faced by corporations when performing analytics [36].

In the next three sections, we discuss various machine learning tools.

7.3 Distributed and Cloud-Based Execution Support in Popular Machine Learning Tools

Annual Nuggets survey [23] shows that R, Python, SQL, and SAS have been rated the preferred languages of choice for the past 3 years. One of the early trends matching cloud computing and data analysis was, around 2010s, the provision of virtual machine images (VMI) for these popular systems (R, Octave, or Maple) integrated within public cloud service providers, such as Amazon Web Services or Rackspace. After several proofs of concept were successfully built, such as

Table 7.1 Distributed processing and storage

Environment	Package	Distributed processing support	Distributed file system access
R	RHadoop	Hadoop	HDFS
	RHIPE	Hadoop	HDFS
	Segue for R	Amazon Elastic MapReduce	–
	RHive	HIVE	HIVE
	Snow	Socket-based, MPI, PVM	–
	H5	–	HDF5
	Pbd*	MPI	NetCDF
Python	pyDoop	Hadoop	HDFS
	Anaconda	Distributed and GPU	HDFS, HDF5
	IPython. parallel	Distributed and parallel	–

Cloudnumbers,[1] CloudStat,[2] Opani,[3] and Revolution R Enterprise,[4] the practice today is to provide VMI through the public cloud providers' marketplaces, such as Amazon Marketplace. One can find Amazon Machine Images (AMI), via the marketplace, for all the popular mathematical and statistics environments. Examples include Predictive Analytics Framework and Data Science Toolbox[5] that support both Python and R, BF Accelerated Scientific Compute for R with accelerated math libraries for boosted performance, or SAS University Edition for SAS Studio.

Much more effort has been invested in the development of plug-ins for the most popular machine learning platforms to allow data scientists to easily create and run time-consuming jobs over clusters of computers. This approach allows ML practitioners to reuse their existing code and adapt it for large data set processing, into the same environment they used for prototyping. It also leverages existing infrastructure (grids, clusters) for large-scale distributed computation and data storage. Table 7.1 synthesizes available plug-ins for distributed storage and processing for the most popular languages of big data: R and Python.

Since R is the preferred option among machine learning practitioners, several packages were developed in order to enable big data processing within R, most of them being available under CRAN[6] package Web page. These R extensions make possible to distribute the computational workload on different types of clusters, while accessing data from distributed file systems. First example is the RHadoop [33], a collection of five R packages, that enables R users to run MapReduce jobs on

[1] http://cloudnumbers.com

[2] http://cs.croakun.com

[3] http://opani.com

[4] www.revolutionanalytics.com

[5] http://datasciencetoolbox.org

[6] http://cran.r-project.org/web/packages/available_packages_by_name.html

Hadoop by writing R functions for mapping and reducing. Similarly, RHIPE[7] is another R package that brings MapReduce framework to R practitioners, providing seamless access to Hadoop cluster from within R environment. Using specific R functions, programmers are able to launch MapReduce jobs on the Hadoop cluster, with results being easily retrieved from HDFS. Segue[8] for R project makes it easier to execute MapReduce jobs from within the R environment on elastic clusters at Amazon Elastic MapReduce,[9] but lacks support for handling large data sets. RHive is an extension enabling distributed computing via HIVE in R, by a seamless integration between HQL (Hive Query Language) and R objects and functions. Snow (Simple Network of Workstations) [41] and its variants (snowfall, snowFT, doSnow) implement a framework that is able to express an important class of parallel computations and is easy to use within an interactive environment like R. It supports three types of clusters: socket based, MPI, and PVM. Support for manipulating large data sets in R is available in H5 plug-in, which provides an interface to the HDF5 API through S4 objects, supporting fast storage and retrieval of R objects to/from binary files in a language-independent format. The pbd* (pbdBASE, pbbMPI, pbdNCDF4, pbdSLAP, etc.) series is a collection of R packages for programming with big data, enabling MPI distributed execution, NetCDF file system access, or tools for scalable linear algebra.

As far as Python is concerned, we should start by mentioning pyDoop,[10] a Python MapReduce and HDFS API for Hadoop [26]. Anaconda[11] is a free, scalable Python distribution for large-scale data analytics and scientific computing. It is a collection of Python packages (NumPy, SciPy, Pandas, IPython, Matplotlib, Numba, Blaze, Bokeh) that enables fast large data set access, GPU computation, access to distributed implementations of ML algorithms, and more. IPython.parallel[12] provides a sophisticated and powerful architecture for parallel and distributed computing [14] that enables IPython to support many different styles of parallelism including single program multiple data (SPMD), multiple program multiple data (MPMD), message passing using MPI, data parallel, and others. In a tutorial at PyCon 2013, Grisel [15] presented how scikit-learn [32], a popular open-source library for machine learning in Python, can be used to perform distributed machine learning algorithms on a cheap Amazon EC2 cluster using IPython.parallel and StarCluster.[13] We should note as well that most of the libraries and frameworks considered in the next sections offer Python language bindings, but we choose not to include them in this section.

[7] http://www.stat.purdue.edu/~sguha/rhipe/doc/html/index.html

[8] http://code.google.com/p/segue

[9] http://aws.amazon.com/elasticmapreduce

[10] https://github.com/crs4/pydoop

[11] https://store.continuum.io/cshop/anaconda

[12] http://ipython.org/ipython-doc/dev/parallel/

[13] http://star.mit.edu/cluster/

Other mathematical and statistics environments have seen similar interest in embracing big data processing. For example, HadoopLink[14] is a package that allows MapReduce programs being implemented in Mathematica and to run them on a Hadoop cluster. It looks more like a proof of concept (PoC), being stalled since 2013. MATLAB has its Parallel Computing Toolbox which extends the capabilities of MATLAB MapReduce and Datastore[15] in order to run big data application. MATLAB Distributed Computing Server also supports running parallel MapReduce programs on Hadoop clusters.[16]

There are extensions to traditional machine learning libraries that enable execution on top of Hadoop or Spark clusters. Weka [16], one of the most popular libraries for data mining, supports both Hadoop and Spark execution through Weka Hadoop integration [17]. There is also a commercial distribution, Pentaho [34], that offers a complete solution for big data analytics, supporting all phases of an analytics process – from preprocessing to advanced data exploration and visualization, which uses distributed Weka execution for analytics. Another example is the KNIME's [4] big data extension,[17] which enables the access to Hadoop via Hive. RapidMiner [20] has Radoop[18] that enables the deployment of workflows on Hadoop.

7.4 Distributed Machine Learning Platforms

After distributed processing and storage environments (Hadoop, Dryad, MPI) reached an acceptable level of maturity, they became an increasingly appealing foundation for the design and implementation of new platforms for machine learning algorithms. These provide users out-of-the-box algorithms, which are run in parallel mode over a cluster of (commodity) computers. These solutions do not use statistics, or mathematics software packages, rather they offer self-contained, optimized implementations in general-purpose programming languages (C/C++, Java) of state-of-the-art ML methods and algorithms. This section focuses on ML platforms specifically designed for distributed and scalable computing. Table 7.2 summarizes some recent platforms.

The IBM Research Lab has been one of the pioneers who invested in distributed machine learning frameworks. Nimble [12] and SystemML [13] are two high-level conceptual frameworks supporting the definition of ML algorithms and their execution on Hadoop clusters. Nimble, a sequel to IBM's Parallel Machine Learning Toolbox [31], features a multilayered framework enabling developers to express

[14] https://github.com/shadanan/HadoopLink

[15] http://www.mathworks.com/help/matlab/large-files-and-big-data.html

[16] http://www.mathworks.com/help/distcomp/big-data.html

[17] https://www.knime.org/knime-big-data-extension

[18] https://rapidminer.com/products/radoop/

Table 7.2 Distributed ML frameworks

Name	License	ML Problem	Distr. Env.	Comm.	Lang.
Petuum	Open source (Sailing Lab)	DL, CLS, CLU, RGR, MET, TOP	Clusters or Amazon EC2, Google CE	Medium	C++
Jubatus	LGPL v2.1	CLS, RGR, ANO, CLU, REC, Graph	Zookeeper	Medium	C++
MLlib (MLBase)	Apache 2.0	RGR, CLS, REC, CLU	Spark	Large	Scala, Java
Mahout	Apache 2.0	Collaborative filtering, CLS, CLU, DR, TOP	Hadoop, Spark, H2O	Medium	Java
Oryx	Apache 2.0	REC, CLS, RGR, CLU	Hadoop, Spark	Low	Java
Trident-ML	Apache 2.0	CLS, RGR, CLU, DR	Storm	Low	Java
H2O	Apache 2.0	DL, RGR, CLS, CLU, DR	Hadoop	Medium	Java
GraphLab Create	Apache 2.0	CLU, CLS, RGR, DL, REC	Hadoop, Spark, MPI	High	C++
Vowpal Wabbit	Ms-PL	CLS, RGR, CLU	Hadoop	Medium	C++
Deeplearning4J	Apache 2.0	DL	Hadoop, Spark, AWS, Akka	Medium	Java, Scala
Julia's MLBase	MIT License	CLS	Julia		Julia
Flink-ML	Apache 2.0	CLS, RGR, CLU, REC	Flink Hadoop	Low	Scala
DryadLINQ			Dryad, Hadoop YARN	None	C#, LINQ
Nimble	NA	CLU, FRQ, ANO,	Hadoop	None	Java
SystemML	NA	RGR, PageRank	Hadoop	None	DML

ANO anomaly detection, *CLS* classification, *CLU* clustering, *DL* deep learning, *DR* dimensionality reduction, *FRQ* frequent pattern, *MET* metrics learning, *REC* recommendation, *RGR* regression, *TOP* topic modeling

their ML algorithms as tasks, which are then passed to the next layer, an architecture-independent layer, composed of one queue of DAGs of tasks, plus worker thread pool that unfolds this queue. The bottom layer is an architecture-dependent layer that translates the generic entities from the upper layer into various runtimes, the only distributed environment supported within the proof of concept being Hadoop alone. The layered architecture of the system hides the low-level control and choreography details of most of the distributed and parallel programming paradigms (MR, MPI, etc.), it allows developers to compose parallel ML algorithms using reusable (serial and parallel) building blocks, but also it enables portability and scalability. SystemML proposes an R-like language (declarative machine learning language) that includes linear algebra primitives and shows how it can be optimized and compiled down to MapReduce. Authors report an extensive performance evaluation on three ML algorithms (group nonnegative matrix factorization, linear regression, PageRank) on varying data and Hadoop cluster sizes. These two systems are purely research endeavors, and they are not available to the community.

Most of the frameworks rely on Hadoop's MapReduce paradigm and the underlying distributed file storage system (HDFS) because it simplifies the design and implementation of large-scale data processing systems. Only a few frameworks (e.g., Jubatus, Petuum, GraphLab Create) have tried to propose novel distributed paradigms, customized to machine learning for big data, in order to optimize the complex, time-consuming ML algorithms.

Recognizing the limitations and difficulties of adapting general-purpose distributed frameworks (Hadoop, MPI, Dryad, etc.) to ML problems, a team at CMU under E. P. Xing lead designed a new framework for distributed machine learning able to handle massive data sets and cope with big models. Petuum[19] (from Perpetuum Mobile) [8, 43] takes advantage of data correlation, staleness, and other statistical properties to maximize the performance for ML algorithms, realized through core features such as a distributed Parameter Server and a distributed Scheduler (STRADS). It may run either on-premise clusters or on cloud computing resources like Amazon EC2 or Google Compute Engine (GCE).

Jubatus[20] is a distributed computing framework specifically designed for online machine learning on big data. A loose model sharing architecture allows it to efficiently train and share machine learning models by defining three fundamental operations, viz., update, mix, and analyze [19]. Comparing to Apache Mahout, Jubatus offers stream processing and online learning, which means that the model is continuously updated with each data sample that is coming in, by fast, not memory-intensive algorithms. It requires no data storage nor sharing, only model mixing. In order to efficiently support online learning, Jubatus operates updates on local models and then each server transmits its model difference that are merged and distributed back to all servers. The mixed model improves gradually thanks to all servers' work.

GraphLab Create,[21] formerly GraphLab project [27], is a framework for machine learning that expresses asynchronous, dynamic, graph-parallel computation while ensuring data consistency and achieving a high degree of parallel performance, in both shared-memory and distributed settings. It is an end-to-end platform enabling data scientists to easily create intelligent apps at scale, from cleaning the data, developing features, training a model, and creating and maintaining a predictive service. It runs on distributed Hadoop/YARN clusters, as

[19] http://petuum.org

[20] http://jubat.us

[21] https://dato.com/products/create

well on local machine or on EC2, and it exposes a Python interface for an easy accessibility.

Apache Mahout [29] is a scalable machine learning framework built on top of Hadoop that features a rich collection of distributed implementations of machine learning and data mining algorithms. Although initially created on top of Hadoop, starting with version 0.10, it supports additional execution engines such as Spark and H20, while Flink[22] is a project in progress. The same release introduces Mahout-Samsara, a new math environment created to enable users to develop their own extensions, using Scala language, based on general linear algebra and statistical operations. Mahout-Samsara comes with an interactive shell that runs distributed operations on a Spark cluster. This makes prototyping or task submission much easier and allows users to customize algorithms with a whole new degree of freedom.

H2O[23] and MLlib [10] are two of the most actively developed projects. Both feature distributed, in-memory computations and are certified for Apache Spark (MLlib being part of Spark), as well as for Hadoop platforms. This in-memory capability means that in some instances these frameworks outperform Hadoop-based frameworks [44]. MLlib has been shown to be more scalable than Vowpal Wabbit. One important distinction when comparing H2O with other MapReduce applications is that each H2O node (which is a single JVM process) runs as a mapper in Hadoop. There are no combiners nor reducers. Also, H2O has more built-in analytical features and a more mature REST API for R, Python, and JavaScript than MLlib.

Vowpal Wabbit[24] [38] is an open-source, fast, out-of-core learning system, currently sponsored by Microsoft research. It has an efficient implementation of online machine learning, using the so-called hash trick [35] as the core data representation, which results in significant storage compression for parameter vectors. VW reduces regression, multiclass, multi-label, or structured prediction problems to a weighted binary classification problem. A Hadoop-compatible computational model called AllReduce [1] has been implemented in order to eliminate MPI and MapReduce drawbacks, which relate to machine learning. Using this model, a 1000-node cluster was able to learn a terafeature data set in one hour [1].

Julia[25] is high-level, high-performance, and dynamic programming language. It is designed for computing and provides a sophisticated compiler, distributed parallel execution, and numerical accuracy and has an extensive mathematical function library. Comparing to traditional MPI, Julia's implementation of message passing is "one sided," thus simplifying the process management. Furthermore, these operations typically do not look like "message send" and "message receive" but rather resemble higher-level operations like calls to user functions. It also provides a

[22] https://flink.apache.org/

[23] http://0xdata.com/product/

[24] https://github.com/JohnLangford/vowpal_wabbit/

[25] http://julialang.org/

powerful browser-based notebook using IPython. It also possesses a built-in package manager and it is able to call C functions directly. It is specially designed for parallelism and distributed computation. It also provides a variety of classification, clustering, and regression analysis packages[26] implemented in Julia.

Another framework focusing on real-time online machine learning is Trident-ML [22], built on top of Apache Storm, a distributed stream-processing framework. It processes batches of tuples in a distributed way, which means that it can scale horizontally. However, Storm does not allow state updates to append simultaneously, a shortage that hinders distributed model learning.

The Apache Oryx 2[27] framework is a realization of the lambda architecture built on top of Spark and Apache Kafka. It is a specialized framework that provides real-time, large-scale machine learning. It consists of three tiers: lambda, machine learning, and application. The lambda tier is further split up into batch, speed, and serving tier, respectively. Currently, it has only three end-to-end implementations for the batch, speed, and serving layers (collaborative filtering, k-means clustering, classification, and regression based on random forest). Although it has only these three complete implementations, its main design goal is not that of a traditional machine learning library but more of a lambda architecture-based platform for MLlib and Mahout. At this point, it is important to note several key differences between Oryx 1[28] and Oryx 2. Firstly, Oryx 1 has a monolithic tier for lambda architecture, while Oryx 2 has three as mentioned in the previous paragraph. The streaming-based batch layer in Oryx 2 is based in Spark, while in the first version, it was a custom MapReduce implementation in the computational layer. Two of the most important differences relate to the deployment of these frameworks. Oryx 2 is faster yet more memory hungry than the previous version because of its reliance on Spark. Second, the first version supported local (non-Hadoop) deployment, while the second version does not.

DryadLINQ[29] [5] is LINQ[30] (Language Integrated Query) subsystem developed at Microsoft Research on top of Dryad [21], a general-purpose architecture for execution of data-parallel applications. A DryadLINQ program is a sequential program composed of LINQ expressions performing arbitrary side effect-free transformations on data sets and can be written and debugged using standard . NET development tools. The system transparently translates the data-parallel portions of the program into a distributed execution plan, which is passed to the Dryad execution platform that ensures efficient and reliable execution of this plan. Following Microsoft's decision to focus on bringing Apache Hadoop to Windows systems, this platform has been abandoned, and Daytona project took off, which has recently became Windows Azure Machine Learning platform.

[26] http://mlbasejl.readthedocs.org/en/latest/

[27] http://oryxproject.github.io/oryx/

[28] https://github.com/cloudera/oryx

[29] http://research.microsoft.com/en-us/projects/dryad/

[30] http://msdn.microsoft.com/netframework/future/linq/

Deeplearning4J[31] is an open-source distributed deep-learning library written in Java and Scala. It is largely based on ND4J library for scientific computation that enables GPU, as well as native code integration. It is also deployable on Hadoop, Spark, and Mesos. The main difference between this library and the others mentioned above is that it is mainly focused on business use cases, not on research. This means that some features, such as parallelism, is automatic, meaning that worker nodes are set up automatically.

Apache SAMOA[32] is a distributed streaming machine learning framework. It contains abstractions for distributed streaming machine learning algorithms. This means that users can focus on implementing distributed algorithms and not worry about the underlying complexities of the stream processing engines it supports (Storm, S4, Samza, etc.).

7.5 Machine Learning as a Service (MLaaS)

This section focuses on Software-as-a-Service providers' provision of machine learning services as MLaaS. These services are accessible via RESTful interfaces, and in some cases, the solution may also be installed on-premise (e.g., ersatz). The favorite class of machine learning problems addressed by these services is predictive modeling (BigML, Google Prediction API, EigenDog), while clustering and anomaly detection receive far less attention. We did not include in this category the fair number of SQL over Hadoop processing solutions (e.g., Cloudera Impala, Hadapt, Hive), because their main target is not machine learning problems, rather fast, elastic, and scalable SQL processing of relational data using the distributed architecture of Hadoop.

Table 7.3 presents current MLaaS solutions as well as some of their key characteristics. We have identified four characteristics: machine learning problem support, data sources, model exporting, and model deployment. It is easily observable that most MLaaS are designed to deal with common problems such as classification, regression, and clustering. When it comes to data acquisition facilities, all platforms support data upload (various formats csv, arff, etc.); some even feature integration with different storage solutions (S3, HDFS, etc.). Predictive model training, verification, and visualization are supported by all the solutions listed in Table 7.3; however, not all support predictive model exporting via PMML.[33] Last but not least, some platforms support local Web services as well as cloud deployment. In the next few paragraphs, we detail some of the more important services from Table 7.3.

[31] http://deeplearning4j.org

[32] https://samoa.incubator.apache.org/

[33] http://www.dmg.org/v4-1/GeneralStructure.html

Table 7.3 Machine learning as a service

Name	ML problems	Data source	Model export	Deployment
Azure ML	CLS, RGR, CLU, ANO	Upload, Azure	None	Cloud
PredictionIO	RGR, CLS, REC, CLU	Upload, Hbase	None	Local, cloud
Ersatz Labs	DL	Upload	None	Cloud, local
ScienceOps[a] (ScienceBox)	RGR, CLS, REC, CLU	S3, Upload	PMML	Cloud, local
Skymind	DL	Upload	None	Cloud, Local
BigML[b]	CLS, RGR, CLU	Upload, S3, Azure, OData	PMML	Cloud
Amazon ML	CLS, RGR, CLU	S3, Redshift Upload	None	Cloud
BitYota[c]	CLS, RGR, CLU	S3, Azure	None	Cloud
Google Prediction API	CLS, RGR, CLU, ANO	Upload, Google Cloud Storage	PMML	Cloud
EigenDog[d]	CLU, RGR	Upload, S3	None	Local, cloud
Metamarkets[e]	CLU, ANO	Upload, HDFS	None	Local (Druid), cloud
Zementis ADAPA[f]	CLS, RGR, CLU	S3, Azure, Upload, SAP HANA	PMML	Local, cloud

[a]https://yhathq.com/products/scienceops
[b]http://bigml.com
[c]http://bityota.com
[d]https://eigendog.com/\#home
[e]http://metamarkets.com/
[f]http://zementis.com/products/adapa/amazon-cloud/

Windows Azure Machine Learning, formerly project Daytona, was officially launched in February 2015 as a cloud-based platform for big data processing. This comes with a rich set of predefined templates for data mining workflows, as well with a visual workflow designer that allows end users to compose complex machine learning workflows. In addition, it supports the integration of R and Python scripts within workflows and is able to run the jobs on Hadoop and Spark platforms. The built models are deployed in a highly scalable cloud environment and can easily be accessed via Web services [39].

PredictionIO[34] is based on an open-source software such as Spark. This means that the solution can be deployed and hosted on any infrastructure. This is in sharp contrast with Azure, which requires the data to be uploaded into Azure. Also, it is possible to write custom distributed data processing tasks in Scala while on Azure custom scripts can only be run on a single node. There are no restrictions on the size of the training data not on the number of concurrent request. It can be deployed on Amazon WS, Vagrant, Docker, or even starting from source code.

[34] https://prediction.io/

The recent popularity of deep learning has resulted in the creation of various services which bundle deep-learning libraries (Theano, pylearn2, Deeplearning4j, etc.) into a MLaaS format. Some good examples are Ersatz Labs[35] and Skymind.[36] These provide similar services and support distributed as well as GPU deployment.

Amazon Machine Learning service[37] allows users to train predictive models in the cloud. It targets a similar use case as Azure Machine Learning from Microsoft and Google's Predictive API. It has similar features to many large-scale learning applications including visualization and basic data statistics. The exact learning algorithm it uses is not known; however, it is similar to Vowpal Wabbit. There are some limitations such as the inability to export the learned model or to access data which is not stored inside Amazon (Amazon S3 or Redshift).

Google Prediction API[38] is Google's cloud-based machine learning tools that can help to analyze data. It is closely connected to Google Cloud Storage[39] where training data is stored and offers its services using a RESTful interface, client libraries allowing programmers to connect from Java, JavaScript, .NET, Ruby, Python, etc. In the first step, the model needs to be trained on data, supported models being classification and regression for now. After the model is built, one can query this model to obtain predictions on new instances. Adding new data to a trained model is called Streaming Training and it is also nicely supported. Recently, PMML preprocessing feature has been added, i.e., Prediction API supports preprocessing your data against a PMML transform specified using PMML 4.0 syntax and does not support importing of a complete PMML model that includes data. Created models can be shared as hosted models in the marketplace.

7.6 Related Studies

Since 1995, when Thearling [40] presented a massively parallel architecture and the algorithms for analyzing time series data, allegedly, one of the first approaches to parallelization of ML algorithms, many implementations were proposed for ML algorithm parallelization for both shared and distributed systems. Consequently, many studies tried to summarize, classify, and compare these approaches. We will address in this section only the most recent ones.

Upadhyaya [42] presents an overview of machine learning efforts since 1995 onward grouping the approaches based on prominent underlying technologies: those employed on GPUs (2000–2005 and beyond), those using MapReduce technique (2005 onward), the ones that did not consider neither MapReduce nor GPUs

[35] http://www.ersatzlabs.com/

[36] http://www.skymind.io/about/

[37] http://docs.aws.amazon.com/machine-learning/latest/mlconcepts/mlconcepts.html

[38] https://developers.google.com/prediction/

[39] https://developers.google.com/storage/

(1999–2000 and beyond), and, finally, few efforts discussing the MapReduce technique on GPU. Contrasting to this extensive overview, we focus on more recent distributed and cloud-based solutions, regardless if they are coming from academia or industry.

The book *Scaling Up Machine Learning: Parallel and Distributed Approaches* by Bekkerman et al. [3] presents an integrated collection of representative approaches, emerged in both academic (Berkeley, NYU, University of California, etc.) and industrial (Google, HP, IBM, Microsoft) environments, for scaling up machine learning and data mining methods on parallel and distributed computing platforms. It covers general frameworks for highly scalable ML implementations, such as DryadLINQ and IBM PMLT, as well as specific implementations of ML techniques on these platforms, like ensemble decision trees, SVM, or k-means. The book is a good starting point, but it does not aim at providing a structured view on how to scale out machine learning for big data applications, which is central to our study.

A broader study is conducted by Assunção et al. [2], who discuss approaches and environments for carrying out analytics on clouds for big data applications. Model development and scoring, i.e., machine learning, is one of the areas they considered, alongside other three: data management and supporting architectures, visualization and user interaction, and business models. Through a detailed survey, they identify possible gaps in technology and provide recommendations for the research community on future directions on cloud-supported big data computing and analytics solutions. With respect to this study, our work goes into deeper details on the specific topic of distributed machine learning approaches, synthesizing and classifying existing solutions to give data scientists a comprehensive view of the field.

Interesting analyses have been made available through online press and blogs [7, 9, 18]; they have reviewed open-source or commercial players for big data analytics and predictions.

7.7 Conclusion and Guidelines

Analyzing big data sets gives users the power to identify new revenue sources, develop loyal and profitable customer relationships, and run the organization more efficiently and cost-effectively – overall, giving them competitive advantage over other competitions. Big data analytics is still a challenging and time-demanding task that requires important resources, in terms of large e-infrastructure, complex software, skilled people, and significant effort, without any guarantee on ROI. After reviewing more than 40 solutions, our key findings are summarized below.

Both, research and industry, have invested efforts in developing "as-a-Service" solutions for big data problems (Analytics-as-a-Service, Data-as-a-Service, Machine-Learning-as-a-Service) in order to benefit of the advantages cloud computing provides such as resources on demand (with costs proportional to the actual usage), scalability, and reliability.

Existing programming paradigms for expressing large-scale parallelism (MapReduce, MPI) are the de facto choices for implementing distributed machine learning algorithms. The initial enthusiastic interest devoted to MapReduce has been balanced in recent years by novel distributed architectures specifically designed for machine learning problems. Nevertheless, Hadoop remains the state-of-the-art platform for processing large data sets stored on HDFS, either in MapReduce jobs or using higher-level languages and abstractions.

Although state-of-the-art tools and platforms provide intuitive graphical user interfaces, current environments lack an interactive process, and techniques should be developed to facilitate interactivity in order to include analysts in the loop, by providing means to reduce time to insight. Systems and techniques that iteratively refine answers to queries and give users more control of processing are desired.

From the perspective of resource management and data processing, new frameworks able to combine applications from multiple programming models (e.g., MPI, MapReduce, workflows) on a single solution need to be further investigated. Optimization of resource usage and energy consumption, while executing data-intensive applications, is another challenging research direction in the next decade.

Developing software that meets the high-quality standards expected for business-critical applications remains a challenge, and quality-driven development methodology and new tools need to be created. Building on the principles of model-driven development and on popular standards, e.g., UML or MARTE, such an approach will guide the simulation, verification, and quality evolution of big data applications.

Further exploiting the scalability, availability, and elasticity of cloud computing for model building and exposing of prediction and analytics as hosted services is opening a competitive and challenging market. Tools and frameworks to support the integration of mobile and sensor data into cloud platforms need to be further developed.

At the end of this chapter, we review some of the best practices that the recently published literature recommends, namely:

- Understand the business problem. Having a well-defined problem, knowing specific constraints available for the problem under investigation, can greatly improve performance of the ML algorithms.
- Understand the ML task. Is it supervised or unsupervised? What activities are required to get the data labeled? The same features (attributes, domains, labels) need to be available at both times, training and testing. Pick a machine learning method appropriate to the problem and the data set. This is the most difficult task, and here are some questions you should consider: Do human users need to understand the model? Is the training time a constraint for your problem? What is an acceptable trade-off between having an accurate answer and having the answer quickly? Keep in mind that there is no single best algorithm; experiment with several algorithms and see which one gives better results for your problem.
- In case of predictive modeling, carefully select and partition the data at hand in training and validation set, which will be used to build the model, versus the test

set that you will use to test the performance of your model. More data for training the model results in better predictive performance. Better data always beats a better algorithm, no matter how advanced it is. Visualize the data with at least univariate histograms. Examine correlations between variables.

- Well prepare your data. Deal with missing and invalid values (misspelled words, values out of range, outliers). Take enough time, because no matter how robust a model is, poor data will yield poor results.
- Evaluate your model using confusion matrix, ROC (receiver operating characteristic) curve, precision, recall, or F1 score. Do not overfit your model, because the power lies in good prediction of unseen examples.
- Use proper tools for your problems. For low-level programming environments you might find difficult to use, try first the machine learning services offered by cloud service providers, which are easy to use and powered by state-of-the-art algorithms.

Acknowledgments This work was supported by the European Commission H2020 co-funded with project DICE (GA 644869).

References

1. Agarwal A, Chapelle O, Dudik M, Langford J (2014) A reliable effective terascale linear learning system. J Mach Learn Res 15:1111–1133
2. Assunção MD, Calheiros RN, Bianchi S, Netto MAS, Buyya R (2014) Big data computing and clouds: trends and future directions. J. Parallel Distrib Comput. http://dx.doi.org/10.10.16/j.jpdc.2014.08.003
3. Bekkerman R, Bilenko M, Langford J (eds) (2012) Scaling up machine learning: parallel and distributed approaches. Cambridge University Press, Cambridge
4. Berthold MR, Cebron N, Dill F, Gabriel TR, Kötter T, Meinl T, Ohl P, Sieb C, Thiel K, Wiswedel B (2008) KNIME: The Konstanz Information Miner. In: Preisach C, Burkhardt H, Schmidt-Thieme L, Decker R (eds) Studies in classification, data analysis, and knowledge organization. Springer, Berlin/Heidelberg
5. Budiu M, Fetterly D, Isard M, McSherry F, Yu Y (2012) Large-scale machine learning using DryadLINQ. In: Bekkerman R, Bilenko M, Langford J (eds) Scaling up machine learning. Cambridge University Press, Cambridge
6. Chandola V, Banerjee A, Kumar V (2009) Anomaly detection: a survey. ACM Comput Surv 41(3):15:1–15:58
7. Charrington S (2012) Three new tools bring machine learning insights to the masses, February, Read Write Web. http://www.readwriteweb.com/hack/2012/02/three-new-tools-bring-machine.php
8. Dai W et al (2015) High-performance distributed ML at scale through parameter server consistency models, AAAI
9. Eckerson W (2012) New technologies for big data. http://www.b-eye-network.com/blogs/eckerson/archives/2012/11/new_technologie.php
10. Franklin M et al (2015) MLlib: Machine Learning in apache Spark
11. Gander M et al (2013) Anomaly detection in the cloud: detecting security incidents via machine learning, trustworthy eternal systems via evolving software, data and knowledge, vol 379. Springer, Berlin/Heidelberg, pp 103–116

12. Ghoting A, Kambadur P, Pednault E, Kannan R (2011) NIMBLE: a toolkit for the implementation of parallel data mining and machine learning algorithms on MapReduce. In: Proceedings of the 17th ACM SIGKDD international conference on Knowledge Discovery and Data mining KDD'11, ACM, New York, NY, USA, pp 334–342
13. Ghoting A, Krishnamurthy R, Pednault E, Reinwald B, Sindhwani V, Tatikonda S, Tian Y, Vaithyanathan S (2011) SystemML: declarative machine learning on MapReduce. In: Proceedings of the 2011 I.E. 27th International Conference on Data Engineering (ICDE '11). IEEE Computer Society, Washington, DC, USA, pp 231–242
14. Granger B, Perez F, Ragan-Kelley M (2011) Using IPython for parallel computing. http://minrk.github.com/scipy-tutorial-2011. Accessed 13 May 2015
15. Grisel O (2013) Advanced machine learning with scikit-learn, PYCON tutorial. https://us.pycon.org/2013/schedule/presentation/23/
16. Hall M et al (2009) The WEKA data mining software: an update. ACM SIGKDD Explor Newsl 11(1):10–18
17. Hall M (2013) Weka and Spark – http://markahall.blogspot.co.nz/. Accessed 13 May 2015
18. Harris D (2015) 5 low-profile startups that could change the face of big data. http://gigaom.com/cloud/5-low-profile-startups-that-could-change-the-face-of-big-data/. Accessed 15 July 2015
19. Hido S, Tokui S, Oda S (2013) Jubauts: an open source platform for distributed online machine learning, NIPS workshop on Big Learning, Lake Taho
20. Hofmann M, Klinkenberg R (2013) RapidMiner: data mining use cases and business analytics applications. Chapman &Hall/CRC, Boca Raton
21. Isard M et al. (2007) Dryad: distributed data-parallel programs from sequential building blocks. SIGOPS Oper Syst Rev 41:59–72. doi:10.1145/1272998.1273005
22. Jain A, Nalya A (2014) Learning storm. Packt Publishing, Birmingham
23. Nuggets KD (2014) http://www.kdnuggets.com/polls/2014/languages-analytics-data-mining-data-science.html. Accessed 15 May 2015
24. Krizhevsky A, Sutskever I, Hinton GE ImageNet (2012) Classification with deep convolutional neural networks. NIPS 2012: neural information processing systems, Lake Tahoe, Nevada
25. Le Q, Ranzato MA, Monga R, Devin M, Chen K, Corrado G, Dean J, Ng A (2012) Building high-level features using large scale unsupervised learning, international conference in machine learning, Edinburgh, UK
26. Leo S, Zanetti G (2010) Pydoop: a Python MapReduce and HDFS API for Hadoop. In: Proceedings of the 19th ACM international symposium on high performance distributed computing, Chicago, IL, USA, pp 819–825
27. Low Y et al. (2012) Distributed GraphLab: a framework for machine learning and data mining in the cloud. In: Proceedings of the VLDB endowment, vol 5, no 8, August 2012, Istanbul, Turkey
28. Mohri M, Rostamizadeh A, Talwalkar A (2012) A foundations of machine learning. The MIT Press, Cambridge, MA
29. Owen S, Anil R, Dunning T, Friedman E (2011) Mahout in action. Manning Publications Co., Shelter Island
30. Patcha A, Park JM (2007) An overview of anomaly detection techniques: existing solutions and latest technological trends. Comput Netw Elsevier, North-Holland, Inc., 51:3448–3470
31. Pednault E, Yom-Tov E, Ghoting A (2012) IBM parallel machine learning toolbox. In: Bekkerman R, Bilenko M, Langford J (eds) Scaling up machine learning. Cambridge University Press, New York
32. Pedregosa F et al (2011) Scikit-learn: machine learning in Python. J Mach Learn Res 12:2825–2830
33. Piccolboni A (2015) RHadoop. https://github.com/RevolutionAnalytics/RHadoop/wiki. Accessed 13 May 2015
34. Roldn MC (2013) Pentaho data integration beginner's guide. Packt Publishing, Birmingham

35. Rosen J et al (2013) Iterative MapReduce for large scale machine learning, CoRR, abs/1303.3517
36. Russom P Big data Analytics (2011) TDWI best practices report, The Data Warehousing Institute (TDWI) Research
37. Sagha H, Bayati H, Millán JDR, Chavarriaga R (2013) On-line anomaly detection and resilience in classifier ensembles. Pattern Recogn Lett, Elsevier Science Inc., 34:1916–1927
38. Shi Q et al (2009) Hash kernels for structured data. J Mach Learn Res JMLR.org, 10:2615–2637
39. Elston SF (2015) Data science in the cloud with Microsoft Azure Machine Learning and R, O'Reilly
40. Thearling KK (1995) Massively parallel architectures and algorithms for time series analysis. In: Nadel L, Stien D (eds) Lectures in complex systems. Addison-Wesley, Reading
41. Tierney L, Rossini AJ, Snow NL (2009) A parallel computing framework for the R system. Int J Parallel Prog 37:78–90. doi:10.1007/s10766-008-0077-2
42. Upadhyaya SR (2013) Parallel approaches to machine learning – a comprehensive survey. J Parallel Distrib Comput 73(3):284–292. ISSN 0743–7315. http://dx.doi.org/10.1016/j.jpdc.2012.11.001
43. Wei D, Wei J, Zheng X, Kim JK, Lee S, Yin J, Ho Q, Xing EP (2013) Petuum: a framework for iterative-convergent distributed ML. arxiv.org/abs/1312:7651
44. Zaharia M et al. Spark: cluster computing with working sets. In: Proceedings of the 2nd USENIX conference on hot topics in Cloud, Computing, USENIX Association, pp 10–10