Available online at www.sciencedirect.com

SCIENCE DIRECT®

Neurocomputing 68 (2005) 225–238

NEUROCOMPUTING

www.elsevier.com/locate/neucom

Letters

A genetic algorithm for solving the inverse problem of support vector machines

Xi-Zhao Wang[a,*], Qiang He[a], De-Gang Chen[b], Daniel Yeung[c]

[a]*Department of Mathematics and Computer Science, Hebei University, No. 88, Hezuo Road, Baoding, Hebei 071002, China*
[b]*Department of Mathematics and Physics, North China Electric Power University, Beijing, China*
[c]*Department of Computing, Hong Kong Polytechnic University, Kawloon, Hong Kong*

Received 10 March 2005; received in revised form 27 April 2005; accepted 1 May 2005
Available online 26 July 2005
Communicated by R.W. Newcomb

Abstract

This paper investigates an inverse problem of support vector machines (SVMs). The inverse problem is how to split a given dataset into two clusters such that the margin between the two clusters attains the maximum. Here the margin is defined according to the separating hyperplane generated by support vectors. It is difficult to give an exact solution to this problem. In this paper, we design a genetic algorithm to solve this problem. Numerical simulations show the feasibility and effectiveness of this algorithm. This study on the inverse problem of SVMs is motivated by designing a heuristic algorithm for generating decision trees with high generalization capability.
© 2005 Elsevier B.V. All rights reserved.

Keywords: Support vector machines; Genetic algorithms; Maximum margin

1. Introduction

Support vector machines (SVMs) are a classification technique of machine learning based on statistical learning theory [6,8]. Considering a classification

*Corresponding author.
E-mail address: wangxz@mail.hbu.edu.cn (X.-Z. Wang).

0925-2312/$ - see front matter © 2005 Elsevier B.V. All rights reserved.
doi:10.1016/j.neucom.2005.05.006

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 [6,7], the maximum of margin implies the extra-ordinary generalization capability and good performances of SVM classifiers [1,2,3]. So far, SVMs have already been successfully applied to many real fields. This paper aims to make a preparation for SVM's application to decision tree generation.

Given a training set, a general procedure for generating a decision tree can be briefly described as follows. The entire training set is first considered as the root node of the tree. Then the root node is split into two sub-nodes based on some heuristic information. If the instances in a sub-node belong to one class, then the sub-node is regarded as a leaf node, else we continue to split the sub-node based on the heuristic information. This process repeats until all leaf nodes are generated. The most popular heuristic information used in the decision tree generation is the minimum entropy. This heuristic information has many advantages such as small number of leaves and less computational efforts. However, it has a serious disadvantage—the poor generalization capability.

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.

This paper has the following organization. Section 2 briefly reviews the basic concept of support vector machines. Section 3 proposes the inverse problem of SVMs and designs a genetic algorithm to solve this problem. Section 4 gives some simulations to demonstrate the feasibility and effectiveness of the genetic algorithm. And the last section briefly concludes this paper.

2. Support vector machines

2.1. The basic problem of SVMs

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

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

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

$(w_0 \cdot x) = \sum_{i=1}^{n} w_0^i \cdot x^i$ is the inner product of the two vectors, where $w_0 = (w_0^1, w_0^2, \ldots, w_0^n)$ and $x = (x^1, x^2, \ldots, x^n)$. The vector W_0 can be determined according to the following quadratic programming [6]

$$\begin{aligned} &\text{Maximum}\ \ W(\alpha) = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{N} y_i y_j \alpha_i \alpha_j \left(x_i \cdot x_j\right), \\ &\text{Subject to}\ \ \sum_{j=1}^{N} y_j \alpha_j = 0; \ \ C \geq \alpha_i \geq 0, \quad i = 1, 2, \ldots, N, \end{aligned} \tag{3}$$

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

$$b_0 = y_i - \left(x_i \sum_{j=1}^{N} y_j \alpha_j^0 x_j \right). \tag{4}$$

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

$$f(x) = \sum_{i=1}^{N} y_i \alpha_i^0 (x_i x) + b_0. \tag{5}$$

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

$$y_i(w_0 x_i - b) \geq 1; \quad i = 1, 2, \ldots, N \tag{6}$$

hold well [6].

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

Procedure 1. The constant C in equation (3) is selected to be large at first.

Step 1. Solving the quadratic programming (3).

Step 2. Determining the separating hyper-plane (5) according to (4).

Step 3. Checking the separability between two subsets according to inequalities (6).

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

Step 5. Computing the maximum margin according to $1/(w_0 \cdot w_0)$ 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 [6] extended the SVMs from the original space to the feature space. The key idea of the extension is that an 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 [6], there exists a relationship between the original space and its feature space for the dot product of two points. That is

$$(z_1 z_2) = K(x_1, x_2), \tag{7}$$

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 [6]. Usually the following three types of kernel functions can be used: polynomial with degree p, radial basis function and sigmoid function [6]. Replacing the inner product $(x_1 \cdot x_2)$ in (5) with the kernel function $K(x_1,x_2)$, the optimal separating hyper-plane becomes the following form:

$$f(x) = \sum_{i=1}^{N} y_i \alpha_i^0 K(x_i, x) + b_0. \tag{8}$$

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

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.

It is an optimization problem. We mathematically formulate it as follows. Let $S = \{x_1, x_2, \ldots, x_N\}$ be a dataset and $x_i \in R^n$ for $i = 1, 2, \ldots, N$, $\Omega = \{f | f \text{ is a function from } S \text{ to } \{1, -1\}\}$. 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 Margin (f). Then the inverse problem is formulated as

$$\text{Maximum}_{f \in \Omega}(\text{Margin}(f)). \tag{9}$$

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 (9). This paper makes an attempt to solve (9) 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 Eq. (9). 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 $100011101 \cdots 01$ 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 $100011101 \cdots 01$) 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.

Suppose that we initially have a dataset $\{x_1, x_2, \ldots, x_N\}$, $x_i \in R^n$, $i = 1, 2, \ldots, N$. A N-dimensional Boolean vector $c = a_1 a_2 \ldots a_N$ is considered as a chromosome, $f(j)$ denotes the fitness value of the jth chromosome. Random(A) denotes a random number uniformly distributed in [0, A]. Specifying the penalty factor C, the population size M, the crossover probability p_c, the mutation probability p_m, the procedure flowchart of this algorithm is shown as Fig. 1. In the last box of Fig 1, $c1 = a_1 a_2 \ldots a_N$ is the optimal chromosome, which corresponds to a function f splitting dataset into two subsets based on the above what has been mentioned in this section. $f(c1)$ is the margin between the two subsets calculated by Procedure 1. The decision function f denotes the optimal or approximately optimal solution of problem (9) when the parameters in GA are selected properly.

4. Numerical examples

To verify the effectiveness of the proposed genetic algorithm, we construct a small dataset with twenty 2-dimesional points (Table 1). Fig. 2 shows the distribution of the 20 points of Table 1. Respectively we use the proposed genetic algorithm and the enumeration method to acquire the maximum margin on Table 1. The parameters specified in the genetic algorithm are shown in Table 2 where the probability of mutation is enlarged to be 0.8. Enlarging the mutation probability aims to enhance the effect of mutation, which is expected to generate more excellent chromosomes and to accelerate the evolutionary process. Table 3 shows the experimental results on the original space for both the genetic algorithm and the enumeration. From Table 3 one can see that the running time of the proposed genetic algorithm is significantly less than the enumeration method, but the performance of the genetic algorithm for searching the second best margin is not very good. This is because that the genetic algorithm is a type of incomplete search on the complete space. Genetic algorithms cannot guarantee obtaining the optimal solution every time, but it is expected to have a big probability for obtaining the optimal or approximately optimal solution. To raise the probability of obtaining optimal solution, one needs to increase the population size or the maximum number of generations, which obviously is at the price of running time increase. Practically we need a reasonable balance between the accuracy and the running time.

Figs. 3–6 show the separating hyper-planes with maximum margin and second maximum margin for both the proposed genetic algorithm and the enumeration on the original space.

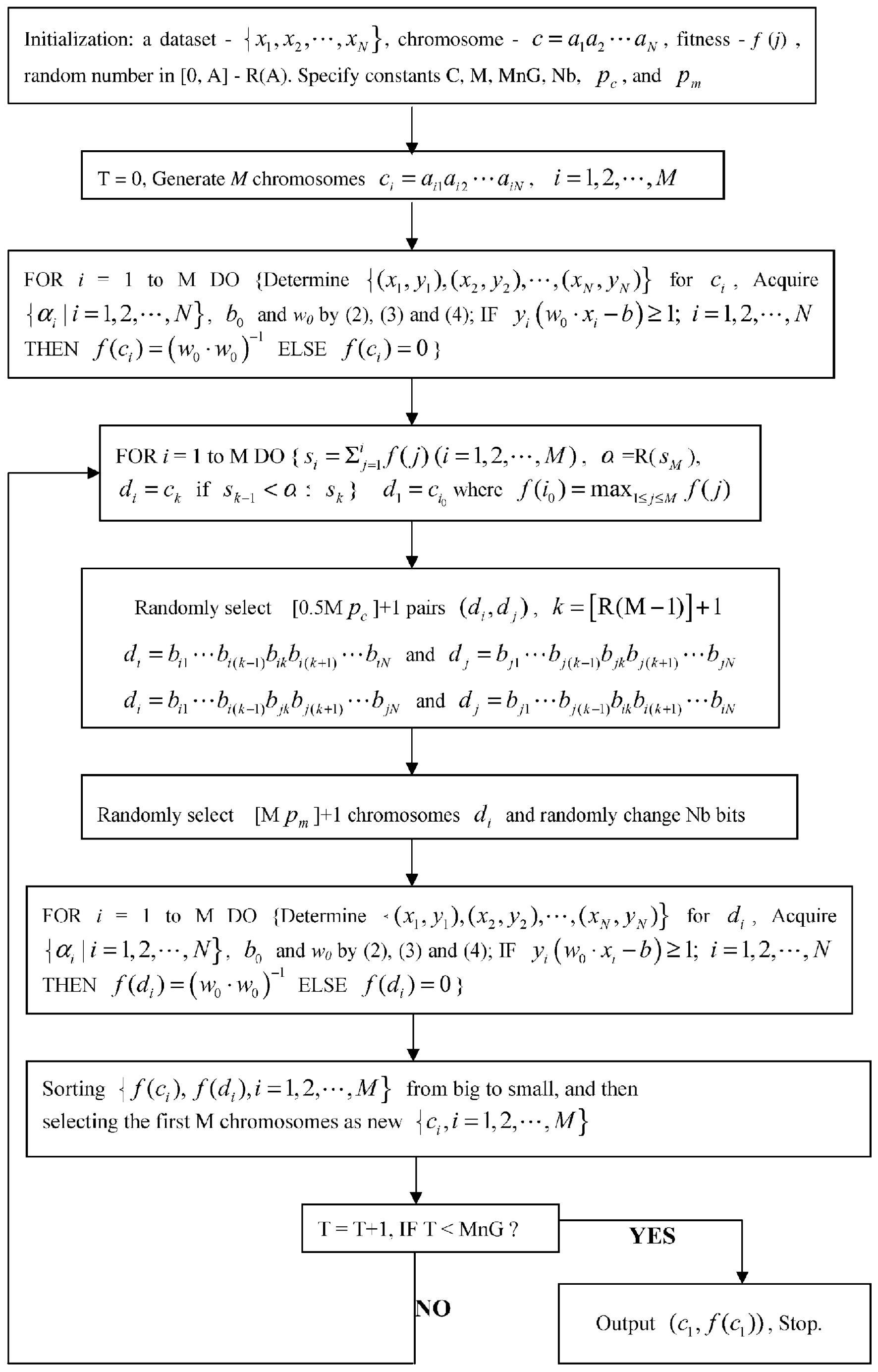
Initialization: a dataset - $\{x_1, x_2, \cdots, x_N\}$, chromosome - $c = a_1a_2\cdots a_N$, fitness - $f(j)$, random number in [0, A] - R(A). Specify constants C, M, MnG, Nb, p_c, and p_m
T = 0, Generate M chromosomes $c_i = a_{i1}a_{i2}\cdots a_{iN}$, $i = 1, 2, \cdots, M$
FOR i = 1 to M DO {Determine $\{(x_1, y_1), (x_2, y_2), \cdots, (x_N, y_N)\}$ for c_i, Acquire $\{\alpha_i \mid i = 1, 2, \cdots, N\}$, b_0 and w_0 by (2), (3) and (4); IF $y_i(w_0 \cdot x_i - b) \geq 1$; $i = 1, 2, \cdots, N$ THEN $f(c_i) = (w_0 \cdot w_0)^{-1}$ ELSE $f(c_i) = 0$ }
FOR i = 1 to M DO { $s_i = \Sigma_{j=1}^{i} f(j)$ $(i = 1, 2, \cdots, M)$, α =R(s_M), $d_i = c_k$ if $s_{k-1} < \alpha$: s_k } $d_1 = c_{i_0}$ where $f(i_0) = \max_{1 \leq j \leq M} f(j)$
Randomly select [0.5Mp_c]+1 pairs (d_i, d_j), $k = [R(M-1)] + 1$
$d_i = b_{i1}\cdots b_{i(k-1)}b_{ik}b_{i(k+1)}\cdots b_{iN}$ and $d_j = b_{j1}\cdots b_{j(k-1)}b_{jk}b_{j(k+1)}\cdots b_{jN}$
$d_i = b_{i1}\cdots b_{i(k-1)}b_{jk}b_{j(k+1)}\cdots b_{jN}$ and $d_j = b_{j1}\cdots b_{j(k-1)}b_{ik}b_{i(k+1)}\cdots b_{iN}$
Randomly select [Mp_m]+1 chromosomes d_i and randomly change Nb bits
FOR i = 1 to M DO {Determine $\{(x_1, y_1), (x_2, y_2), \cdots, (x_N, y_N)\}$ for d_i, Acquire $\{\alpha_i \mid i = 1, 2, \cdots, N\}$, b_0 and w_0 by (2), (3) and (4); IF $y_i(w_0 \cdot x_i - b) \geq 1$; $i = 1, 2, \cdots, N$ THEN $f(d_i) = (w_0 \cdot w_0)^{-1}$ ELSE $f(d_i) = 0$ }
Sorting $\{f(c_i), f(d_i), i = 1, 2, \cdots, M\}$ from big to small, and then selecting the first M chromosomes as new $\{c_i, i = 1, 2, \cdots, M\}$
T = T+1, IF T < MnG ?
YES
NO
Output $(c_1, f(c_1))$, Stop.

Fig. 1. Flowchart for solving the inverse problem of SVMs.

Table 1
A small dataset

Case	Feature1	Feature2	Case	Feature1	Feature2
1	0.428	0.064	11	0.12	0.552
2	0.416	0.082	12	0.15	0.53
3	0.388	0.098	13	0.128	0.562
4	0.444	0.12	14	0.09	0.592
5	0.468	0.092	15	0.174	0.58
6	0.762	0.272	16	0.466	0.786
7	0.784	0.302	17	0.442	0.802
8	0.75	0.314	18	0.438	0.764
9	0.728	0.272	19	0.502	0.742
10	0.712	0.332	20	0.482	0.82

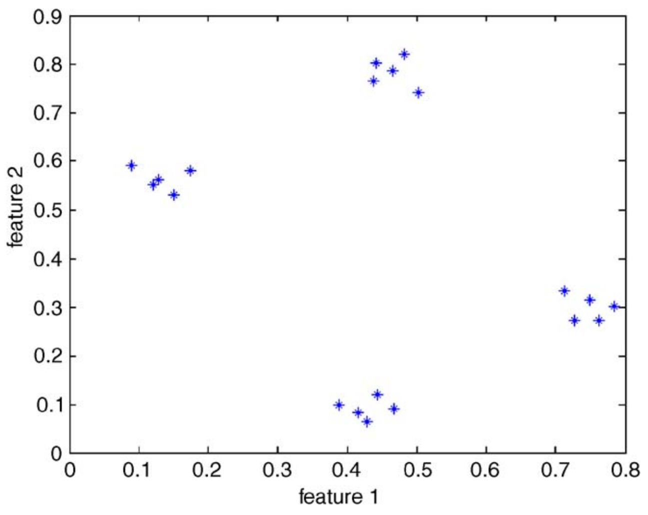

Fig. 2. Sample distribution of Table 1.

Table 2
Parameters in genetic algorithm

POPSIZE = 150	Size of population
PC = 0.7	Probability of crossover
PM = 0.8	Probability of mutation
NB = 0.3	Gen mutation proportion
$G = 0.6$	Generation gap
MAXGENERATION = 20;	Maximum generation

Table 3
Experimental results on the original space

	Time (min)	The best margin	The second best margin
Numeration method	922.45	0.4596	0.3156
Genetic algorithm	3.37	0.4596	0.2546

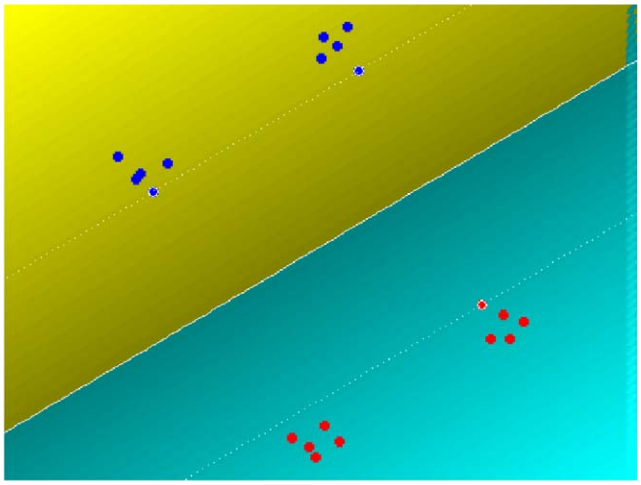
Fig. 3. Max margin by enumeration on the original space.

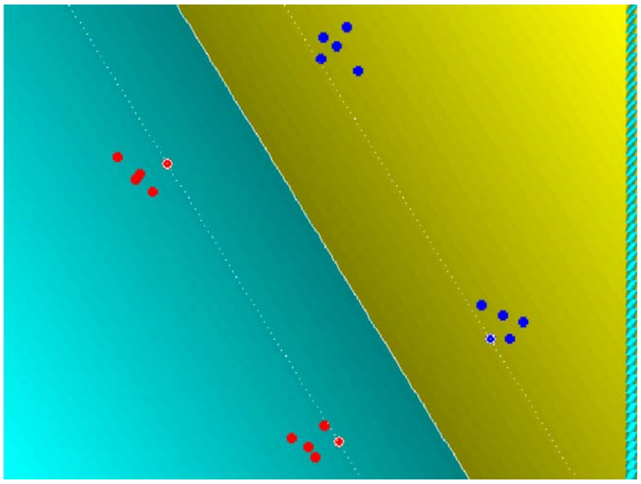
Fig. 4. Second max margin by enumeration on the original space.

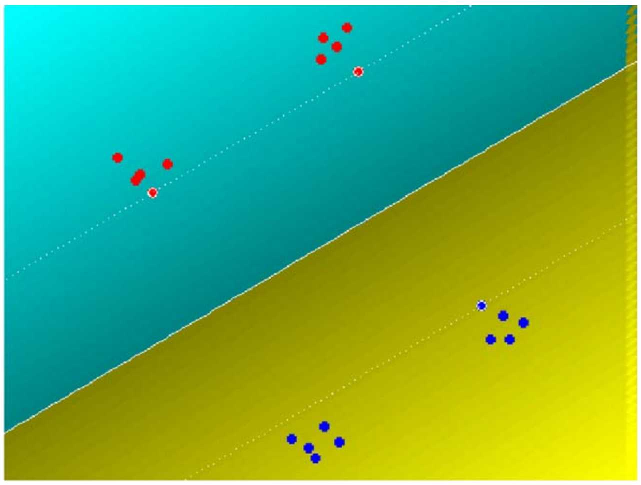

Fig. 5. Max margin by GA on the original space.

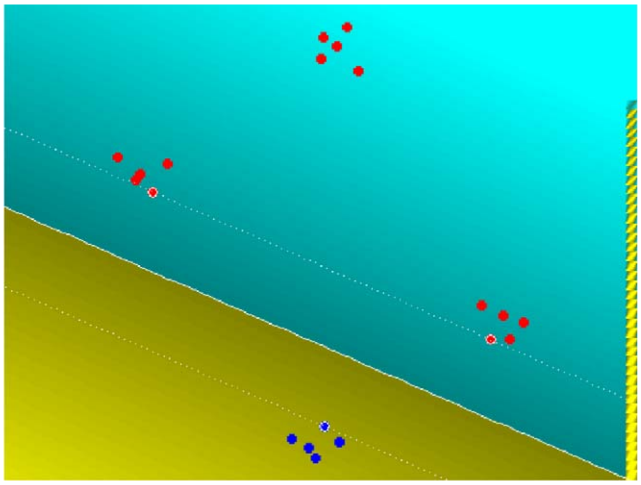

Fig. 6. Second max margin by GA on the original space.

Selecting the 3rd polynomial as the kernel function and keeping the parameters of the GA unchanged, we repeat the experiments on the feature space for the small dataset given in Table 1. The experimental results are shown in Table 4 and Figs. 7–10.

Comparing Figs. 3–6 with Figs. 7–10, we find a similar consequent. Due to the use of kernel functions, Fig. 10 shows a nonlinear boundary. This is the essential difference between the original space and the feature space. From Tables 3 and 4 we find that, for the small dataset both in the original space and in the feature space, the running time of the genetic algorithm is significantly less than the enumeration. It

Table 4
Experimental results on the feature space

	Time (min)	The best margin	The second best margin
Numeration method	1051.4	0.98694	0.6285
Genetic algorithm	3.57	0.98694	0.60667

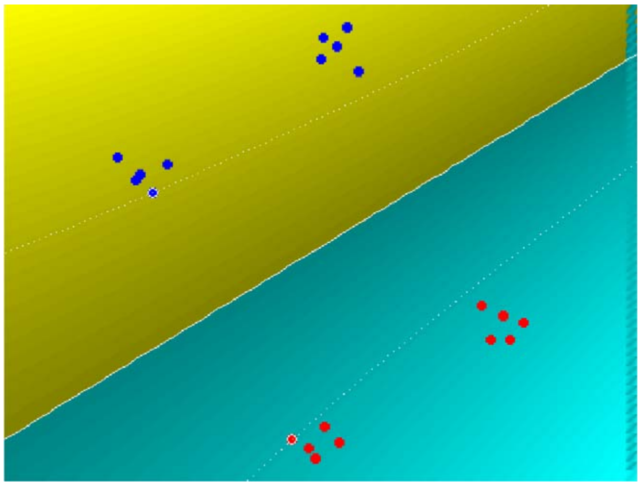

Fig. 7. Max margin by enumeration on the feature space.

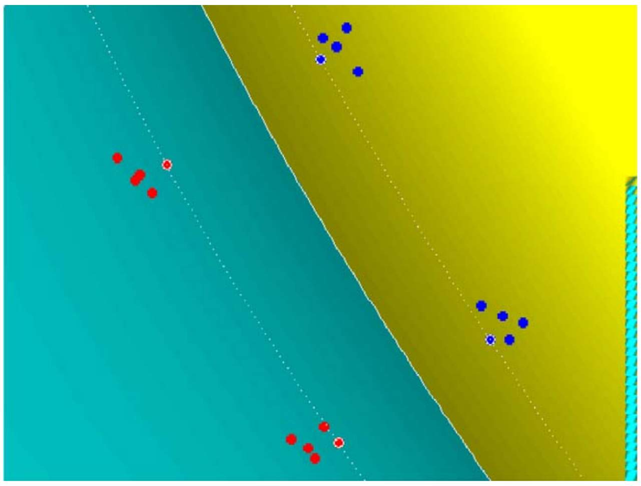

Fig. 8. Second max margin by enumeration on the feature space.

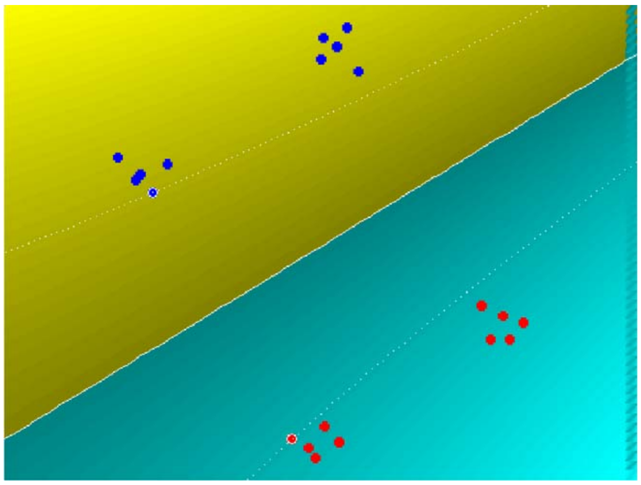

Fig. 9. Max margin by GA on the feature space.

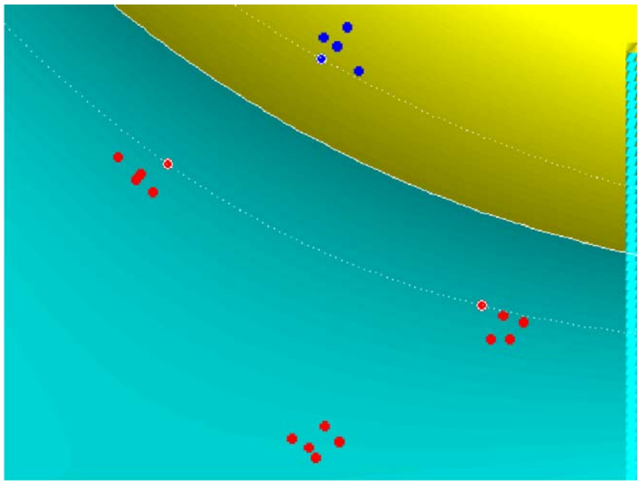

Fig. 10. Second max margin by GA on the feature space.

implies that the proposed genetic algorithm has much smaller time-complexity than the enumeration. However, with an increase of dataset size, the time complexity of the proposed genetic algorithm is increasing rapidly. It is the main defect of this type of genetic algorithms.

Since the main difficulty in the genetic algorithm is the searching time, we now experimentally check the time change with the increase of samples.

A well-known dataset called Iris [5] is selected to verify the relationship between the running time and the number of samples. We use 100 samples of the dataset (the second class and the third class) for the verification. Table 5 shows the running

Table 5
Running time with the increase of samples in Iris dataset

Sample number	Running time (min)	Sample number	Running time (minutes)
5	0.42943	55	25.455
10	0.90247	60	30.925
15	1.7772	65	36.793
20	3.3708	70	42.832
25	5.4337	75	50.649
30	7.2144	80	59.036
35	10.013	85	67.508
40	12.96	90	77.152
45	16.646	95	87.893
50	21.045	100	98.792

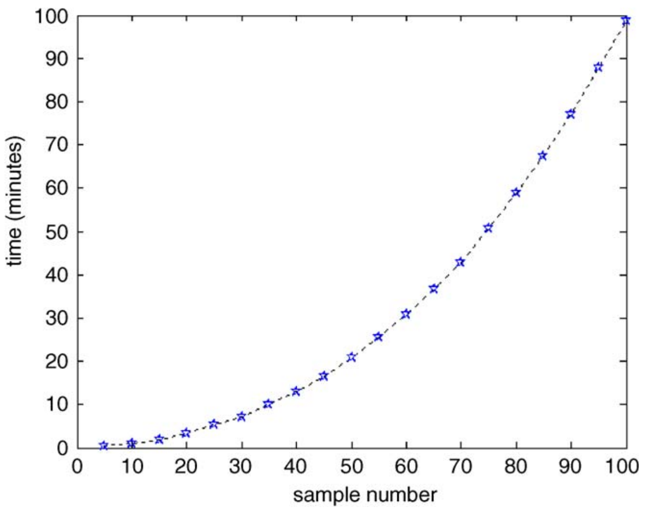

Fig. 11. Running time change with the increase of sample.

time change with the increase of samples. From Table 5 and Fig. 11 we observe that the running time rapidly increase with the samples. The increase seems to be exponential.

One reason that the proposed genetic algorithm has large time complexity is the process of solving quadratic programs. We do not know whether the quadratic programming can be replaced with an approximate and fast algorithm for acquiring the maximum margin. It is an issue really being valuable to investigate.

The large time complexity downgrades the applicability of the proposed genetic algorithm. How to reduce the time complexity of the algorithm for large databases is a very important issue to be investigated further.

5. Concluding remarks

Motivated by designing a new heuristic procedure of generating decision trees with higher generalization capability, this paper 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. Practically it is expected to give an improved version with time complexity reduction.

Acknowledgements

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

References

[1] B.E. Boser, I.M. Guyon, V.N. Vapnik, A training algorithm for optimal margin classifiers, in: D. Guyon, Haussler (Eds.), Proceedings of the Fifth Annual Workshop on Computational Learning Theory, ACM Press, Pittsburgh, 1992, pp. 144–152.
[2] R. Schapire, Y. Freund, P. Bartlett, W. Sun Lee, Boosting the margin: a new explanation for the effectiveness of voting methods, Ann. Statist. 26 (5) (1998) 1651–1686.
[3] J. Shawe-Taylor, N. Cristianini, On the generalization of soft margin algorithms, IEEE Trans. Inform. Theory 48 (10) (2002) 2721–2735.
[4] M. Srinivas, L.M. Patnaik, Genetic algorithms: a survey, Computer 27 (6) (1994) 17–26.
[5] UCI Repository of machine learning databases and domain theories. FTP address: ftp://ftp.ics.uci.edu/pub/machine-learning-databases
[6] V.N. Vapnik, Statistical Learning Theory, Wiley, New York, ISBN 0-471-03003-1, 1998.
[7] V.N. Vapnik, An overview of statistical learning theory, IEEE Trans. Neural Networks 10 (5) (1999) 88–999.
[8] V.N. Vapnik, The Nature of Statistical Learning Theory, Springer, New York, ISBN 0-387-98780-0, 2000.

Xi-Zhao Wang received the B.Sc. and M.Sc. degrees in Mathematics from Hebei University, Baoding, China, in 1983 and 1992, respectively, and the Ph.D. degree in Computer Science from the Harbin Institute of Technology, Harbin, China, in 1998.

From 1983 to 1998, he worked as a Lecturer, an Associate Professor, and a Full Professor in the Department of Mathematics, Hebei University. From 1998 to 2001, he worked as a Research Fellow at the Department of Computing, Hong Kong Polytechnic University, Kowloon. Since 2001, he has been the Dean and Professor of the Faculty of Mathematics and Computer Science, Hebei University. His main research interests include inductive learning with fuzzy representation, fuzzy measures and integrals, neuro-fuzzy systems and genetic algorithms, feature extraction, multi-classifier fusion, and applications of machine learning. So far, he has published over 30 international journal papers. He is an IEEE Senior Member and is an Associate Editor of IEEE Transactions on SMC part B.

Prof. Xi-Zhao Wang is the General Co-Chair of the 2002, 2003, and 2004 International Conference on Machine Learning and Cybernetics, co-sponsored by IEEE SMC.

Qiang He received the B.Sc. and M.Sc. degrees in Mathematics from the Hebei University, Baoding, China, in 2000 and 2003, respectively.

From 2003 to now, he worked as a Teaching Assistant in the Faculty of Mathematics and Computer Science, Hebei University. In 2004, he worked as a Research Assistant at the Department of Computing, Hong Kong Polytechnic University, Kowloon. His main research interests include inductive learning, genetic algorithms and statistical learning theory.

Degang Chen received the M.S. degree from the Northeast Normal University, Changchun, Jilin, China, in 1994, and the Ph.D. degree from the Harbin Institute of Technology, Harbin, China, in 2000.

He has worked as a Postdoctoral Fellow with the Xi'an Jiaotong University, Xi'an, China, from 2000 to 2002 and with the Tsinghua University, Tsinghua, China, from 2002 to 2004. From 1994 to 2004, he had been a Teacher at the Bohai University, Jinzhou, Liaoning, China, and since 2005, he has worked as a Teacher at the North China Electric Power University, Beijing, China. His research interests include fuzzy group, fuzzy analysis, rough sets, and SVM.

Daniel S. Yeung (M'89–SM'99) received the Ph.D. degree in Applied Mathematics from the Case Western Reserve University, Cleveland, OH, in 1974. In the past, he has worked as an Assistant Professor of Mathematics and Computer Science at Rochester Institute of Technology, Rochester, NY, as a Research Scientist in the General Electric Corporate Research Center, and a System Integration Engineer at TRW, Inc. Currently, he is a Chair Professor of Computing, The Hong Kong Polytechnic University, Hong Kong. His current research interests include neural-network sensitivity analysis, expert neural-network hybrid systems, off-line handwritten Chinese character recognition, and fuzzy expert systems.

Dr. Yeung was the President of the IEEE Computer Chapter of Hong Kong for 1991 and 1992. He now chairs the technical committees on Cybernetics of the IEEE Systems, Man, and Cybernetics Society. He is also an Associate Editor for the IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS and the IEEE TRANSACTIONS ON NEURAL NETWORKS.