Neural Networks 101 (2018) 94–100

Contents lists available at ScienceDirect

Neural Networks

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

Distributed support vector machine in master–slave mode

Qingguo Chen, Feilong Cao *

Department of Applied Mathematics, College of Sciences, China Jiliang University, Hangzhou 310018, Zhejaing Province, PR China

ARTICLE INFO

Article history:
Received 21 October 2017
Received in revised form 31 January 2018
Accepted 6 February 2018
Available online 15 February 2018

Keywords:
Support vector machine (SVM)
Alternating direction method of multipliers (ADMM)
Distributed algorithm
Master–slave mode

ABSTRACT

It is well known that the support vector machine (SVM) is an effective learning algorithm. The alternating direction method of multipliers (ADMM) algorithm has emerged as a powerful technique for solving distributed optimisation models. This paper proposes a distributed SVM algorithm in a master–slave mode (MS-DSVM), which integrates a distributed SVM and ADMM acting in a master–slave configuration where the master node and slave nodes are connected, meaning the results can be broadcasted. The distributed SVM is regarded as a regularised optimisation problem and modelled as a series of convex optimisation sub-problems that are solved by ADMM. Additionally, the over-relaxation technique is utilised to accelerate the convergence rate of the proposed MS-DSVM. Our theoretical analysis demonstrates that the proposed MS-DSVM has linear convergence, meaning it possesses the fastest convergence rate among existing standard distributed ADMM algorithms. Numerical examples demonstrate that the convergence and accuracy of the proposed MS-DSVM are superior to those of existing methods under the ADMM framework.

1. Introduction

As an effective supervised learning algorithm, the support vector machine (SVM) (Vapnik, 1998) has been widely used in many research fields, such as medical imaging (Codella et al., 2015), bioinformatics (Bao, Hua, Yuan, & Huang, 2017; Huang & Du, 2008; Liu, Qian, Dai, & Zhang, 2013a; Zheng & Lu, 2011), speech processing (Han, Park, & Lee, 2016; Trabelsi & Ellouze, 2016), facial recognition (Li & Huang, 2008), handwriting recognition (Mustafa & Prof, 2015), and radar image recognition (Huang, 1999). In the real world, the amount of data requiring processing has exploded. Data are typically acquired in various distributed forms, which leads to the failure of the conventional SVM algorithm running on a single computer. Therefore, it is necessary to integrate the traditional SVM with various effective distributed algorithms.

In recent years, the alternating direction method of multipliers (ADMM) (Boyd, Parikh, Chu, Peleato, & Eckstein, 2011; Ouyang, He, Tran, & Gray, 2013) has become one of the most widely used approaches to solving various distributed optimisation problems. ADMM is an algorithm that combines the decomposability of dual ascent with the strong convergence properties of the multipliers method. Recent work (Shi, Ling, Yuan, Wu, & Yin, 2014) has shown that the distributed ADMM has a linear convergence rate under the assumption that local objective functions are strongly convex and have Lipschitz continuous gradients. Deng and Yin (2016) and Iutzeler, Bianchi, Ciblat, and Hachem (2016) observed the same convergence rates using different assumptions and provided some equivalence conditions. In Goldstein, Donoghue, Setzer, and Baraniuk (2014), it was proved that the dual objective value of a modified ADMM converges at $O(1/k^2)$ under the assumptions that two sub-problems are solved exactly and both objective functions are strongly convex. Simone et al. (Scardapane, Wang, & Panella, 2016; Scardapane, Wang, Panella, & Uncini, 2015) addressed distributed learning for random vector functional-link networks (RVFL) and echo state networks via ADMM.

Generally, topologies for distributed ADMM can be divided into the following two categories according to the types of network connections:

- Peer-to-peer mode: Every node is connected to at least one other node, and nodes calculate local data and exchange information with their neighbours (Iutzeler, Bianchi, Ciblat, & Hachem, 2013; Mokhtari, Shi, Ling, & Ribeiro, 2016; Wei & Ozdaglar, 2012; Xi & Khan, 2015; Xi, Wu, & Khan, 2017).
- Master–slave mode: A unique master node is connected to slave nodes and collects results from the slave nodes (Boyd et al., 2011).

This paper focuses on distributed SVM via ADMM in a master–slave mode.

The distributed SVM has attracted significant research interest because of its high efficiency. Graf, Cosatto, Bottou, Durdanovic, and Vapnik (2004) developed a filtering process that can be parallelised efficiently to eliminate non-support vectors early in optimisation. They utilised SVMs as filters. There have also been similar

* Corresponding author.
E-mail address: icteam@163.com (F. Cao).

https://doi.org/10.1016/j.neunet.2018.02.006
0893-6080/

works on parallel designs for centralised SVMs when training sets are prohibitively large (Bordes, Ertekin, Weston, & Bottou, 2005; Chang et al., 2007; Do & Poulet, 2006). However, the convergence of these parallel designs is typically not guaranteed for any given partitioning of a dataset (Bordes et al., 2005; Graf et al., 2004). Another class of distributed SVMs relies on local support vectors that are broadcasted to neighbours to obtain a discriminant function (Flouri, Lozano, & Tsakalides, 2006, 2008; Lu, Roychowdhury, & Vandenberghe, 2008). However, these schemes cannot ensure adequate algorithm performance. Scardapane, Fierimonte, Di, Panella, and Uncini (2016) employed a distributed gradient descent algorithm and recently developed framework for in-network non-convex optimisation (NEXT) to implement a distributed semi-supervised SVM. One of the most noteworthy research directions for distributed SVMs is to integrate various big data algorithms, such as MapReduce (You, Yu, Zhu, Li, & Wen, 2014).

Although there have been many studies on distributed SVM, few have focused on how to integrate distributed SVM and ADMM. Forero, Cano, and Giannakis (2010) first proposed a method to train an SVM via distributed ADMM in peer-to-peer mode by decomposing the original problem as a series of sub-problems. They demonstrated that a distributed SVM is equivalent to a traditional non-distributed SVM. However, they did not analyse the convergence of the distributed SVM algorithm. Although the concept of distributed SVM via ADMM in the master–slave mode was mentioned in Boyd et al. (2011), no details were presented.

This paper addresses the problem of distributed SVM in the master–slave mode (MS-DSVM), where all slave nodes are connected to only one master node. Local data are calculated and the results are broadcasted to the master node. Because there is no communication between slave nodes in MS-DSVM, the distributed SVM can be regarded as a distributed convex optimisation problem, where the original problem is broken down into several sub-problems via ADMM. However, solving these sub-problems is still difficult. In order to accelerate the convergence rate, we solve them as dual problems. Furthermore, we demonstrate that the proposed MS-DSVM has linear convergence, which is the fastest convergence among existing ADMM algorithms.

The remainder of this paper is organised as follows. Section 2 introduces the traditional SVM, standard ADMM, and distributed ADMM frameworks. Section 3 describes the proposed MS-DSVM algorithm, presents its convergence analysis, and proposes the relaxed MS-DSVM. Section 4 presents numerical results. Section 5 concludes this paper and presents some final remarks.

2. Relevant background

2.1. Support vector machine

Here, we briefly introduce the SVM. A more detailed description can be found in Liu, Qian, Dai, and Zhang (2013b). Considering the binary classification of samples $\{(X_i, Y_i)\}_{i=1}^n$, where $X_i \in \mathbf{R}^{1\times p}$ are training data and $Y_i \in \{-1, 1\}$ are the corresponding labels, the essence of the SVM is to find the optimal hyperplane that separates the two classes of data points with the largest margin:

$$\begin{aligned} \min_{w,b} \quad & \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{n}\xi_i \\ \text{s.t.} \quad & Y_i(X_i w + b) \geq 1 - \xi_i \\ & \xi_i \geq 0,\ i = 0, 1, 2, \ldots, n. \end{aligned} \tag{1}$$

Here, ξ_i are slack variables and C is a tuneable positive scalar. It can be equivalently converted into hinge loss with an ℓ_2 norm penalty format:

$$\min_{w,b} \quad \sum_{i=1}^{n}(1 - Y_i(X_i w + b))_+ + \frac{\lambda'}{2}\|w\|^2, \tag{2}$$

where the loss function $(1 - \cdot)_+ = \max(1 - \cdot, 0)$ is called hinge loss and λ' is a positive regularisation parameter corresponding to C in problem (1), which controls the balance between the loss and penalty.

2.2. Alternating direction method of multipliers

Typically, the ADMM algorithm addresses the following optimisation problem:

$$\begin{aligned} &\min\ f(x) + g(z) \\ &\text{s.t.}\ Ax + Bz = c \end{aligned} \tag{3}$$

with $x \in \mathbf{R}^n$ and $z \in \mathbf{R}^m$, where $A \in \mathbf{R}^{p\times n}$, $B \in \mathbf{R}^{p\times m}$, $c \in \mathbf{R}^p$, and $f(x)$ and $g(z)$ are convex functions. ADMM utilises the following iterative solutions:

$$\begin{aligned} x^{k+1} &= \arg\min_x\ f(x) + \frac{\rho}{2}\|Ax + Bz^k - c + u^k\|_2^2, \\ z^{k+1} &= \arg\min_z\ g(z) + \frac{\rho}{2}\|Ax^{k+1} + Bz - c + u^k\|_2^2, \\ u^{k+1} &= u^k + Ax^{k+1} + Bz^{k+1} - c. \end{aligned} \tag{4}$$

This algorithm is effective when x- and z-minimisations can be calculated effectively, for example, when they have closed-form expressions. The advantage of ADMM is that there is only one parameter ρ. Furthermore, the algorithm has been proved to converge for all values of the parameter under certain mild conditions.

2.3. Distributed ADMM for consensus problems

According to Boyd et al. (2011), research on consensus problems has a long history. Recently, there have been more works regarding survey, and several applications on signal processing and wireless communications, such as Mateos, Bazerque, and Giannakis (2010), Qin, Ma, Shi, and Wang (2016), Schizas, Ribeiro, and Giannakis (2008) and Zhu, Cano, and Giannakis (2010).

If the optimisation objective can be split into N parts:

$$f(x) = \sum_{i=1}^{N} f_i(x), \tag{5}$$

where $x \in \mathbf{R}^n$ is a global variable and $f_i(x)$ are convex functions, then problem (5) can be equivalently reformulated for the local variables $x_i \in \mathbf{R}^{n'}$ and a common global variable z:

$$\begin{aligned} \min \quad & \sum_{i=1}^{N} f_i(x_i) \\ \text{s.t.} \quad & x_i - z = 0, \quad i = 1, 2, \ldots, N. \end{aligned} \tag{6}$$

This is called the global consensus problem because the constraints guarantee that all local variables will be equal. The distributed version of the algorithm is as follows:

$$\begin{aligned} x_i^{k+1} &= \arg\min_{x_i}\left(f_i(x_i) + y_i^{kT}(x_i - z^k) + \frac{\rho}{2}\|x_i - z^k\|_2^2\right), \\ z^{k+1} &= \frac{1}{N}\sum_{i=1}^{N}\left(x_i^{k+1} + \frac{1}{\rho}y_i^k\right), \\ y_i^{k+1} &= y_i^k + \rho\left(x_i^{k+1} - z^{k+1}\right). \end{aligned} \tag{7}$$

Each slave node only solves for its local variable x_i and corresponding dual variable y_i, which can be accomplished independently for each $i = 1, \ldots, N$. Then, the master node collects all the x_is from the slave nodes to update the global variable z. Next, the updated z is broadcast to all the slave nodes for updating the dual variable y_i and local variable x_i in the next iteration. This process is repeated until all local variables agree with the global variables (i.e., local variables and global variable are equal).