DEALING WITH “VERY LARGE” DATASETS
An Overview of a Promising Research Line: Distributed Learning

Diego Peteiro-Barral, Bertha Guijarro-Berdiñas and Beatriz Pérez-Sánchez
Dept. of Computer Science, Faculty of Informatics, University of A Coruña, Campus de Elviña s/n, 15071 A Coruña, Spain

Keywords: Machine learning, Large-scale learning, Very large dataset, Data fragmentation, Distributed learning.

Abstract: Traditionally, a bottleneck preventing the development of more intelligent systems was the limited amount of data available. However, nowadays in many domains of machine learning, the size of the datasets is so large that the limiting factor is the inability of learning algorithms to use all the data to learn with in a reasonable time. In order to handle this problem a new field in machine learning has emerged: large-scale learning, where learning is limited by computational resources rather than by the availability of data. Moreover, in many real applications, “very large” datasets are naturally distributed and it is necessary to learn locally in each of the workstations in which the data are generated. However, the great majority of well-known learning algorithms do not provide an admissible solution to both problems: learning from “very large” datasets and learning from distributed data. In this context, distributed learning seems to be a promising line of research with which to deal with both situations, since “very large” concentrated datasets can be partitioned among several workstations. This paper provides some background regarding distributed environments as well as an overview of distributed learning for dealing with “very large” datasets.

1 INTRODUCTION

In the year 2000, the total amount of information on the Web varied somewhere between 25 and 50 terabytes (School of Information and Management and Systems, 2000). In 2003, in (School of Information and Management and Systems, 2003) its size was estimated at 167 terabytes, i.e. it more than tripled in just 3 years. By 2005, the total size was approximately 600 terabytes (D-Lib Magazine, 2006), i.e. another fourth times greater in two years. Nowadays, the total amount of information on the Web is almost incalculable and, what is more, the “deep” Web, which consists of specialized Web-accessible databases and dynamic web sites, is anywhere from 400 to 550 times larger than the information on the “surface” (School of Information and Management and Systems, 2003). Furthermore, the unrestrainable growth of data in fields such as bioinformatics, intrusion detection in computer networks, text classification (spam, no-spam) or engineering problems in which data are continuously recorded in a SCADA system such as hydroelectric or nuclear power stations, opens the way for new applications of machine learning as automatic data analyzers are needed since a human, even an expert, cannot look at a “very large” dataset and plausibly find a good solution for a given problem based on those data. In this situation, new challenges are raised regarding the scalability of current learning algorithms in order to be able to efficiently work with “very large” datasets.

1.1 Large-scale Learning

Users of “very large” datasets include researchers and practitioners from diverse fields such as database systems or machine learning. This begs the question, how large is “very large”? The answer is very different for each user in spite of sharing a research area in some way related. In the first place, the database community generally deals with gigabytes of data. “Very large” to a database practitioner usually means databases of 100 gigabytes or larger (Agrawal and Srikant, 1994). On the other hand, researchers in machine learning generally deal with flat files and algorithms that run in minutes or even seconds on a reasonably priced computer. For them, $100,000$ data (instances $\times$ features) is the beginning for the range of “very large” datasets (Provost and Kolluri, 1999). Nevertheless, in order to provide a single answer, we will take an algorithmic perspective on the issue of “very large” since this work is focused on learning

Peteiro-Barral D., Guijarro-Berdiñas B. and Pérez-Sánchez B..
DEALING WITH “VERY LARGE” DATASETS - An Overview of a Promising Research Line: Distributed Learning.
DOI: 10.5220/0003288804760481
In *Proceedings of the 3rd International Conference on Agents and Artificial Intelligence* (ICAART-2011), pages 476-481
ISBN: 978-989-8425-40-9

algorithms. So, for most published work on algorithms, $1,000,000$ data is considered to a be a “very large” dataset (somewhere between 100 megabytes and 1 gigabyte). Furthermore, this concurs with Huber’s assessment from a statistical perspective, given in this KDD-97 invited talk (Huber, 1997): “Somewhere around data sizes of 100 megabytes or so, qualitatively new, very serious scaling problems begin to arise, both on the human and on the algorithmic side”. More recently, a sample of the increasing interest generated by “very large” datasets was revealed with the organization of the workshop *PASCAL Large Scale Learning Challenge* (Sonnenburg et al., 2009). This workshop was concerned with the scalability and efficiency of existing learning algorithms with respect to computational and memory resources. Thus, “very large” datasets (more than 25 million data each) were used during the challenge in order to assess the scalability of learning algorithms. In this sense, most current machine learning algorithms are able to deal with medium-size datasets but they cannot be applied over datasets with more than $1,000,000$ data. In many cases, learning algorithms are not able to process the whole training dataset and, in practice, preprocessing techniques are needed since frequently most of them are limited to $200,000$ data due to time or memory restrictions. However, and that is the point, the need for preprocessing techniques (e.g. subsampling) is a constraint on learning algorithms by themselves and is not a conceptual constraint for processing “very large” datasets. Moreover, increasing the size of the training set of learning algorithms often increases the accuracy of the classification models (Catlett, 1991).

In order to handle “very large” datasets, a new and active research field emerges: large-scale learning (Bottou and Bousquet, 2008; Sonnenburg et al., 2007). This new field, which is located within machine learning, intends to develop efficient and scalable algorithms with regard to requirements of computation, memory, time and communications. Thus, for scaling up learning algorithms, the issue is not so much as one of speeding up a slow algorithm as one of turning an impracticable algorithm into a practicable one, i.e. the crucial issue is seldom “how fast” you can run a particular problem, but rather “how large” a problem you can deal with (Provost and Kolluri, 1999). However, practically all existing implementations of algorithms operate with the training set entirely in main memory. If the computational complexity of the algorithm exceeded the main memory then the algorithm will not scale well or will be unfeasible to run. Finally, even if the scalability of learning algorithms is achieved, there is still the question of its impact on the goal of learning. Evaluating and comparing performance becomes complicated if a degradation in the quality of the learning is allowed. Thus, we are mostly interested in methods that scale up without a substantial decrease in the quality of learning.

1.2 Scaling Up Learning Algorithms

Large-scale learning has received considerable attention in the recent years and many successful techniques have been proposed and implemented (Moretti et al., 2008; Krishnan et al., 2008; Raina et al., 2009). The different techniques proposed in the literature can be categorized into three main approaches where, in most cases, techniques from separate categories are independent and can be applied simultaneously. The three main approaches are: a) design a fast algorithm, b) use a relational representation, and c) partition the data (Provost and Kolluri, 1999).

1.2.1 Fast Algorithms

The most straightforward approach to scaling up learning algorithms is to produce more efficient algorithms or to increase the efficiency of the existing ones, including a wide variety of techniques for reducing asymptotic complexity, for optimizing the search and representation, or for finding approximate solutions rather than an exact one. However, it is usually the case where a fast algorithm may not be sufficient for dealing with “very large” problems.

1.2.2 Relational Representations of Data

Most existing learning algorithms were not designed to handle “very large” datasets. In particular, the great majority were designed under the assumption that the dataset would be represented as a single memory-resident table. Unfortunately, producing flat files from real-world datasets is usually unrealistic, since the data is usually stored in relational databases and the flattening-out process may not be feasible due to time or memory space restrictions. The relational representation approach addresses data that cannot be feasibly treated as a flat file by learning directly from any relational database. Thus, the ability of relational databases to compress data is critical in order to learn from “very large” datasets. In this sense, the development of methods able to learn from a database is needed in order to deal with “very large” datasets.

1.2.3 Data Partitioning

Data partitioning involves breaking up the dataset into subsets, learning from one or more of them and, finally, combining the results. Thus, running algorithms on "very large" datasets is avoided since the size of the subsets is usually many times smaller than the whole dataset. Data partitioning can be categorized based on whether they separate the subsets by instances or by features, being a particular case those that choose a subset of instances or a subset of features, i.e. instance selection and feature selection, respectively. Equally, multiple subsets can be chosen and they can be processed in sequence (incremental learning) or concurrently (distributed learning).

2 DISTRIBUTED LEARNING

One of the most promising research lines for large-scale learning is distributed learning since allocating the learning process among several workstations is a natural way of scaling up learning algorithms. Furthermore, the current trend of reducing the speed of processor in favor of computer clusters and multi-core processors leads to a suitable context for distributed learning.

2.1 A Distributed Environment

A distributed computing environment can be defined as a set of workstations physically separated but mutually connected by a communications network, including in this definition multi-core processors. In this context, if the data is distributed across the workstations there are at least three possible kinds of fragmentation of the dataset: a) horizontal fragmentation, whereby subsets of instances are stored at different workstations; b) vertical fragmentation, whereby subsets of features are stored at different workstations; c) a combination of both (Caragea et al., 2001). In this context, if subsets of data are denoted by D_1, D_2, $\ldots$, D_n, and the corresponding complete dataset by D, then a horizontally fragmented dataset has the following property:

$$D_1 \cup D_2 \cup \cdots \cup D_n = D \qquad (1)$$

where $\cup$ denotes the set union. Hence, L being a non-distributed algorithm, the challenge is to develop a distributed learning algorithm L_d which guarantees the following property:

$$L_d(D_1, D_2, \ldots, D_n) = L(D_1 \cup D_2 \cup \cdots \cup D_n) \qquad (2)$$

Similarly, a vertically fragmented dataset has the following property:

$$D_1 \times D_2 \times \cdots \times D_n = D \qquad (3)$$

where $\times$ denotes the join operation. In a similar manner, the challenge is to develop a distributed learning algorithm L_d which guarantees the following property:

$$L_d(D_1, D_2, \ldots, D_n) = L(D_1 \times D_2 \times \cdots \times D_n) \qquad (4)$$

The great majority of distributed datasets are horizontally fragmented since it constitutes the most suitable approach for most applications. Vertical fragmentation is solely useful where the representation of data could vary by adding new features, e.g. novel laboratory experiments.

While machine learning applications are now most often distributed, most existing learning algorithms cannot handle this circumstance. Thus, gathering the distributed datasets in a single node for central processing is required for the great majority of them. However, this is usually either inefficient or unfeasible for the following reasons (Tsoumakas, 2009):

- *Storage cost*: the cost of storing a central dataset is much larger than the sum of the cost of storing smaller parts of the dataset. For example, considering a multinational corporation, with thousands of establishments throughout the world, who wants to store data regarding all purchases of all its customers. The central storage of this data would require a huge database at an enormous monetary cost.
- *Computational cost*: in some way related to the storage cost, the cost of learning on a "very large" dataset is much larger than the sum of the cost of learning on smaller parts of the dataset, that could also be done in parallel. Following the example above, the best way to quickly develop a successful business strategy is to analyze the data in a distributed way, since analyzing the data in a centralized way takes too long due to the "very large" number of instances.
- *Communication cost*: the exchange of a huge volume of data over a network could take a very long time and also could require a enormous monetary cost. Note also that it is common to have frequently updated databases and communication could be a continuous overhead. For example, following the above example and even when the storage and computational costs were not too high, the communication of a "very large" volume of data takes too long for quickly developing a successful business strategy.
- *Private and sensitive data*: there are many real-world applications that deal with sensitive data such as medical or financial records. Thus, even if the communication cost is not so high, the exchange of raw data might not be desirable since its privacy could be at risk. For example, finan-

cial corporations who want to cooperate in preventing fraudulent intrusion into their computing systems (Kargupta et al., 2000). The data stored by financial corporations is sensitive and cannot be exchanged with outsiders. Moreover, even if it were possible, the corporations could be rivals and they may want to only exchange knowledge without the exchange of raw data.

The development of distributed learning algorithms seems necessary and for this it is receiving considerable attention. In recent years, several distributed learning algorithms were proposed in the literature (Wolpert, 1992; Chan and Stolfo, 1993; Tsoumakas and Vlahavas, 2002; Lazarevic and Obradovic, 2002; Guijarro-Berdiñas et al., 2009) in order to learn from distributed datasets in an effective and efficient way. However, none of them so far have achieved wide popularity since distributed learning is a relatively new field and many solutions are implemented ad hoc.

2.2 Learning from Subsets of Data

Most distributed learning algorithms have their foundations in ensemble learning (Dietterich, 2000). Ensemble learning builds a set of classifiers in a centralized way in order to enhance the accuracy of a single classifier. Although there are other methods, the most common one builds the set of classifiers by training each one on different subsets of data. Afterwards, the classifiers are combined in a concrete way defined by the ensemble algorithm. Thus, the ensemble approach is almost directly applicable to a distributed environment since a classifier can be trained at each workstation, which stores a subset of data, and then the classifiers can be eventually aggregated using ensemble strategies. In this sense, the following advantages of distributed learning come from the advantages of ensemble learning (Guo and Sutiwaraphun, 1999):

- *Learning accuracy*: using different learning processes to train several classifiers from distributed datasets increases the possibility of achieving higher accuracy especially on a large-size domain. This is because the integration of such classifiers can represent an integration of different learning biases which possibly compensate one another for their inefficient characteristics. Hansen & Salamon (Hansen and Salamon, 1990) have shown that, for an ensemble of artificial neural networks, if all classifiers have the same probability of making error of less than 0.5 and if all of them make errors independently, then the overall error must decrease as a function of the number of classifiers.
- *Execution time and memory limitations*: learning in a distributed way provides a natural solution for large-scale learning where algorithm complexity and memory limitation are always the main obstacles. If several workstations or a multi-core processor are available, then each workstation or processor, respectively, can work on a different partition of data in order to independently derive a classifier. Therefore, the memory requirements as well as the execution time (assuming some minor communication overhead) become smaller since the computational cost of training several classifiers on subsets of data is lower than training one classifier on the whole dataset (see Section 2.1).
- *Scalability*: distributed learning is inherently scalable since the growing amount of data may be offset by increasing the number of workstations. In this manner, scalability of distributed learning is in some way related to execution time and memory limitations.

Since several classifiers are trained during the distributed learning process, the last outstanding issue is related to the way in which they can be combined. In distributed learning as well as in ensemble learning, there are in general two types of information to be combined: on the one hand, the classifiers by themselves and, on the other hand, the predictions of the classifiers. In the first case, most learning algorithms are concerned with learning concept descriptions in the form of a decision tree or a set of rules, among others, expressed in terms of the originally given attributes. Nevertheless, the type of learning technique employed at one workstation might be different from the one employed at another, since there is no restriction on this aspect. Consequently, the learning algorithms could have different representations and, in order to combine the classifiers by themselves, we need to define a uniform representation to which the different classifiers are translated. It is difficult to define such a representation to encapsulate all other representations without losing a significant amount of information during the translation. Furthermore, due to this difficulty, one could restrict, to a large degree, the information supported by the classifier. For example, it is difficult to define a uniform representation to merge a distance-based learning algorithm with a rule-based learning algorithm and, even if it were possible, the amount of information lost during translation might be unacceptable.

An alternative strategy to combine classifiers is to merge their predictions instead of the classifiers themselves. In this way, the representation of the classifiers and their internal organization is completely transparent since the type of information is based on the predictions which are the outputs of the classifiers for a particular dataset, i.e. a hypothesized class for each instance. The predictions can be categorical or non-categorical (associated with some numeric measure like probabilities or distances) but, in this case, the difficulty to define a uniform framework is much less severe. Thus, non-categorical predictions can be treated as categorical by simply choosing the class where the measure reaches the highest value. The opposite is not considered, since converting categorical predictions into predictions with numeric measures is undesirable or impossible. The great majority of learning algorithms published in the literature focus their development on combining the predictions of the classifiers, since any classifier can be employed in this case, avoiding potential problems with concept descriptions and knowledge representation.

2.3 Distributed Learning Algorithms

In recent years, distributed learning has received considerable attention in the literature. However, few distributed algorithms have been proposed so far since distributed learning is a relatively new field of research. To date, distributed algorithms have not yet achieved as high a level of popularity as artificial neural networks or support vector machines in centralized learning. Moreover, some of the distributed learning algorithms proposed focus only on distributing a dataset into multiple processors to exploit parallel processing to speed up learning. Only a few of them take into account issues related to naturally distributed datasets as privacy-preserving computation or communication costs.

The aim of this work is not to provide a deep review of existing distributed learning algorithms but to discuss some relevant issues in distributed learning. However, in order to facilitate the work for interested readers, some references regarding distributed learning algorithms are detailed below. Some of them were originally developed for ensemble learning, but they can be used as distributed learning with a few modifications (basically, communication issues have to be taken into account): *Fixed rules* (Kittler, 1998), *Stacked generalization* (Wolpert, 1992), *Meta-learning* (Chan and Stolfo, 1993), *Knowledge probing* (Guo and Sutiwaraphun, 1999), *Pasting votes* (Chawla et al., 2002), *DAGGER* (Davies et al., 2000), *Effective stacking* (Tsoumakas and Vlahavas, 2002), *Effective voting* (Tsoumakas et al., 2004b), *Distributed boosting* (Lazarevic and Obradovic, 2002), *Distributed clustering* (Tsoumakas et al., 2004a), and *DEvoNet* (Guijarro-Berdiñas et al., 2009).

3 DISCUSSION

Although some progress was achieved regarding distributed learning, many of the algorithms presented in the literature perform simulated experiments on a single computer and do not take into account restrictions regarding some distributed environments, e.g. data privacy or communication costs. Moreover, usually the assessment of these algorithms is performed on public machine learning datasets that are partitioned in order to simulate a distributed environment. This does not represent the singularities showed in naturally distributed datasets where the data distributions among partitions may not be identical. The performance of the algorithms is affected by this fact but most algorithms cannot handle it. Finally, comparisons among distributed learning algorithms presented in the literature are usually focused on assessing a few algorithms on a few datasets and, what is worse, they usually involve different evaluation criteria. As a result, it is difficult to determine how a method behaves and compare with others in terms of accuracy, spatial and time complexity, or scalability.

4 CONCLUSIONS

An overview of distributed learning was presented in this work. Distributed learning seems essential in order to provide solutions for learning from both “very large” datasets (large-scale learning) and distributed datasets. On the one hand, “very large” datasets can be scattered among distributed workstations, turning an impracticable algorithm into a practicable one. In this sense, distributed learning provides a scalable solution to deal with “very large” datasets since the growing volume of data may be offset by increasing the number of workstations. On the other hand, distributed learning avoids the necessity of gathering distributed data into a single workstation for central processing, saving time and money. Moreover, situations where learning was not possible become possible by keeping intact the privacy of the data. For future work, a deep study of distributed learning algorithms appears necessary.

ACKNOWLEDGEMENTS

This research was partially funded by the Xunta de Galicia by projects 2008/060 and PGIDT-08TIC012105PR, and the Ministerio de Ciencia e Innovación by project TIN2009-10748.

REFERENCES

Agrawal, R. and Srikant, R. (1994). Fast algorithms for mining association rules. In *Proceedigns of the 20th International Conference on Very Large Data Bases (VLDB)*, volume 1215, pages 487–499.

Bottou, L. and Bousquet, O. (2008). The tradeoffs of large scale learning. *Advances in neural information processing systems*, 20:161–168.

Caragea, D., Silvescu, A., and Honavar, V. (2001). Analysis and synthesis of agents that learn from distributed dynamic data sources. *Emergent neural computational architectures based on neuroscience*, pages 547–559.

Catlett, J. (1991). *Megainduction: machine learning on very large databases*. PhD thesis, School of Computer Science, University of Technology, Sydney, Australia.

Chan, P. and Stolfo, S. (1993). Toward parallel and distributed learning by meta-learning. In *AAAI in Knowledge Discovery in Databases*, pages 227–240.

Chawla, N., Hall, L., Bowyer, K., Moore, T., and Kegelmeyer, W. (2002). Distributed pasting of small votes. *Multiple Classifier Systems*, pages 52–61.

D-Lib Magazine (2006). A Research Library Based on the Historical Collections of the Internet Archive. http://www.dlib.org/dlib/february06/arms/02arms.html. [Online; accessed 27-Oct.-2010].

Davies, W., Edwards, P., and Scotland, U. (2000). Dagger: A new approach to combining multiple models learned from disjoint subsets. *Machine Learning*, 2000:1–16.

Dietterich, T. (2000). Ensemble methods in machine learning. *Multiple classifier systems*, pages 1–15.

Guijarro-Berdiñas, B., Martínez-Rego, D., and Fernández-Lorenzo, S. (2009). Privacy-Preserving Distributed Learning Based on Genetic Algorithms and Artificial Neural Networks. *Distributed Computing, Artificial Intelligence, Bioinformatics, Soft Computing, and Ambient Assisted Living*, pages 195–202.

Guo, Y. and Sutiwaraphun, J. (1999). Probing knowledge in distributed data mining. *Methodologies for Knowledge Discovery and Data Mining*, pages 443–452.

Hansen, L. and Salamon, P. (1990). Neural network ensembles. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(10):993–1001.

Huber, P. (1997). From large to huge: A statistician's reaction to KDD and DM. In *Proceedings of the 3rd International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 304–308.

Kargupta, H., Park, B., Hershberger, D., and Johnson, E. (2000). Collective Data Mining: A New Perspective Toward Distributed Data Mining. *Advances in Distributed and Parallel Knowledge Discovery*, pages 131–178.

Kittler, J. (1998). Combining classifiers: A theoretical framework. *Pattern Analysis & Applications*, 1(1):18–27.

Krishnan, S., Bhattacharyya, C., and Hariharan, R. (2008). A randomized algorithm for large scale support vector learning. In *Proceedings of Advances in Neural Information Processing Systems (NIPS)*, pages 793–800.

Lazarevic, A. and Obradovic, Z. (2002). Boosting algorithms for parallel and distributed learning. *Distributed and Parallel Databases*, 11(2):203–229.

Moretti, C., Steinhaeuser, K., Thain, D., and Chawla, N. (2008). Scaling up classifiers to cloud computers. In *Proceedings of the 8th IEEE International Conference on Data Mining (ICDM)*, pages 472–481.

Provost, F. and Kolluri, V. (1999). A survey of methods for scaling up inductive algorithms. *Data mining and knowledge discovery*, 3(2):131–169.

Raina, R., Madhavan, A., and Ng, A. (2009). Large-scale deep unsupervised learning using graphics processors. In *Proceedings of the 26th Annual International Conference on Machine Learning (ICML)*, pages 873–880.

School of Information and Management and Systems (2000). How much information? http://www2.sims.berkeley.edu/research/projects/how-much-info/internet.html. [Online; accessed 27-September-2010].

School of Information and Management and Systems (2003). How much information? http://www2.sims.berkeley.edu/research/projects/how-much-info-2003/internet.htm. [Online; accessed 27-Sept.-2010].

Sonnenburg, S., Franc, V., Yom-Tov, E., and Sebag, M. (2009). PASCAL Large Scale Learning Challenge. *Journal of Machine Learning Research*.

Sonnenburg, S., Ratsch, G., and Rieck, K. (2007). Large scale learning with string kernels. *Large Scale Kernel Machines*, pages 73–104.

Tsoumakas, G. (2009). Distributed Data Mining. *Database Technologies: Concepts, Methodologies, Tools, and Applications*, pages 157–171.

Tsoumakas, G., Angelis, L., and Vlahavas, I. (2004a). Clustering classifiers for knowledge discovery from physically distributed databases. *Data & Knowledge Engineering*, 49(3):223–242.

Tsoumakas, G., Katakis, I., and Vlahavas, I. (2004b). Effective voting of heterogeneous classifiers. *Machine Learning: ECML 2004*, pages 465–476.

Tsoumakas, G. and Vlahavas, I. (2002). Effective stacking of distributed classifiers. In *ECAI 2002: 15th European Conference on Artificial Intelligence*, page 340.

Wolpert, D. (1992). Stacked generalization. *Neural networks*, 5(2):241–259.