Neurocomputing 124 (2014) 2–12

Contents lists available at ScienceDirect

Neurocomputing

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

Distributed machine learning in networks by consensus

Leonidas Georgopoulos [a,*], Martin Hasler [b]

[a] Avenue de l'Eglise Anglaise 12, CH-1006 Lausanne, Switzerland
[b] School of Computer and Communication Sciences, École Polytechnique Fédéral de Lausanne (EPFL). EPFL IC ISC LANOS, Station 14, CH-1015 Lausanne, Switzerland

ARTICLE INFO

Article history:
Received 15 September 2011
Received in revised form
28 September 2012
Accepted 3 December 2012
Available online 8 April 2013

Keywords:
Distributed machine learning
Parallel machine learning
Gradient descent
Consensus
Peer-to-peer learning
Neural networks

ABSTRACT

We propose an algorithm to learn from distributed data on a network of arbitrarily connected machines without exchange of the data-points. Parts of the dataset are processed locally at each machine, and then the consensus communication algorithm is employed to consolidate the results. This iterative two stage process converges as if the entire dataset had been on a single machine. The principal contribution of this paper is the proof of convergence of the distributed learning process in the general case that the learning algorithm is a contraction. Moreover, we derive the distributed update equation of a feed-forward neural network with back-propagation for the purpose of verifying the theoretical results. We employ a toy classification example and a real world binary classification dataset.

1. Introduction

We consider the case of supervised machine learning, but in a distributed setting where the dataset is partitioned and a part resides at each machine in the communication network. We assume that for some reason, we are unwilling to communicate the data between machines, or gather it centrally for computation. Instead of just using the local data at each machine, we would like to learn from the entire dataset but without exchange of data-points. This is accomplished by employing the consensus algorithm, which is briefly reviewed in this paper, Section 1.2.

There are numerous occasions where the entire dataset may not be available. We conceive these as combinations of the following four basic cases. First, the dataset is too large to be handled by a single machine due to either hardware, software implementation, and or algorithmic limitations. Second, data is intrinsically distributed. That is the case when data is generated by a set of machines which acquire data by observation or examination (e.g. a set of sensors for environmental monitoring). Third, when data has to remain private but decisions are better performed globally (e.g. when working with clinical patient data). Finally, when data cannot be collected. Hence, it is inaccessible or its access is practically infeasible. This might be due to many reasons, among those just a few are communication costs and failures, energy consumption, and machine downtime. A few of the possible applications where the problem arises are wireless sensor networks, data mining in large datasets, distributed databases, social networks, robotic applications, and inference from confidential or private data.

1.1. Problem layout

For the rest of this paper we assume a dataset, partitioned and distributed over different machines in an arbitrary manner. Our purpose is to learn from the dataset such that any of the machines when presented with an example from the same generating process can successfully classify it. Moreover, we wish that the classification performed by any machine is identical and the performance equivalent to a centralised case. Moreover, this has to be achieved without exchanging any data-points between the machines. Specifically, any machine can communicate with other machines but not necessarily with every other machine. However, we require that the connection graph has no disconnected components. Obviously, disconnected components cannot be brought to agreement by means of communication algorithms.

In our algorithm, the elementary learning process, risk computation and model update, is modified, and becomes a two phase process. The first phase consists of learning with the dataset available locally. This is performed simultaneously at every machine. Therefore, identical learning machines are trained locally but with different data drawn from the same generating process. In the second phase, the parameters of the model are

* Corresponding author. Tel.: +41 786272711.
E-mail addresses: g24l@ieee.org (L. Georgopoulos).
martin.hasler@epfl.ch (M. Hasler).

0925-2312/$ - see front matter © 2013 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.neucom.2012.12.055

communicated to initiate the consensus algorithm and estimate the mean of the learned parameters. This iterated two phase process, has roughly the same effect as if a single classifier was trained on the entire dataset.

The use of the consensus algorithm is certainly not a prerequisite. The necessary step is to compute the mean of the model updates at each step, and compute the new update with the computed mean at each machine. The proof that such a process converges as the non-distributed counterpart, in Section 2.1, is the main contribution of this paper. Such a computation, may in general be achieved with other communication protocols as well. One may consider from broadcasting to gathering the values to a central hub and then broadcasting the results back to the machines. Although these approaches have their predicaments, such as congestion, need for routing protocols, and increased administration complexity with respect to the number of nodes on the network and its topology, it is not in the interest of this study to compare these different approaches in depth. The main advantage of the consensus algorithm, which justifies its use in this study, is that the algorithm is so simple that it can be implemented in numerous scenarios; thus separating the theoretical analysis from the implementation details, without making the work inapplicable. Nonetheless, to demonstrate that distributed machine learning can be achieved in even the most simplistic scenarios of ad-hoc communication networks brings additional value to this work, and extends the list of possible applications.

1.2. Consensus algorithm

Assume a network of machines coupled in an arbitrary manner. Let the communication graph be $\mathcal{G}(\mathcal{V},\mathcal{E})$ such that it is connected but not fully. Suppose, every vertex i in the graph has an associated scalar value $x_i \in \mathbb{R}$. The consensus algorithm computes the arithmetic mean of these values at each vertex. This is possible just by local communication between connected vertices.

The linear consensus algorithm as presented in [1] consists of a simple vertex-local update equation $x_i(t+1) = \sum_j w_{ij} x_j(t)$, where $w_{ij} \in \mathbb{R}_+$ are coefficients associated with the edges of the graph. Specifically, $w_{ij} \neq 0$ when vertices i and j are connected and $w_{ij} = 0$ otherwise. These coefficients guarantee the coherence of the local estimates. The global update equation is

$$\mathbf{x}(t+1) = \mathbf{W}\mathbf{x}(t) \tag{1}$$

where the elements of $\mathbf{W} \in \mathbb{R}^{n\times n}$ are the coefficients w_{ij} on the edges. The process is presented in Algorithm 1.

Algorithm 1. Consensus Algorithm, $\mathcal{S}(\mathbf{x}, q)$.

1: Execute the while loop for every i-th machine simultaneously
2: **for** $t = 1$ to q **do**
3: $\quad x_i \leftarrow \sum_{j=1}^{n} w_{ij} x_j$
4: **end for**

The convergence of the algorithm depends solely on the selection of these coefficients. Sufficient conditions for convergence are $\mathbf{W}^{\mathrm{T}} = \mathbf{W}$, $\mathbf{W}\mathbf{1} = \mathbf{1}$, $\rho(\mathbf{W} - \mathbf{1}\mathbf{1}^{\mathrm{T}}/n) < 1$ where $\mathbf{1} = (1, 1, 1, \ldots, 1)^{\mathrm{T}} \in \mathbb{R}^n$ and $\rho(\cdot)$ denotes the spectral radius. This dynamical system has asymptotic convergence to the arithmetic mean $(1/n)\mathbf{1}\mathbf{1}^{\mathrm{T}}\mathbf{x}$. Practically, the number of iterations $q \in \mathbb{Z}^+$ affects the precision of estimation of the mean and the level of agreement, [2].

Advantages of employing the consensus algorithm include and may not be limited to being inherently distributed and robust, having no need for routing tables, sub-network wide switching, and packet switching. In the simplest case, what is necessary, but usually trivial to employ, is the need to determine time-slots only between networked neighbours. Moreover, the consensus algorithm can be very well applied to both digital and analog networks [3], and to LAN and WAN networks, with little effort. Moreover, it is so simple to employ that it permits the application of this work in even the simplest networks, such as ad-hoc wireless sensor networks, robot swarms, and simple peer-to-peer networks, without interfering with employed communication protocol stack. Finally, the algorithm needs no central coordination center, e.g. a switch to broadcast the packet, which distinguishes it among its counterparts; thus allowing it to operate in completely decentralised fashion.

1.3. Definitive consensus algorithm

The main drawback for the application of the consensus algorithm is the large number of communications needed to reach consensus. This can be alleviated if the communication coefficients w_{ij} are switched in a timely manner [4]. This can be achieved with the definitive consensus algorithm that in fact permits the network to reach consensus in fixed and finite number of iterations. The coefficients can be obtained by numerically solving the equation below

$$\mathbf{W}_d \mathbf{W}_{d-1} \ldots \mathbf{W}_1 = \frac{\mathbf{1}\mathbf{1}^{\mathrm{T}}}{n} \tag{2}$$

where d is the graph diameter, and $\mathbf{W}_d, \mathbf{W}_{d-1}, \ldots, \mathbf{W}_1$ are weight matrices corresponding to $\mathcal{G}$. The solutions to this equation are easy to retrieve up to medium sized graphs with a numerical solver [4].

1.4. Related work

Related work with the problem at hand may be found in the field of distributed optimisation [5]. However, the interest of the researchers there is different, i.e. accuracy, quality of solution, convergence speed, violation of constraints, whereas in machine learning, it is generalisation, model selection, bias, and over-fitting. Thus optimisation cannot be considered to be equivalent to a learning problem. In the case of optimisation, the optimised function is known a priori. This allows for the computation of the Jacobian, the Hessian, and the retrieval of KKT conditions that designate the proper execution of the algorithm, and the quality of the obtained solution; especially of the sub-gradient computed with the consensus algorithm, which can be enforced to be within bounds by strict knowledge of qualitative benchmarks of the update step, such that the result is equivalent with the non-distributed case of evaluating the gradient. In contrast, in a machine learning process such facilities are not available since the model of the data is unknown, and one cannot evaluate the quality of a solution.

Outlining, a central problem of statistical learning theory is if a family of learning models imposed by a learning algorithm is appropriate to model the data; to shatter the data [6]. In the distributed setting the data-points locally available are a subset of those globally available. A learning machine that may be able to adequately learn on a given data set, may not learn the true underlying model on a subset of the data. The problem that arises in distributed machine learning, is if by performing machine learning distributively, and combining the local results, permits to shatter the data, as in the case of the non-distributed counterpart. This central question, lies far from previous work, and especially from approaches in the distributed optimisation literature. We base our theoretical results on the fact that a large number of machine learning algorithms are contractions on sets of learning data. This approach however does not provide guarantees