Classification Using Support Vector Machines with Graded Resolution

Lipo Wang*†, Bing Liu† and Chunru Wan†
* College of Information Engineering, Xiangtan University, Xiangtan, Hunan, China
Email: elpwang@ntu.edu.sg
† School of Electrical and Electronic Engineering, Nanyang Technology University,
Block S1, 50 Nanyang Avenue, Singapore 639798

***Abstract*— A method which we call support vector machine with graded resolution (SVM-GR) is proposed in this paper. During the training of the SVM-GR, we first form data granules to train the SVM-GR and remove those data granules that are not support vectors. We then use the remaining training samples to train the SVM-GR. Compared with the traditional SVM, our SVM-GR algorithm requires fewer training samples and support vectors, hence the computational time and memory requirements for the SVM-GR are much smaller than those of a conventional SVM that use the entire dataset. Experiments on benchmark data sets show that the generalization performance of the SVM-GR is comparable to the traditional SVM.**

Keywords: granular support vector machine, granular computing

I. Introduction

SVMs have shown great capabilities in solving various classification problems [1]–[6]. In many applications, SVMs have outperformed many other machine learning methods [1] and have established themselves as a powerful tool for classification problems. When performing a pattern recognition task, the SVM first maps the input data into a high-dimensional feature space and then finds an optimal separating hyperplane to maximizes the margin between two classes in this high-dimensional space. Maximizing the margin is a quadratic programming (QP) problem and can be solved by using some optimization algorithms [7]. However, for large-scale training sets, two major problems still exist for standard SVMs with traditional optimization algorithms, such as Newton and Quasi-Newton [8]. First, most computation consumed in the support vector learning is to solve a full dense matrix Q, where $Q \in R^{l \times l}$ is a positive semi-definite matrix and l is the number of training samples. Therefore, the time complexity is exponential with respect to l. [9], [10]. Second, solving the QP problem needs storing the matrix Q. Hence, the memory requirements will grow with square of the size of training sample, which may become too large for large data sets. [8]

One major method to solve these problems is the decomposition method [11], which solves a sequence of smaller-sized QP problems so that the the memory and computation difficulties can be avoided. However, for huge problems with many support vectors, the training speed of the decomposition method is still very slow [8]. Recently, Lee and Mangasarian [9] proposed the reduced SVM (RSVM) to reduce the computation complexity. The key idea of the RSVM is to randomly select a subset of training data as candidates of support vectors and use the whole training set as constraints, in order to obtain a smaller QP problem. But in this method, the 'support vectors' are chosen randomly, which leads to the unstable results. Zheng et al. [10] extended the RSVM by applying unsupervised clustering methods to select candidates of support vectors. They stated that the modified RSVM algorithm achieves more stable results and higher classification accuracy in comparison to the RSVM. However, it should be noticed that many unsupervised clustering methods also have high time complexity. Therefore, the modified RSVM algorithm may also be slower than the RSVM and not suitable for large problems.

In this paper, we propose a novel learning method called SVM with graded resolution (SVM-GR), which uses ideas in granular computing method to solve the QP problem. Granular computing is an emerging conceptual paradigm of computing with information granules [12] [13]. Information granules are collections of entities, usually originating at the numeric level, and then arranged together due to their similarity, functional adjacency, indistinguishability, coherency or alike [14]. There are a number of formal models of of information granules including interval analysis, fuzzy sets, rough sets, shadowed sets and probabilistic sets. Information granules are processed as follows [12]. Depending upon the problem, we usually group granules of similar "size" (or called granularity) together in a single layer. If more detailed processing is required, smaller information granules are sought and then these granules are arranged in another layer. Recently, granular computing has been successfully applied to many areas, such as data mining [15], image compression [16], signal analysis [17] and neural network design [18], [19]. In 2004, Tang et al. [20] proposed the GSVM algorithm that used the granular computing methods to build SVMs. In [20], the whole feature space was split into a set of subspaces and for each subspace a hyperplane was built. Training the GSVM required to maximize the margin width for each subspace simultaneously.

The proposed SVM-GR method is inspired by the essential result in support vector learning, i.e., the optimal separating hyperplane is determined by only the support vectors. Since all the non-support-vector samples have no contribution to set up the optimal hyperplanes, we can filter these samples in advance and use the remaining data to perform support vector learning. The development of the SVM-GR involves two main phases as follows:

1) Rough Filtering: Divide each input dimension into r equal segments. The input space thus forms r^n data granules, where n is the dimensionality of the input space. Use the average within each granule to train a SVM. The non-support vector granules are filtered (ignored in further training).
2) Elaborate Learning: Release the training samples from the remaining support vector granules. These training samples are used for final support vector learning.
3) If the data set is very large, rather than using all data in the support vector granules for training the SVM directly in step 2, we can form smaller granules within these larger granules obtained in Step 1 and filter non-support-vector granules again. The number of levels in this hierarchical granulation may be as large as needed.

Although support vector learning is carried out in all phases, the sizes of training datasets are much smaller than that of the whole training dataset and hence the time complexity can be reduced greatly. Experimental results show that, for a large-scale problem with up to tens of thousands of data, the size of training data in the SVM-GR algorithm is decreased to one tenth of the original size and hence the SVM-GR algorithm provides a much faster speed than the conventional SVMs. Experimental results also indicate that the SVM-GR algorithm produces a comparable generalization performance in comparison to the traditional SVMs.

Although both the proposed SVM-GR and the GSVM are based on the granular computing concept, the training for these two algorithms are quite different. Different from our paper, there are no filtering process and final training process in [20]. Since all data have to be used simultaneously to train SVMs, the speed of the GSVM [20] cannot be very fast.

Our paper is organized as follows. A brief review of the conventional theory will be described in Section 2. In Section 3, we propose the SVM-GR algorithm. Performance evaluation of the proposed SVM-GR is presented in Section 4. Finally, discussions and conclusions are given in Section 5.

II. SVMs for Classification

Given a training set $(\mathbf{x}_i, y_i)$, $i = 1, 2, ..., l$, where $\mathbf{x}_i \in R^n$ and $y_i \in \{-1, 1\}$, the traditional SVM algorithm is summarized as the following optimization problem:

$$\min_{w,b,\xi} \left\{ \frac{1}{2} w^T w + C \left(\sum_{i=1}^{l} \xi_i^2 \right) \right\}$$

$$subject\ to: \ y_i \left(w^T \phi(\mathbf{x}_i) + b \right) \geq 1 - \xi_i \tag{1}$$

where $\phi(\mathbf{x})$ is a nonlinear function that maps $\mathbf{x}$ into a higher dimensional space. w, b and ξ_i are the weight vector, bias and slack variable, respectively. C is a constant and determined *a priori*. Searching the optimal hyperplane in (1) is a QP problem, which can be solved by constructing a Lagrangian and transformed into dual

$$\max\ Q(\alpha) = \sum_{i=1}^{l} \alpha_i - \frac{1}{2} \sum_{i=1}^{l} \sum_{j=1}^{l} \alpha_i \alpha_j y_i y_j K(\mathbf{x}_i, \mathbf{x}_j)$$

$$subject\ to: \sum_{i=1}^{l} \alpha_i y_i = 0; \ 0 \leq \alpha_i \leq C, \ for\ i = 1, 2, ..., l \tag{2}$$

where $K(x_i, x_j) = \phi(x_i)^T \phi(x_j)$ is the kernel function, $\alpha = (\alpha_1, \alpha_2, ..., \alpha_l)$ is the vector of nonnegative Lagrange multipliers.

Suppose the optimum values of the Lagrange multipliers are denoted as $\alpha_{o,i}$ $(i = 1, 2, ..., l)$, we may determine the corresponding optimum value of the linear weight vector $\mathbf{w}_o$ and the optimal hyperplane as in (3) and (4), respectively:

$$\mathbf{w}_o = \sum_{i=1}^{l} \alpha_{o,i} y_i \phi(\mathbf{x}_i) \tag{3}$$

$$\sum_{i=1}^{l} \alpha_{o,i} y_i K(\mathbf{x}, \mathbf{x}_i) = 0 \tag{4}$$

and the decision function is

$$f(\mathbf{x}) = sign \left(\sum_{i=1}^{l} \alpha_{o,i} y_i K(\mathbf{x}_i, \mathbf{x}) + b \right) \tag{5}$$

III. The SVM-GR Algorithm

A. Data Granules

We use the interval analysis to form the data granules [12]. An interval vector $[\mathbf{b}_k]$ in R^n is defined as a Cartesian product of n intervals as follows:

$$[\mathbf{b}_k] = [b_{k,1}] \times [b_{k,2}] \times ... \times [b_{k,n}], \ k = 1, 2, ..., N \tag{6}$$

where $b_{k,i}$ are intervals, $[b_{k,i}] = [b_{k,i}^{min}, b_{k,i}^{max}]$. $1 \leq i \leq n$ and n is the dimensionality of the input space. The interval vectors in R^2 are showed in Figure 1.

Assume $\mathbf{X}$ is a measurable compact subset of the $n-$dimensional Euclidean space R^n and all training data $\mathbf{x}_i \in \mathbf{X}$. $\mathbf{X}$ can be represented as the union of interval vectors, i.e., $\mathbf{X} = [\mathbf{b}_1] \cup [\mathbf{b}_2] \cup ... \cup [\mathbf{b}_{l_1}]$, and $[\mathbf{b}_k] \cap [\mathbf{b}_j] = \phi$ for any $k \neq j$, $1 \leq k, j \leq l_1$, where l_1 is the total number of data granules. Then we define the data granules $(\mathbf{a}_k, t_k)$ as follows:

$$\mathbf{a}_k = mean\{x_i | x_i \in [\mathbf{b}_k]\} \tag{7}$$

$$t_k = sign \left(\sum_{x_i \in [\mathbf{b}_k]} y_i \right) \tag{8}$$

In our SVM-GR algorithm, we assume the widths of all intervals are same, i.e., $b_{k,i}^{max} - b_{k,i}^{min} = d$, where d is the parameter to determine granularity of data. Assume all $\mathbf{x}_i \in$

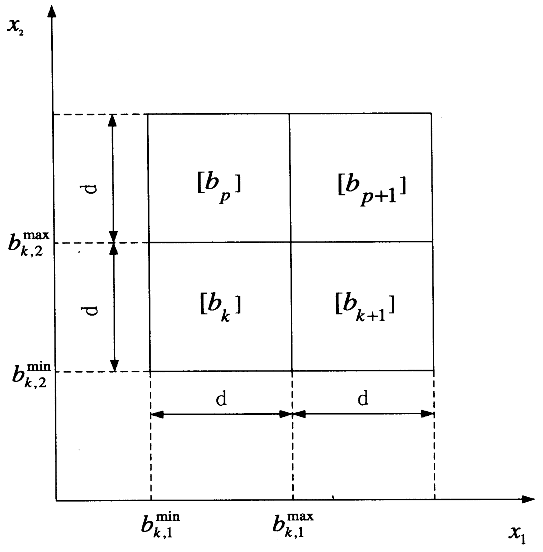

Fig. 1. Interval vectors in R^2. The widths of all intervals are same and equal to d.

$[d_{min}, d_{max}]^n$ and $d_{max} - d_{min} = r \cdot d$, then the interval vector $[\mathbf{b}_k]$ is formed as follows:

$$\begin{aligned}
&[\mathbf{b}_1] = [d_{min}, d_{min} + d]^n; \\
&[\mathbf{b}_2] = [d_{min}, d_{min} + 2d] \times [d_{min}, d_{min} + d]^{n-1}; \\
&\vdots \\
&[\mathbf{b}_k] = [d_{min} + s_1 \cdot d, d_{min} + (s_1 + 1) \cdot d] \times ... \\
&\times [d_{min} + s_n \cdot d, d_{min} + (s_n + 1) \cdot d]; \\
&\vdots \\
&[\mathbf{b}_{l_1}] = [d_{max} - d, d_{max}]^n,
\end{aligned} \tag{9}$$

where

$$\begin{aligned}
&s_n = (k-1) \ mod \ (r^{n-1}) + 1; \\
&s_{n-1} = \left[(k-1) - (s_n - 1) \cdot r^{n-1}\right] \ mod \ r^{n-2} + 1; \\
&\vdots \\
&s_m = \left[(k-1) - \textstyle\sum_{i=m+1}^{n} (s_i - 1) \cdot r^{i-1}\right] \ mod \ r^{m-1} + 1; \\
&\vdots \\
&s_1 = \left[(k-1) - \textstyle\sum_{i=2}^{n} (s_i - 1) \cdot r^{i-1}\right] + 1,
\end{aligned} \tag{10}$$

where $1 \leq m \leq n$.

B. The SVM-GR Algorithm

The SVM-GR algorithm involves two or more phases. In the first step, we carry out the data preprocessing to filter the samples roughly which have no contribution to the optimal hyperplane. The data granules are formed by using the interval analysis method. Then the non-empty data granules are used to carry out support vector learning. The parameter d controls the number of data granules. That is, if the d is large enough, meaning that the larger data granules are sought, the size of non-empty information granules will be much smaller than that of the whole training dataset and hence the computation is reduced. We only keep those support vector granules and filter the remainders. In the second step, we perform the support vector learning elaborately. We release the training samples from the support vector granules. These samples are used for finally support vector learning. Noticed that many training samples are filtered in the first step, hence the number of training samples for SVM in this phase is much smaller and the computation can be reduced.

If the dataset in the second step is still very large, we can form smaller granules within these large granules obtained in step 1 and filter non-support-vector granules again. The number of levels in this hierarchical granulation may be as large as needed. In this way, we can assure that the training dataset in each phase is not very large and hence the computation and storage difficulties can be solved.

The proposed SVM-GR algorithm can be summarized as follows:

Algorithm 1: Given a training set $\varphi_0 = \{(\mathbf{x}_i, y_i) | \mathbf{x}_i \in R^n, y_i \in \{-1, 1\}, i = 1, 2, ..., l\}$, we conduct the following:

1) Normalize x_i $(i = 1, 2, ..., l)$ to the range $[d_{min}, d_{max}]^n$;
2) Choose the interval vector $[\mathbf{b}_k]$ as (9) and (10), then form the data granules $(\mathbf{a}_k, t_k)$ as (7) and (8). Non-empty data granules form the new training dataset defined as $\varphi_1 = \{(\mathbf{a}_k, t_k) | \mathbf{a}_k \in [d_{min}, d_{max}]^n, t_k \in \{-1, 1\}, k = 1, 2, ..., l_1\}$.
3) Carry out support vector learning using the dataset φ_1. The support vector granules form a set Θ. Then release the training samples from Θ and form another training set defined as $\varphi_2 = \{(\mathbf{x}_i, y_i) | \mathbf{x}_i \in \mathbf{b}_k, \mathbf{a}_k \in \Theta\}$.
4) If the size of φ_2 is smaller than M_0 then turn to Step 2, otherwise turn to Step 5. Here M_0 is the maximum size of the training set that is determined *a priori*.
5) Carry out support vector learning using the dataset φ_2 and predict the test set.

IV. Experimental Results

In this section, we conducted experiments on some commonly used problems. For the classification problems, we conduct experiments on three data sets from UCI database [21] (satimage, letter and shuttle). The problem statistics are given in Table I.

TABLE I
PROBLEM STATISTICS

Problems	Training Data	Testing Data	Class	Attribute
Satimage	4435	2000	6	36
Letter	15000	5000	26	16
Shuttle	43500	14500	7	9

The performance of our SVM-GR algorithm was compared with conventional algorithms for SVMs. All the simulations were carried out in MATLAB 6.5 environment running in a Pentium 4, 2.4 GHZ CPU with 256 MB of RAM. We used functions in the LibSVM MATLAB interface [22] to design a traditional SVM that implements a simple decomposition method. In the SVM-GR, the training includes two phases and the support vector learning was carried out using the decomposition method in [22]. All training and testing data were linearly scaled to be in [0,1], respectively. We estimated the generalized accuracy using different $C = [2^{12}, 2^{11}, 2^{10}, ..., 2^{-2}]$ and $\gamma = [2^4, 2^3, 2^2, ..., 2^{-10}]$. For each pair of (C, γ), the validation performance was measured by training the entire training data and testing the validating data. Then we used the pair of (C, γ) that achieves the best validation accuracy to design a SVM and predict the test set. We used the same training, validating and testing data sets as in [8] so that we can compared our results with theirs.

The comparison of the test accuracy for the SVM-GR, the classical SVM and RSVM [8] are shown in Table II. For the RSVM algorithm, we selected the best results in [8] to compare with our results. Seen from Table II, the test accuracy for the SVM-GR algorithm is always higher than the RSVM and comparable to the classical SVMs.

TABLE II
TEST ACCURACY (%) FOR THE SVM-GR, THE CLASSICAL SVM AND RSVM

Problems	SVM-GR	Classical SVM	RSVM [8]
Satimage	90.05	91.85	90.0
Letter	97.88	97.90	95.9
Shuttle	99.84	99.92	99.81

Table III shows the size of training data, the size of support vectors, and computing time used in the SVM-GR and the traditional SVMs. For the SVM-GR algorithm, Table III shows the number of training samples and support vectors in the first and second training step. Note that the time reported in Table III is the total training and testing time for solving the optimal model. Since the training time may vary for different parameter sets, results here may not directly imply which method is faster. However, generally we can see for the problems with at least one thousand of data and the percentage of support vectors is not high, SVM-GR runs faster than the traditional SVM using a decomposition method. We note that the speeds for the SVM-GR algorithm on shuttle data set are about 5 times faster than the traditional SVM. The main reason behind this is that the percentage of support vectors in this case (about 0.01) is very low. We can remove a lot of training samples without decreasing the test accuracy much after the first training step in the SVM-GR algorithm. Therefore, the training data size in the second phase of the SVM-GR is much smaller than the total training set size. Note that we can control the training set size of the first phase in the SVM-GR by using small granularity parameter r. In this way, we can reduce lots of computational cost for the shuttle data set. This also imply that the proposed SVM-GR algorithm will be useful for large problems with a low percentage of support vectors.

V. CONCLUSIONS

In this paper, we proposed a novel learning algorithm called SVM-GR that solves the QP problem using granular computing method and makes prediction with SVMs. The main characteristic of SVM-GR is to break up the original support vector network into two or more smaller ones. Hence the original QP problem can be reduced to two or more smaller problems and the matrix Q are reduced from $l \times l$ to $(l_1 \times l_1 + l_2 \times l_2 + ... + l_p \times l_p)$, where l_1, l_2 and l_p are the sizes of training samples in the first, second and pth training phase. Compared with the the decomposition method for SVMs, which is notable for its fast speed and good generalization ability, the proposed SVM-GR algorithm shows a faster speed, especially for the large-scale problems. According to our results, for the problems with up to thousands of data, SVM-GR is several to tens times faster than the popular learning method for SVMs.

TABLE III
GENERALIZATION PERFORMANCE FOR THE SVM-GR AND THE CLASSICAL SVM ALGORITHMS ON CLASSIFICATION PROBLEMS. FOR THE SVM-GR, THE SIZE OF TRAINING DATA AND SUPPORT VECTORS IN THE FIRST AND SECOND TRAINING PHASE ARE DENOTED AS 'DATA1/DATA2' AND 'SVS1/SVS2', RESPECTIVELY. FOR THE TRADITIONAL SVM, THE SIZE OF TRAINING DATA AND SUPPORT VECTORS ARE DENOTED AS 'DATA' AND 'SVS', RESPECTIVELY.

Problems	Methods	Data Data1/Data2	SVs SVs1/SVs2	Time
Satimage	SVM-GR	3463/1868	1396/864	18.90
	SVM	4435	1613	25.19
Letter	SVM-GR	1474/14509	1324/7574	341.33
	SVM	15000	7742	438.40
Shuttle	SVM-GR	316/10258	116/135	9.84
	SVM	43500	278	48.06

Experiments on these classification problems demonstrate that the test accuracy of SVM-GR is always comparable to or a little lower than the traditional SVMs. We can expect this because in the SVM-GR algorithm, we filter some samples in the first training phase, thus we cannot ensure if some important support vectors are screened in advance and excluded in the final support vector learning. This seems to also imply that if the problems are not too large, the proposed SVM-GR method may achieve a lower test rate for classification problems in comparison to the classical SVMs. Compared with the RSVM, the proposed SVM-GR produced a higher test accuracy, which indicated that the SVM-GR used more useful and important support vectors for training in comparison to the RSVM algorithm.

The SVM-GR algorithm contains another new granularity parameter r. In our experiments, when $r = 1$, there will be no samples contained in the first training phase. However, if r is large enough, the training samples in the first phase become the whole dataset. Hence the SVM-GR can be reduced to a traditional SVM for these two cases. Experiments show that the SVM-GR with a small r may achieve a desirable test accuracy with a small training time.

References

[1] C. Burges, "A turtorial on support vector machines for pattern recognition," *Data Mining and Knowledge Discovery*, vol. 2, no. 2, pp. 121–167, 1998.

[2] C. Cortes and V. N. Vapnik, "Support vector networks," *Machine Learning*, vol. 20, pp. 273–297, 1995.

[3] V. N. Vapnik, *The Nature of Statistical Learning Theory*. New York: Springer-Verlag, 1995.

[4] B.Scholkopf, C. Burges, and A. Smola, *Advances in Kernel Methods: Support Vector Learning*. Cambridge, MA: MIT Press, 1999.

[5] L. P. Wang and X. Fu, *Data Mining with Computational Intelligence*. Berlin, Springer, 2005.

[6] L. P. Wang, ed., *Support Vector Machines: Theory and Applications*. Berlin, Springer, 2005.

[7] C.-F. Lin and S.-D. Wang, "Fuzzy support vector machines," *IEEE Trans. Neural Networks*, vol. 13, no. 2, pp. 464–471, 2002.

[8] K. Lin and C. Lin, "A study on reduced support vector machines," *IEEE Trans. Neural Networks*, vol. 14, no. 6, pp. 1449–1459, Nov. 2003.

[9] Y.-J. Lee and O. L. Mangasarian, "Rsvm: reduced support vector machines," *Proc. 1st SIAM Int. Conf. Data Mining*, 2001.

[10] S. Zheng, X.Lu, N. Zheng, and W. Xu, "Unsupervised clustering based reduced support vector machines," in *Proc. IEEE Conference on Acoustics, Speech, and Signal Processing*, pp. 821–824, 2003.

[11] J. C. Platt, "Fast training of support vector machines using sequential minimal optimazation," in *Advances in kernel Methods–Support Vector Learning*, 1998.

[12] A. Bargiela and W. Pedrycz, eds., *Granular computing: an introduction*. Kluwer Academic Publishers, 2003.

[13] T. Y. Lin, "Granular computing: Structures, representations, applications and future directions," in *Proceedings of 9th International Conference, RSFDGrC 2003, Chongqing, China, Lecture Notes on Artificial Intelligence LNAI 2639, Springer-Verlag*, p. 16, 2003.

[14] W. Pedrycz, "Granular computing: an introduction," in *IFSA World Congress and 20th NAFIPS International Conference*, pp. 1349–1354, 2001.

[15] Y. Y. Yao, "On modeling data mining with granular computing," in *25th Annual International Conference on Computer Software and Applications*, pp. 638–643, 2001.

[16] N. Karayiannis and P. Pai, "Fuzzy vector quantization algorithms and their application in image compression," *IEEE Trans. Image Processing*, vol. 4, pp. 1193–1201, 1995.

[17] K. Cios, W. Pedrycz, and R. Swiniarski, *Data Mining Techniques*. Kluwer Academic Publishers, Boston, 1998.

[18] S. Dick and A. Kandel, "Granular weights in a neural network," in *IFSA World Congress and 20th NAFIPS International Conference*, pp. 1708–1713, 2001.

[19] W. Pedrycz and G. Vukovich, "Granular neural networks," *Neurocomputing*, vol. 36, pp. 205–224, 2001.

[20] Y. Tang, B. Jin, Y. Sun, and Y. Q. Zhang, "Granular support vector machines for medical binary classification problems," in *Proceedings of the 2004 IEEE Symposium on Computational Intelligence in Bioinformatics and Computational Biology*, pp. 73–78, 2004.

[21] C. Blake and C. Merz, "Uci repository of machine learning databases," in *http://www.ics.uci.edu/ mlearn/MLRepository.html, Department of Information and Computer Science, University of California, Irvine, USA*, 1998.

[22] C.-C. Chang and C.-J. Lin, "LIBSVM: a library for support vector machines," in *Software available at* `http://www.csie.ntu.edu.tw/~cjlin/libsvm`, 2001.