The Big Data Landscape: Hurdles and Opportunities

Divyakant Agrawal[1,2] and Sanjay Chawla[1,3]

[1] Qatar Computing Research Institute, Qatar
[2] University of California Santa Barbara, USA
[3] University of Sydney, Australia

Abstract. Big Data provides an opportunity to interrogate some of the deepest scientific mysteries, e.g., how the brain works and develop new technologies, like driverless cars which, till very recently, were more in the realm of science fiction than reality. However Big Data as an entity in its own right creates several computational and statistical challenges in algorithm, systems and machine learning design that need to be addressed. In this paper we survey the Big Data landscape and map out the hurdles that must be overcome and opportunities that can be exploited in this paradigm shifting phenomenon.

1 Introduction

Big data has emerged as one of the most promising technology paradigm in the past few years. Availability of large data arises in numerous application contexts: trillions of words in English and other languages, hundreds of billions of text documents, a large number of translations of documents in one language to other languages, billions of images and videos along with textual annotations and summaries, thousands of hours of speech recordings, trillions of log records capturing human activity, and the list goes on. During the past decade, careful processing and analysis of different types of data has had transformative effect. Many applications that were buried in the pages of science fiction have become a reality, e.g., driverless cars, language agnostic conversation, automated image understanding, and most recently deep learning [6] to simulate a human brain.

In the technology context, Big Data has resulted in significant research and development challenges. From a systems perspective, scalable storage, retrieval, processing, analysis, and management of data poses the biggest challenge. From an application perspective, leveraging large amounts of data to develop models of physical reality becomes a complex problem. The interesting dichotomy is that the bigness of data in the system context makes some of the known data processing solutions that were "acceptable" to "not acceptable." For example, standard algorithms for carrying out join processing may have to be revisited in the Big Data context. In contrast, the bigness of data allows many applications to move from being "not possible" to "possible." For example real time, automated, high quality and robust language translation seems entirely feasible. Thus, new approaches are warranted to develop scalable technologies for processing and managing Big Data [7]. In the same vein, designing and developing robust models for learning from big data remains a significant research challenge.

In this paper, we explore the Big Data problem both from the system perspective as well as from the application perspective. In the popular press, "bigness" or the size of data is touted as a desirable property. Our goal is to clearly comprehend the underlying complexity that must be overcome with the increasing size of data and clearly delineate the hurdles and opportunities in this fast developing space.

The rest of the paper is structured as follows. In Section 2 we use the record de-duplication application to delve into some of the intrinsic computational, systems and statistical challenges when operating in a Big Data environment. In Section 3 we use the classical clustering problem to highlight how simple machine learning tasks can result in complex computational problems and the resulting trade-offs in learning in a Big Data environment. We briefly review the dictionary learning problem in Section 4 and its connects with deep and representational learning. We conclude in Section 5 with a discussion and directions for future work.

2 The Big Data Problem

To understand the "Big Data" problem consider a table $S(r, c)$ whose rows (r) and columns (c) can grow infinitely. Assume the growth rate of the table (in terms of r and c) outstrips the corresponding increase in unit computational processing power and storage capacity. We refer to this setting as the Big Data Operating Paradigm (BDOP).

A task $\mathbb{T}$ on table S is an operation which maps S onto another entity E, where E can be another table or the state of a mathematical model specified as part of the task $\mathbb{T}$. The Big Data problem is to understand the feasibility of carrying out $\mathbb{T}$ both in terms of computational tractability and statistical effectiveness in BDOP as S grows.
Example: Let $S(r, c)$ be a merged table of customer records from two databases. Let $\mathbb{T}$ be the task of record de-duplication, i.e., identify records in S which belong to the same customer. The computational challenge arises because all pairs of records $(O(|r|^2))$ in S have to be compared. The statistical hardness comes into fore because of the high dimensionality of the problem as the number of columns $(|c|)$ increases.

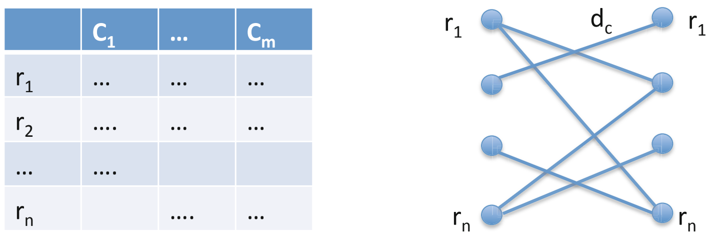
2.1 Distributed Computational Complexity

For concreteness consider the record de-duplication task [3] which can be specified as

$$\mathbb{T} : S \times S \rightarrow_d \{0, 1\}$$

Here, d is a "dis-similarity" function between two records which is either specified by a domain expert or learnt separately from the data. As noted earlier, the computational complexity of $\mathbb{T}$ is $O(|r|^2)$.

In a BDOP setting it becomes necessary that S is stored in a distributed environment and the task $\mathbb{T}$ carried out in parallel. However, if we assume that in a given time window, data grows by an order of magnitude but unit processing speed and storage capacity are constant, then for the $O(|r|^2)$ de-duplication task $\mathbb{T}$, the number of computation nodes required to maintain the same response time grows at least quadratically! This might appear as a paradox as the resources required to maintain the same quality of service is an order of magnitude greater than the corresponding increase in data size. However, in practical real data settings, even if $|S \times S|$ grows quadratically, the computational complexity of $\mathbb{T}$ is governed by $|S \bowtie S|$ which grows at the rate $O(\alpha.|r|)$, where α is typically a small fraction. Thus the task $\mathbb{T}$ becomes feasible as the corresponding bi-partite graph is sparse. However, while α maybe small, the distribution of the degree of the bi-partite graph is highly non-uniform (often Zipfian). Thus the processing time in a distributed environment is lower bounded by the size of the largest partition which can be large. Designing the appropriate trade-off between data partitioning and task parallelism in a cloud environment becomes a major design and research challenge.



	C_1	...	C_m
r_1	...	...	...
r_2	....	...	...
...	....		
r_n		....	...

Fig. 1. From a computational and systems perspective, a Big Data task reduces to processing a table as a bi-partite graph in a distributed setting, where the edge similarity is derived as a function of the columns of a table. In a BDOP setting, computational is feasible only because the number of edges grow at constant rate as the number of rows increase, i.e., the bi-partitite graph is sparse.

2.2 Statistical Effectiveness

From a statistical perspective, we focus on the columms of S. For the de-duplication task the objective is to *infer* a function $f : \{C\} \to 2^{\{C\}}$ on the columns such that for any pair of records r_1, r_2

$$\pi_{f(C)}(r_1) = \pi_{f(C)}(r_2) \to r_1 \equiv r_2$$

We can interpret $f(C)$ as a subset of columms of S. Now, given the statistical nature of data, every subset of columns has a small probability p of being selected by the inference function f. However as $|c|$ increases then the probability that an

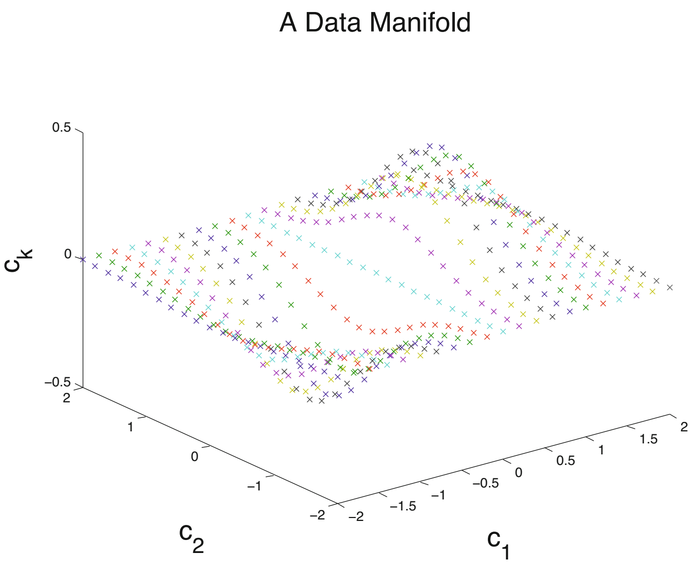


Fig. 2. Observational data tends to be highly redundant, i.e., there is a large degree of correlation (possibly non-linear) between the different columns of table S. This is often referred to as "data lives in a low-dimensional manifold."

arbitrary subset of columns will be selected by f is given $1-(1-p)^{2^{|c|}} \to 1$. Thus from a statistical perspective, Big Data settings can lead to situations where the danger of inferring spurious relationships becomes highly likely. However, in practice, observational data (i.e., data collected serendipitously and not as part of an experimental design), tends to be highly redundant. High redundancy implies that the number of degrees of freedom which govern data generation is small. This is often referred to as "observational data lives in a low-dimensional manifold." The manifold structure of the data explains the widespread use of dimensionality reduction techniques like PCA and NMF in machine learning and data mining.

3 Machine Learning

Machine Learning tasks are often formulated as optimization problems. Even the simplest of tasks can result in hard and intractable optimization formulations. In BDOP, the constraint on the solution is often determined by a "time budget," i.e., obtain the best possible approximate solution of the optimization problem within time T. We highlight the trade-off between large data and the quality of the optimization solution using two examples: clustering and dictionary learning. The latter in intimately tied to "deep learning."

3.1 Clustering

Consider a set of one dimensional data points: $D = \{x_1, \ldots, x_n\}$. For example, D could record the height of a group of n people. How do we summarize D ? One obvious answer is the mean (average) of D, but lets cast the summarization task as an optimization problem. Thus, the objective is to find a y which minimizes the following objective function:

$$\min_{y \in \mathbb{R}} \sum_{i=1}^{n} (x_i - y)^2$$

If we denote $F(y)$ as $\sum_{i=1}^{n} (x_i - y)^2$, then a necessary condition to obtain y is to set $\frac{dF}{dy} = 0$ and solve for y to obtain:

$$y^* = \frac{1}{n} \sum_{i=1}^{n} x_i$$

Thus, as expected, the "optimal" summarization of D is to use average or mean of the data set. Now suppose we would like to summarize D with two data points, y_1 and y_2. The motivation for this task is to possibly obtain a representative male and female height in the group. Note, the data set D is not labeled, i.e.,we are not given which data point records a male or female height. In this situation, what might be an appropriate objective function? The first instinct is perhaps to set up the optimization problem where the aim is to find y_1 and y_2 which minimizes

$$G(y_1, y_2) = \sum_{i=1}^{n} \sum_{j=1}^{2} (x_i - y_j)^2$$

However, an examination of G reveals that it may not be an appropriate objective function. For example, we do not want to take the difference between every pair of x_i and y_j but *only* between x_i and its most *representative* y_j, which of course is not known. Thus a more suitable objective is to minimize:

$$H(y_1, y_2) = \sum_{i=1}^{n} \min_{j \in \{1,2\}} (x_i - y_j)^2 \tag{1}$$

The appearance of the min inside the summation, makes the objective non-convex and is a typical optimization pattern in many machine learning problem formulations. In Figure 3, the objective functions G and H are plotted which clearly show that H (the relevant objective function) is non-convex.

3.2 Clustering in BDOP

Having specified a clustering objective function in Equation 1, the question remains how to solve the resulting optimization problem and understand the impact of Big Data on the solution. An important observation is that the clustering

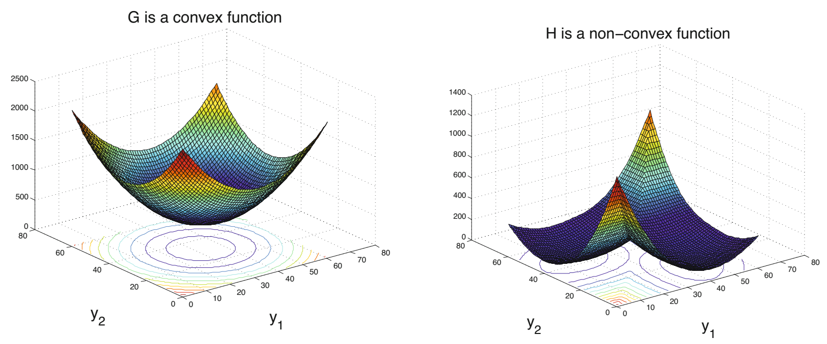


Fig. 3. The figure on the left is the shape of the objective function $G(y_1, y_2)$. G is convex but not appropriate for the clustering task. The figure on the right is the shape of $H(y_1, y_2)$ which is more suitable for clustering but non-convex.

problem is inherently imprecise. Even in the case where each data point represents a person's height and the goal is to obtain separate clusters for males and females, there are bound to be misclassified data points, i.e., males will be assigned to the female cluster and vice-versa. The availability of large amounts of data will not mitigate the imprecision and thus a case can be made to design an algorithm where the ability to trade-off between exactness of the solution and the time to obtain the solution is transparent [2]. For example, instead of optimizing $H(y_1, y_2)$, perhaps a more reasonable objective is to optimize

$$\mathbb{E}_x[H(y_1, y_2, x)] \tag{2}$$

where the expecation is over different samples of x from the same underlying (but unknown) distribution $P(x)$. Bottou et. al. [2] have precisely investigated the decomposition of Equation 2 in a general case which we customize for the case of clustering.

3.3 Error Decomposition for Big Data Learning

Notice that the objective function $H(y_1, y_2, x)$ is highly specialized as we have used $(x_i - y_j)^2$ term inside the summation. In a high-dimensional setting, this is equivalent to enforcing the resulting clusters to be spherical. Thus if the original clusters were elliptical then the choice of the objective function already results in an error which is *independent of the size of the data* ! This is known as the *approximation error*. Now because our objective is to minimize $\mathbb{E}_x[H(y_1, y_2, x)]$ but we only have access to finite number of samples from $P(x)$, the solution of any algorithm will result in value H_n such that $H_n \leq \mathbb{E}_x[H(y_1, y_2, x)] + \rho$. The discrepancy (bounded by ρ) is known as the *estimation error*. In a Big Data environment, where a given time budget may force us to stop the optimization task before all the samples are processed will lead to an objective value of $\hat{H}_n$, which is an approximation of H_n. This is known as the *computation error*.

3.4 Stochastic Gradient Descent

A simple but extremely versatile optimization algorithm where the trade-off between the estimation and computation error can be controlled is the Stochastic Gradient Descent Algorithm (SGDA). Recall that in order to optimize

$$\min_y f(x, y)$$

the gradient descent algorithm iterates on the following step till an approximate fixed point is obtained

$$y_{t+1} = y_t - \gamma \nabla f(y_t, x).$$

Now since in many machine learning formulations, $f(x, y)$ is of the form $f(x, y) = \sum_i^n g(x_i, y)$, the cost of the gradient step is $O(n)$. However, in stochastic gradient a random sample x_r from the training set is selected in each iteration to approximate the gradient

$$\hat{\nabla} f(y) \approx \nabla g(x_r, y)$$

The above approximation reduces the computation cost of the gradient computation from $O(n)$ to $O(1)$ and it has been shown that as $n \to \infty, y_{t+1} = y_t - \gamma \nabla g(x_r, y)$ approaches the optimal solution of $\min_y \mathbb{E}_x[f(x, y)]$.

The SGDA algorithm can easily be adapted to solve the clustering problem with objective function $H(y_1, y_2)$ as given in Equation 1. Here are the steps [1]:

1. Initialize y_1 and y_2 and set $n_1, n_2 = 0$.
2. Randomly permute D to obtain $\{x_{i_1}, \ldots x_{i_n}\}$
3. **For** $j = 1 : n$

   (a) Choose y_k closest to x_{i_j}:

   $$k^* = \text{arg min}_k (x_{i_j} - y_k)^2$$

   (b) Increment the count associated with y_{k^*}

   $$n_{k^*} \leftarrow n_{k^*} + 1$$

   (c) Update y_{k^*} (the gradient descent step):

   $$y_{k^*} \leftarrow y_{k^*} + \frac{1}{n_{k^*}}(x_{i_j} - y_{k^*})$$

4. **End For**

Notice that SGDA is especially suitable in a Big Data setting compared to the traditional batch k-means algorithm as it is online and can start emitting results incrementally.

4 Dictionary Learning

Some of the most promising applications of Big Data include language translation, speech recognition, image search and cross-modality querying, i.e., given an image automatically generate a textual description of the image and given a piece of text find the most appropriate image. Deep learning (aka neural networks) have been re-emerged as the algorithm of choice for these applications. However, the fundamental reason for success in these applications is because of the ability for algorithms to *learn* the appropriate *representation* in the presence of Big Data. The learning of an appropriate representation is often called the Dictionary Learning or sparse coding problem. We briefly describe the dictionary learning problem and note its similarity with the clustering problem described above. While we will not provide the algorithm, it should become clear that SGDA can be used to solve the dictionary learning problem.

We again start with a data set $D = \{\mathbf{x_1}, \mathbf{x_2}, \ldots, \mathbf{x_n}\}$, where each $\mathbf{x_i} \in \mathbb{R}^m$. For example the set D may be a collection of images and $\mathbf{x_i}$ is a vector of pixel values. In signal and image processing, data is often represented as a linear combination of *pre-defined* basis functions using Fourier or Wavelet transforms. In dictionary learning (also called sparse coding), the objective is to find a set of data-dependent basis functions $\mathbf{E} \in \mathbb{R}^{m \times k}$, and a set of $k-$dimensional sparse vectors $\{\alpha_{\mathbf{i}}\}_{i=1}^{n}$,such that

$$\mathbf{x_i} \approx \mathbb{E}\alpha_{\mathbf{i}} \;\; \forall i = 1, \ldots, n$$

The columns of $\mathbb{E}$ are the basis function and α's are the weights. Note that the columns of E are not restricted to be orthogonal and this provides a degree of flexibility that is not available in the case of Fourier or Wavelet transform or an SVD decomposition.

In order to infer both $\mathbb{E}$ and α, we can setup an objective function which has to be minimized

$$g_n(\mathbb{E}, \mathbf{x}) = \sum_{i=1}^{n} \min_{\alpha_{\mathbf{i}}} \|\mathbf{x}_i - \mathbb{E}\alpha_{\mathbf{i}}\|_2^2 + \lambda\|\alpha_i\|_1 \tag{3}$$

Notice the similarity between the clustering objective in Equation 1 and the dictionary learning objective in Equation 3. Both objectives have a min inside the sum function and in the case of dictionary learning there is a coupling (product) between the two unknowns ($\mathbb{E}$ and α) which makes the objective non-convex. More details about dictionary and represenational learning can be found in [9,5].

4.1 Distributed Stochastic Gradient Descent

Many machine learning problems are formulated as optimization problems with a very specific form. For example a typical optimization pattern will be of the form

$$\sum_{i=1}^{n} \ell(x_i, \mathbf{w}) + \Omega(\mathbf{w}) \tag{4}$$

Here, $\ell(x, \mathbf{w})$ is the loss function which accounts for the mismatch between the data and the model. The regularization term $\Omega(\mathbf{w})$ is often used to prevent overfitting of the model to the data. An important observation is that the loss function is applied pointwise to each data point and this can be used to design efficient distributed implementations.

A common distributed computation design pattern for machine learning is the *parameter server* pattern which consists of the following components: (i) data is partitioned (horizontally) and distributed across computational nodes, (ii) a parameter server node maintains the global state of the variable $\mathbf{w}$, (iii) each computation node i applies (stochastic) gradient descent to its data partition and computes a gradient $\nabla_i(w)$, (iv) the parameter server periodically polls the compute nodes and pulls ∇_i from each node. It then carries out a global 'sync' operation and pushes the updated $\mathbf{w}$ vector to the nodes [4,8].

The nature of machine learning problems tasks i that they can afford to tolerate a level of imprecision which is not available in other application domains (e.g., airplane engine simulation). Together with specific optimization pattern that emerges in many machine learning provides an opportunity for creating "near embarrassingly parallel" implementations.

5 Discussion and Conclusion

The availability of Big Data across many application domains has led to the promise of designing new applications which hitherto were considered out of reach. For example, Big Data has been instrumental in designing algorithms for real time language translation, high quality speech recognition and image and video search. Large amount of FMRI and MEG brain data collected while people are carrying out routine tasks holds the promise of understanding the working of the brains.

To fully utilize the promise of Big Data several hurdles in the computational, systems and algorithmic aspects of data processing have to be overcome. In the Big Data Operating Paradigm (BDOP) the growth rate of data is higher than the corresponding increase in computational processing speed and storage capacity. This necessarily leads to an environment where data has to be processed in a distributed manner. Since many algorithm for data processing and machine learning are super-linear, increase in data size results in a much higher increase in infrastructure resources - if quality of service constraints have to be met. From a computational perspective many data analytic tasks can be abstracted as the processing of a bipartite graph where the nodes are rows of a table and the edges are determined by a similarity function based on the columns. If the edges of the graph were uniformly distributed then processing in BDOP would be impossible. However bipartite graphs of real data tends to be highly sparse and skewed. Sparsity provides an opportunity to process data efficiently in BDOP but skewness results in a lower bound on the computation. The interplay between sparsity and skewness is a major systems challenge that is currently addressed on a case by case basis.

From a statistical perspective, availability of Big Data does not necessarily result in better quality or stable results. In fact the danger of deriving spurious relationships are greater in a high-dimensional than a low-dimensional setting. For real data it has been observed that even though the ambient dimension of the data may be high, its intrinsic dimension is low. This is often referred to as "data living in a low dimensional manifold." The low intrinsic dimensionality explains the widespread use of dimensionality reduction techniques like SVD and NMF. Most dimensionality reduction techniques have high computational complexity (often $O(n^3)$). Sometimes the use of random projection in conjunction with dimensionality reduction can lead to improved efficiency without substantial loss in accuracy as machine learning task output can afford a degree of imprecision which is greater than in many other domains (like aircraft engine design or exactness of OLTP query result).

Machine Learning tasks are often cast as optimization problems. Even the simplest of tasks, like data summarization, can result in complex optimization formulations. The intrinsic coupling between the approximation, estimation and computational error while solving the optimization task has several implications including the ability to use simple first order algorithms like stochastic gradient descent which trade-off between estimation and computational error in a transparent manner. For example, the typical cost of solving the k-means algorithm on a data set of size n is $O(nI)$ where I is the number of iterations. In BDOP the standard k-means algorithms is near impossible to use as it requires multiple passes over the data. However, SGDA can start producing reasonably accurate clusters within one pass and time budget can be used to settle on the quality of the result. Furthermore the form of a typical optimization objective function and tolerance of a certain amount of imprecision makes machine learning tasks amenable to highly efficient distributed and parallel implementations.

In the last five years there has been a resurgence of interest in deep learning. While deep learning is synonymous with neural networks, the key insight is to *infer* a representation of raw data specific for the task at hand. This is often referred to as dictionary learning or sparse coding. The dictionary learning problem can be cast as optimization problem where the objective has a similar form like the clustering problem. SGDA algorithms have been successfully employed for dictionary learning which attest to their versatility.

A grand challenge in the Big Data landscape continues to be a lack of a system which has the robustness and scalability of a traditional relational database management system while offering the expressive power to model a large and customizable class of machine learning problems.

References

1. Bottou, L., Bengio, Y.: Convergence properties of the k-means algorithms. In: Advances in Neural Information Processing Systems Conference, NIPS, Denver, Colorado, USA, pp. 585–592 (1994)

2. Bottou, L., Bousquet, O.: The tradeoffs of large scale learning. In: Proceedings of the Twenty-First Annual Conference on Neural Information Processing Systems, Advances in Neural Information Processing Systems 2007, Vancouver, British Columbia, Canada, December 3-6, pp. 161–168 (2007)
3. Christen, P.: A survey of indexing techniques for scalable record linkage and deduplication. IEEE Trans. Knowl. Data Eng. 24(9), 1537–1555 (2012)
4. Dean, J., Corrado, G., Monga, R., Chen, K., Devin, M., Mao, M.: Large scale distributed deep networks. In: Pereira, F., Burges, C., Bottou, L., Weinberger, K. (eds.) Advances in Neural Information Processing Systems, vol. 25, pp. 1223–1231 (2012)
5. Goodfellow, I.J., et al.: Challenges in representation learning: A report on three machine learning contests. In: Lee, M., Hirose, A., Hou, Z.-G., Kil, R.M. (eds.) ICONIP 2013, Part III. LNCS, vol. 8228, pp. 117–124. Springer, Heidelberg (2013)
6. Hinton, G.E.: Deep belief nets. In: Encyclopedia of Machine Learning, pp. 267–269 (2010)
7. Kraska, T., Talwalkar, A., Duchi, J.C., Griffith, R., Franklin, M.J., Jordan, M.I.: Mlbase: A distributed machine-learning system. In: Proceedings of the Sixth Biennial Conference on Innovative Data Systems Research, CIDR 2013, Asilomar, CA, USA, January 6-9 (2013)
8. Li, M., Andersen, D.G., Park, J.W., Smola, A.J., Ahmed, A., Josifovski, V., Long, J., Shekita, E.J., Su, B.-Y.: Scaling distributed machine learning with the parameter server. In: 11th USENIX Symposium on Operating Systems Design and Implementation, OSDI 2014, Broomfield, CO, USA, October 6-8, pp. 583–598 (2014)
9. Mairal, J., Bach, F., Ponce, J., Sapiro, G.: Online learning for matrix factorization and sparse coding. Journal of Machine Learning Research 11, 19–60 (2010)