Neurocomputing 174 (2016) 187–193

Contents lists available at ScienceDirect

Neurocomputing

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

An efficient active set method for optimization extreme learning machines

Ming-hua Zhao *, Xiao-feng Ding, Zheng-hao Shi, Quan-zhu Yao, Yong-qin Yuan, Rui-yang Mo

School of Computer Science and Engineering, Xi'an University of Technology, Xi'an 710048, China

ARTICLE INFO

Article history:
Received 15 August 2014
Received in revised form
3 December 2014
Accepted 5 January 2015
Available online 3 September 2015

Keywords:
Optimization extreme learning machines
Quadratic programming
Active set
Piecewise projected gradient

ABSTRACT

In this paper an efficient active set algorithm is presented for fast training of Optimization Extreme Learning Machines (OELMs). This algorithm suggests the use of an efficient identification technique of active set and the value reassignment technique for quadratic programming problem. With these strategies, this algorithm is able to drop many constraints from the active set at each iteration, and it can converge to the optimal solution with less iterations. The global convergence properties of the algorithm as well as its theoretical properties are analyzed. The effectiveness of the algorithm is demonstrated via benchmark datasets from many sources. Experiment results indicate that the quadratic programming problem which keeps the number of constraints in the active set as small as possible is computationally most efficient.

1. Introduction

Optimization Extreme Learning Machines (OELMs) are a recently proposed regularized machine learning technique and their goal is mapping an original input space into Extreme Learning Machine (ELM) feature space in which the training samples can be separable [1]. It has been shown that OELMs are less sensitive to user specified parameters and offer excellent performance in many applications. The training stage for OELMs involves at its core a dense convex quadratic programming (QP) problem. Tackling this optimization problem is computationally expensive, primarily due to the dense Hessian matrix. Active set method is preferable when QP of the constraint matrix of the QP problem is dense [2]. The main purpose of this paper is to exploit the structure of QP of OELMs and design a fast solution based on an active set method.

First, there is the problem of binary classification. The sample set consists of N examples $\{(\mathbf{x}_i, t_i)\}_{i=1}^{N}$, where $\mathbf{x}_i \in \mathbb{R}^m$ and $t_i \in \{-1, +1\}$. An Optimization Extreme Learning Machine (OELM) algorithm with penalization of the training errors consists of solving the following equation:

$$\begin{aligned} &\text{minimize} \quad \frac{1}{2}\|\boldsymbol{\beta}\|^2 + C\sum_{i=1}^{N}\xi_i \\ &\text{subject to} \quad t_i\boldsymbol{\beta}\cdot h(\mathbf{x}_i) \geq 1-\xi_i \\ &\quad \xi_i \geq 0, \quad i = 1, \ldots, N \end{aligned} \tag{1}$$

where $\boldsymbol{\beta}$ is the normal vector of the weight theory, $h(\mathbf{x}_i)$ actually maps the training examples from the m-dimensional input space to the ELM feature space, and C is a user-specified value.

The standard way to train an OELM is to introduce Lagrange multipliers and optimize them by solving the following equation:

$$\begin{aligned} &\text{minimize} \quad f(\boldsymbol{\alpha}) = \frac{1}{2}\boldsymbol{\alpha}^T K\boldsymbol{\alpha} - \boldsymbol{e}^T\boldsymbol{\alpha} \\ &\text{subject to} \quad 0 \leq \alpha_i \leq C, \quad i = 1, \ldots, N \end{aligned} \tag{2}$$

where $\boldsymbol{\alpha} \in \mathbb{R}^N$ represents a vector of non-negative Lagrange multipliers corresponding to the inequality constraints, $\boldsymbol{e}$ is the vector of all 1's of length N, $K = K_{ELM}\boldsymbol{t}^T\boldsymbol{t} \in \mathbb{R}^{N\times N}$ is positive semi-definite kernel matrix, and $\boldsymbol{t}$ is N vector with components t_i and .

Eq. (2) is similar to Support Vector Machine's (SVM) dual problem [3]. SVM dual problem needs to satisfy both equality constraints and inequality constraints, while OELM problem only needs to satisfy the inequality constraints. Obviously SVM tends to find a sub-optimal solution compared to OELM.

The essential part of an active set method is a procedure for determining which of the r inequality constraints will be active (that is, treated as equalities). In the active set method, variables at their boundary correspond to the active constraints. Several active set methods have been proposed for solving QP of SVM and OELM [2,4–6]. Although these methods provide an efficient implementation, two disadvantages still exist:

(1) Methods of this type are efficient for training set with relatively small examples, but are typically unsuitable for large training

* Corresponding author.

http://dx.doi.org/10.1016/j.neucom.2015.01.092
0925-2312/

set with abundant examples. The main reason is the set of active constraints is changed by the addition or deletion of no more than one constraint at each iteration. So if, for example, 100 constraints are binding at the point of convergence and an interior starting point is selected, then the method will require at least 100 iterations (and possible $O(3n/2)$) to converge.

(2) Methods of this type update the active set until the correct set is confirmed at the optimum. Although only a subset of the constraints joins the computations at each iteration, the same constraint may be added and dropped repeatedly. Such method may usually waste time minimizing the function with "wrong" constraint.

In order to overcome the above limitations, two directions we are explored in this paper:

(1) piecewise project gradient technique is employed to add several constraints to the active set at each iteration. Steepest descent direction can be obtained by adjusting the appropriate stepsize;
(2) value reassignment method is used to reduce the phenomenon of repeated iterations. At the current iteration, although a constraint has been removed from the active set, the value of a corresponding variable has not been changed yet. It is quite possible that this constraint is added to the active set in iterations again.

The organization of this paper is as follows. The rest of this section introduces notation. Section 2 describes a basic active set algorithm. Section 3 gives a detailed statement of the researchers' algorithm. In Section 4, the theoretical properties of the researchers' algorithm are then investigate. Performance evaluation is presented in Section 5. Discussions and conclusions are given in Section 6.

This section concludes by providing a list of the notation employed. Throughout the paper, $\|\cdot\|$ indicates the Euclidean vector norm. If K is an $N\times N$ matrix with elements K_{ij}, $i,j=1,\ldots N$, and I is an index set such that $I\subseteq\{1,\ldots,N\}$, it is denotes by K_{II} the $|I|\times|I|$ submatrix of K consisting of elements K_{ij}, $i\in I$, $j\in I$. If $\boldsymbol{\alpha}$ is an N vector, it is denotes by $\boldsymbol{\alpha}_I$ the subvector with components α_i, $i\in I$. The gradient $\nabla f(\boldsymbol{\alpha})$ is a row vector while $g(\boldsymbol{\alpha})=\nabla f(\boldsymbol{\alpha})^T$ is a column vector and T denotes transpose. A superscript (k in general) indicates iteration numbers.

2. The Basic active set (BAS) algorithm

Following the usual terminology in constrained optimization problems, vector $\bar{\boldsymbol{\alpha}}\in\mathbb{R}^N$ is a stationary point of Eq. (2) if vector $\bar{\boldsymbol{\alpha}}$ exists, then it solves the Karush–Kuhn–Tucker (KKT) system [7]

$$\begin{aligned} &g(\bar{\boldsymbol{\alpha}})_i=0, \quad \forall\, i\in S\\ &g(\bar{\boldsymbol{\alpha}})_i\geq 0, \quad \forall\, i\in L\\ &g(\bar{\boldsymbol{\alpha}})_i\leq 0, \quad \forall\, i\in U \end{aligned} \tag{3}$$

where $g(\bar{\boldsymbol{\alpha}})_i$ is the i-th component of the gradient vector of f at $\bar{\boldsymbol{\alpha}}$.

Here, we first define the feasible region of Eq. (2), that is, the set of all $\boldsymbol{\alpha}$ to search for the optimal solution. Given a point $\boldsymbol{\alpha}$ in the feasible region, a constraint $0\leq\alpha_i\leq C$ is call active at $\boldsymbol{\alpha}$ if $\alpha_i=0$ or $\alpha_i=C$. $L=\{i|\alpha_i=0\}$ and $U=\{i|\alpha_i=C\}$ are defined as the indices set of the active constraints, and the set $L\cup U$ of bound variables are called the active set at the current point. Also, $S=\{i|0<\alpha_i<C\}$ is defines as the free set, and the set S of variables are corresponding to free variables.

The vector of $\boldsymbol{\alpha}$ variables whose indices belong to set L will be denoted by $\boldsymbol{\alpha}_L$, and other $\boldsymbol{\alpha}$ variables will be denoted by $\boldsymbol{\alpha}_U$ and $\boldsymbol{\alpha}_S$ respectively. Corresponding to the choice of index set L, U and S, the matrix K is partitioned and rearranged as follows:

$$K=\begin{bmatrix} K_{LL} & K_{LS} & K_{LU}\\ K_{SL} & K_{SS} & K_{SU}\\ K_{UL} & K_{US} & K_{UU}\end{bmatrix}$$

As vector $\boldsymbol{\alpha}_L$ makes no contribution to compute the value of objective function f, solving Eq. (2) is equivalent to solving the following equation:

$$\begin{aligned} &\text{minimize} \quad f(\boldsymbol{\alpha}_S)=\frac{1}{2}\boldsymbol{\alpha}_S^T K_{SS}\boldsymbol{\alpha}_S+\boldsymbol{\alpha}_U^T K_{US}\boldsymbol{\alpha}_S-\boldsymbol{e}^T\boldsymbol{\alpha}_S\\ &\text{subject to} \quad 0\leq\alpha_i\leq C, \quad i\in S \end{aligned} \tag{4}$$

With notation S defined above, a box constrained Eq. (4) is reduced eventually to an unconstrained minimization problem

$$\text{minimize} \quad f(\boldsymbol{\alpha}_S)=\frac{1}{2}\boldsymbol{\alpha}_S^T K_{SS}\boldsymbol{\alpha}_S+\boldsymbol{\alpha}_U^T K_{US}\boldsymbol{\alpha}_S-\boldsymbol{e}^T\boldsymbol{\alpha}_S \tag{5}$$

The essence of active set algorithms is to handle an unconstrained optimization problem with a fixed active constraint set. A typical iteration of these methods can be described as follows. Given a feasible point $\boldsymbol{\alpha}_S^{(k)}$ (i.e. a point such that $0<\boldsymbol{\alpha}_S^{(k)}<C, k=0$):

Algorithm BAS

Step 1: Identify the set $U^{(k)}=\{i|\alpha_i^{(k)}=C\}$ of active constraints;
Step 2: Compute a descent direction $d^{(k)}=\boldsymbol{\alpha}_S^*-\boldsymbol{\alpha}_S^{(k)}$, where $\boldsymbol{\alpha}_S^*$ is the solution of Eq. (4);
Step 3: Drop one constraint if some dropping rule allows it. If it is the case, compute a new direction $d^{(k)}$ as above;
Step 4: Perform a line search along $d^{(k)}$ with maximal stepsize $\mu_1^{(k)}$ not breaking constraints $0\leq\boldsymbol{\alpha}_i^{(k)}\leq C$, $i\in S$ in order to get a new feasible point $\boldsymbol{\alpha}_S^{(k+1)}=\boldsymbol{\alpha}_S^{(k)}+\mu_1^{(k)}d^{(k)}$.

Different methods differ by the choice of a descent direction and the relaxing rule. In [4,6], active set algorithms follow the search direction $d^{(k)}$ until a constraint is broken and remove the corresponding variable from the free set S. The largest possible stepsize $\mu_1^{(k)}$ is chosen that do not lead to breaking of the box constraints of problem (2). It is obvious that all components of $\boldsymbol{\alpha}_S^{(k)}$ are set the same stepsize $\mu_1^{(k)}$. However, only one constraint is allowed to drop from the active set at each iteration, may lead more iterations to converge.

3. Proposed active set algorithm

3.1. The piecewise projected gradient

In this section, a piecewise projected gradient method is described whereby more than one constraint could be dropped from the active set between iterations. A piecewise projected gradient method has been discussed for the mathematical optimization problem in theory many times [8–10]. Theoretically, by a series of projections, the active constraints can be identified in a finite number of iterations.

To formalize the piecewise projected method, the notation of a projection operator Π is introduced and defined componentwise by

$$\Pi(\boldsymbol{\alpha})_i=\begin{cases} 0 & \text{if } \alpha_i\leq 0\\ \alpha_i & \text{if } 0<\alpha_i<C\\ C & \text{if } \alpha_i\geq C\end{cases} \tag{6}$$

With this notation, $\boldsymbol{\alpha}$ can be updated by $\boldsymbol{\alpha}:=\Pi(\boldsymbol{\alpha}+\mu d)$, where μ is the stepsize, and d is the search direction.