Proceedings of the Fifth International Conference on Machine Learning and Cybernetics, Dalian, 13-16 August 2006

LINEAR PROGRAMMING APPROACH FOR THE INVERSE PROBLEM OF SUPPORT VECTOR MACHINES

QIANG HE[1], XUE-JUN SONG[2], GANG YANG[1]

[1] Faculty of Mathematics & Computer Science, Hebei University, Baoding, Hebei, China, 071002
[2] Faculty of Physics Science & Information Engineering, Hebei Normal University, Shijiazhuang, Hebei, China, 050016
E-MAIL: heq@mail.hbu.cn

Abstract:

It is well recognized that support vector machines (SVMs) would produce better classification performance in terms of generalization power. A SVM constructs an optimal separating hyper-plane through maximizing the margin between two classes in high-dimensional feature space. The inverse problem is how to split a given dataset into two clusters such that the margin between the two clusters attains the maximum. It is difficult to give an exact solution to this problem, so a genetic algorithm is designed to solve this problem. But the proposed genetic algorithm has large time complexity for the process of solving quadratic programs. In this paper, we replace the quadratic programming with a linear programming. The new algorithm can greatly decrease time complexity. The fast algorithm for acquiring the maximum margin can upgrade the applicability of the proposed genetic algorithm.

Keywords:

Support vector machines; linear programming; genetic algorithms; maximum margin

1. Introduction

Support vector machines (SVMs) are a classification technique of machine learning based on statistical learning theory [1, 2]. Considering a classification problem with two classes, SVMs are to construct an optimal hyper-plane that maximizes the margin between two classes. According to Vapnik statistical learning theory[1], the maximum of margin implies the extraordinary generalization capability and good performances of SVM classifiers. So far, SVMs have already been successfully applied to many real fields.

The investigation to the inverse problem of SVMs is motivated by designing a new decision tree generation procedure to improve the generalization capability of existing decision tree programs based on minimum entropy heuristic. Due to the relationship between the margin of SVMs and the generalization capability, the split with maximum margin may be considered as the new heuristic information for generating decision trees.

It is difficult to give an exact solution to this problem, so a genetic algorithm is designed to solve this problem. But the proposed genetic algorithm has large time complexity for the process of solving quadratic programs. In this paper, we replace the quadratic programming with a linear programming. Our proposed techniques in this paper can greatly decrease time complexity. The fast algorithm for acquiring the maximum margin can upgrade the applicability of the proposed genetic algorithm.

This paper has the following organization. Section 2 briefly reviews the basic concept of support vector machines. Section 3 introduces the inverse problem of SVMs and a genetic algorithm to solve this problem. In section 4, we introduce linear programming approach to the inverse problem of support vector machines. And the last section briefly concludes this paper.

2. Support Vector Machines

2.1. The basic problem of SVMs

Let $S=\{(x_1,y_1),(x_2,y_2),\cdots,(x_N,y_N)\}$ be a training set, where $x_i \in R^n$ and $y_i \in \{-1,1\}$ for $i=1,2,\cdots,N$. The optimal hyper-plane of S is defined as $f(x)=0$, where

$$f(x)=\left(w_0 \cdot x\right)+b_0 \tag{1}$$

$$w_0=\sum_{j=1}^{N} y_j \alpha_j^0 x_j \tag{2}$$

$\left(w_0 \cdot x\right)=\sum_{i=1}^{n} w_0^i \cdot x^i$ is the inner product of the two vectors, where $w_0=(w_0^1,w_0^2,\cdots,w_0^n)$ and $x=(x^1,x^2,\cdots,x^n)$. The vector W_0 can be determined according to the following quadratic programming [1]

$$\begin{aligned} \min \quad & \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{N}\xi_i \\ s.t. \quad & y_i((w\cdot x_i)+b) \geq 1-\xi_i \\ & \xi_i \geq 0, \quad i=1,2,\ldots,N \end{aligned} \tag{3}$$

Its dual problem is following:

$$\begin{cases} \text{Maximum } W(\alpha) = \sum_{i=1}^{N}\alpha_i - \frac{1}{2}\sum_{i,j=1}^{N} y_i y_j \alpha_i \alpha_j \left(x_i \cdot x_j\right) \\ \text{Subject to } \sum_{j=1}^{N} y_j\alpha_j = 0;\, C \geq \alpha_i \geq 0, \ i=1,2,\cdots,N \end{cases} \tag{4}$$

where C is a positive constant. The constant b_0 is given by

$$b_0 = y_i - \left(x_i \cdot \sum_{j=1}^{N} y_j \alpha_j^0 x_j\right) \tag{5}$$

Substituting (2) for w_0 in (1), we have

$$f(x) = \sum_{i=1}^{N} y_i \alpha_i^0 \left(x_i \cdot x\right) + b_0 \tag{6}$$

We can know separability of two subsets through checking whether the following inequalities

$$y_i\left(w_0 \cdot x_i - b\right) \geq 1; \ i=1,2,\cdots,N \tag{7}$$

hold well[1].

A procedure to compute maximum margin for two subsets is described below.

Procedure 1. The constant C in equation (4) is selected to be large at first.

Step 1. Solving the quadratic programming (4).

Step 2. Determining the separating hyper-plane (6) according to (5).

Step 3. Checking the separability between two subsets according to inequalities (7).

Step 4. Let the margin be 0 if the two subsets are not separable.

Step 5. Computing the maximum margin according to $1/\sqrt{(w_0 \cdot w_0)}$ for the separable case where the vector w is determined by (2).

2.2. Generalization in Feature Space

Practically the performance of SVMs based on the previous section may not be very good for the nonlinear-separable cases in the original space. To improve the performance and to reduce the computational load for the nonlinear separable datasets, Vapnik [1] extended the SVMs from the original space to the feature space. The key idea of the extension is that a SVM first maps the original input space into a high-dimensional feature space through some nonlinear mapping, and then constructs an optimal separating hyper-plane in the feature space. Without any knowledge of the mapping, the SVM can find the optimal hyper-plane by using the dot product function in the feature space. The dot function is usually called a kernel function. According to Hilbert-Schmidt theorem [1], there exists a relationship between the original space and its feature space for the dot product of two points. That is

$$(z_1 \cdot z_2) = K(x_1, x_2) \tag{8}$$

where it is assumed that a mapping Φ from the original space to the feature space exists, such that $\Phi(x_1) = z_1$ and $\Phi(x_2) = z_2$, and $K(x_1, x_2)$ is conventionally called a kernel function satisfying the Mercer theorem [1]. Usually the following three types of kernel functions can be used: polynomial with degree p, radial basis function and sigmoid function [1]. Replacing the inner product $(x_1 \cdot x_2)$ in (6) with the kernel function $K\left(x_1, x_2\right)$, the optimal separating hyper-plane becomes the following form:

$$f(x) = \sum_{i=1}^{N} y_i \alpha_i^0 K\left(x_i, x\right) + b_0 \tag{9}$$

It is worth noting that the conclusion of section 2.1 is still valid in the feature space if we substitute $K\left(x_1, x_2\right)$ for the inner product $\left(x_1 \cdot x_2\right)$.

3. An inverse problem of SVMs and its solution based on genetic algorithms

For a given dataset of which no class labels are assigned to instances, we can randomly split the dataset into two subsets. Suppose that one is the positive instance subset and the other is the negative instance subset, we can calculate the maximum margin between the two subsets according to Procedure 1 where the margin is equal to 0 for the non-separable case. Obviously the calculated margin depends on the random split of the dataset. Our problem is how to split the dataset such that the margin calculated according to Procedure 1 attains the maximum.[5]

It is an optimization problem. We mathematically formulate it as follows. Let $S = \left\{x_1, x_2, \cdots, x_N\right\}$ be a dataset and $x_i \in R^n$ for $i = 1,2,\cdots,N$, $\Omega = \left\{f \mid f \text{ is a function from } S \text{ to } \{1,-1\}\right\}$. Given a function $f \in \Omega$, the dataset can be split to two subsets and then the margin can be calculated by Procedure 1. We denote the calculated margin (the functional) by $\text{Margin}(f)$. Then the inverse problem is formulated as

$$\text{Maximum}_{f\in\Omega}\left(\text{Margin}(f)\right) \tag{10}$$

Due to the exponentially increased complexity, it is not feasible to enumerate all possible functions in Ω for calculating their margins according to Procedure 1. It is difficult to give an exact algorithm for solving the optimization problem (10). This paper makes an attempt to solve (10) by designing a genetic algorithm [4]. Because of the limit of the paper length, we only present the encoding mechanics and fitness function of our proposed genetic algorithm to solve equation (10). For details about genetic algorithms, one can refer to [4].

Each function $f \in \Omega$ corresponds to a binary partition of the dataset *S*. Therefore each *f* can be viewed as a N-dimensional vector such as $100011101\cdots01$ with *N* bits. Each bit taking value 0 or 1 is regarded as a gene corresponding to an instance in *S*. Thus each chromosome (a bit string such as $100011101\cdots01$) consisting of *N* genes represents a function in Ω where if a bit is 1 it means that the corresponding instance is positive; and a value 0 represents that the corresponding instance is negative. The fixed length of each chromosome's coding is N, the number of instances of the initial dataset.

Noting that each chromosome corresponds to a training set given in section 2, we define the fitness value for each chromosome as the margin value computed by Procedure 1. Here the fitness value is 0 if the chromosome corresponds to a non-separable training set, and is the real margin of the SVM if the chromosome corresponds to a separable training set.

Due to the limit of paper length, we omit the formulation of the algorithm. For details, one can refer to [5].

4. Linear programming approach for the inverse problem of support vector machines

In reference [5], a well-known dataset called Iris is selected to verify the relationship between the running time and the number of samples. 100 samples of the dataset (the second class and the third class) are used for the verification. Figure 1 shows the running time change with the increase of samples. The increase seems to be exponential.

From the experiment result, we know that that the proposed genetic algorithm has large time complexity, the main reason is the process of solving quadratic programs. In this paper, we replace the quadratic programming with linear programming. In reference [6], quadratic programming (3) is replaced with the following linear programming:

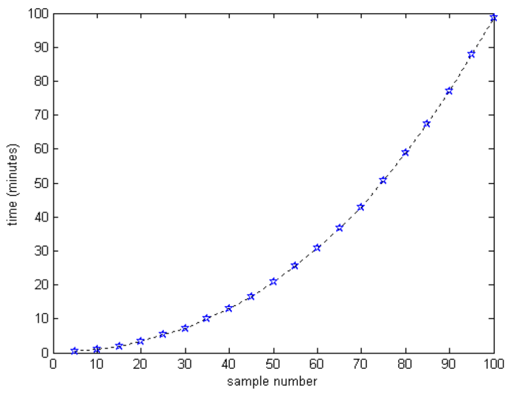

Figure 1. Running time change with the increase of samples

$$\begin{aligned}
\min \quad & L = -r + C\sum_{i=1}^{N}\xi_i \\
s.t. \quad & y_i f(x_i) = y_i\left(\sum_{j=1}^{N}\alpha_j y_j (x_j \cdot x_i) + b\right) \geq r - \xi_i \\
& r \geq 0 \\
& -1 \leq \alpha_i \leq 1 \\
& \xi_i \geq 0, \quad i = 1,2,\ldots,N
\end{aligned} \tag{11}$$

Through introducing linear programming, the time complexity of genetic algorithm for solving the inverse problem of support vector machines is improved, and the applicability of the proposed inverse problem of support vector machines.

5. Concluding remarks

Motivated by designing a new heuristic procedure of generating decision trees with higher generalization capability, reference [5] proposes a genetic algorithm for solving an inverse problem of SVMs. The algorithm is effective and efficient for small datasets. The main disadvantage of this algorithm is its large time complexity. In this paper, we replace the quadratic programming with linear programming, which improving greatly the time complexity, and the applicability of the proposed inverse problem of support vector machines. We have the following remarks:

(1) We need to have some experiments to verify the effectiveness of linear programming support vector machines.

(2) Whether the linear programming support vector machines has the loss in the generalization.

(3) If there is the loss in the linear programming support vector machines, we must evaluate whether it is worthy although linear programming SVM improves the time complexity.

Acknowledgements

The first author acknowledges the financial support from the National Natural Science Foundation of China (Project No. 60473045 & 60573069).

References

[1] V. N. Vapnik, Statistical learning theory, New York, Wiley, 1998, ISBN: 0-471-03003-1.

[2] V. N. Vapnik, The Nature of Statistical Learning Theory, Springer-Verlag, New York, 2000, ISBN: 0-387-98780-0.

[3] V. N. Vapnik, An Overview of Statistical Learning Theory, IEEE Transactions on Neural Networks, Vol. 10, No.5, Pages 88-999, 1999.

[4] Srinivas, M.; Patnaik, L.M.; Genetic algorithms: a survey, Computer, Volume: 27, Issue: 6, June 1994, Pages: 17-26.

[5] Wang XZ, He Q, Chen DG, et al. A genetic algorithm for solving the inverse problem of support vector machines, NEUROCOMPUTING 68: 225-238 OCT 2005

[6] Weida Zhou; Li Zhang; Licheng Jiao, Linear programming support Vector Machines, PATTERN RECOGNITION, Volume 35, 2002, Page 2927-2936.