Neural Networks 80 (2016) 43–52

Contents lists available at ScienceDirect

Neural Networks

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

Distributed semi-supervised support vector machines

Simone Scardapane [a,*], Roberto Fierimonte [a], Paolo Di Lorenzo [b], Massimo Panella [a], Aurelio Uncini [a]

[a] *Department of Information Engineering, Electronics and Telecommunications (DIET), "Sapienza" University of Rome, Via Eudossiana 18, 00184 Rome, Italy*

[b] *Department of Engineering, University of Perugia, Via G. Duranti 93, 06125, Perugia, Italy*

ARTICLE INFO

Article history:
Received 23 October 2015
Received in revised form 12 April 2016
Accepted 17 April 2016
Available online 27 April 2016

Keywords:
Semi-supervised learning
Support vector machine
Distributed learning
Networks

ABSTRACT

The semi-supervised support vector machine (S^{3}VM) is a well-known algorithm for performing semi-supervised inference under the large margin principle. In this paper, we are interested in the problem of training a S^{3}VM when the labeled and unlabeled samples are distributed over a network of interconnected agents. In particular, the aim is to design a distributed training protocol over networks, where communication is restricted only to neighboring agents and no coordinating authority is present. Using a standard relaxation of the original S^{3}VM, we formulate the training problem as the distributed minimization of a non-convex social cost function. To find a (stationary) solution in a distributed manner, we employ two different strategies: (i) a distributed gradient descent algorithm; (ii) a recently developed framework for In-Network Nonconvex Optimization (NEXT), which is based on successive convexifications of the original problem, interleaved by state diffusion steps. Our experimental results show that the proposed distributed algorithms have comparable performance with respect to a centralized implementation, while highlighting the pros and cons of the proposed solutions. To the date, this is the first work that paves the way toward the broad field of distributed semi-supervised learning over networks.

1. Introduction

Semi-supervised learning (SSL) algorithms are a family of techniques for performing inference in the presence of both labeled and unlabeled data (Chapelle, Schölkopf, & Zien, 2006). Among them, in the binary classification setting the semi-supervised support vector machine (S^{3}VM) has attracted a large amount of attention over the last decades (Chapelle, Sindhwani, & Keerthi, 2008). The S^{3}VM is based on the idea of minimizing the training error and maximizing the margin over both labeled and unlabeled data, whose labels are included as additional variables in the optimization problem. Since its first practical implementation in Joachims (1999), inspired by previous work on transductive learning by Vapnik (1998), numerous researchers have proposed alternative solutions for solving the resulting mixed integer optimization problem, including branch and bound algorithms (Chapelle, Sindhwani, & Keerthi, 2006), convex relaxations (Chapelle & Zien, 2005; Li, Tsang, & Kwok, 2013), convex–concave procedures (Fung & Mangasarian, 2001), and others. It has been applied to a wide variety of practical problems, such as text inference (Joachims, 1999), and it has given birth to numerous other algorithms, including semi-supervised least-square SVMs (Adankon, Cheriet, & Biem, 2009), and semi-supervised random vector functional-link networks (Scardapane, Comminiello, Scarpiniti, & Uncini, in press).

In this paper, we are interested in designing algorithms for solving the S^{3}VM optimization problem, in the case where the training data is distributed across a network of interconnected agents (Scardapane, Wang, Panella, & Uncini, 2015). In the fully supervised case, this is a well-known scenario, which has been investigated extensively in multiple research fields, including peer-to-peer (P2P) (Ang, Gopalkrishnan, Hoi, & Ng, 2013) and sensor networks (Barbarossa, Sardellitti,& Di Lorenzo, 2014; Predd, Kulkarni, & Poor, 2006), robotic swarms, and many others. In all of these settings, the underlying network of agents is generally unstructured, and no centralized authority can coordinate the overall process. Thus, distributed training algorithms are designed based on simple local exchanges of information among

* Corresponding author. Tel.: +39 06 44585495; fax: +39 06 4873300.
E-mail addresses: simone.scardapane@uniroma1.it (S. Scardapane), robertofierimonte@gmail.com (R. Fierimonte), paolo.dilorenzo@unipg.it (P. Di Lorenzo), massimo.panella@uniroma1.it (M. Panella), aurelio.uncini@uniroma1.it (A. Uncini).

http://dx.doi.org/10.1016/j.neunet.2016.04.007
0893-6080/© 2016 Elsevier Ltd. All rights reserved.

neighboring agents. A large number of decentralized algorithms have been developed for training a supervised SVM in such a distributed scenario (Forero, Cano, & Giannakis, 2010; Lu, Roychowdhury, & Vandenberghe, 2008; Navia-Vázquez, Gutierrez-Gonzalez, Parrado-Hernández, & Navarro-Abellan, 2006), and they are briefly summarized in the next section.

To the best of our knowledge, however, there is a lack of distributed training algorithms for the SSL case over networks. Indeed, this problem has been addressed only for specific cases, such as localization in a WSN (Chen, Wang, Sun, & Shen, 2011). Nonetheless, as we argue in Fierimonte, Scardapane, Uncini, and Panella (submitted for publication), there is a large number of realistic applications where the agents can benefit from the inclusion of additional unlabeled data in the training process. As an example, consider a distributed medical application, where multiple clinical institutions possess similar databases, but privacy concerns do not allow them to share it with a centralized institution (Clifton, Kantarcioglu, Vaidya, Lin, & Zhu, 2002). In this case, labeled data is generally scarce, while each institution has access to a large amount of unlabeled samples. Using currently available distributed algorithms, however, would imply discarding these unlabeled databases, resulting in a possible loss of generalization accuracy.

To simplify our derivation, in this paper, we focus on the *linear* S^3VM formulation, whose decision boundary corresponds to an hyperplane in the input space. It is known that non-linear decision boundaries can be obtained with the use of kernel functions. In that case, however, the resulting SVM model is expressed in terms of all examples, which in a decentralized setting are distributed among the different agents. This is a notoriously complex problem (Predd et al., 2006), which in many contexts hinders the applicability of the resulting algorithms. In an alternative publication (Fierimonte et al., submitted for publication), we have explored the problem of training a semi-supervised Laplacian SVM using a distributed computation of the underlying kernel matrix. However, the resulting algorithm requires a large amount of computational and/or communication resources. The algorithms presented in this paper, instead, can be implemented even on agents with stringent requirements in terms of power, such as sensors in a WSN. At the same time, limiting ourselves to a linear decision boundary can be reasonable, as the linear S^3VM can perform well in a large range of settings, due to the scarcity of labeled data (Chapelle et al., 2008).

Specifically, starting from the smooth approximation to the original S^3VM presented in Chapelle and Zien (2005), we show that the distributed training problem can be formulated as the joint minimization of a sum of non-convex cost functions. This is a complex problem, which has been investigated only very recently in the distributed optimization literature (Bianchi & Jakubowicz, 2013; Di Lorenzo & Scutari, in press). In our case, we build on two different solutions. The first one is based on the idea of diffusion gradient descent (DGD) (Di Lorenzo & Sayed, 2013; Sayed, 2014a, 2014b), arisen from previous work in the context of distributed filtering applications (Lopes & Sayed, 2008). The main idea of DGD is to interleave gradient descent steps at every node, with local averaging of the estimates. The resulting algorithm leads to an extremely efficient implementation. Nevertheless, since it is a gradient-based algorithm exploiting only first order information of the objective function, it generally suffers from slow practical convergence speed, especially in the case of non-convex and large-scale optimization problems. Recently, it was showed in Di Lorenzo and Scutari (in press), Facchinei, Scutari, and Sagratella (2015) and Scutari, Facchinei, Song, Palomar, and Pang (2014) that exploiting the structure of nonconvex functions by replacing their linearization (i.e., their gradient) with a "better" approximant can enhance practical convergence speed. Thus, we propose a distributed algorithm based on the recently proposed In-Network Successive Convex Approximation (NEXT) framework (Di Lorenzo & Scutari, in press). The method hinges on successive convex approximation techniques while leveraging dynamic consensus (Zhu & Martínez, 2010) as a mechanism to distribute the computation among the agents as well as diffuse the needed information over the network. Both algorithms are provably convergent to stationary points of the non-convex optimization problem. Moreover, as shown in our experimental results, NEXT exhibits a faster practical convergence speed with respect to DGD, which is paid by a larger computation cost per iteration.

To summarize, our main contributions with respect to the current literature on distributed learning are two-fold. Firstly, to the best of our knowledge, this is the first work dealing explicitly with (fully) distributed implementations of semi-supervised routines and, more specifically, semi-supervised SVMs, paving the way to a large number of possible domains which can benefit from the availability of these techniques. Additionally, the present work is one of the first successful applications of optimization protocols explicitly designed for distributed *non-convex* costs, while the majority of works on distributed learning has focused on models giving rise to convex optimization problems.

The rest of the paper is structured as follows. Section 2 goes briefly over previous works on distributed SVMs in the fully supervised case. Then, Section 3 introduces the S^3VM model together with the approximation presented in Chapelle and Zien (2005). In Section 4, we first formulate the distributed training problem for S^3VMs, and subsequently we derive our two proposed solutions. Then, Section 5 details an extensive set of experimental results and, finally, Section 6 concludes the paper.

Notation

In the rest of the paper, vectors are denoted by boldface lowercase letters, e.g. $\mathbf{a}$, while matrices are denoted by boldface uppercase letters, e.g. $\mathbf{A}$. All vectors are assumed column vectors. Symbol a_i denotes the ith element of vector $\mathbf{a}$, and A_{ij} the (i, j) entry of the matrix $\mathbf{A}$. The operator $\|\cdot\|_2$ is the standard L_2 norm on an Euclidean space. Finally, the notation $a[n]$ is used to denote dependence with respect to a time-instant n in an iterative procedure. Other notation is introduced in the text when appropriate.

2. Related works

We start by briefly reviewing some works on distributed SVM algorithms in the fully supervised case. Similar overviews can be found in Scardapane, Wang, and Panella (in press, Section 2.1) and Wang and Zhou (2012). Initial works in this field were sparked by realizing that the set of support vectors represents an efficient way of 'compressing' data to be sent among the neighbors. In practice, this is complicated by the fact that each agent has no principled way of knowing whether a specific example is a support vector of the full problem. Thus, in Navia-Vázquez et al. (2006) the real set of support vectors is approximated by a specific set chosen *a priori*, whose weights are updated based on a least-square procedure. On the contrary, Lu et al. (2008) solve the problem considering the real set of support vectors, with assured convergence in a finite number of steps. Both approaches, however, are hindered by the necessity of sending the examples throughout the network on a Hamiltonian cycle. The most efficient procedure up-to-date is presented in Forero et al. (2010), where the problem is recast as multiple convex subproblems at every node, and solved with the use of the alternating direction method of multipliers (ADMM), an efficient procedure for distributed optimization of convex cost functions. Indeed, ADMM is among the most widely