European Journal of Operational Research 224 (2013) 560–565

Contents lists available at SciVerse ScienceDirect

European Journal of Operational Research

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

Decision Support

An incremental least squares algorithm for large scale linear classification

A. Cassioli [a,c,*], A. Chiavaioli [a], C. Manes [b], M. Sciandrone [a]

[a] Università degli Studi di Firenze, Dipartimento di Sistemi e Informatica, Via di S. Marta 3, 50139 Firenze, Italy
[b] Università degli Studi dell'Aquila, Dip. di Ingegneria e Scienze dell'Informazione e Matematica, Via Vetoio, 67100 Coppito (AQ), Italy
[c] Radiation Oncology - Massachusetts General Hospital and Harvard Medical School, 30 Fruit Street, Boston, MA 02114, USA

ARTICLE INFO

Article history:
Received 5 December 2011
Accepted 2 September 2012
Available online 19 September 2012

Keywords:
Large scale optimization
Machine learning
Linear classification
Incremental algorithms

ABSTRACT

In this work we consider the problem of training a linear classifier by assuming that the number of data is huge (in particular, data may be larger than the memory capacity). We propose to adopt a linear least-squares formulation of the problem and an incremental recursive algorithm which requires to store a square matrix (whose dimension is equal to the number of features of the data). The algorithm (very simple to implement) converges to the solution using each training data once, so that it effectively handles possible memory issues and is a viable method for linear large scale classification and for real time applications, provided that the number of features of the data is not too large (say of the order of thousands). The extensive computational experiments show that the proposed algorithm is at least competitive with the state-of-the-art algorithms for large scale linear classification.

1. Introduction

Machine learning models are useful tools for many real life problems and their construction, based on a set of data called the training set, requires very often the solution of optimization problems (as deeply examined in [4,2,18], this latter issue represents the strong interaction of machine learning and mathematical programming).

In classification the training data are labeled, that is, each instance is associated to one of two or more classes. Starting with the training set, the machine learning task for classification is to construct a predictive model able to correctly predict as possible the class of new instances (generalization capability).

Support Vector Machines (SVMs) are widely used as a simple and efficient tool for linear and nonlinear classification (as well as for regression problems). The basic training principle of SVM, motivated by the statistical learning theory [23], is that the expected classification error for unseen test samples is minimized, so that SVM define good predictive models. The design of efficient and reliable optimization methods for SVM training is an active and dynamic research area as proved by many papers devoted to the topic and published in the last decade (see, e.g., [1,14,13,22,16,17,7,19,12]).

In this work we focus on linear classification, which is a useful tool in many real applications and represents a very active topic (see, e.g., [9,24,20]). The aim is to develop fast optimization algorithms for training linear classifiers on large scale data (in particular, data may be larger than the memory capacity).

Given a finite set (the training set) of data (not necessarily linearly separable)

$$TS = \{(\mathbf{u}^i, d^i) : \mathbf{u}^i \in \mathbb{R}^n, \quad d^i \in \{-1, 1\}, \quad i = 1, \ldots, m\},$$

where the label d^i denotes the class of the vector $\mathbf{u}^i$, we consider the problem of training a linear machine defined by the decision function.

$$y(\mathbf{u}) = sgn(\mathbf{w}^T\mathbf{u} + \theta), \tag{1}$$

where $\mathbf{u} \in \mathbb{R}^n$ is the input vector, $\mathbf{w} \in \mathbb{R}^n$ is the vector of weights, $\theta \in \mathbb{R}$ is the threshold, $sgn : \mathbb{R} \to \{-1, 1\}$ is the sign function such that

$$sgn(t) = \begin{cases} 1 & t \geqslant 0 \\ -1 & t < 0 \end{cases}$$

We assume that the number m of training data is huge.

The standard SVM linear classifier is obtained by computing the solution of the problem

$$\begin{aligned} \min_{\mathbf{w}\in\mathbb{R}^n, \theta\in\mathbb{R}, \xi\in\mathbb{R}^m} \quad & \tfrac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{m}\xi^i \\ & d^i(\mathbf{w}^T\mathbf{u}^i + \theta) \geqslant 1 - \xi^i \quad i = 1, \ldots, m \\ & \xi^i \geqslant 0 \quad i = 1, \ldots, m, \end{aligned} \tag{2}$$

where $\|\cdot\|$ is the Euclidean norm. The term $\sum_{i=1}^{m}\xi^i$ is an upper bound on the training error. The parameter $C > 0$ trades off margin size (related to the generalization capability) and training error, and is usually determined by standard cross-validation tuning procedures.

* Corresponding author. Tel.: +39 055 4796 464.
E-mail addresses: cassioli@dsi.unifi.it, cassioliandre@gmail.com (A. Cassioli), alessandrochiavaioli@gmail.com (A. Chiavaioli), costanzo.manes@univaq.it (C. Manes), sciandro@dsi.unifi.it (M. Sciandrone).

0377-2217/$ - see front matter © 2012 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.ejor.2012.09.004

Robustness with respect to the parameter C is a very desirable property that, as reported in Section 4, the classifier proposed in this work seems to possess.

Problem (2), usually referred as primal problem, is equivalent to a non-smooth unconstrained optimization problem of the form

$$\min_{\mathbf{w}\in\mathbb{R}^n,\theta\in\mathbb{R}} \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{m}\max\{0, 1 - d^i(\mathbf{w}^T\mathbf{u}^i + \theta)\} \tag{3}$$

Using Lagrangian duality, we have that the dual problem of (2) is a convex quadratic problem (whose dimension is equal to the number m of training data) with one linear equality constraint and box constraints. There exist both (primal) algorithms for solving Problem (3) and (dual) algorithms for solving its dual.

A more general primal form is

$$\min_{\mathbf{w}\in\mathbb{R}^l,\theta\in\mathbb{R}} \rho(\mathbf{w},\theta) + C\sum_{i=1}^{m}\Xi^i(\mathbf{w},\theta,\mathbf{u}^i,d^i), \tag{4}$$

where $\rho(\mathbf{w},\theta)$ is the regularization term and $\Xi^i(\mathbf{w}, \mathbf{u}^i, y^i, \theta)$ is the loss function associated with the observation $(\mathbf{u}^i, d^i)$. There exist algorithms (see, e.g., [6,15,20]) which refer to problem (4) and others (see, e.g., [10,25]) for solving the corresponding dual. For what concerns the loss function $\Xi^i(\mathbf{w}, \mathbf{x}_i, y_i, b)$, two common forms are

$$\Xi^i(\mathbf{w},\theta,\mathbf{u}^i,d^i) = \max\{0, 1 - d^i(\mathbf{w}^T\mathbf{u}^i + \theta)\}$$
$$\Xi^i(\mathbf{w},\theta,\mathbf{u}^i,d^i) = \max\{0, 1 - d^i(\mathbf{w}^T\mathbf{u}^i + \theta)\}^2.$$

For the regularization term $\rho(\cdot)$ standard choices are the following

$$\rho(\mathbf{w},b) = \|\mathbf{w}\|_2^2$$
$$\rho(\mathbf{w},b) = \|\mathbf{w}\|_1.$$

We are assuming that the number m of training data is so huge that either the whole training set cannot be stored in the computer memory or, in the case that memory is enough, loading data to memory can be too expensive. Therefore, we focus on incremental methods which use some observations at a time rather than using the whole training set. We address the reader to [25] for the state of art concerning linear large-scale classification, we only cite two efficient optimization methods for large scale linear classification proposed in [11,5], respectively. The former is a batch method using the whole training set, the latter is based on an incremental strategy. Both methods, as the one proposed in this work are able to solve problems with a huge number of observations having a relatively small number of features.

In this work we introduce a linear classifier obtained by solving a linear least squares problem corresponding to a primal formulation of the form (4), and we adopt an incremental recursive least-squares algorithm as training algorithm. The results of the extensive computational experiments reported in the paper show that the presented methodology (very simple to implement) may be an effective tool for large scale linear classification whenever the number n of features is not huge (say n of the order of thousands).

The paper is organized as follows. In Section 2 we recall the structure of a recursive linear least-squares algorithm. In Section 3 we present the linear classifier and the training algorithm. In Section 4 we report the results of extensive computational experiments performed on large dimensional classification problems and we show the comparison with many other linear classifiers and training algorithms.

2. Recursive linear least-squares algorithm

Let us consider the linear least-squares problem

$$\min_{\mathbf{x}\in\mathbb{R}^n}\|A_k\mathbf{x} - \mathbf{b}_k\|^2, \tag{5}$$

where $A_k \in \mathbb{R}^{k\times n}$ with $rank\ (A_k) = n$, $\mathbf{b}_k \in \mathbb{R}^k$. The solution $\mathbf{x}_k$ of (5) satisfies the normal equations

$$A_k^TA_k\mathbf{x} = A_k^T\mathbf{b}_k.$$

Assume that a new equation

$$\mathbf{a}_{k+1}^T\mathbf{x} = b_{k+1}$$

is added, so that the new problem to be solved is

$$\min_{\mathbf{x}\in\mathbb{R}^n}\|A_{k+1}\mathbf{x} - \mathbf{b}_{k+1}\|^2, \tag{6}$$

where

$$A_{k+1} = \begin{bmatrix} A_k \\ \mathbf{a}_{k+1}^T \end{bmatrix} \qquad \mathbf{b}_{k+1} = \begin{bmatrix} \mathbf{b}_k \\ b_{k+1} \end{bmatrix}.$$

Letting $H_i^{-1} = \left(A_i^TA_i\right)^{-1}$ and denoting by $\mathbf{x}_{k+1}$ the solution of (6), we have the recursive updating formulae (see, e.g., [3])

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \mathbf{v}_{k+1}(b_{k+1} - \mathbf{a}_{k+1}^T\mathbf{x}_k), \tag{7}$$

where

$$\mathbf{v}_{k+1} = H_{k+1}^{-1}\mathbf{a}_{k+1} \tag{8}$$

and

$$H_{k+1}^{-1} = H_k^{-1} - \frac{\mathbf{s}_k\mathbf{s}_k^T}{1 + \mathbf{a}_{k+1}^T\mathbf{s}_k}, \qquad \mathbf{s}_k = H_k^{-1}\mathbf{a}_{k+1}. \tag{9}$$

Eqs. (7)–(9) allow us to define an incremental algorithm for a linear least-squares problem

$$\min_{\mathbf{x}\in\mathbb{R}^n}\|A\mathbf{x} - \mathbf{b}\|^2,$$

where $A \in \mathbb{R}^{m\times n}$ and $\mathbf{b} \in \mathbb{R}^m$ with $m > n$ and $rank(A) = n$. The algorithm requires to store a positive definite $n \times n$ matrix, and at each iteration updates the estimate of the solution using a single observation. Once initialized H_0^{-1}, the algorithm requires exactly $m - n$ iterations for determining the solution of the problem. In Section 3 we will present the training problem of a linear classifier formulated as a linear least-squares problem and the incremental training algorithm, which can be conveniently adopted for solving large-scale problems.

3. The linear classifier and the training algorithm

Consider the problem of designing a linear classifier using a given training set

$$TS = \{(\mathbf{u}^i, d^i) : \mathbf{u}^i \in \mathbb{R}^n, d^i \in \{-1, 1\}, i = 1,\ldots,m\}.$$

As introduced in Section 1, a standard linear SVM can be trained by solving the unconstrained non-smooth optimization problem (3). Note that the classification constraints are imposed by means of linear inequalities, and transferred by a penalty term in the objective function.

We consider here the regularized least-squares formulation (see, e.g., [21] as introduction to least-squares Support Vector Machines). In particular, the classification constraints, whose violation must be penalized, are defined by linear equalities, that is,

$$\mathbf{w}^T\mathbf{u}^i + \theta = d^i \qquad i = 1,\ldots,m,$$

so that, the linear least-squares formulation corresponding to (4) becomes

$$\min_{\mathbf{w}\in\mathbb{R}^n,\theta\in\mathbb{R}} \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{m}(\mathbf{w}^T\mathbf{u}^i + \theta - d^i)^2. \tag{10}$$