Proceedings of the Third International Conference on Machine Learning and Cybernetics, Shanghai, 26-29 August 2004

COMPARISON OF PARALLEL AND CASCADE METHODS FOR TRAINING SUPPORT VECTOR MACHINES ON LAREGE-SCALE PROBLEMS

BAO-LIANG LU[1], KAI-AN WANG[1], YI-MIN WEN[1,2]

[1]Department of Computer Science & Engineering, Shanghai Jiao Tong University,
1954 Hua Shan Road, Shanghai, 200030, China
[2]Hunan Industry Polytechnic, Changsha, 400052, China
E-MAIL: blu@cs.sjtu.edu.cn, {kaianwang, wenyimin}@sjtu.edu.cn

Abstract:

We have proposed two different methods for training support vector machines (SVMs) on large-scale pattern classification problems, namely min-max-modular SVM (M3-SVM) and cascade SVM (C-SVM). For speeding up the training of SVMs with new computing infrastructure such as cluster and grid systems, both methods decompose a large-scale two-class problem to a number of relatively smaller two-class sub-problems which can be implemented in a parallel way, but they use different decomposition and combination strategies. In this paper, we conduct a comprehensive investigation in the two methods to compare their generalization performance and training time. Our experiments show that M3-SVM needs shorter training time, but has a little lower generalization performance than the standard SVM and Cascade SVM. The experiments also indicate that cascade SVM has the least number of support vectors among these three SVMs.

Keywords:

Support vector machines; parallel learning; problem decomposition; module combination; text categorization

1. Introduction

In recent years, support vector machines (SVMs) [1], [2] have been applied to various pattern classification problems, such as handwritten digit recognition [1], text classification [3] and face detection [4]. However, how to efficiently train support vector machines on problems with a large training data set is still an ongoing research issue.

Given l training data $\{(x_1, y_1), \ldots, (x_l, y_l)\}$, where $x_i \in R^n$, $y_i \in \{-1, 1\}$, $i=1, \ldots, l$, the SVM solves the following quadratic programming problem[2]:

$$\begin{aligned} \max_{\alpha} \quad & \tfrac{1}{2}\alpha^T Q\alpha - e^T\alpha \\ \textit{subject to} \quad & y^T\alpha = 0 \\ & \alpha_i \geq 0, \quad i = 1,\ldots\ldots,l. \end{aligned} \tag{1}$$

where e is the vector of all ones, Q is an l by l matrix, $Q_{ij} = y_i y_j K(x_i, x_j)$, and $K(x_i, x_j)$ is the kernel function.

For large-scale problems, the difficulties of solving (1) result from the matrix Q, which cannot be saved in the main memory, so traditional optimization algorithm like Newton method can not be directly used. Currently one major method for dealing with this problem is to decompose the quadratic programming problem to a sequence of smaller-sized quadratic programming problems which are solved sequentially [5], [6]. However, for huge problems, especially those with a large number of support vectors, this method needs a large number of iterations and suffers from slow convergence [7], [8].

In many real-world applications such as text categorization and geography information classification, however, the size of training data sets are usually huge. For example, the Yomiuri News Corpus used in this paper contains 2,190,512 documents collected from Yomiuri Newspapers dated from 1987-2001, with 75 unique categories assigned to those documents [9].

For dealing with large-scale problems, two efficient training methods [8], [9] were proposed in our previous work. Both methods decompose a large-scale problem to a number of relatively smaller sub-problems, so we can solve these sub-problems in a parallel way by using new computing infrastructure such as cluster and grid systems [10] to speed up the training process.

In this paper, we conduct a serious and comprehensive investigation in the two methods to compare their generalization performance with the regular SVM. Moreover, as the training efficiency is the main motivation of developing M^3-SVMs and C-SVMs, we would like to compare their training time with the standard SVM.

This paper is organized as follows. In Section 2, we introduce the M^3-SVM reported by [9]. In Section 3, we outline the cascade SVM proposed by [8]. Implementation issues are described in Section 4. In Section 5, we perform a comparative study on a large-scale text categorization problem.

Conclusions are outlined in Section 6.

2. M³-Support Vector Machine

We have combined the min-max modular Neural Network [11], [12] with SVMs and proposed a min-max modular SVM (M³-SVM) [9]. The method decomposes the training data set for a large-scale problem to a series of smaller sub-problems, which are independent from each other in training phase. As a result, these sub-problems can be learned parallelly. After training, all the trained SVMs are integrated into a M³-SVM following two module combination principles.

2.1. Task decomposition

Let χ^+ and χ^- be the given positive and negative training data set for a two-class problem T,

$$\chi^+ = \{(x_i^+, +1)\}_{i=1}^{l^+} \quad (2)$$

$$\chi^- = \{(x_i^-, -1)\}_{i=1}^{l^-} \quad (3)$$

Where $x_i \in R^n$ is the input vector, l^+ and l^- are the total numbers of positive training data and negative training data, respectively.

According to [9], χ^+ can be randomly partitioned into N^+ subsets in the form,

$$\chi_j^+ = \{(x_i^{+j}, +1)\}_{i=1}^{l_j^+} \quad (4)$$

$$\text{for } j = 1, \ldots, N^+$$

where $\cup_{j=1}^{N^+} \chi_j^+ = \chi^+, 1 \le N^+ \le l^+$ and $\chi_i^+ \cap \chi_j^+ = \phi, i \ne j$. When $N^+ = l^+$, each subset only contains one training sample.

Also, χ^- can be randomly partitioned into N^- subsets in the form,

$$\chi_j^- = \{(x_i^{-j}, -1)\}_{i=1}^{l_j^-} \quad (5)$$

$$\text{for } j = 1, \ldots, N^-$$

where $\cup_{j=1}^{N^-} \chi_j^- = \chi^-, 1 \le N^- \le l^-$ and $\chi_i^- \cap \chi_j^- = \phi, \ i \ne j$.

In practical applications of SVMs, an appropriate value of N^+ and N^- might depend on two main factors, such as the number of training data belonging to each class and the available computational resource [9].

After decomposing the positive training data set χ^+ and negative training data set χ^-, the original two-class problem T can be divided into $N^+ \times N^-$ relatively smaller and more balanced two-class sub-problems $T_{i,j}$ as follows:

$$(T_{i,j})^+ = \chi_i^+ \text{ and } (T_{i,j})^- = \chi_j^- \quad (6)$$

where $(T_{i,j})^+$ and $(T_{i,j})^-$ denote the positive training data set and the negative training data set of $T_{i,j}$, respectively.

According to above discussions, we know that the task decomposition method is simple and straightforward, and neither domain specialists nor prior knowledge of the problem is required.

After task decomposition, each of the two-class sub-problems can be treated as a completely independent, non-communicating problem in learning phase [9], [12]. Therefore, all the two-class sub-problems can be efficiently learned in a massively parallel way.

2.2. Combination method

After training individual SVMs assigned to learn associated two-class sub-problems, all the trained SVMs are integrated into M³-SVM with the MIN and MAX units according to two module combination laws, namely the minimization and maximization principle [9], [12].

2.2.1. Min unit and Max unit

We introduce two integrating units, namely MIN and MAX units, respectively, which are the elements for connecting individual trained SVMs [12].

The basic function of a MIN unit is to find a minimum value from its multiple inputs. The transfer function of the MIN unit is given by

$$q = \min(p_1, p_2, \ldots, p_n) \quad (7)$$

where $p_1, p_2, \ldots, p_n$ are the inputs and q is the output which is the smallest one among the inputs.

The basic function of a MAX unit is to find a maximum value from its multiple inputs. The transfer function of the MAX unit is given by

$$q = \max(p_1, p_2, \ldots, p_n) \quad (8)$$

where $p_1, p_2, \ldots, p_n$ are the inputs and q is the output which is the largest one among the inputs.

2.2.2. Module combination laws

The $N^+ \times N^-$ trained SVMs will be integrated into a M³-SVM according to the following two module combination principles [9], [12]:

Minimization Principle: The SVMs, which are trained on the data sets which contain the same positive training data $(T_{i,j})^+$, should be integrated by the MIN unit.

Maximization Principle: The SVMs, which are trained

on the data sets which have the same negative training data $(T_{i,j})^-$, should be integrated by the MAX unit

According to the above two principles, the $N^+ \times N^-$ smaller SVMs are integrated first with N^+ MIN units and one MAX unit as illustrated in Figure 1,

$$T_i(x) = \min_{j=1}^{N^-}(T_{i,j}(x)) \quad \text{for} \quad i = 1,...,N^+ \tag{9}$$

$$T(x) = \max_{i=1}^{N^+} T_i(x) \tag{10}$$

where $T_{i,j}(x)$ denotes the transfer function of the trained SVM corresponding to the two-class sub-problem $T_{i,j}$ and $T_i(x)$denotes the transfer function of a combination of N^- SVMs integrated by the MIN unit.

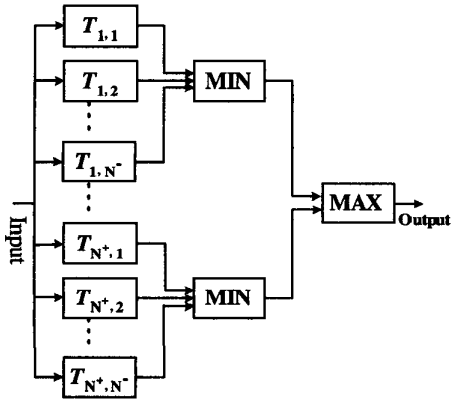

Figure 1. Illustration of the integration of $N^+ \times N^-$ trained SVMs with N^+ MIN units and one MAX unit in M^3-SVM

3. A cascade method for training SVM

Recently, we have proposed a cascade training method of SVMs based on the essence of support vector (SVs) [8]. It is reported that the method not only speeds up the training of SVMs but also reduces the number of SVs.

Differing from M^3-SVM, which has only one training phase, the cascade training method has threes steps of training phase as illustrated in Figure 2.

First, the positive and negative training data sets of the original two-class problem T, χ^+ and χ^-, are divided into two subsets by the same ratio r (0<r<1) respectively as follows,

$$\chi_1^+ = \{(x_i^+, +1)\}_{i=1}^{l_1^+}, \quad \chi_2^+ = \{(x_i^+, +1)\}_{i=1}^{l_2^+},$$

$$\chi_1^- = \{(x_i^-, -1)\}_{i=1}^{l_1^-}, \text{ and } \chi_2^- = \{(x_i^-, -1)\}_{i=1}^{l_2^-} \tag{11}$$

where $l_1^+ = \left| r * l^+ \right|$, $l_2^+ = l^+ - l_1^+$ and $l_1^- = \left| r * l^- \right|$, $l_2^- = l^- - l_1^-$.

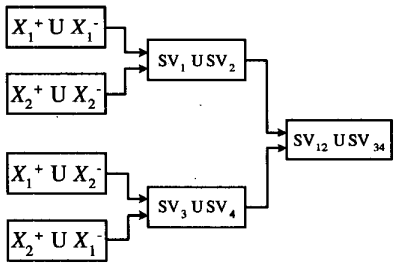

Figure 2. Illustration of the process of obtaining a set of support vectors as training data from the original training data set by using a cascade-parallel way. Note that each frame denotes a problem to be trained by an SVM

Therefore, the original two-class problem T is divided into four two-class sub-problems in the bottom layer as follows,

$$T_1 = \chi_1^+ \cup \chi_1^-, \quad T_2 = \chi_2^+ \cup \chi_2^-,$$
$$T_3 = \chi_1^+ \cup \chi_2^-, \text{ and } T_4 = \chi_2^+ \cup \chi_1^- \tag{12}$$

Apparently, the four sub-problems are independent from each other, so the four two-class sub-problems can be learned by standard SVM in a parallel way. After the first training step, four sets of support vectors, SV_1, SV_2, SV_3, and SV_4, are obtained from SVM_1, SVM_2, SVM_3, and SVM_4, which correspond to sub-problems T_1, T_2, T_3, and T_4 respectively.

Second, we construct two training data sets from the SV_1, SV_2, SV_3, and SV_4, as follows,

$$T_{12} = SV_1 \cup SV_2$$
$$T_{34} = SV_3 \cup SV_4 \tag{13}$$

After collecting T_{12} and T_{34}, all the sets of support vectors, SV_1, SV_2, SV_3, and SV_4, can be discarded, that is, all the SVMs, SVM_1, SVM_2, SVM_3, and SVM_4, can be discarded. The reason is that the SVMs generated in the first step are just used to filter out the non-support-vector data.

Since the two problems defined by (13) are independent from each other, the two problems can be parallelly learned. After the second training step, another two new sets of support vectors, SV_{12} and SV_{34}, are produced.

Finally, we carry out a union operation between the two sets of support vectors, SV_{12} and SV_{34}, to construct a training data set corresponding to the original two-class problem as follows,

$$T = SV_{12} \cup SV_{34} \tag{14}$$

It should be noted that the two sets of support vectors, SV_{12} and SV_{34}, may contain the same support vectors. Following the three steps mentioned above, each of the sub- problems defined by (12) can be furthered decomposed into four sub-problems in the same way if they are too large to solve [8].

4. Implementation issues

Although we have described the two parallel SVMs in detail, there are still some issues need to be clarified in practical implementation such as the determination of the number of the sub-problems in M^3-SVM and the approaches for the multi-class problems.

4.1. Determination of N^+ and N^- in M^3-SVM

Paper [9] suggested a method which N^+ may be different from N^- as follows,

Step 1: Set the desired number of the training data for two-class sub-problems, p.

Step 2: Determine the number of sub training data sets N^+ and N^-, as follows,

$$N^{\kappa} = \begin{cases} \left\lfloor \frac{2l^{\kappa}}{p} \right\rfloor & 2l^{\kappa} > p \\ 1 & \text{otherwise} \end{cases} \tag{15}$$

$$\text{for } \kappa = \{+,-\}$$

Using this method, the number of training data for each of sub-problems is about:

$$\left\lfloor \frac{l^+}{N^+} \right\rfloor + \left\lfloor \frac{l^-}{N^-} \right\rfloor \tag{16}$$

The feature of this method is that N^+ and N^- are determined by the size of positive training data set and negative training data set except for p, the larger the data set is, the more sub-sets will be decomposed to, so the sub-problems may be more balanced.

In addition to this method, we implemented another method which just simply divides the positive and negative training data sets to two data sub-sets as C-SVM does, because we would like to use the same condition to compare M^3-SVM with C-SVM. By using this method, the number of training data for each of the sub-problems is about:

$$\left\lfloor \frac{l^+}{2} \right\rfloor + \left\lfloor \frac{l^-}{2} \right\rfloor \tag{17}$$

4.2. A Revised version of C-SVM

We notice that the first step in C-SVM filters out most of the non-support-vector data, while the second step only filter out a little proportion of the data.

Therefore, in this paper, we modify the original C-SVM with three main training steps to a method with only two training steps for further speeding up the training of C-SVMs. We combine the last two training steps of C-SVM as illustrated in Figure 3. That is, we merge directly the SV sets, SV_1, SV_2, SV_3, and SV_4, to construct a final training data set corresponding to the original two-class problems as follows,

$$T = SV_1 \cup SV_2 \cup SV_3 \cup SV_4 \tag{18}$$

As a result, we only need to train a SVM on the training data set. We name this revised method as C-SVM* for short. We implement both C-SVM and C-SVM* and compare their training time, testing time and generalization performance.

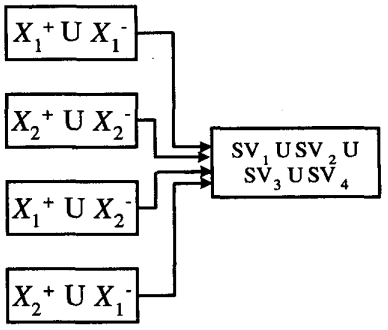

Figure 3. Illustration of the process of obtaining a set of support vectors in C-SVM*

4.3. Multi-class M^3-SVM and C-SVM

Although SVM was originally designed for two-class classification problems, several methods have been proposed to solve multi-class classification. A major method is to decompose the multi-class problem to a number of two-class problems which are learned independently [13]. The method in general can also be applied to the two parallel SVM.

There are two popular decomposition approaches: one-against-one and one-against-the-rest [13], in which a K-class problem is decomposed to K two-class sub-problems and K(K-1)/2 two-class sub-problems, respectively. One-against-one performs better than one-against-the-rest in practice [13], so we use for our implementation here. In testing phase, the K(K-1)/2 sub-problems are integrated with voting strategy [13].

We have known that one-against-one method

decomposes a K-class problem to K(K-1)/2 two-class problems, and a two-class problem can be further divided into $N^+ \times N^-$ two-class sub-problems by using part-versus-part method [9]. Therefore the total number of sub-problems in M3-SVM, is,

$$\sum_{i=1}^{K-1}\sum_{j=i+1}^{K} N_i^+ \times N_j^- \qquad (19)$$

In C-SVM, the numbers of sub-problems for a K-class problem in each step can be expressed as $K(K\text{-}1)/2\times 4$ for the first step; $K(K\text{-}1)/2\times 2$ for the second step; $K(K\text{-}1)/2$ for the third step.

5. Experiments and results

In this section we present experimental results on a text categorization problem to compare M^3-SVMs and C-SVMs with the standard SVMs. We use Yomiuri News Corpus for this study.

Table 1
Distributions of training and test data of the top ten classes in a subset of Yomiuri News Corpus

#	Category	#data	
		Training	Test
C_1	Crime News	103607	24374
C_2	Sport News	79726	17610
C_3	Asian-Pacific News	41374	5943
C_4	North-South American News.	36275	6109
C_5	Health News	35932	7004
C_6	Accident News	34044	8483
C_7	By-time News	33590	7702
C_8	Society News	31790	6830
C_9	Finance News	27972	3213
C_{10}	Politics News	27282	4465
	Total	451592	91733

There are 2,190,512 documents in the full collection from the years 1987 to 2001. We used 913,118 documents dated 1996-2000 as a training set and 181,863 documents dated July-December of 2001 as a test set in this study. In the simulations, we selected the top ten classes as shown in Table 1. A χ^2 statistic (CHI) feature selection method was used for preprocessing the documents after the morphological analysis was performed with ChaSen. The number of features is 5,000. In simulations, C and r of Gaussian kernel are selected as 64 and 2 for training all the methods. All of the simulations were performed on a 2.5G Pentium 4 PC with 500M RAM.

To compare the performance of the standard SVM, M^{3}SVM, C-SVM, and C-SVM*, We perform six experiments, A_1, A_2, A_3, A_4, A_5 and A_6, on the ten-class text categorization problem as shown in Table 1. A_1 is conducted for the standard SVM. In A_2, A_3, and A_4, the text categorization was learned by M^3-SVMs whose detail information is shown on Table 2. In A_2 and A_3, N^+ and N^- is calculated by using (15), however, the decomposition method used in A_4 is just simply to divide the positive and negative training data set to two data sub-sets directly, that is, the values of N^+ and N^- were set to two. A_5 and A_6 are trained by C-SVM and C-SVM*, respectively.

Table 2
Three different ways of dividing the training data set of the text categorization problem used in M^3-SVM

#	p	Number of subsets for each class			
		N_1	N_2	N_3	others
A_2	90000	2	2	1	1
A_3	70000	3	2	2	1
A_4	---	2	2	2	2

Table 3
Performance the four methods

Method	#	train		accuracy (%)	#SV
		time (h.)	Speed up		
SVM	A_1	6.74	---	82.09	491166
M^3-SVM	A_2	2.91	2.32	81.82	614766
	A_3	2.75	2.45	81.47	663810
	A_4	1.61	4.19	81.44	1059987
C-SVM	A_5	4.83	1.40	82.05	479201
C-SVM*	A_6	3.79	1.78	82.02	483201

Table 3 presents the results of the six experiments. From Table 3, we can see that M^3-SVM, C-SVM and C-SVM* are faster than the standard SVM in a parallel way, and still can remain almost the same generalization performance. M^3-SVM in A_4 is most fast, and M^3-SVM can achieve more higher training speed if the original problem is decomposed to much more sub-problems, however, it may sacrifice generalization performance. The faster M^3-SVM trains, the lower generalization it achieves, so one can choose the number of sub-problems according to his preference, training time or test accuracy.

Table 3 also shows that both C-SVM and C-SVM* have smaller number of support vectors than the standard SVM, while M^3-SVM has far more number of support vectors in comparison with the standard SVM. Although the revised method C-SVM* is faster than C-SVM, it has more number of support vectors. So C-SVM has the least

number of support vectors. Because test time controlled by the number of support vectors, the smaller the number of SVs is, the shorter the test time is, so C-SVM may be the fastest in test phase among these four methods.

It should be noted that M^3-SVM in A_4 shares the same sub-problems with C-SVM and C-SVM* in the bottom layer, however, the three methods have one, two and three training steps respectively. With different steps of training, different generalization performance, training time and support vectors can be achieved. C-SVM and C-SVM* with more training steps is slower in training, but get higher performance and smaller number of support vectors. On the contrary, M^3SVM in A_4 is faster, but achieve a little lower test accuracy and more number of support vectors.

6. Conclusions and discussions

In this paper, we have investigated the performance, including training time, generalization capability and the number of support vectors, of the four methods, namely SVM, M^3-SVM, C-SVM, and C-SVM*. We draw the following conclusions:

a) As reported by [12], the advantages of M^3-SVMs are its parallelism and scalability. One can decompose a large-scale problem to a number of sub-problems as small as needed. The method is the fastest among the four approaches, however, it may have a little lower generalization performance, especially when the original problem is divided into too many sub-problems.

b) C-SVM is faster than SVM, but slower than M^3-SVM, however, the merit of this approach is that it gets the smallest number of support vectors among these four methods after training. Therefore, C-SVM may be the fastest in the test phase.

Acknowledgements

The authors would like to thank Drs. H. Isahara and M. Utiyama for providing the Yomiuri News Corpus and performing morphological analysis. The authors appreciate Ms. Hong Shen for the help on processing the training and test data sets. This work was supported in part by the National Natural Science Foundation of China via the grant NSFC 60375022.

References

[1] C. Cortes and V. Vapnik, "Support-Vector Networks", *Machine Learning*, vol. 20, pp. 273-297, 1995.

[2] V. Vapnik, *Statistical Learning Theory*, John Wiley and Sons, New York, 1998.

[3] T. Joachims. "Text categorization with support vector machine: learning with many relevant features", *Proceeding of 10th European Conference on Machine Learning*, pp. 137-142, 1998.

[4] E. Osuna, R. Freund and R. Girosi, "Training support vector machines: An application to face detection", *Proceeding of the 1997 conference on Computer Vision and Pattern Recognition (CVPR'97)*, Puerto Rico, pp. 130-136, 1997.

[5] T. Joachims, "Making large-scale SVM learning practical", In B. Scholkopf, C. J. C. Burges, and A. J. Smola eds. *Advances in Kernel Methods – Support Vector Learning*, Cambridge, MA: MIT press, 1998.

[6] J. C. Platt, "Fast training of support vector machines using sequential minimal optimization", In B. Scholkopf, C. J. C. Burges, and A. J. Smola eds. *Advances in Kernel Methods – Support Vector Learning*, Cambridge, MA: MIT press, 1998.

[7] K. M. Lin and C. J. Lin, "A study on reduced support vector machines", *IEEE Transaction on Neural Networks*, vol.14, pp. 1449-1469, 2003.

[8] Y. M. Wen and B. L. Lu, "A cascade method for reducing training time and the number of support vectors", *Proceeding of International Symposium on Neural Network (ISNN2004)*, Dalian, Aug. 15-19,2004 (to appear).

[9] B. L. Lu, K. A. Wang, M. Utiyama and H. Isahara, "A part-versus-part method for massively parallel training of support vector machines", *Proceeding of IJCNN*, Budapast, Jul. 25-29, 2004 (to appear).

[10] I. Foster and C. Kesselman, "The Grid: Blueprint for a New Computing Infrastructure", Morgan Kaufmann, San Francisco, 1998.

[11] B. L. Lu and M. Ito, "Task decomposition based on class relations: a modular neural network architecture for pattern classification", In J. Mira, R. Moreno-Diaz and J. Cabestany eds. *Biological and Artificial Computation: From Neuroscience to Technology, Lecture Notes in Computer Science*, vol. 1240, pp. 330-339, Springer, 1997.

[12] B. L. Lu and M. Ito, "Task decomposition and module combination based on class relations: a modular neural network for pattern classification", *IEEE Transaction on Neural Networks*, vol. 10, No. 5, pp. 1244-1256, 1999.

[13] C. W. Hsu and C. J. Lin, "A comparison of methods for multi-class support vector machines", *IEEE Transaction on Neural Networks*, vol. 13, No. 2, pp. 415-425,2002.