A Parallel Training Algorithm of Support Vector Machines Based on the MTC Architecture

Lei Wang, Huading Jia
School of Economics Information Engineering
Southwest University of Finance and Economics
Chengdu, China
{wanglei_t, jhd}@swufe.edu.cn

***Abstract*—For accelerating the training speed of support vector machines (SVM), a novel "multi-trifurcate cascade (MTC)" architecture was proposed in this paper, which held the advantages of fast feedback, high utilization rate of nodes, and more feedback support vectors. Then, a parallel algorithm for training SVM was designed based on the MTC architecture, and it was proven to converge to the optimal solution strictly. The experimental results showed that the proposed algorithm obtained very high speedup and efficiency, and needed significantly less training time than the Cascade SVM algorithm.**

I. Introduction

Support vector machines are modern learning systems that deliver state-of-the-art performance in real world pattern recognition applications, such as text categorization, handwritten character recognition etc., which put their basis on the statistics learning theory (STL) [1]. However, training SVM requires resolving quadratic programming under constraints of inequality, which results in calculation and storage difficulties when the number of samples gets larger.

Recently, there were some works on developing parallel implementation for training SVM, and they were proved more suitable and efficient than chunking and decomposition approaches, especially on large-scale datasets [2]-[6]. Zanghirati and Zanni proposed a parallel implementation of SVM$^{\text{light}}$, in which the quadratic programming problem was split into smaller subproblems and then solved by a variable projection method [5]. Cao et al. found that over 90% of the total computational time of the sequential SMO algorithm was used for updating F_i array and calculating b_{up} and b_{low} in each step, so they parallelized these tasks using multiple CPU processors and obtained a great speedup than sequential SMO on different datasets [4].

Different with above methods, several researchers investigated another way to parallelize SVM, they trained multiple SVM classifiers using subsets of training samples and then combined them by a multilayer perceptron or another SVM. Unfortunately, such methods cannot guarantee to converge to the global optimal solution of SVM [2][3], in particular to high-dimension problems. More recently, Graf et al. utilized a special binary cascade architecture to train SVM on multiple processors, and corresponding parallel algorithm is called Cascade SVM. They proved that the global optimal solution can be guaranteed with several passes through such architecture.

In this paper, we propose a novel "multi-trifurcate cascade (MTC)" architecture, which holds the superiorities of fast feedback, high utilization rate of nodes, and more feedback support vectors than the binary cascade one. Then, we develop a MTC-SVM algorithm for training SVM in parallel based on the MTC, and prove its convergence to the global optimum. Finally, we run the proposed algorithm on at most 27 processors simultaneously under different large-scale datasets, the experimental results show that the obtained speedup and efficiency are both high, and the training speed is much faster than Graf's Cascade SVM.

II. Brief review of Cascade SVM

The binary cascade architecture is developed based on decomposing the problem into a number of independent smaller subproblems, and the partial results are combined pairwisely in later layers in a hierarchical fasion [6].

As shown in Fig. 1, the original training dataset S is decomposed into p disjoint subsets $S_1, S_2, \cdots, S_p$ in the first layer, and they are distributed to p computational nodes (CPU processors) to train p local SVM classifiers in parallel. Then, in the next layer, sets of support vectors from two SVMs are combined pairwisely into $p/2$ training subsets, and $p/2$ local SVMs are learned on them. This continues until only one SVM is left in the last layer. But if such SVM can't reach to the global optimum, its support vectors should be fed back into the first layer and a new pass through this cascade architecture is needed.

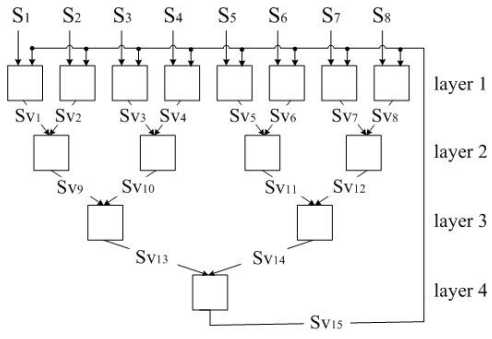

Fig. 1. The binary cascade architecture ($p = 2^3$)

According to the comments of Graf, the Cascade SVM

algorithm will converge to the global optimum within several passes for most classification tasks.

However, we analyze that there are two main deficiencies of the binary cascade architecture, as follows.

(1) Feedback of support vectors in the last layer is too slow.

we can easily verify that the binary cascade architecture has $k+1$ layers if the number of nodes is $p = 2^k$. That is to say, at least one node has to train $k+1$ local SVMs before it receives the feedback from previous pass. Obviously, if we can reduce the layers of the cascade architecture, the support vectors will be fed back more early and the algorithm will converge more quickly.

(2) Utilization ratio of computational nodes is too low.

Take Fig. 1 for example, all the 8 nodes need to train local SVM in the first layer, but the number reduces to 4 in the second layer, while 2 and 1 in the last two layers. Hence, many nodes are idle during a pass, which causes severe computational inefficiency.

In next section, we propose a promising multi-trifurcate cascade architecture to ease above deficiencies, and based on which develop an efficient parallel algorithm for training SVM.

III. The Multi-Trifurcate Cascade Architecture

For accelerating the feedback of support vectors, we adopt a trifurcate cascade architecture, as Fig. 2.

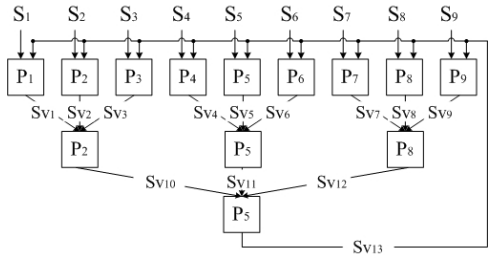

Fig. 2. The trifurcate cascade architecture ($p = 3^2$)

It's easy to observe from Fig. 2 that: (1) In the first layer, the original training dataset S is decomposed into $p = 3^2$ disjoint subsets and distributed to computational nodes $P_1, P_2, \cdots, P_p$ to train p local SVMs. (2) In the second layer, the nodes P_2, P_5, P_8 assemble their new training subsets respectively by collecting support vectors from their two adjacent nodes, together with their local sets of support vectors, and then $p/3$ local SVMs are trained on these new subsets. (3) In the third layer, the node P_5 assembles its new training subset to train the final SVM by collecting support vectors from nodes P_2 and P_8 together with its local support vectors. (4) If the final SVM doesn't reach to global optimum, its support vectors are fed back into all the nodes in the first layer, and then the next pass starts.

Obviously, each nodes needs to train at most 3 local SVMs before receives the feedback support vectors from node P_5 in Fig. 2, while the number is 4 in Fig. 1. Therefore, the trifurcate cascade architecture holds fast feedback speed than the binary one.

However, we notice that the node $P_1, P_3, P_4, P_6, P_7, P_9$ are all idle in the second and the third layers, which causes severe computational inefficiency. Here, we let these nodes collect their adjacent sets of support vectors and train local SVMs in the second and the third layer, according to the other two isomorphic trifurcate cascade architectures in Fig. 3. Further, we are easy to analyze that the newly added architectures won't disturb existing computational tasks in nodes P_2, P_5, P_8, but just make full use of the idle nodes $P_1, P_3, P_4, P_6, P_7, P_9$, namely such three architectures are independent with each other during a pass.

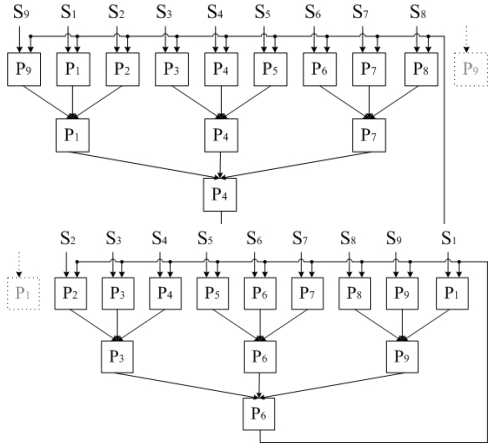

Fig. 3. The other two isomorphic trifurcate cascade architectures ($p = 3^2$)

Thus, we obtain the proposed "multi-trifurcate cascade architecture (MTC)" in this paper immediately after superposing above three isomorphic and independent trifurcate cascade architectures in Fig. 2 and Fig. 3. Furthermore, we can easily extend the MTC architecture to more general cases where $p = 3^k, k = 2, 3, 4, \cdots$, and find that it obeys the following rules in the j-th pass:

(1) In the first layer, the training subset of each node P_i is the union of S_i and the feedback set of support vectors $\mathcal{U}^j$ ($\mathcal{U}^0 = \emptyset$ in the first pass).

(2) In the r-th layer ($1 < r \leq k$), the training subset of each node P_i is composed of the sets of support vectors from P_{z1} and P_{z2}, as well as its local set, where $z1 = (i - 3^{r-2}) \bmod p$ and $z2 = (i + 3^{r-2}) \bmod p$.

(3) In the $(k+1)$-th layer, only $p/3$ nodes need to train local SVMs, with subscripts satisfying $p/3 < i \leq 2p/3$. Similarly, the each P_i builds its training subset in the same way as in rule (2), just with $z1 = (i - 3^{k-1}) \bmod p$ and $z2 = (i + 3^{k-1}) \bmod p$. Then, the feedback set $\mathcal{U}^{j+1}$ is the union of the sets of support vectors from such $p/3$ nodes.

Hence the MTC is equivalent to the superposition of $p/3$ isomorphic trifurcate cascade architectures. Compared with Graf's binary cascade architecture, it holds three main advantages as follows.

(1) Faster feedback speed.

Since trifurcate cascade architectures are independent with each other, the needed training time of MTC within a pass is

comparable with that of a single trifurcate cascade architecture. Hence, the feedback speed of MTC is faster than Graf's binary cascade architecture.

(2) Higher utilization ratio of computational nodes.

Except for only $p/3$ nodes in the last layer (see also P_4, P_5, P_6 in Fig. 2 and Fig. 3), all nodes in MTC have participated in training local SVMs. Hence, the utilization ratio of MTC is much higher.

(3) More feedback support vectors.

Since the feedback set in MTC is composed of $p/3$ sets of support vectors in the last layer, while that in binary cascade architecture is composed of only one set of support vectors, hence more support vectors can be fed back into next pass. According to the comments of Graf [6], this means faster convergence of parallel training process.

IV. Training SVMs based on the MTC

A. The proposed MTC-SVM algorithm

Based on the proposed MTC architecture, we develop a parallel algorithm for training SVM efficiently, which is called "MTC-SVM" as follows.

Initialization: Divide the training set S into p equivalent subsets $S_1, S_2, \cdots, S_p$ with $p = 3^k$, and distribute them to p nodes respectively. Let $\mathcal{U}^0 = \emptyset$ and $t = 1$.

Step 1: Each node $P_i (1 \le i \le p)$ prepares its current training subset S_i^t as the following way:

(a) If $t \bmod (k+1) = 1$, S_i^t is the union of S_i and the feedback set $\mathcal{U}^{t-1}$.

(b) If $t \bmod (k+1) = r$ and $1 < r \le k$, the current subset S_i^t is equivalent to $S_i^t = Sv_i^{t-1} \bigcup Sv_{z1}^{t-1} \bigcup Sv_{z2}^{t-1}$. Here, Sv_j^{t-1} is the set containing all support vectors of S_j^{t-1}, and subscript j $(= z1, z2)$ satisfies $z1 = (i - 3^{r-2}) \bmod p$ and $z2 = (i + 3^{r-2}) \bmod p$.

(c) If $t \bmod (k+1) = 0$, only nodes belonging to $\{P_i \mid p/3 < i \le 2p/3\}$ need to prepare their current training subsets. Here, S_i^t is equivalent to $S_i^t = Sv_i^{t-1} \bigcup Sv_{z1}^{t-1} \bigcup Sv_{z2}^{t-1}$, and subscripts $z1, z2$ satisfy $z1 = (i - 3^{k-1}) \bmod p$ and $z2 = (i + 3^{k-1}) \bmod p$.

Step 2: Each node P_i trains local SVM on subset S_i^t and gains current set of support vectors Sv_i^t.

Step 3: If $t \bmod (k+1) = 1$ and $Sv_i^t \subset \mathcal{U}^{t-1}$ are satisfied for each P_i, algorithm stops.

Step 4: If $t \bmod (k+1) = 0$, set $\mathcal{U}^t = \bigcup Sv_i^t$ with $p/3 < i \le 2p/3$.

Step 5: Let $t = t + 1$, return to *step 1*.

B. Analysis of Convergence

The following theorems show that MTC-SVM converges to the global optimum within finite steps. Let T denote a subset of training set S, $W(T)$ is the optimal (or minimal) objective function of SVM over T, and let $Sv(T) \subset T$ be the subset that contains all the support vectors of T.

Since the solution of SVM is determined only by support vectors [1], we have $W(Sv(T)) = W(T)$. Furthermore, let $Sv(S)$ be set of support vectors when SVM reaches to the global optimal solution on S, we have $W(Sv(S)) = W(S)$. Thus, we can easily deduce the following equation for $\forall T \subset S$:

$$W(T) = W(Sv(T)) \ge W(Sv(S)) = W(S) \tag{1}$$

Let $\mathcal{G}$ represent a family of subsets of S. The set $T_{\mathcal{G}}^*$ that achieves the smallest $W(T)$ will be called the best subset in family $\mathcal{G}$. Thus, we further define $W(\mathcal{G})$ as

$$W(\mathcal{G}) = \min_{T \in \mathcal{G}} W(T) = W(T_{\mathcal{G}}^*) \ge W(S) \tag{2}$$

Theorem 1: Let us consider two families $\mathcal{F}$ and $\mathcal{G}$ of subsets of S, where $T_{\mathcal{G}}^*$ is the best subset of $\mathcal{G}$. Then, if there exists a subset $T_{\mathcal{F}} \in \mathcal{F}$ that contains all support vectors of $T_{\mathcal{G}}^*$, we have $W(\mathcal{G}) \ge W(\mathcal{F})$.

Proof: Since $Sv(T_{\mathcal{G}}^*) \subset T_{\mathcal{F}}$, we have $W(T_{\mathcal{G}}^*) = W(Sv(T_{\mathcal{G}}^*)) \ge W(T_{\mathcal{F}})$. Therefore, $W(\mathcal{G}) = W(T_{\mathcal{G}}^*) \ge W(T_{\mathcal{F}}) \ge W(\mathcal{F})$. ■

Theorem 2: Let us consider two families $\mathcal{F}$ and $\mathcal{G}$ of subsets of S, where $T_{\mathcal{G}}^*$ is the best subset of $\mathcal{G}$. Assume that each subset $T_{\mathcal{F}} \in \mathcal{F}$ contains all support vectors of $T_{\mathcal{G}}^*$. Then, if $W(\mathcal{G}) = W(\mathcal{F})$, we have $W(T_{\mathcal{G}}^*) = W(\bigcup T_{\mathcal{F}})$.

Proof: Since any subset $T_{\mathcal{F}} \in \mathcal{F}$ contains $Sv(T_{\mathcal{G}}^*)$, we have $W(T_{\mathcal{F}}) \le W(Sv(T_{\mathcal{G}}^*))$ according to *Theorem* 1. We also have $W(T_{\mathcal{F}}) \ge W(\mathcal{F}) = W(\mathcal{G}) = W(Sv(T_{\mathcal{G}}^*))$ for any subset $T_{\mathcal{F}} \in \mathcal{F}$ according to the given condition. Therefore, $W(T_{\mathcal{F}}) = W(Sv(T_{\mathcal{G}}^*))$, which implies that any subset $T_{\mathcal{F}} \in \mathcal{F}$ have the same set of support vectors as $Sv(T_{\mathcal{G}}^*)$

Further, we can easily analyze that the union of subsets $\bigcup T_{\mathcal{F}}$ also have the same set of support vectors as $Sv(T_{\mathcal{G}}^*)$, namely $W(T_{\mathcal{G}}^*) = W(Sv(T_{\mathcal{G}}^*)) = W(\bigcup T_{\mathcal{F}})$. ■

During the running of MTC-SVM algorithm, we let all the training subsets S_i^t in t-th step compose a family $\mathcal{G}^t$ of subsets of S, and label all the families in sequence as $\mathcal{G}^1, \mathcal{G}^2, \cdots, \mathcal{G}^t, \cdots$.

Theorem 3: The MTC-SVM algorithm converges to the global optimal solution of SVM in finite steps.

Proof: According to the character of MTC architecture, each family $\mathcal{G}^t (t > 1)$ has at least one subset $S_i^t \in \mathcal{G}^t$ that contains all the support vectors of $S_j^{(t-1)*}$, where $S_j^{(t-1)*}$ is the best subset of previous family $\mathcal{G}^{t-1}$. Then, from *Theorem 1*, we can easily verify that the sequence $W(\mathcal{G}^t), t = 1, 2, \cdots$ is monotonically decreasing and is bound by $W(S)$.

Supposing that current family $\mathcal{G}^t$ is composed of all the training subsets $S_i^t = S_i \bigcup \mathcal{U}^{t-1}$ in the first layer of MTC architecture, where $\mathcal{U}^{t-1}$ is the feedback set of previous pass, we can prove that:

(1) If all subsets $S_i^t \in \mathcal{G}^t$ satisfy $W(S_i^t) = W(\mathcal{U}^{t-1})$, we have $W(\mathcal{U}^{t-1}) = W(\bigcup S_i^t) = W(S)$ according to *Theorem 2*, which implies that the optimal solution of SVM can be trained on $\mathcal{U}^{t-1}$. Hence, the algorithm can stop now.

(2) If any subset $S_i^t \in \mathcal{G}^t$ satisfies $W(S_i^t) < W(\mathcal{U}^{t-1})$, we have $W(\mathcal{U}^{t-1}) > W(S_i^t) \ge W(\mathcal{G}^t) \ge W(S)$, which means that the optimal solution cannot be trained on $\mathcal{U}^{t-1}$. However, we

can easily get $W(\mathcal{G}^{t-1}) \geq W(\mathcal{U}^{t-1}) > \mathcal{G}^t$ according to (1). Since $W(\mathcal{G}^t)$ takes $W(S)$ as its lower bound, there must exist a constant $d > 0$ that all subsets $S_i^{t+b} \in \mathcal{G}^{t+b}$ satisfy $W(S_i^{t+b}) = W(\mathcal{U}^{t+b-1})$ after d training passes, where $b = d \cdot (k+1)$. Hence, the algorithm can stop at the global optimal solution after d pass.

Therefore, above analyses prove that the MTC-SVM algorithm will converge to the global optimum of SVM in finite steps. ■

V. Experimental results

To verify the advantages of MTC-SVM, we test it extensively against Graf's Cascade algorithm using several UCI[1] datasets. Both algorithms are implemented in C and run on a cluster which has a total of 27 nodes, with each node being a IBM P4 2.66GHz Processor.

The first experiment investigates the performance of MTC-SVM algorithm with different number of nodes ($p = 3, 9, 27$) on *splice*, *letter*, *adult* and *shuttle* datasets. For each dataset, about 10% samples are used for testing, while the rest is divided equally into p parts and distributed to p nodes for training SVM in parallel. In each computational node, the local SVM classifiers are trained by the SVM$^{\text{light}}$ algorithm [7] with Gaussian kernel, and kernel parameter σ and penalty parameter C are set as Table I.

In this experiment, five indices are evaluated for each dataset, including total training time, speedup, efficiency, testing accuracy and total number of passes. Main experimental results are reported as Table I, where $p = 1$ represents results of SVM$^{\text{light}}$ on entire training samples and single node.

TABLE I
Testing accuracy(accu.), training time(time), total number of passes(#pas.), speedup(spee.) and efficiency(effi.) of MTC-SVM algorithm under different numbers of parallel nodes

datasets	# nodes	accu. (×100%)	time (sec.)	# pas.	spee./effi.
splice (σ=0.5, C=1.0)	$p = 1$	90.60	183.3	—	—/—
	$p = 3$	90.60	120.8	4	1.52/0.51
	$p = 9$	90.60	10.1	2	18.15/2.02
	$p = 27$	90.60	3.8	2	48.24/1.79
letter (σ=2.0, C=10)	$p = 1$	86.45	517.9	—	—/—
	$p = 3$	86.45	631.0	6	0.82/0.27
	$p = 9$	86.45	60.4	3	8.57/0.95
	$p = 27$	86.45	21.3	3	24.31/0.90
adult (σ=1.0, C=10)	$p = 1$	85.21	2974.0	—	—/—
	$p = 3$	85.21	2619.4	5	1.14/0.38
	$p = 9$	85.21	305.7	3	9.73/1.08
	$p = 27$	85.21	60.5	2	49.16/1.82
shuttle (σ=10.0, C=1000)	$p = 1$	99.91	481.2	—	—/—
	$p = 3$	99.91	231.0	3	2.08/0.69
	$p = 9$	99.91	24.8	2	19.40/2.16
	$p = 27$	99.91	7.6	2	62.92/2.33

From the obtained results of Table I, we observe that MTC-SVM reaches to the optimal testing accuracy (same with that of SVM$^{\text{light}}$ on single node) in all cases. This confirms *Theorem* 3 that MTC-SVM can converge to the global optimum of SVM.

[1] Available at http://www.ics.uci.edu/~mlearn/MLRepository.html

We also observe that MTC-SVM can improve training speed remarkably and obtains very high speedup and efficiency, especially as $p = 9$ and $p = 27$. For example, the efficiency is greater than one on 3 datasets. This phenomenon is easy to explain according to our previous theory analyses. The MTC is "three-trifurcate cascade architecture" and "nine-trifurcate cascade architecture" respectively as $p = 9$ and $p = 27$, on one hand, the costs for training local SVM classifiers on each node are decreased since the reduced scale of training subsets. And on the other hand, the total number of passes of MTC-SVM before convergence is decreased greatly since more support vectors are fed back into next pass. Therefore, MTC-SVM algorithm can significantly reduce the training time.

For a better understanding of the superiority of MTC-SVM on training speed, we compare it deeply against Cascade SVM algorithm on large-scale *forest* dataset in our second experiment, where MTC-SVM runs on a cluster of 9 nodes while the latter utilizes 8 nodes.

In the experiment, about 10% samples in *forest* dataset are used for testing, and the rest are distributed to computational nodes equally for training. The training parameters are set with $\sigma = 0.1$ and $C = 10$. Table II reports the main results of both algorithms after each pass, where column "N_F" presents the number of support vectors in current feedback set.

TABLE II
Comparison of training time(time), testing accuracy(accu.) and N_F between MTC-SVM and Cascade SVM algorithms

no. pass	MTC-SVM			Cascade SVM		
	time (sec.)	accu. (×100%)	N_F	time (sec.)	accu. (×100%)	N_F
1	1,476	94.81	9,970	2,171	94.62	9,503
2	2,352	95.12	10,797	3,204	94.90	10,366
3	2,533	95.16	10,832	3,518	95.05	10,680
4		——		3,670	95.14	10,808
5				3,650	95.16	10,825
SUM	6,291	95.16		16,213	95.16	

The results in Table II show that the needed total training time of MTC-SVM algorithm is significantly less than that of Cascade SVM, at the cost of only one more computational node. The reasons for such superiority can further be explained directly from Table II that: (1) MTC-SVM needs much less passes; (2)The training time of MTC-SVM for each pass is remarkably less than that of the latter.

The reason for less passes of MTC-SVM is obviously. Since the MTC is equivalent to the superposition of multiple isomorphic trifurcate cascade architectures, its feedback set is composed of the support vectors of each trifurcate cascade architecture in the last layer during any pass. Therefore, the feedback set always contains more support vectors than that of binary cascade architecture, which accounts for the faster convergence ratio. For example, after the first pass the MTC-SVM already obtains a testing accuracy of 94.91% and corresponding feedback set contains 9,970 support vectors, while those of Cascade SVM are only 94.62% and 9,503 respectively.

The reason for less training time of each pass is also easy to undertand. Since the MTC has fewer layers than binary cascade architecture when the number of nodes is comparable, the maximal number of local SVMs that each node needs to train during a single pass is smaller than than that of the latter. Therefore, the total training time of each pass for MTC-SVM algorithm is much less.

VI. Conclusion

One efficient way to accelerate the training speed of SVM is to implement it in parallel. For conquering the deficiencies of binary cascade architecture[6], such as slow feedback of support vectors and low utilization ratio of computational nodes, we propose a novel and efficient "multi-trifurcate cascade architecture" and develop corresponding parallel algorithm with it for training SVM quickly. Theoretical analysis shows that the proposed MTC-SVM algorithm can converge to the global optimal solution of SVM in finite steps. Experimental results show that it can reach to very high speedup and efficiency, and obtains much faster training speed than Graf's Cascade SVM algorithm.

This work is very useful for the research where a cluster of CPU processors is available. Future work needs to extend the parallel MTC-SVM algorithm from classification to regression estimation by implementing the same methodology for SVM regressors.

Acknowledgment

This paper is sponsored by the National Natural Science Foundation of China (NSFC) under Grant No. 69732010.

References

[1] V. N. Vapnik, The Nature of Statistical Learning Theory, Springer, New York, 1995.

[2] R. Collobert, S. Bengio and Y. Bengio, "A parallel mixture of SVMs for very large scale problems," Neural Computation, vol. 14, pp. 1105-1114, 2002.

[3] J. X. Dong, A. Krzyzak and C. Y. Suen, "Fast SVM training algorithm with decomposition on very large data sets," IEEE Transaction on Pattern Analysis and Machine Intelligence, vol. 27, pp. 603-618, 2005.

[4] L. J. Cao, S. S. Keerthi, et al., "Parallel sequential minimal optimization for the training of support vector machines," IEEE Transaction on Neural Networks, vol. 17, pp. 1039-1049, 2006.

[5] G. Zanghirati, L. Zanni, "A parallel solver for large quadratic programs in training support vector machines," Parallel Computing, vol. 29, pp. 535-551, 2003.

[6] H. P. Graf, E. Cosatto, L. Bottou, I. Durdanovic and V. Vapnik, "Parallel support vector machines: the cascade SVM," in Advances in Neural Information Processing Systems. MIT Press, Cambridge, 2005, pp. 521-528.

[7] T. Joachims, "Making large-scale SVM learning practical," in Advances in Kernel Methods: Support Vector Learning, B. Scholkopf, C. Burges and A. Smola, Eds. MIT Press, Cambridge, 1999, pp. 169-184.