Neurocomputing 237 (2017) 350–361

Contents lists available at ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom

Brief papers

# Machine learning on big data: Opportunities and challenges

Lina Zhou[a,*], Shimei Pan[a], Jianwu Wang[a], Athanasios V. Vasilakos[b]

[a] Information Systems Department, UMBC, Baltimore, MD 21250, United States
[b] Department of Computer Science, Electrical and Space Engineering, Luleå University of Technology, SE-931 87 Skellefteå, Sweden

ARTICLE INFO

Communicated by Guoliang Wei

*Keywords:*
Machine learning
Big data
Data preprocessing
Evaluation
Parallelization

ABSTRACT

Machine learning (ML) is continuously unleashing its power in a wide range of applications. It has been pushed to the forefront in recent years partly owing to the advent of big data. ML algorithms have never been better promised while challenged by big data. Big data enables ML algorithms to uncover more fine-grained patterns and make more timely and accurate predictions than ever before; on the other hand, it presents major challenges to ML such as model scalability and distributed computing. In this paper, we introduce a framework of ML on big data (MLBiD) to guide the discussion of its opportunities and challenges. The framework is centered on ML which follows the phases of preprocessing, learning, and evaluation. In addition, the framework is also comprised of four other components, namely big data, user, domain, and system. The phases of ML and the components of MLBiD provide directions for identification of associated opportunities and challenges and open up future work in many unexplored or under explored research areas.

## 1. Introduction

Machine learning (ML) techniques have generated huge societal impacts in a wide range of applications such as computer vision, speech processing, natural language understanding, neuroscience, health, and Internet of Things. The advent of big data era has spurred broad interests in ML. ML algorithms have never been better promised and also challenged by big data in gaining new insights into various business applications and human behaviors. On the one hand, big data provides unprecedentedly rich information for ML algorithms to extract underlying patterns and to build predictive models; on the other hand, traditional ML algorithms face critical challenges such as scalability to truly unleash the hidden value of big data. With an ever-expanding universe of big data, ML has to grow and advance in order to transform big data into actionable intelligence.

ML addresses the question of how to build a computer system that improves automatically through experience [1]. A ML problem is referred to as the problem of learning from experience with respect to some tasks and performance measures. ML techniques enable users to uncover underlying structure and make predictions from large datasets. ML thrives on efficient learning techniques (algorithms), rich and/or large data, and powerful computing environments. Thus, ML has great potential for and is an essential part of big data analytics [2].

This paper focuses on ML techniques in the context of big data and modern computing environments. Specifically, we aim to investigate opportunities and challenges of ML on big data. Big data presents new opportunities for ML. For instance, big data enables pattern learning at multi-granularity and diversity, from multiple views in an inherently parallel fashion. In addition, big data provides opportunities to make causality inference based on chains of sequence. Nevertheless, big data also introduces major challenges to ML such as high data dimensionality, model scalability, distributed computing, streaming data [3], adaptability, and usability. In this paper, we introduce a framework of ML on big data (MLBiD) to guide the discussion of its opportunities and challenges. The framework is centered on ML which follows the phases of preprocessing, learning, and evaluation. In addition, the framework is also comprised of four other components that both influence and are influenced by ML, namely big data, user, domain, and system. The components of MLBiD and the phases of ML provide directions for identification of opportunities and challenges and open up future work in many unexplored or under explored research areas.

## 2. A framework of machine learning on big data

The framework of ML on big data (MLBiD) is shown in Fig. 1. MLBiD is centered on the *machine learning (ML) component*, which interacts with four other components, including *big data*, *user*, *domain*, and *system*. The interactions go in both directions. For instance, *big data* serves as inputs to ML and the latter generates outputs, which in turn become a part of big data; *user* may interact with ML by providing domain knowledge, personal preferences and usability feedback, and by leveraging learning outcomes to improve

* Corresponding author.
*E-mail addresses:* zhoul@umbc.edu (L. Zhou), shimei@umbc.edu (S. Pan), jianwu@umbc.edu (J. Wang), athanasios.vasilakos@ltu.se (A.V. Vasilakos).

http://dx.doi.org/10.1016/j.neucom.2017.01.026
Received 13 September 2016; Received in revised form 20 December 2016; Accepted 5 January 2017
Available online 12 January 2017
0925-2312/© 2017 Elsevier B.V. All rights reserved.

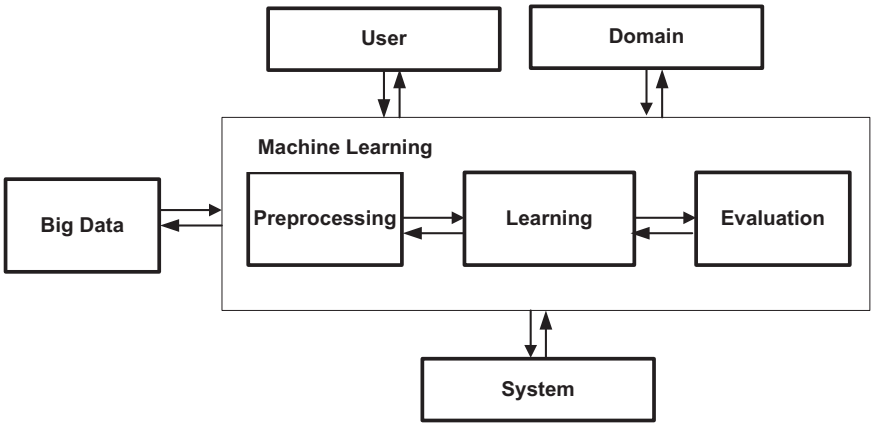


**Fig. 1.** A framework of machine learning on big data (MLBiD).

decision making; *domain* can serve both as a source of knowledge to guide ML and as the context of applying learned models; *system architecture* has impact on how learning algorithms should run and how efficient it is to run them, and simultaneously meeting ML needs may lead to a co-design of system architecture. Next, we introduce each component of MLBiD separately.

### 2.1. Machine learning

ML typically goes through data preprocessing, learning, and evaluation phases (see Fig. 1). Data preprocessing helps prepare raw data into the "right form" for subsequent learning steps. The raw data is likely to be unstructured, noisy, incomplete, and inconsistent. The preprocessing step transforms such data into a form that can be used as inputs to learning through data cleaning, extraction, transformation, and fusion. The learning phase chooses learning algorithms and tunes model parameters to generate desired outputs using the preprocessed input data. Some learning methods, particularly representational learning, can also be used for data preprocessing. The evaluation follows to determine the performance of the learned models. For instance, performance evaluation of a classifier involves dataset selection, performance measuring, error-estimation, and statistical tests [4]. The evaluation results may lead to adjusting the parameters of chosen learning algorithms and/or selecting different algorithms.

ML can be characterized in multiple dimensions: nature of learning feedback, target of learning tasks, and timing of data availability. Accordingly, we propose a multi-dimensional taxonomy of ML, as shown in Fig. 2.

- Based on the nature of the feedback available to a learning system, ML can be classified into three main types: supervised learning, unsupervised learning, and reinforcement learning [5]. In supervised learning, a learning system is presented with examples of input-output pairs, and the goal is to learn a function that maps inputs to outputs. In unsupervised learning, the system is not provided with explicit feedback or desired output, and the goal is to uncover patterns in the input. As in unsupervised learning, a reinforcement learning system is not presented with input-output pairs. Like supervised learning, the reinforcement learning is given feedback on its previous experiences. Unlike supervised learning, however, the feedback in reinforcement learning is rewards or punishments associated with actions instead of desired output or explicit correction of sub-optimal actions. Semi-supervised learning falls between supervised and unsupervised learning, where the system is presented with both a small number of input-output pairs and a large number of un-annotated inputs. The goal of semi-supervised learning is similar to supervised one except that it learns from both annotated and un-annotated data.
- Based on whether the target of learning is specific tasks using input features or the features themselves, ML can be categorized into representational learning and task learning. Representational learning aims to learn new representations of data that make it easier to extract useful information when building classifiers or other predictors [6]. A good representation is one that disentangles the underlying factors of variation. It is often one that captures the posterior distribution of underlying exploratory factors for the observed output in case of probabilistic models [6].

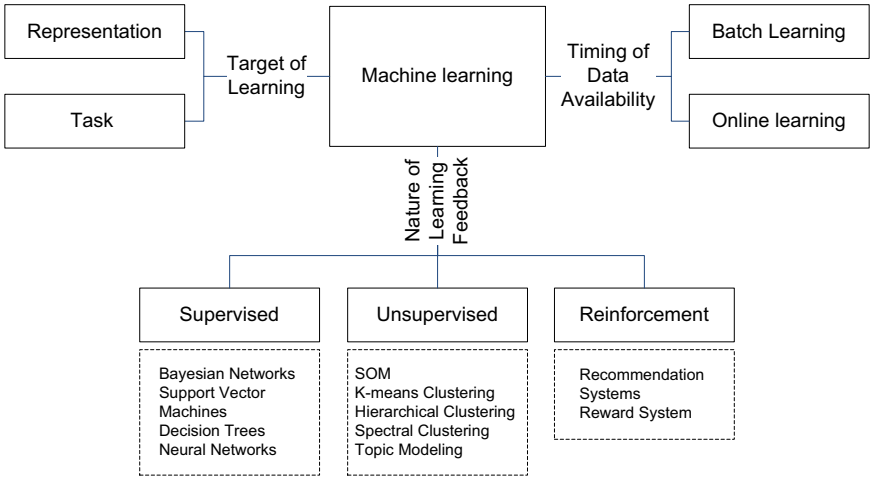


**Fig. 2.** A multi-dimensional taxonomy of machine learning.

Representation learning is often entangled with density estimation and dimensionality reduction. Density estimation finds the underlying probability density function of a random variable. Dimensionality reduction maps inputs from a high-dimensional space to a space of lower-dimensionality. It is often difficult to establish a clear objective or target in representation learning. In contrast, task learning typically has desired outputs, and accordingly is categorized into classification, regression, and clustering. In classification, ML techniques produce a model that assigns unseen inputs to one or more pre-defined classes. Regression differs from classification in that its outputs are continuous rather than discrete values. Clustering produces groups of data, and these groups are not known in advance, which distinguishes itself from classification. Traditionally, classification and regression are referred to as supervised learning, and clustering as unsupervised learning. Their representative algorithms are also shown in Fig. 2.

- Based on the timing of making training data available (e.g., whether the training data are available all at once or one at a time), ML can be classified into batch learning and online learning. Batch learning generates models by learning on the entire training data, whereas online learning updates models based on each new input. A batch learning algorithm assumes that data are independent and identically distributed or drawn from the same probability distribution, which is usually not satisfied by real data. Online learning typically makes no statistical assumptions about the data [7]. Although a batch learning algorithm is expected to generalize, there is no notion of generalization for online learning because the algorithm is only expected to accurately predict the labels of examples that it receives as input [7]. Online learning is used when it is computationally infeasible to train over the entire dataset and/or when the data is being generated over time and a learning system needs to adapt to new patterns in the data.

Each ML algorithm can be categorized in multiple dimensions. For instance, conventional decision trees belong to supervised batch learning algorithms.

### 2.2. Big data

Big data has been characterized by five dimensions: volume (quantity/amount of data), velocity (speed of data generation), variety (type, nature, and format of data), veracity (trustworthiness/quality of captured data), and value (insights and impact). We organized the five dimensions into a stack, consisting of *big*, *data,* and *value* layers starting from the bottom (see Fig. 3). The *big* layer is the most fundamental and the *data* layer is central to big data, and the *value* aspect characterizes impact of big data real world applications. The lower layer (e.g., volume and velocity) depends more heavily on technological advances, and higher layer (e.g., value) is more oriented toward applications that harness the strategic power of big data. In order to realize the value of big data analytics and to process big data efficiently, existing ML paradigms and algorithms need to be adapted.

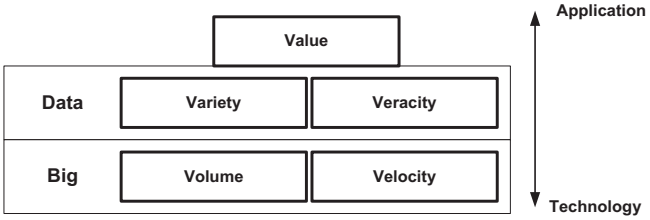


**Fig. 3.** Big data stack.

### 2.3. Other Components

#### 2.3.1. Users

There are a variety of stakeholders of ML systems such as domain experts, end-users, and ML researchers and practitioners. Traditionally, it is the ML practitioners who make most decisions in applying ML, starting from data collection all the way to performance evaluation. End-user involvement during this process has been limited to providing data labels, answering domain-related questions, or giving feedback about the learned results, which is typically mediated by the practitioners, leading to lengthy and asynchronous iterations [8]. However, end-users are inclined to provide more than just data labels. They value transparency in the design of a learning system, which in turn helps them understand the system and provide better labels/feedback. Involving users in ML can potentially lead to more effective learning systems and better user experiences [8]. For instance, interactive ML [8] allows users to interactively examine the impact of their actions and adapt subsequent inputs to steer ML behaviors to obtain desired outputs.

#### 2.3.2. Domain

Domain knowledge facilitates ML in discovering interesting patterns that may not be discoverable from datasets alone. Training datasets may not be sufficiently large and/or representative to enable the discovery of all patterns. It is also costly and even infeasible to obtain sufficient and representative data, possibly due to great domain variation and application-specific requirements. Domain knowledge can help improve the generality and robustness of patterns induced from datasets [9]. There are several ways to incorporate prior domain knowledge into inductive ML [10]: (1) preparing training examples; (2) generating hypotheses or hypothesis space; (3) modifying the search objective; and (4) augmenting the search. These learned patterns can in turn be used to update and refine the domain knowledge.

#### 2.3.3. System

System architecture or platform, which consists of both software and hardware, creates an environment in which ML algorithms can run. For instance, compared with their simpler counterparts, a multi-core machine with distributed architecture is expected to improve the efficiency of ML. New framework and system architecture such as Hadoop/Spark have been proposed to address the challenges of big data. Nevertheless, migrating existing ML algorithms to distributed architecture requires modifications to how ML algorithms are implemented and deployed. In addition, the unique needs and values of ML may inspire the design and development of new system architecture.

Based on the MLBiD framework, we identify important opportunities and key challenges. We discuss them for each of the three phases in ML– preprocessing, learning, and evaluation, separately.

## 3. Data preprocessing opportunities and challenges

Much of the actual effort in deploying an ML system goes into the design of preprocessing pipelines and data transformations that result in a representation of data that can support effective ML [6]. Data preprocessing aims to address a number of issues such as data redundancy, inconsistency, noise, heterogeneity, transformation, labeling (for (semi-)supervised ML), data imbalance and feature representation/selection. Data preparation and preprocessing is usually costly, due to the requirement of human labor and a large number of options to choose from. Additionally, some conventional data assumptions do not hold for big data, consequently some preprocessing methods become infeasible. On the other hand, big data creates the opportunity of reducing the reliance on human supervision by learning from massive, diverse, and streaming data sources directly.

### 3.1. Data redundancy

Duplication arises when two or more data samples represent the same entity. The impact of data duplication or inconsistency on ML can be severe. Despite a range of techniques for identifying duplicates developed in the past 20 years [11], traditional methods such as pairwise similarity comparison is no longer feasible for big data. In addition, the traditional assumption that duplicated pairs are minority compared with non-duplicated pairs no longer holds. To this end, Dynamic Time Warping can be much faster than the state-of-the-art Euclidean distance algorithms [12].

### 3.2. Data noise

Missing and incorrect values, data sparsity, and outliers can introduce noise to ML. Traditional solutions to noisy data problem face challenges in dealing with big data. For instance, manual methods are no longer feasible due to its lack of scalability; replacement by mean would lose the advantages of the richness and fine granularity of big data. In some cases, interesting patterns may lie in these noisy data, so simple deletion may not be a wise alternative. Accurate predictive analytics of big data can be used to estimate missing values, such as replacing incorrect readings due to malfunctioned sensors or broken communication channels. To address considerable bias that may be introduced into predictions by collective influence methods, maximum entropy constraint has been imposed on the inference step, forcing the predictions to have the same distribution as observed labels [13]. Despite that data sparsity may remain and even be aggravated by big data, the sheer size of big data creates unique opportunities to enable predictive analytics because there could be sufficient frequency accumulated for different sub-samples. There have been efforts to scale up outlier detection (e.g., ONION [14]) to enable analysts to effectively explore anomalies in large datasets [14].

### 3.3. Data heterogeneity

Big data promise to offer multi-view data from different types of repositories, in disparate formats, and from different samples of the population and thus are highly heterogeneous. These multi-view heterogeneous data (e.g., unstructured text, audio, and video formats [15]) might have varying level of importance for a learning task. Thus, concatenating all the features by treating them equally important will unlikely lead to optimal learning outcomes. Big data present an opportunity for learning from multiple views in parallel and then ensembling multiple results by learning the importance of feature views to the task. The method is expected to be robust to data outliers and can address optimization difficulty and convergence issues [16].

### 3.4. Data discretization

Some ML algorithms such as decision trees and Naïve Bayes can only deal with discrete attributes. Discretization translates quantitative data into qualitative data, procuring a non-overlapping division of a continuous domain. The purpose of attribute discretization is to find concise data representations as categories, which are adequate for the learning task to retain as much information in the original continuous attribute as possible. However, when coping with big data, most of existing discretization approaches will not be efficient. To address the big data challenges, standard discretization methods have been parallelized by developing a distributed version of the entropy minimization discretizer based on Minimum Description Length Principle in big data platforms, boosting both performance and accuracy [17]. In another study [18], the data is first sorted based on the value of a numerical attribute, and then split into fragments of the original class attribute. These fragments, which are summarized by the percentage composition of different classes, are viewed as super instances and the target of discretization.

### 3.5. Data labeling

Traditional data annotation methods are labor-intensive. Several alternative methods have been suggested to address the challenge of big data. For instance, online crowd-generated repositories can serve as a source for free annotated training data, which can capture a large variety in terms of both class number and intra-class diversity [19]. In addition, human-level concept learning can be achieved through probabilistic program induction [20]. Furthermore, the ability of labeling data is built into ML algorithms such as semi-supervised learning, transfer learning, and active learning (e.g., [21,22]). By using active learning as the optimization strategy for labeling tasks in crowd-sourced databases, one can minimize the number of questions asked to the crowd, allowing crowd-sourced applications to scale. However, designing active learning algorithms for a crowd-sourced dataset poses many practical challenges, such as generality, scalability, and ease of use [23]. Another issue is that such a dataset may not cover all user-specific contexts, which may often result in significantly worse performance than that of user-centric training [19].

### 3.6. Imbalanced data

The problem of imbalanced data has been addressed by traditional stratified random sampling methods. However, the process can be very time-consuming if it involves iterations of sub-sample generation and error metrics calculation. In addition, traditional sampling methods cannot efficiently support data sampling over a user-specified subset of data that includes value-based sampling. Big data necessitates parallel data sampling. For instance, a parallel sampling framework has been proposed to generate sample dataset out of the original dataset based on multiple distributed index files [24]. The parallel level can be selected based on the dataset size and the available processes.

### 3.7. Feature representation and selection

The performance of ML is heavily dependent upon the choice of data representation or features [6]. A ML algorithm's generalizability depends on the dataset, which also indirectly depends on the features that represent a salient structure of the dataset. Feature selection helps enhance the performance of ML by identifying prominent features. It essentially selects different subsets of features and data, and aggregates them at different levels of granularity, which contributes to reducing the amount of big data. However, feature engineering requires prior domain knowledge and human ingenuity and is often labor-intensive [6]. To address the weakness of current feature engineering algorithms when dealing with big data, various solutions have been proposed, such as distributed feature selection [25]; a low-rank matrix approximation (e.g., standard Nyström method [26]); representation learning to make learning algorithms less dependent on feature engineering by learning a generic prior [6]; adaptive feature scaling scheme for ultra high-dimensional feature selection, which iteratively activates a group of features and solves a sequence of multiple kernel learning sub-problems [27]; a unified framework for feature selection based on spectral graph theory, which is able to generate families of algorithms for both supervised and unsupervised feature selection [28]; fuzzy clustering prior to classification, where classification is realized with the center of groups, followed by de-clustering and classification via reduced data [29]; and reducing the size of data dimensions and volumes (e.g., Random Forest-Forward Selection Ranking and Random Forest-Backward Elimination Ranking [30], and linguistic hedges neuro-fuzzy classifier with selected features [31]). Recently, deep neural network-based autoencoding has proven to be very effective in learning video, audio and textual features [32,33].

**Table 1**
A taxonomy of methods/platforms for machine learning on big data.

| Parallelism | Target | Techniques | Sample Studies* |
|---|---|---|---|
| Non-parallel | | Optimization | [41,42] [43] |
| Parallel | data | MapReduce | BN [44,45], DT [38], TM [46], GP [47,48] [49] [50] [51] |
| | | DistributedGraph | GA [52] |
| | | Others | SVM [37], NN [53], GP [36,39] |
| | model/ parameter | Multi-threading | SVM [37] |
| | | MPI/OpenMP | NN [40], TM [46] |
| | | GPU | NN [40,53,54] |
| | | Others | SVM [55], NN [56], GP [36,39] |

* *BN* Bayesian network learning, *DT* decision tree, *TM* topic modeling, *GP* generic platform, *SVM* support vector machine, *NN* neural network

## 4. Learning opportunities and challenges

Developing scalable ML algorithms that is capable to handle large datasets has been a long-standing research theme in the ML community prior to the advent of "big data" era. To better organize the discussion of opportunities and challenges, we propose a taxonomy of methods/platforms for ML on big data, as shown in Table 1.

In the taxonomy, we first categorize studies based on whether any parallelism is considered in their algorithms/platforms. Methods in the non-parallelism category aim to have much faster optimization methods that can deal with big data without any parallelism. Traditionally, ML scalability was mainly focus on developing novel algorithms that can run much more efficiently (e.g., with significantly better time complexity and/or space complexity). For instance, stochastic gradient descent, is a classic example of a scalable ML algorithm that in principle can process massive datasets without the requirement of huge memory [34]. There is also a tradeoff between scalability and convexity, a desirable algorithm property that is amenable to theoretical analysis [34]. It has shown that trading convexity can provide scalability advantages in SVM inference. Similarly, it has been argued [35] that deep architectures such as multi-layer neural network with several hidden layers are more efficient and thus more scalable in representing typical learning tasks such as prediction, visual perception and language understanding than shallow architectures exemplified by modern kernel machines such as SVMs.

The parallelism category reflects the majority of state-of-the-art scalable ML methods. To deal with emerging big data characterized by huge feature dimensions and sample size, methods in this category exploit data geometry in the input and/or algorithm/model space [30]. Specifically, we further classify parallelism methods that make ML algorithms more scalable into two sub-categories: (1) data parallelism: leveraging existing big data architecture, partitioning input data vertically, horizontally, or even arbitrarily into manageable pieces, and then computing on all subsets simultaneously, and (2) model/parameter parallelism: creating parallelized versions of learning algorithms by first dividing the learning model/parameters and then computing on each structural block concurrently. We note that some efforts such as [36–39] support both data parallelism and model/parameter parallelism. Some other efforts such as [40] support more than one type of parallelism techniques. In the following, we discuss each type of the methods and key opportunities and challenges in big data learning.

### 4.1. Non-parallelism

Optimization lies at the heart of most ML approaches. Traditional optimization methods are categorized into combinatorial optimization (greedy search, beam search, branch-and-bound) and continuous optimization [57]. The latter is further grouped into unconstrained (e.g., gradient descent, conjugate gradient, quasi-Newton methods) and constrained optimization (e.g., linear programming, quadratic programming). Constrained optimization is often costly especially when the training dataset is large. One possible solution is to compute an approximate optimum. The state-of-the-art large-scale optimization algorithms use various stochastic gradient descent, stochastic coordinate descent and distributed optimization, and particularly randomized approximation algorithms, to learn from large-scale data [41]. Nevertheless, stochastic gradient descent methods are difficult to tune and parallelize [58] and unlikely to have amazing performance for large-scale problems.

Parameter optimization is computationally challenging for learning methods with many hyperparameters. For large scale learning problems, it is desirable to obtain the optimal model parameters by going through the data in only one pass [42]. To this end, second order stochastic gradient and averaged stochastic gradient are asymptotically efficient after a single pass on the training set [43]. In addition, pushing analytical functions (mapping models) into big data architecture is another alternative to making parameter optimization feasible on a massive scale [59]. There have been studies on how to parallelize constrained optimization methods used in many learning algorithms such as SVMs, nonnegative least square problems, and L1 regularized regression (LASSO) problems [29]. By converting these learning problems into a series of matrix-vector multiplication operations, parallelization can be implemented straightforwardly using MapReduce or GPU parallelization programming models. Furthermore, the use of limited memory BFGS and conjugate gradient with line search can significantly simplify and speed up the process of unsupervised feature learning and pretraining deep algorithms using stochastic gradient descent methods, particularly when considering sparsity regularization and GPUs or computer clusters [58].

### 4.2. Data parallelism

Existing ML models could utilize big data techniques to achieve scalability. Such efforts can be classified into two categories. One is to provide a general middleware layer that re-implements existing learning tasks so they can run on a big data platform such as Hadoop and Spark. Such a middleware layer often provides general primitives/operations that are useful for many learning tasks. This approach is suitable for users who want to try different learning tasks/algorithms within the same framework. The other category is to transform individual learning algorithms to run on a big data platform. These implementations are normally built directly on top of a big data engine and could achieve better scalability or result.

#### 4.2.1. General big data middleware for existing learning algorithms

Spark MLlib [47] and Mahout [48] are two representative open-source projects/packages that support many scalable learning algorithms. Many common learning algorithms, including classification, regression, clustering, collaborative filtering and dimensionality reduction, are supported by both Spark MLlib and Mahout. Because they provide an independent layer that separates front-end algorithms from a back-end execution engine, it is easy to switch from one big data engine to another. For instance, Mahout supports Hadoop, Spark and H2O as its big data engines. Further, although these algorithms can be used to process large datasets in a distributed environment, their usage is very similar to those that run on a single machine with a small dataset. In addition, this independent layer enables optimization between logic plans from users and physical plans that can be executed in a distributed environment. There are also projects for large-scale stream data learning, such as SAMOA [71]. Yet they are still at an early stage.

#### 4.2.2. Efforts on specific algorithms with parallel data

Although the above middleware supports many common learning algorithms for big data, there are still both practical needs and research interests to extend individual ML algorithms to support big data, especially for less widely-used or new algorithms. Many ML algorithms can be implemented by MapReduce, including linear regression, k-means, logistic regression, Naive Bayes, SVM, ICA, PCA, EM, Neural network, etc. [49]. Simple multiplicative algorithms can be used to straightforwardly implement ML algorithms such as SVMs and non-negative least square problems in parallel computational environments using MapReduce [60]. MapReduce has also been used to achieve parallel spatial data co-location mining [61], nearest neighbor classification [62], and Bayesian Network learning [44,45]. Many of these new directions utilize increasingly complex ML workflows/pipelines which require systems to use a combination of state-of-the art tools and techniques [63].

### 4.3. Models/parameter parallelism

A host of efforts have been put on how to parallelize ML algorithms (e.g., [64]) or seek performance guarantees on various parallelized algorithms (e.g., [65]). These efforts are warranted because many ML algorithms are at best trivially parallel [66–68]. In addition, big data ML is not simply a scaled-up version of small data ML. It requires different formulations and novel algorithms to address its associated technical challenges. Parallelization of learning algorithms has roots in distributed ML and large-scaled ML. Thus, we discuss opportunities and challenges of ML on big data from the following perspectives: distributed ML, parallelization in several primary ML paradigms, and deep learning.

#### 4.3.1. Distributed machine learning

Distributed ML can naturally solve the algorithm complexity and memory limitation problem in large-scale ML [69]. To address the inability of ML algorithms to use all the data to learn within a reasonable amount of time, distributed ML scales up learning algorithms by allocating the learning process on multiple computers or processors [69], and solving a distributed optimization problem [70]. Distributed ML can achieve not only efficiency by parallel data loading but also fault tolerance by replicating data across machines. Moreover, using different learning processes to train several classifiers from distributed data sets increases the possibility of achieving higher accuracy especially on a large domain [69]. Another advantage of distributed algorithms is that they can be integrated with other parts of data management (e.g., [71]). However, designing and implementing efficient and provably correct parallel algorithms is extremely challenging [52]. Additionally, traditional parallel ML algorithms distribute computation to nodes, which works well in dedicated parallel machines with fast communication among nodes but not so well when data are transferred across networks, which leads to high communication cost due to network latencies. Thus, accessing data from local disks is highly preferred. But most ML algorithms are not designed to achieve good data locality. Further, the communication delays between different machines may cause problems in convergence even though a non-distributed algorithm shows a good convergence rate [70].

GraphLab is a parallel framework for ML which exploits the sparse structure and common computational patterns of ML algorithms [52]. It enables ML experts to easily design and implement efficient scalable parallel algorithms by composing problem-specific computation, data-dependencies, and scheduling, in a shared-memory multiprocessor setting. For example, *bagging* trains several subsets and assembles the results of several linear classifications. Nevertheless, scaling up ensemble ML techniques to large and distributed data remains a research challenge. Adaboosting of Extreme Learning Machine has been explored by leveraging the power of MapReduce to build a reliable predictive bag of classification models [72]. These models can produce good generalization performance and efficiency. A weight based ensemble algorithm is proposed to learn a Bayesian Network structure from an ensemble of local results, which also employs Kepler scientific workflow to build the whole learning process [44].

#### 4.3.2. Parallelization of traditional ML algorithms

Here we discuss the parallelization of several traditional supervised ML algorithms, including SVM with Gaussian and polynomial kernels, Bayesian networks, and decision trees.

Support vector machines (SVMs) have been promising classification methods due to their solid mathematical foundations, which exhibit two prominent properties: margin maximization and nonlinear classification using kernels. However, kernel-based SVM algorithms suffer from a scalability problem as they require computing a kernel matrix with $O(N^2)$ time and space complexity [73]. SVM was recently adapted to the field of high performance computing through power/performance prediction, auto-tuning, and runtime scheduling [37]. Parallel SVMs has become one of the best out-of-the-box classification methods [74]. The key idea is to introduce a parallel optimization step to quickly remove most of nonsupport vectors, where block diagonal matrices are used to approximate the original kernel matrix so that the original problem can be split into hundreds of subproblems which can be solved more efficiently. In addition, some effective strategies such as kernel caching and efficient computation of kernel matrix are integrated to speed up the training process [55], and to substantially reduce computation and memory overhead required to compute the kernel matrix, without significantly impacting results accuracy [73].

Bayesian networks are a powerful probabilistic representation (Graphical models). There have been substantial recent developments on adaptive, flexible and scalable Bayesian learning. Big Bayesian learning includes nonparametric Bayesian methods for adaptively inferring model complexity, regularized Bayesian inference for improving the flexibility via posterior regularization, and scalable algorithms and systems based on stochastic subsampling and distributed computing for dealing with large-scale applications [75].

Decision trees have been known for superior interpretability of their learning results. Random forest has demonstrated its effectiveness for predictive analytics on high-dimensional data in various applications (e.g., [76]). Moreover, parallel learning of tree ensembles using MapReduce on computer clusters has been used to construct scalable classification and regression trees [38].

#### 4.3.3. Deep learning

Recently, deep neural network-based learning becomes one of the fastest growing and most exciting areas of ML with big data. Neural networks are a family of models inspired by biological neural networks that consist of interconnected neurons whose connections can be tuned and adapted to inputs. Deep neural networks can be simply viewed as neural networks with many large hidden layers, or deep-layered architecture with each layer applying a nonlinear transformation from its input to its output. Recently, big data and novel techniques to train deeper networks, along with the availability of more powerful computers (e.g., faster CPUs and the advent of general purpose GPUs), faster network connectivity, and better software infrastructure have created great opportunities for deep learning research. For example, various deep learning software and libraries including Theano [53], Caffe [54], Torch7 [40], Tensorflow [36], are created to empower innovative GPU-accelerated deep learning applications. Among them, Theano was originally developed as a symbolic math processor for symbolic differentiation or integration, on complicated non-linear functions. It has later been widely adopted by neural network and ML researchers as a useful environment for developing new ML algorithms. The Caffe framework includes a large repository of pre-trained neural network models suitable for a variety of image classification tasks. Moreover, Google's Tensorflow is an open source software library for numerical computation using data flow graphs. With Tensorflow, computation

can be deployed to one or more CPUs or GPUs efficiently. In the past few years, deep learning has witnessed tremendous growth in a wide range of applications, including image processing and computer vision [77–81], speech and natural language processing [82,83], health [84], and so on.

Typically, deep neural networks can be trained in two different modes (1) supervised training in which a large number of task-related labeled ground truth data is available (2) self-taught learning (sometimes also called unsupervised learning) in which training data can be automatically generated from unlabeled data without much human effort [85]. For example, ImageNet [79] is an image dataset with over 14 million images labeled with over 20 thousand concepts. Images of each concept are quality-controlled and human-annotated. This labeled dataset is frequently used in training many of the state-of-the-art image recognition systems that employ deep learning [77]. As of 2015, a rough rule of thumb is that a supervised deep learning algorithm will generally achieve acceptable performance with around 5000 labeled examples per category, and will match or exceed human performance when trained with a dataset containing at least 10 M labeled examples [86].

Deep learning algorithms can also take advantage of a huge amount of unsupervised data to automatically learn complex representation [3]. The best results obtained on supervised learning tasks often involve an unsupervised feature learning step [87]. For example, in Natural Language Processing (NLP), unsupervised learning of word embeddings [88] has proven to be very effective in many NLP tasks. Although no annotated data are required to train a model, the system automatically learns a neural network model that is capable of deriving a vector representation of a word based on how well it can correctly predict the neighboring words in its context. In image processing, an autoencoder consisting of an encoder and a decoder is often used for unsupervised feature learning where the encoder uses raw data (e.g., image) as input and produces feature or representation as output, and the decoder uses the extracted feature from the encoder as input and reconstructs the original raw input data as output. The goal is to automatically learn an image representation to minimize the differences between the raw and the reconstructed image.

Deep neural networks displace kernel machines with manually designed features in part because the time and memory cost of training a kernel machine is quadratic in the size of a dataset, and datasets have grown to be large enough for this cost to outweigh the benefits of convex optimization. On the other hand, the availability of labeled data varies greatly from one domain to another. Thus, one key challenge of applying deep learning is to generalize well from smaller datasets by taking advantage of large quantities of unlabeled data, with unsupervised or semi-supervised learning techniques.

### 4.4. Hybrid approaches

Hybrid approaches combine model and data parallelism by partitioning both data and model variables simultaneously. This not only leads to faster learning on distributed clusters, but also enables ML applications to work efficiently when both data and model are too large to fit in the memory of a single machine [46]. For example, DistBelief is a software framework designed for distributed training and learning of deep networks with very large models (e.g., a few billion parameters) and very large data sets. It leverages large clusters of machines to manage both data and model parallelism via multithreading, message passing, synchronization as well as communication between machines [56]. SystemML aims at declarative, large-scale ML on top of MapReduce, in which ML algorithms are expressed as higher-level language scripts. This higher-level language exposes several constructs that constitute key building blocks for a broad class of supervised and unsupervised ML algorithms. The algorithms expressed in SystemML are compiled and optimized into a set of MapReduce jobs that can run on a cluster of machines [50]. One key challenge is how to efficiently combine both types of parallelism for arbitrary ML scripts and workloads.

### 4.5. Key opportunities and challenges

In addition to detailed discussion of opportunities and challenges that big data present to ML throughout this section, here we highlight a few key opportunities and challenges.

ML on big data requires a new way of thinking and novel algorithms to address many technical challenges [89]. Big data is one of the key enablers of deep learning, which has improved the state-of-the-art performance in various applications. Deep learning can typically recognize at least 1000 different categories, which is at least 2 orders of magnitude higher than the typical number of categories handled by a traditional neural network [86]. In addition, big data enables learning at multi-granularity. Furthermore, big data provides opportunities to make causality inference based on chains of sequence, to enable effective decision support.

The need of ML on big data presents unique opportunities for co-design of system and ML. ML can affect how systems are designed. Since many ML programs are fundamentally optimization-centric and admit error-tolerant, iterative-convergent algorithmic solutions, an integrative system design may consider issues such as bounded-error network synchronization and dynamic scheduling based on ML program structure [39]. Hardware accelerations, including a new supercomputer, are under development that only target ML tasks [90,91].

Learning on big data promises a great opportunity for research on workflow management and task scheduling. This is because one key issue in ML on big data is how to divide/schedule the task/data and then integrate multiple predictions. Database query optimization techniques can be utilized to identify effective execution plans, and the resulting runtime plans can be executed on a single unified data-parallel query processing engine [51,92]. A slowdown predictor can be embedded in the map-reduce infrastructure to improve the agility and timeliness of scheduling decisions [93]. The latest Spark MLlib can assemble a sequence of algorithms into a single pipeline, or workflow. It supports scalable execution and automatic parameter tuning for all algorithms within the pipeline.

ML on big data presents an unprecedented opportunity for learning with humans in the loop for several reasons. First, ML on big data requires people with background in both ML algorithms and parallelization techniques, which is very challenging for most users. Thus, there is an increasing attention to the design of ML systems that are easy to understand and easy to use. Second, solely relying on ML algorithms may not bring out the full potential of big data because the algorithms may discover many spurious relationships. Thus, it would be particularly beneficial for ML algorithms to leverage the complementary strengths of human knowledge/expertise. Third, in traditional ML, users often play a passive role (as consumers of ML results). To engage users and help them gain insight into big data, we need to move more toward interactive ML and away from batch ML. Effective interactive ML relies on the design of novel interaction techniques based on an understanding of end-user capabilities, behaviors, and needs [8]. By learning interactively from end-users, ML systems can reduce the need for supervision by experts and empower end-users to create big data ML systems to meet their own needs.

ML on big data also highlights the importance of privacy-preserving ML. Big data may be highly personal [1]. For instance, healthcare data may be collected from multiple organizations that have different privacy policies, and may not explicitly share their data publicly. Since joint ML may sometimes becomes necessary, how to share big data among distributed ML entities while mitigating privacy concerns becomes a challenging problem. For instance, privacy-preservation ML has been achieved by employing the data locality property of Hadoop architecture and only a limited number of cryptographic operations at the Reduce steps are needed [94]. A privacy-preserving solution for

SVM classification has also been proposed [95].

Big data enhances the real-world impact of ML. The applications in which ML has created real world impact range from science (e.g., physical design, biological science, earthquake prediction) to business (e.g., financial systems, post-approval drug monitoring, self-driving cars), and from public platforms (e.g., social media) to organizational realms (e.g., intrusion networks, healthcare systems).

Among the existing challenges for ML on big data, one key issue is to improve the efficiency for iterations. Existing parallel frameworks are not particularly designed for ML algorithms. Usually big data tools perform computation in batch-mode and are not optimized for tasks with iterative processing and high data dependency among operations (e.g., due to heavy disk I/O). Iterative subtasks (i.e., processing steps which are executed repetitively until a convergence condition is met) dominate both categories of algorithms. Optimizing cluster resource allocations among multiple workloads of iterative algorithms often require an estimation of their runtime, which in turn requires: (a) predicting the number of iterations and (b) predicting the processing time of each iteration [96]. The Hadoop infrastructure can both avoid extremely slow, or straggler tasks and handle them at runtime (through speculative execution). Spark [92] supports not only MapReduce and fault tolerance but also cache data in memory between iterations. On a related note, methods have been developed to improve computational efficiency on big data without sacrificing ML performance, which hold only small pieces of the data rather than all data in fast memory, and build a predictor on each small piece and then combine these predictors together [97]. In addition, graph-based architectures and in-memory big data tools have been developed to minimize the I/O cost and optimize iterative processing [98].

Another challenge is to minimize the feedback/communication from/with classifiers. The problem of learning the optimal classifier chain at run-time has been modeled as a multi-player multi-armed bandit problem with limited feedback [99]. It does not require distributed local classifiers to exchange any information except limited feedback on mining performance to enable the learning of an optimal classifier chain [99].

A third challenge is to address the velocity aspect of big data in ML. Current (de-facto standard) solutions for big data analysis are not designed to deal with evolving streams [100]. A ML system must be able to cope with the influx of changing data in a continual manner. Lifelong Machine Learning is in contrast with the traditional one-shot learning [101]. To this end, online learning has been exploited to make kernel methods efficient and scalable for large-scale learning applications. For instance, two different online kernel ML algorithms – Fourier Online Gradient Descent and Nystrom Online Gradient Descent algorithms have been explored to tackle three online learning tasks: binary classification, multi-class classification, and regression [102,103].

A fourth challenge is to address the variety aspect of big data in ML. Most traditional ML algorithms can only take certain type of input, such as numerical, text or images. In many cases, data that could be used for a single ML goal may come in different types and formats. It will result in an explosion of features to be learned and is sometimes referred as a “Big Dimensionality” challenge [104]. For instance, one ML algorithm might need learn from: (1) a mixture of large volume of data and high speed stream data, or (2) large volume of data with image, text, acoustic, and motion features.

A fifth challenge is increased problem complexity (e.g., in multi-class classification and new classes). In document and image clustering/classification, in addition to a large number of data points and their high dimensionality, the number of clusters/classes is also large. Therefore, there is a need to gradually expand the capacity of ML models to predict an increasingly large number of new classes [105].

ML on big data presents numerous other challenges. For instance, optimization in conventional ML focuses on average performance, but hard to prevent poor outcomes. Most traditional ML algorithms are not designed for data that are not loaded into memory completely. Additionally, it is complex to set objective functions due to the large number of component terms and various trade-offs among performance measures. Noise is a bigger issue for big data because patterns typically reside in a small subset of data (e.g., spam and online attacks).

## 5. Evaluation opportunities and challenges

Traditional ML has an established set of metrics for performance evaluation, such as accuracy, error rate, precision, recall, squared error, likelihood, posterior probability, information gain, K-L divergence, cost, utility, margin, optimization error, estimation error, approximation, and mean and worst outcome. These metrics focus on the prediction accuracy of ML. In addition, scalability, traditionally used to evaluate a parallel program, is the emphasis of big data analytics. Scalability has been operationalized as such metrics as data I/O performance, fault tolerance, real-time processing, memory usage, data size supported, iterative task support, and throughput [106].

Evaluating big data ML is not a simple combination of the two types of metrics. It needs to address both trade-offs within each type of the metrics and complex trade-offs between them. For instance, precision and recall, accuracy and response time, are classical performance trade-offs. The support of iterative tasks goes against fault tolerance in supporting scalability (e.g., MapReduce supports fault tolerance but not iteration). Additionally, non-iterative algorithms (e.g., Nystrom approximation) scale better than iterative ones (e.g., Eigen decomposition) but with slightly worse performance. Although it is faster to train a linear SVM than a non-linear one, it is more difficult to parallelize the former than the latter. Further, the conventional tradeoff between computation and communication is particularly applicable to big data ML. Algorithms should be carefully designed so that time saved on computation can compensate the cost associated with communication/loading. Take parallel SVM as an example; although its computational cost is high (not linear), its data communication/loading cost is less of a concern. Many methods (e.g., stochastic gradient descent or coordinate descent) are inherently sequential, and consequently communication cost becomes a main concern. In addition, traditionally ML research only considers running time (depends on the number of operations) but ignores data loading time (depends on the number of access). For linear algorithms, loading time could be bigger than running time. The opposite is true for kernel methods.

The complexity of existing ML algorithms is often overwhelming because many (layman) users do not understand the trade-offs and the challenges in parameterization and choosing between different learning techniques. For example, to run an ML algorithm, users often need to set hyper-parameters. Since the values of hyper-parameters may significantly impact execution time and results, choosing proper parameters is critical in ML applications. However, existing ML systems typically offer little or no help on how to set parameters. In addition, since these algorithms can be hard to understand for people who do not have a strong background in ML or distributed systems, finding the right parameters can be very challenging [107].

There is an increasing attention to the usability of ML models in terms of interpretability, ease of use, stability, and so on. Among them, ease of use is the most commonly used usability metrics. Some ease of use metrics include complexity in setting objective functions, resilience, average error, and pattern “diversity”. However, interpretability and stability [99,100] are not the main design considerations for many well-performed ML algorithms. Understanding and explaining the underlying process from which observable information is generated are important challenges for statistical ML. In contrast, rule-based ML models are intuitive and able to express cause-effect relationships. Nevertheless, rule-based models face their own set of research challenges such as rule generation, evaluation, execution, optimization, and maintenance.

Declarative approach to ML is one way to make ML more accessible to non-experts [108]. To address the lack of support of data independence and declarative specification in big data solutions, MLbase was

**Table 2**
Open Research Issues in Machine Learning on Big Data.

| Main Component | Aspects | Open research issues |
|---|---|---|
| Big Data | Volume | • Cleaning and compressing big data<br>• Large scale distributed feature selection<br>• Workflow management and task scheduling |
| | Velocity | • Real time online learning for streaming data |
| | Variety | • Multi-view learning for heterogeneous multimedia data<br>• Multimedia neural semantic embedding |
| | Veracity | • Assessing data veracity<br>• Learning with unreliable or contradicting data |
| | Value | • Explainable ML for decision support<br>• Multi-user collaborative decision support based on big data analysis |
| User | Labeling | • Crowd sourced active learning for effective large scale data annotation |
| | Evaluation | • Comprehensive evaluation measures for ML (e.g, usability-based measures) |
| | Privacy | • Privacy preserving distributed ML |
| | User Interface | • Visualizing big data<br>• Intelligent user interfaces for interactive ML<br>• Declarative ML |
| Domain | Domain knowledge | • Incorporating general domain knowledge (e.g., ontology, first-order logic, business rules) in ML |
| System | Infrastructure | • New infrastructure that seamlessly provides decision support based on real time analysis of large amount of heterogeneous and unreliable data.<br>• General big data middleware |

developed to harness the power of ML for both non-expert end-users and ML researchers [107]. The system provides: (1) a simple declarative way to specify ML tasks, (2) a novel optimizer to select and dynamically adapt the choice of learning algorithms (transforms a declarative ML task into a sophisticated learning plan), (3) a set of high-level operators to enable ML researchers to implement a wide range of ML methods without deep system knowledge, and (4) a new run-time optimized for the data-access patterns of these high-level operators [107].

Although it is less theoretically interesting to design an algorithm that is slightly worse in accuracy but has greater ease of use and system reliability, the latter can be very valuable in practice [109]. Therefore, developing usable ML on big data will facilitate the training of data scientists (e.g., in parameter tuning, workflow optimization, and data preparation) and a wide adoption of ML in practice.

One conventional way to explain data is through charts, graphs and other visualization techniques, because humans can readily make decisions based on patterns and comparisons. Nevertheless, as data volume goes up, this method is reaching its limits [110].

In order for big data ML to receive wide societal acceptance to exert impacts, data ethics issues such as data privacy, security, ownership, liability, and behavioral targeting should also be addressed [111,112].

In summary, since commonly accepted evaluation metrics are the main driving force behind new ML algorithm development, there is an urgent need to establish more comprehensive evaluation metrics that are beyond typical accuracy and scalability based measures. For example, comprehensive usability-based evaluation metrics will help guide the development of new ML algorithms that simultaneously balance multiple usability factors such as interpretability, efficiency, accuracy, stability, robustness and ease of use.

## 6. Future research and conclusion

This paper presents an overview of opportunities and challenges of ML on big data. Big data creates numerous challenges for traditional ML in terms of scalability, adaptability, and usability, and presents new opportunities for inspiring transformative and novel ML solutions to address many associated technical challenges and create real-world impacts. These opportunities and challenges serve as promising directions for future research in this area. We further highlight some open research issues in ML on big data according to the components of the MLBiD framework, as shown in Table 2.

Most existing work on ML for big data focused on the *volume*, *velocity* and *variety* aspects, but there has not been much work addressing the remaining two aspects of big data: *veracity* and *value*. To handle data veracity, one promising direction is to develop algorithms that are capable of accessing the trustworthiness or credibility of data or data sources so that untrustworthy data can be filtered during data pre-processing; and another direction is to develop new ML models that can inference with unreliable or even contradicting data. To realize the value of big data in decision support, we need to help users understand ML results and the rationale behind each system's decision. Thus, explainable ML will be an important future research area. Moreover, to support human-in-the-loop big data ML, we need to address fundamental research questions such as how to effectively acquire large amount of annotated data through crowd sourcing; how to evaluate an ML algorithm based not only on its prediction accuracy or scalability, but also on its overall capability to support end users in performing their tasks (e.g., usability-based measure). Further, additional open research issues include: (1) how to protect data privacy while performing ML; (2) how to make ML more declarative so that it is easier for non-experts to specify and interact with; (3) how to incorporate general domain knowledge into ML; and (4) how to design new big data ML architecture that seamlessly provides decision support based on real-time analysis of large amount of heterogeneous data that may not be reliable.

In summary, ML is indispensible to meet the challenges posed by big data and uncover hidden patters, knowledge, and insights from big data in order to turn its potential into real value for business decision making and scientific exploration. The marriage of ML and big data points to prosperous future in a new frontier.

## Acknowledgements

This work was supported in part by the National Science Foundation [grant number 1527684]. The authors would like to thank

Yueyang Jiang for his assistance with retrieving and downloading some of the references cited in this paper from various databases.

## References


[1] M.I. Jordan, T.M. Mitchell, Machine learning: trends, perspectives, and prospects, Science 349 (2015) 255–260.
[2] C.-W. Tsai, C.-F. Lai, H.-C. Chao, A.V. Vasilakos, Big data analytics: a survey, J. Big Data 2 (2015) 1–32.
[3] M.M. Najafabadi, F. Villanustre, T.M. Khoshgoftaar, N. Seliya, R. Wald, E. Muharemagic, Deep learning applications and challenges in big data analytics, J. Big Data 2 (2015) 1–21.
[4] N. Japkowicz, M. Shah, Evaluating Learning Algorithms: a Classification Perspective, Cambridge University Press, New York, NY, USA, 2011.
[5] S. Russell, P. Norvig, Artificial Intelligence: A Modern Approach, 3rd ed., Prentice Hall, Upper Saddle River, New Jersey, USA, 2010.
[6] Y. Bengio, A. Courville, P. Vincent, Representation learning: a review and new perspectives, IEEE Trans. on Pattern Anal. Mach. Intell., Trans. 35 (2013) 1798–1828.
[7] O. Dekel, From Online to Batch Learning with Cutoff-AveragingNIPS (2008), 2008, pp. 377–384.
[8] S. Amershi, M. Cakmak, W.B. Knox, T. Kulesza, Power to the people: the role of humans in Interactive machine learning, AI Mag. 35 (2014) 105–120.
[9] V. Mirchevska, M. Luštrek, M. Gams, Combining domain knowledge and machine learning for robust fall detection, Expert Syst. 31 (2014) 163–175.
[10] T. Yu, Incorporating Prior Domain Knowledge into Inductive Machine LearningComputing Sciences, University of Technology Sydney, Sydney, Augtralia, 2007.
[11] Q. Chen, J. Zobel, K. Verspoor, Evaluation of a machine learning duplicate detection method for bioinformatics Databases, Proc. ACM Ninth Int. Workshop Data Text. Min. Biomed. Inform. (2015) 4–12.
[12] T. Rakthanmanon, B. Campana, A. Mueen, G. Batista, B. Westover, Q. Zhu, et al., Addressing Big data time series: mining Trillions of time series subsequences Under dynamic time Warping, ACM Trans. Knowl. Discov. Data 7 (2013) 10.
[13] J.J.Pfeiffer , III, J.Neville, P.N.Bennett, Overcoming relational learning biases to accurately predict preferences in large scale networks, in: Proceedings of the 24th International Conference on World Wide Web, 2015, pp. 853–863.
[14] L.Cao, M.Wei, D.Yang, E.A.Rundensteiner, Online outlier exploration over large datasets, in: Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2015, pp. 89–98.
[15] A. Gandomi, M. Haider, Beyond the hype: Big data concepts, methods, and analytics, Int. J. Inf. Manag. 35 (2015) 137–144.
[16] X.Cai, F.Nie, H.Huang, Multi-view K-means clustering on big data, in: Proceedings of the Twenty-Third international joint conference on Artificial Intelligence, 2013, pp. 2598–2604.
[17] S. Ramírez-Gallego, S. García, H. Mouriño-Talín, D. Martínez-Rego, V. Bolón-Canedo, A. Alonso-Betanzos, et al., "Data discretization: taxonomy and big data challenge," Wiley Interdisciplinary Reviews, Data Mining and Knowledge Discovery, vol. 6, pp. 5-21, 2016.
[18] Y.Z.Y.-M.Cheung, Discretizing Numerical Attributes in Decision Tree for Big Data Analysis, in: Proceedings of the 2014 IEEE International Conference on Data Mining Workshop (ICDMW), 2014.
[19] L.-V. Nguyen-Dinh, M. Rossi, U. Blanke, G. Tröster, Combining crowd-generated media and personal data: semi-supervised learning for context recognition, Proc. 1st ACM Int. Workshop Pers. data meets Distrib. Multimed. (2013) 35–38.
[20] B.M. Lake, R. Salakhutdinov, J.B. Tenenbaum, Human-level concept learning through probabilistic program induction, Science 350 (2015) 1332–1338.
[21] G. Zhang, S.-X. Ou, Y.-H. Huang, C.-R. Wang, Semi-supervised learning methods for large scale healthcare data analysis, Int. J. Comput. Healthc. 2 (2015) 98–110.
[22] J. Suzuki, H. Isozaki, and M. Nagata, Learning condensed feature representations from large unsupervised data sets for supervised learning, in: Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics, Human Language Technologies, short papers, 2, 2011, pp. 636–641.
[23] B. Mozafari, P. Sarkar, M. Franklin, M. Jordan, S. Madden, Scaling up crowd-sourcing to very large datasets: a case for active learning, Proc. VLDB Endow. 8 (2014) 125–136.
[24] Y. Su, G. Agrawal, J. Woodring, K. Myers, J. Wendelberger, J. Ahrens, Effective and efficient data sampling using bitmap indices, Clust. Comput. 17 (2014) 1081–1100.
[25] V. Bolón-Canedo, N. Sánchez-Maroño, A. Alonso-Betanzos, Distributed feature selection, Appl. Soft Comput. 30 (2015) 136–150.
[26] S. Sun, Jing Zhao, J. Zhu, A review of Nyström methods for large-scale machine learning, Inf. Fusion 26 (2015) 36–48.
[27] M. Tan, I.W. Tsang, L. Wang, Towards ultrahigh dimensional feature selection for big data, J. Mach. Learn. Res. 15 (2014) 1371–1429.
[28] Z.Zhao, H.Liu, Spectral feature selection for supervised and unsupervised learning, in: Proceedings of the 24th international conference on Machine learning, 2007, pp. 1151–1157.
[29] J. Cervantes, X. Li, W. Yu, Support vector machine classification based on fuzzy clustering for large data sets, in: Proceedings of the 5th MICAI, 2015, pp. 572–582.
[30] O. Y. S. Al-Jarrah, A., M. Elsalamouny, P. D. Yoo, S. Muhaidat, and K. Kim, Machine-Learning-Based Feature Selection Techniques for Large-Scale Network Intrusion Detection, in: Proceedings of the 2014 IEEE 34th International Conference on in Distributed Computing Systems Workshops (ICDCSW).
[31] A.T. Azar, A.E. Hassanien, Dimensionality reduction of medical big data using neural-fuzzy classifier (04/01/2015)Soft Comput. - A Fusion Found., Methodol. Appl. 19 (2015) 1115–1127.
[32] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, P.-A. Manzagol, Stacked denoising Autoencoders: learning useful representations in a deep network with a local denoising criterion, J. Mach. Learn. Res. 11 (2010) 3371–3408.
[33] C.-Y. Liou, W.-C. Cheng, J.-W. Liou, D.-R. Liou, Autoencoder for words, Neurocomputing 139 (2014) 84–96.
[34] R. Collobert, F. Sinz, J. Weston, L. Bottou, Trading convexity for scalability, Proc. 23rd Int. Conf. Mach. Learn. (2006) 201–208.
[35] Y. Bengio, Y. LeCun, Scaling learning algorithms towards, AI (ed), in: L. Bottou, O. Chapelle, D. DeCoste, J. Weston (Eds.), Large Scale Kernel Machines, MIT Press, Cambridge, MA, 2007.
[36] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, et al., "TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems," CoRR, 2016.
[37] Y. You, H. Fu, S.L. Song, A. Randles, D. Kerbyson, A. Marquez, et al., Scaling support vector machines on modern HPC platforms, J. Parallel Distrib. Comput. 76 (2015) 16–31.
[38] B. Panda, J.S. Herbach, S. Basu, R.J. Bayardo, PLANET: massively parallel learning of tree ensembles with MapReduce, Proc. VLDB Endow. 2 (2009) 1426–1437.
[39] E. Xing, Q. Ho, W. Dai, J.-K. Kim, J. Wei, S. Lee, et al., Petuum: a new platform for distributed machine learning on Big data, IEEE Trans. Big Data (2015) 49–67.
[40] R. Collobert, K. Kavukcuoglu, and C. Farabet, Torch7: A Matlab-like Environment for Machine Learning, in: Proceedings of the Neural Information Processing Systems (NIPS) Workshop on BigLearn, 2011.
[41] T.Yang, Q.Lin, R.Jin, Big data analytics: Optimization and randomization, in: Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2015, pp. 2327–2327.
[42] W. Xu, Towards Optimal one pass large scale learning with averaged stochastic gradient descent, 2011. Available at: arXiv:1107.2490.
[43] L. Bottou, Large-Scale Machine Learning with Stochastic Gradient Descent, in: Proceedings of COMPSTAT, 2010, pp. 177–186.
[44] J. Wang, Y. Tang, M. Nguyen, I. Altintas, A Scalable data Science workflow approach for Big data Bayesian network learning, Proc. 2014 IEEE/ACM Int. Symp. Big Data Comput. (2014) 16–25.
[45] K. Yue, H. Wu, X. Fu, J. Xu, Z. Yin, W. Liu, A data-intensive approach for discovering user similarities in social behavioral interactions based on the bayesian network, Neurocomputing 219 (2017) 364–375.
[46] A. Kumar, A. Beutel, Q. Ho, E.P. Xing, Fugue: Slow-Worker-Agnostic Distributed Learning for Big Models on Big Data, in: Proceedings of the 17th International Conference on Artificial Intelligence and Statistics (AISTATS), Reykjavik, Iceland, 2014, pp. 531–539.
[47] K. Sankar, H. Karau, Fast Data Processing with Spark, Second ed., Packt Publishing, 2015.
[48] S. Owen, R. Anil, T. Dunning, E. Friedman, Mahout in Action, Manning Publications Co., 2011.
[49] C.T. Chu, S.K. Kim, Y.A. Lin, Y. Yu, G.R. Bradski, A.Y. Ng, et al., Map-reduce for machine learning on multicore, *NIPS* (2006) 281–288.
[50] A.K.Ghoting, R.E.Pednault, B.Reinwald, V.Sindhwani, S.Tatikonda, Y.Tian, et al., SystemML: Declarative machine learning on MapReduce, in: Proceedings of the 27th International Conference on Data Engineering (ICDE), 2011.
[51] V.R. Borkar, Y. Bu, M.J. Carey, J. Rosen, N. Polyzotis, T. Condie, et al., Declarative systems for large-scale machine learning, *IEEE Data Eng. Bull.* 35 (2012) 24–32.
[52] Y. Low, D. Bickson, J. Gonzalez, C. Guestrin, A. Kyrola, J.M. Hellerstein, Distributed GraphLab: a framework for machine learning and data mining in the cloud, Proc. VLDB Endow. 5 (2012) 716–727.
[53] Theano Development Team, Theano: A Python framework for fast computation of mathematical expression. Available: arXiv:1605.02688.
[54] Y.Jia, E.Shelhamer, J.Donahue, S.Karayev, J.Long, R.Girshick, et al., Caffe: Convolutional Architecture for Fast Feature Embedding, in: Proceedings of the 22nd ACM international conference on Multimedia, Orlando, Florida, USA, 2014.
[55] J.-x. Dong, A. Krzyzak, C.Y. Suen, Fast SVM training algorithm with decomposition on very large data sets, *IEEE* Trans. Pattern Anal. Mach. Intell. 27 (2005) 603–618.
[56] J. Dean, G. S. Corrado, R. Monga, K. Chen, M. Devin, Q. V. Le, et al., Large scale distributed deep networks, in: Proceedings of the Neural Information Processing Systems, Lake Tahoe, Nevada, United States, 2012, pp. 1232–1240.
[57] J.E. Mason, I. Traoré, I. Woungang, Machine Learning Techniques for Gait Biometric Recognition: Using the Ground Reaction Force, Springer, Switzerland, 2016.
[58] Q.V.Le, J.Ngiam, A.Coates, A.Lahiri, B.Prochnow, A.Y.Ng, On optimization methods for deep learning, in: Proceedings of the 28th International Conference on Machine Learning, Bellevue, WA, USA, 2011.
[59] Y. Ganjisaffar, T. Debeauvais, S. Javanmardi, R. Caruana, C.V. Lopes, Distributed tuning of machine learning algorithms using MapReduce Clusters, Proc. Third Workshop Large Scale Data Min.: Theory Appl. (2011) 2.
[60] C.Dijun Luo, Ding, H.Huang, Parallelization with ultiplicative algorithms for big data mining, in: Proceedings of the 12th International Conference on Data Mining (ICDM), 2012, pp. 489–498.
[61] J.S.Yoo, D.Boulware, D.Kimmey, A Parallel Spatial Co-location Mining Algorithm Based on MapReduce, in: proceedings of the 2014 IEEE International Congress on Big Data, 3rd, pp. 25–31.
[62] I. Triguero, D. Peralta, J. Bacardit, S. García, F. Herrera, MRPR: A MapReduce

solution for prototype reduction in big data classification, Neurocomputing 150 (2015) 331–345 Part A.
[63] S. Landset, T.M. Khoshgoftaar, A.N. Richter, T. Hasanin, A survey of open source tools for machine learning with big data in the Hadoop ecosystem, J. Big Data 2 (2015) 1–36.
[64] R.Gemulla, E.Nijkamp, P.J.Haas, Y.Sismanis, Large-scale matrix factorization with distributed stochastic gradient descent, in: Proceedings of the 17th ACM SIGKDD international conference ion Knowledge discovery and data mining, San Diego, California, USA, 2011, pp. 69–77.
[65] D. Hsu, N. Karampatziakis, J. Langford, A.J. Smola, Parallel online learningScaling up machine learning: Parallel and distributed approaches, Cambridge University Press, 2011.
[66] P.Domingos, G.Hulten, A General Method for Scaling Up Machine Learning Algorithms and its Application to Clustering, presented at Proceedings of the Eighteenth International Conference on Machine Learning, 2001, pp. 106–113.
[67] R. Bekkerman, M. Bilenko, J. Langford (Eds.), Scaling up Machine Learning: Parallel and Distributed Approaches, Cambridge University Press, New York, 2012.
[68] C. Parker, Unexpected challenges in large scale machine learning, Proc. 1st Int. Workshop Big Data, Streams Heterog. Source Min.: Algorithms, Syst., Program. Models Appl. (2012) 1–6.
[69] D. Peteiro-Barral, B. Guijarro-Berdiñas, A survey of methods for distributed machine learning, Prog. Artif. Intell. 2 (2013) 1–11.
[70] K.L.C.Zhu, M.Savvides, Distributed class dependent feature analysis — A big data approach, in: proceedings of the 2014 IEEE International Conference on Big Data, 2014.
[71] M. Yui, I. Kojima, A database-Hadoop hybrid approach to Scalable machine learning, IEEE Int. Congr. Big Data (BigData Congr.) (2013) 1–8.
[72] F.Ö. Çatak, Classification with boosting of extreme learning machine over arbitrarily partitioned data, Soft Comput. (2015) 1–13.
[73] M. Hefeeda, F. Gao, and W. Abd-Almageed, Distributed approximate spectral clustering for large-scale datasets, in: Proceedings of the 21st international symposium on High-Performance Parallel and Distributed Computing, 2012, pp. 223–234.
[74] G. Cavallaro, M. Riedel, M. Richerzhagen, J.A. Benediktsson, A. Plaza, On Understanding Big data impacts in remotely sensed image classification using support vector machine methods, IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens. 8 (2015) 4634–4646.
[75] J.Zhu, J.Chen, W.Hu, Big Learning with Bayesian Methods. Available: ⟨http://arxiv.org/pdf/1411.6370⟩, 2014.
[76] L.Bagheri, H.Goote, A.Hasan, G.Hazard, Risk adjustment of patient expenditures: A big data analytics approach, in Proceedings of the 2013 IEEE International Conference on Big Data, 2013.
[77] A. Krizhevsky, I. Sutskever, G. Hinton, Imagen. Classif. Deep convolutional Neural Netw. (2012).
[78] Y. LeCun, K. Kavukcuoglu, and C. Farabet, Convolutional networks and applications in vision, in: Proceedings of IEEE International Symposium on Circuits and Systems, 2010, pp. 253–256.
[79] J. Deng, K. Li, M. Do, H. Su, L. Fei-Fei, Construction and analysis of a large scale image ontology, Vis. Sci. Soc. 1 (2009).
[80] Y. Guo, Y. Liu, A. Oerlemans, S. Lao, S. Wu, M.S. Lew, Deep learning for visual understanding: a review, Neurocomputing 187 (2016) 27–48.
[81] X. Jiang, Y. Pang, X. Li, J. Pan, Speed up deep neural network based pedestrian detection by sharing features across multi-scale models, Neurocomputing 185 (2016) 163–170.
[82] R. Socher, A. Perelygin, J. Wu, J. Chuang, C. Manning, A. Ng, et al., Recursive deep models for semantic compositionality over a sentiment treebank, in: Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP), 2013.
[83] S. Zhou, Q. Chen, X. Wang, Active deep learning method for semi-supervised sentiment classification, Neurocomputing 120 (2013) 536–546.
[84] N. Zeng, Z. Wang, H. Zhang, W. Liu, F.E. Alsaadi, Deep belief networks for quantitative analysis of a gold immunochromatographic strip, Cogn. Comput. 8 (2016) 684–692.
[85] R.Raina, A.Battle, H.Lee, B.Packer, A.Y.Ng, Self-taught learning: transfer learning from unlabeled data, in: Proceedings of the 24th international conference on Machine learning, Corvalis, Oregon, USA, 2007.
[86] I. Goodfellow, Y. Bengio, A. Courville, Deep Learning, MIT Press, 2016.
[87] D. Erhan, Y. Bengio, A. Courville, P.-A. Manzagol, P. Vincent, S. Bengio, Why does Unsupervised Pre-training help deep learning?, *The* J. Mach. Learn. Res. 11 (2010) 625–660.
[88] T.Mikolov, I.Sutskever, K.Chen, G.S.Corrado, J.Dean, Distributed Representations of Words and Phrases and their Compositionality, presented at the NIPS, Stateline, NV, 2013.
[89] X.-w. Chen, X. Lin, Big data deep learning: challenges and perspectives, Access, IEEE 2 (2014) 514–525.
[90] Y. Chen, T. Luo, S. Liu, S. Zhang, L. He, J. Wang, et al., DaDianNao: a machine-learning Supercomputer, 47th Annu. IEEE/ACM Int. Symp. Micro. (2014) 609–622.
[91] D. Mahajan, J. Park, E. Amaro, H. Sharma, A. Yazdanbakhsh, J.K. Kim, et al., TABLA: a unified template-based framework for accelerating statistical machine learning, *IEEE* Int. Symp. High. Perform. Comput. Archit. *(HPCA)* (2016) 14–26.
[92] M.Zaharia, M.Chowdhury, M.J.Franklin, S.Shenker, I.Stoica, Spark: cluster computing with working sets, presented at in: Proceedings of the 2nd USENIX conference on Hot topics in Cloud Computing, Boston, MA, 2010.
[93] E.Bortnikov, A.Frank, E.Hillel, S.Rao, Predicting execution bottlenecks in map-reduce clusters, in: Proceedings of the 4th USENIX conference on Hot Topics in Cloud Ccomputing, 2012, pp. 18–18.
[94] K. Xu, H. Yue, L. Guo, Y. Guo, Y. Fang, Privacy-preserving machine learning algorithms for big data systems, in: Proceedings of the 2015 IEEE 35th International Conference on Distributed Computing Systems (ICDCS), 2015, pp. 318–327.
[95] J. Vaidya, H. Yu, X. Jiang, Privacy-preserving SVM classification, Knowledge Inf. Syst. 14 (2008) 161–178.
[96] A.D. Popescu, A. Balmin, V. Ercegovac, A. Ailamaki, PREDIcT: towards predicting the runtime of large scale iterative analytics, Proc. VLDB Endow. 6 (2013) 1678–1689.
[97] L. Breiman, Pasting small votes for classification in large databases and On-Line, Machine Learn. 36 (1999) 85–103.
[98] H. Kashyap, H.A. Ahmed, N. Hoque, S. Roy, D.K. Bhattacharyya, Big Data Anal. Bioinforma.: A Mach. Learn. Perspect. (2015).
[99] J.Xu, C.Tekin, M.van der Schaar, Learning optimal classifier chains for real-time big data mining, in Proceedings 51st Annu. Allerton *C*onference Comm., Control and Comput. (Allerton'13), 2013.
[100] G.De Francisci Morales, SAMOA: a platform for mining big data streams, in: Proceedings of the 22nd International Conference on World Wide Web, 2013, pp. 777–778.
[101] Q.Yang, Big data, lifelong machine learning and transfer learning, in: Proceedings of the sixth ACM international conference on Web search and data mining, 2013, pp. 505–506.
[102] J. Lu, S.C. Hoi, J. Wang, P. Zhao, Z.-Y. Liu, Large scale online kernel learning, J. Mach. Learn. Res. 17 (2016) 1–43.
[103] Z. Wang, K. Crammer, S. Vucetic, Breaking the curse of kernelization: budgeted stochastic gradient descent for large-scale SVM training, T*he J. Mach. Learn. Res.* 13 (2012) 3103–3131.
[104] Y. Zhai, Y.S. Ong, I.W. Tsang, The emerging big dimensionality, IEEE Comput. Intell. Mag. 9 (2014) 14–26.
[105] T.Xiao, J.Zhang, K.Yang, Y.Peng, Z.Zhang, Error-Driven Incremental Learning in Deep Convolutional Neural Network for Large-Scale Image Classification, in: Proceedings of the ACM International Conference on Multimedia, 2014, pp. 177–186.
[106] D. Singh, C.K. Reddy, A survey on platforms for big data analytics, J. Big Data 2 (2014) 1–20.
[107] T.Kraska, A.Talwalkar, J.Duchi, R.Griffith, M.J.Franklin, M.I.Jordan, MLbase: A Distributed Machine-learning System, in: Proceedings of the 6th Biennial Conference on Innovative Data Systems Research, Asilomar, California, USA, 2013.
[108] V. Markl, Breaking the chains: on declarative data analysis and data independence in the big data era, Proc. VLDB Endow. 7 (2014) 1730–1733.
[109] S. Tong, Lessons learned developing a practical large scale machine learning system 2016, Google Research Blog, 2010.
[110] T.R. Armes, M, Using Big data and predictive machine learning in aerospace test environments, IEEE Autotestcon (2013).
[111] B.Thuraisingham, Big Data Security and Privacy, in: Proceedings of the 5th ACM Conference on Data and Application Security and Privacy, San Antonio, Texas, USA, 2015.
[112] B.Nelson, T.Olovsson, Security and Privacy for Big Data: A Systematic Literature Review, in: Proceedings of the 2016 IEEE International Conference on Big Data, Washington, D.C, 2016, pp. 3693–3702.

**Lina Zhou** is an associate professor of Information Systems at the University of Maryland, Baltimore County. She has published more than 150 referred papers in academic journals and conferences. Her current research interests include information extraction, machine learning, online deception, intelligent human-computer interaction. She currently serves on the editorial boards of seven international journals.

**Shimei Pan** received a Ph.D. in Computer Science from Columbia University. Before joining UMBC, Dr. Pan was a research scientist at IBM Watson Research Center in New York. Her primary research interests are large-scale text mining, social media analytics, and their applications in human behavior modeling. She is also interested in human-centered text mining and intelligent interactive systems. Dr. Pan has authored more than 70 peer-reviewed papers in major international conferences and journals. She has also served on various technical program committees for major international conferences such as IJCAI, EMNLP, IUI, CIKM and RecSystem.

**Jianwu Wang** received a Ph.D. degree from the Institute of Computing Technology, Chinese Academy of Sciences, in 2007. He currently is an Assistant Professor with the Department of Information Systems, University of Maryland, Baltimore County. He is also an Adjunct Professor at North China University of Technology, China. His research interests include big data, scientific workflow, distributed computing, service-oriented computing, and end-user programming. He has published 60+ papers with more than 600 citations.

**Athanasios V. Vasilakos** is Professor with the Lulea University of Technology, Sweden. He served or is serving as an Editor for many technical journals, such as the IEEE Transactions on Network and Service Management; IEEE Transactions on Cloud Computing, IEEE Transactions on Information Forensics and Security, IEEE Transactions on Cybernetics; IEEE Transactions on Nanobioscience; IEEE Transactions on Information Technology in Biomedicine; ACM Transactions on Autonomous and Adaptive Systems; the IEEE Journal on selected areas in communications.