Optik 126 (2015) 2405–2410

Contents lists available at ScienceDirect

Optik

journal homepage: www.elsevier.de/ijleo

A method to improve support vector machine based on distance to hyperplane

Shu-yin Xia[a,*], Zhong-yang Xiong[a], Yue-guo Luo[b], Li-mei Dong[c]

[a] College of Computer Science, Chongqing University, Chongqing 400044, China
[b] Network Information Center, Yangtze Normal University, Chongqing 408100, China
[c] Institute of Electrical Engineering and Information, Sichuan University, Chengdu 400015, China

ARTICLE INFO

Article history:
Received 22 April 2014
Accepted 2 June 2015

Keywords:
Support vector machine
Hyperplane
Hilbert space

ABSTRACT

As SVM (support vector machine) has good generalizability, it has been successfully implemented in a variety of applications. Yet in the process of resolving its mathematical model, SVM needs to compute the kernel matrix. The dimension of the kernel matrix is equal to the number of records in the training set, so computing it is very costly in terms of memory. Although some improved algorithms have been proposed to decrease the need for memory, most of these algorithms need iterative computations that cost too much time. Since the existing SVM models fail to perform well regarding both runtime and space needed, we propose a new method to decrease the memory consumption without the need for any iteration. In the method, an effective measure in kernel space is proposed to extract a subset of the database that includes the support vectors. In this way, the number of samples participating in the training process decreases, resulting in an accelerated training process which has a time complexity of only O(*nlogn*). Another advantage of this method is that it can be used in conjunction with other SVM methods. The experiments demonstrate effectiveness and efficiency of SVM algorithms that are enhanced with the proposed method.

1. Introduction

The support vector machine (SVM) is a relatively new tool for classification and regression [1,2], but it has already been proven useful in many applications. The original model of SVM is a convex optimization conducted by minimizing the structural risk instead of the empirical risk that previous methods were based on, thus enabling it to obtain global solutions and overcome the problem of overfitting. That means it has good generalization performance. But the original model has a time complexity of $O(n^3)$ which is too high for most practical purposes. Moreover, it needs to compute the kernel matrix whose dimension is equal to the number of training samples, which requires a very large amount of memory. Therefore, some improved algorithms have been proposed over the years.

Suykens and Vandewalle proposed Least squares support vector machine (LS-SVM) [3]. LS-SVM uses squared errors instead of inequality constraint that conventional SVM used. As a result, the quadratic problem is converted into a linear system that accelerates the runtime of SVM. Especially in approximately linear and small-scale datasets, SVM and LS-SVM demonstrate better performances than various other method including statistical algorithms, decision tree based algorithms and instance based learning methods. However, LS-SVM still needs to compute an $(n+1)$th order square matrix in which n is the number of training samples. To tackle large scale problems, an iterative training algorithm based on the conjugate gradient (CG) was proposed by Suykens [4,5]. Chua proposed an algorithm based on Sherman–Morrison–Woodbury (SMW) matrix [6]. A reduced set of linear equations for function estimation was proposed by Chu [7]. By selecting a pair of samples violating the KKT optimality conditions, the sequence of minimal optimization (SMO) algorithm was proposed in [8]. Furthermore, Second order SMO algorithm was proposed in [9]. In this method, second order approximation information of the dual function is used to search the second index so that the number of iterations can be decreased greatly. The second order rules have been derived based on the dual gain [10]. These iterative algorithms enable solving large scale classification and regression problems. But all of them are very costly in terms of time since they need to run for much iteration.

The geometric properties of SVM can also be used to extract subsets including support vectors. In [11], the maximum-margin hyperplane is proposed to find the nearest neighbors in the convex hull of each class, but it is valid only in separable cases. A method based on the measurement of neighborhood entropy is proposed

* Corresponding author. Tel.: +86 013883929561.
E-mail address: 380835019@qq.com (S.-y. Xia).

http://dx.doi.org/10.1016/j.ijleo.2015.06.010
0030-4026/

in [12]. In [13], fuzzy C-means clustering is used to detect samples including support vectors around the decision curve. Since the support vectors are located in local extremes or near extremes on the border of the convex hull, the concept of convex hull is proposed to reduce the training points in [14,15]. Yet in high dimensional data, no methods can obtain the precise border, which means that some support vectors cannot be detected.

In this paper, to improve the performance of SVM for binary classification problems, we first use the kernel function to map the training samples into linear space. In the linear space, the vertical hyperplane of the straight line consisting of centers of two types of training samples is considered as approximate decision hyperplane. Then the distance from training samples to the vertical hyperplane is computed and is used to produce the subset including support vectors. The smaller the distance is, the more the possibility that the sample will be a support vector. In an average dataset, the percentage of support vectors in the whole dataset is usually lower than 30%. Thus, the number of samples participating in the training process can be decreased greatly. In this way, we produce a new algorithm to accelerate various SVM methods.

2. Support vector machine

Given a training set $\{x_i, y_i\}$, where $x_i \in R_n (1 \le i \le l, 1 \le j \le l)$ is the training sample data and $y_i \in \{-1, +1\}$ is the corresponding label for classification. The mapping function is introduced to convert input space R_n to Hilbert space:

$$\phi : \chi \subset R^n \to H$$
$$x \qquad \to \phi(x)$$

Training samples can be mapped into Hilbert space by this mapping function. The original problem in a linear dataset can be denoted by the following:

$$\begin{aligned} &\min \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{l}\xi_i, \\ &s.t \quad y_i((w \cdot x_i) + b) \ge 1 - \xi_i, \\ &\quad \xi_i \ge 0, \quad i = 1, \ldots, l \end{aligned} \tag{1}$$

where ξ_i are slack variables to relax the constraints, C is the penalty coefficient to avoid overfitting, $\sum_{i=1}^{l}\xi_i$ is the penalty term accounting for the presence of outliers. Suppose the inner function in the Hilbert space is $K(.,.)$, then problem (1) is changed into:

$$\begin{aligned} &\min \quad \frac{1}{2}\sum_{j=1}^{l}\sum_{i=1}^{l}\alpha_i\alpha_j y_i y_j K(x_i, x_j) - \sum_{1}^{l}\alpha_i \\ &s.t \quad \sum_{i=1}^{l} y_i\alpha_i = 0, \\ &\quad 0 \le \alpha_i \le C, \quad 1 \le i \le l \end{aligned} \tag{2}$$

when C is large enough in problem (2), the upper bound of Lagrange coefficient α_i disappears. This transforms problem (2) into a separable problem, where the decision function can be expressed as: $f(x) = sgn\left(\sum_{i=1}^{l} y_i a_i^* k_\theta(x_i, x) + b^*\right)$, where $b^* = y_j - \sum_{i=1}^{l} y_i a_i^* k_\theta(x_i, x_j)$, $j \in \{j | 0 < a_j^* < C\}$, a^* is the best solution of the Lagrange coefficients. The support vectors are those samples whose Lagrange coefficient is greater than 0.

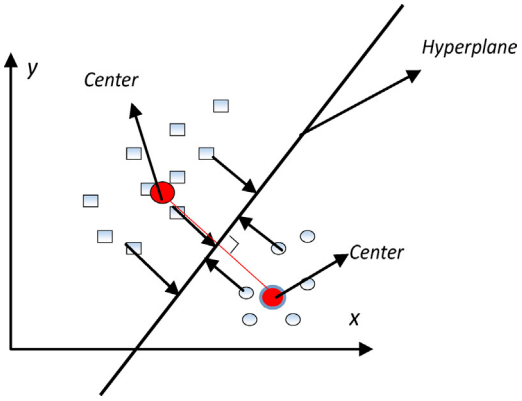

Fig. 1. A linear space mapped from original inseparable space.

3. Hyperplane based support vector machine

As described above in Section 1, various different methods have been developed to improve the performance of SVM. In this paper, the concept of distance to hyperplane is introduced to measure the probability of a sample data of being considered as a support vector.

As shown in Fig. 1, the mapped space obtained by kernel function is a linear space. The hyperplane of the line consisting of two centers of two kinds of mapped points in mapped space can be considered as the approximation of decision hyperplane. The points closer to the hyperplane are more likely to be support vectors. These points are used to replace the entire dataset for cross validation so that the process of cross validation can be accelerated. As the number of support vectors usually does not exceed 30% of the total samples in the entire dataset, the number of samples in the set including support vectors can be set to 30% of the total samples in the training set.

The database D consists of two kinds of points: $D+$ and $D-$; for point $x_i \in D+$, and $y_i \in D-$, the number of points in $D+$ and $D-$ are l_1 and l_2 respectively. x_i and y_i in dataset D are mapped into a new Hilbert space by kernel function φ, and the resulting samples are $\varphi(x_i)$ and $\varphi(y_i)$ respectively. The two centers of two types of points in the mapped space are:

$$\frac{1}{l_1}\sum_{i=1}^{l_1}\phi(x_i), \frac{1}{l_2}\sum_{i=1}^{l_2}\phi(y_i),$$

The hyperplane of the line consisting of the two centers can be expressed by:

$$\begin{aligned} &z - 0.5 * \left(\frac{1}{l_1}\sum_{i=1}^{l_1}\phi(x_i) + \frac{1}{l_2}\sum_{i=1}^{l_2}\phi(y_i)\right) * \left(\frac{1}{l_1}\sum_{i=1}^{l_1}\phi(x_i) - \frac{1}{l_2}\sum_{i=1}^{l_2}\phi(y_i)\right) = 0 \\ &\Rightarrow z\left(\frac{1}{l_1}\sum_{i=1}^{l_1}\phi(x_i) - \frac{1}{l_2}\sum_{i=1}^{l_2}\phi(y_i)\right) - 0.5 * \left(\frac{1}{l_1}\sum_{i=1}^{l_1}\phi(x_i) + \frac{1}{l_2}\sum_{i=1}^{l_2}\phi(y_i)\right) * \left(\frac{1}{l_1}\sum_{i=1}^{l_1}\phi(x_i) - \frac{1}{l_2}\sum_{i=1}^{l_2}\phi(y_i)\right) = 0 \\ &\Rightarrow z\left(\frac{1}{l_1}\sum_{i=1}^{l_1}\phi(x_i) - \frac{1}{l_2}\sum_{i=1}^{l_2}\phi(y_i)\right) = 0.5 * \left(\frac{1}{l_1}\sum_{i=1}^{l_1}\phi(x_i) + \frac{1}{l_2}\sum_{i=1}^{l_2}\phi(y_i)\right) * \left(\frac{1}{l_1}\sum_{i=1}^{l_1}\phi(x_i) - \frac{1}{l_2}\sum_{i=1}^{l_2}\phi(y_i)\right) \end{aligned} \tag{3}$$