An Improved Algorithm for Parallelizing Sequential Minimal Optimization

C.R. LI & J. GUO
Computer Center, East China Normal University, Shanghai, China

ABSTRACT: In our previous work, a parallelizing sequential minimization optimization was proposed, where the algorithm was executed successfully but its convergence cannot be guaranteed in some cases. In this paper, an improved version is proposed, which can avoid falling into the endless loops. In the proposed method, the multiple violation pairs are selected in each step, and depending on the decrement value of the objective function, a single-pair update or multiple-pair update is determined. Experimental results show that the proposed method is more effective than the previous methods. The parallel algorithm is well executed while the accuracy is maintained and the convergence is completely guaranteed.
KEYWORDS: SVM; SMO; parallel algorithm; convergence

1 INTRODUCTION

The Support vector machines (SVMs) [1-2] have been proposed on the basis of structural risk minimization and the theory of VC bounds. SVMs are widely used in artificial intelligence area. For binary classification problems, to train a standard SVM is to construct two parallel hyper-planes which maximize the width between the two classes. It is a constrained quadratic programming (QP) problem where the number of variables equals to the size of dataset. The QP matrix is usually too huge to be stored.

Decomposition methods [3-5] were proposed to deal with such difficulties, in which only a subset of the whole Lagrange multipliers, which is called working set B, is modified and the rest set N is regarded as fixed. Chunking [3] uses the fact that zero multipliers have no effect on the value of the objective function. It keeps every non-zero multiplier in B from the previous step, discards the zeros and imports the worst multipliers that violate the Karush-Kuhn-Tucker (KKT) conditions. When all the non-zero multipliers are identified, the last step solves the QP problem. Chunking greatly reduces the size of the QP matrix, but it cannot handle large-scale training because the reduced matrix may still be too large. In Osuna's method [4], while there are some violating multipliers in N, any of them is replaced with one from B in each step. Osuna has proved that moving a multiplier from B to N doesn't change objective function while moving a violating multiplier from N to B leads to a strict improvement when B is re-optimized. Osuna's algorithm definitely guarantees convergence to the global optimal solution. Note that chunking method obeys the conditions of Osuna's theorem, hence it converges too.

Sequential minimal optimization (SMO) [5] restricts B to have only two multipliers which can be solved analytically and requires no extra matrix storage. There are two heuristics for choosing which multipliers to be optimized. The first choice heuristic concentrates on unbound multipliers that are most likely to violate the KKT conditions. Once a first Lagrange multiplier is chosen, the second choice heuristic chooses the second Lagrange multiplier that maximizes the difference value of the two prediction errors. But it's inefficient to re-compute the threshold value after each step.

Some researchers focus on how to parallelize SMO algorithm to decrease training time, [6] is based on the single program multiple data (SPMD) model. It splits the whole dataset into smaller subsets, and updates each subset's error array in parallel using multiple processors. In cascade SVM [7], multiple layers of SVMs work as filters to extract support vectors. Two sets of support vectors from the previous layer are combined as input for next layer. The filtering process continues until only one subset is left. All the support vectors from last layer are sent back to the first layer to test global convergence. Different from the above methods, the parallel version of SMO (PSMO) proposed in [8] parallelizes the selection of the working set, where two violating pairs are chosen and updated simultaneously in each step, then the whole set's gradients are updated. The key idea is to reduce time

cost by reducing the number of iterations. As the stopping criterion in PSMO is the same to the traditional SMO, the prediction accuracy is kept. Experiments show that PSMO does accelerate the training speed in many cases. However, in some cases, PSMO makes the training time longer or even fails to converge.

In this paper, we propose an improved version of PSMO (called IPSMO) by intelligently choosing between single-pair update and multi-pair update in each step, depending on which way has larger decrement of the objective function. We also expand it to a 4-way parallelization. Experiments show that using the 2-way parallelization, IPSMO not only performs as well as PSMO in the datasets where PSMO outperforms SMO, but also solves the convergence issues on those datasets for which PSMO fails to converge. The advantages of IPSMO are more obvious in the 4-way parallelization.

This paper is organized as follows: the brief introduction to SMO is given in Section 2. In Section 3, the difference between PSMO and the improved version IPSMO is discussed. Detailed experiments are shown in Section 4. Finally, Section 5 concludes this work.

2 SEQUENTIAL MINIMAL OPTIMIZATION

Given a binary classification problem with instances $x_i,\ i=1,...,l$ and labels $y_i \in \{-1,+1\}$, the main task in the training step of SVM is to solve the following quadratic programming problem:

$$\min_{\alpha}\ \mathrm{Q}(\alpha) = -\sum_{i=1}^{l}\alpha_i + \frac{1}{2}\sum_{i=1}^{l}\sum_{j=1}^{l} y_i y_j \alpha_i \alpha_j K(x_i, x_j) \quad (1)$$

Subject to:

$$\forall_{i,}\ 0 \le \alpha_i \le C,\ \ \sum_{i=1}^{l} y_i \alpha_i = 0$$

Where α is the vector of Lagrange multipliers, C restricts the upper bound of all the multipliers and K is the kernel function calculating the dot product of two samples after mapped to higher dimension space.

The KKT conditions are necessary and sufficient conditions for solving the QP problem. According to [9], when the optimality holds, for each i, the following constraints are satisfied:

$$\begin{cases} \alpha_i = 0,\ \mathrm{y}_i(F_i - \mathrm{b}) \ge 0 \\ 0 < \alpha_i < C,\ \mathrm{y}_i(F_i - \mathrm{b}) \approx 0 \\ \alpha_i = C,\ \mathrm{y}_i(F_i - \mathrm{b}) \le 0 \end{cases} \quad (2)$$

Where:

$$F_i(\alpha) = y_i \frac{\partial Q(\alpha)}{\partial(\alpha_i)} = y_i\Big(\sum_{j=1}^{l} y_i\, \mathrm{y}_j\, \alpha_j K(\mathrm{x}_j, \mathrm{x}_i)\ \text{-1}\ \Big) \quad (3)$$

Eq. (2) can be written as:

$$\max_{i\in I_{up}}\ -F_i(\alpha) \le \min_{j\in I_{low}}\ -F_j(\alpha)$$

Where:

$$\begin{cases} I_{up} = \{t \mid y_t = 1, \alpha_t < C \ \text{ or } y_t = -1, \alpha_t > 0\} \\ I_{low} = \{t \mid y_t = -1, \alpha_t < C \ \text{ or } y_t = 1, \alpha_t > 0\} \end{cases}$$

We define a pair of indices (i, j) as a τ-violating pair if the following constrain holds:

$$i \in I_{up}(\alpha),\ \mathrm{j} \in I_{low}(\alpha)\ \ -F_i(\alpha) \ge -F_j(\alpha) + \tau$$

The optimality holds if and only if no violating pair exists.

SMO uses first order (i.e., gradient) information of Eq. (3) to select the violating pair via "maximal violating pair" rule:

$$\begin{cases} i \in \arg\max_t(-F_t(\alpha) \mid \mathrm{t} \in \mathrm{I}_{up}) \\ j \in \arg\min_t(-F_t(\alpha) \mid \mathrm{t} \in \mathrm{I}_{low}) \end{cases}$$

While LIBSVM [10-11] uses second order information to select j which directly relates to decrement of Eq. (1):

$$\begin{cases} j \in \arg\min_t(-b_{i,t}^2 / a_{i,t} \mid \mathrm{t} \in I_{low,}\ -F_i(\alpha) \ge -F_t(\alpha) + \tau) \\ b_{i,t} = -F_i(\alpha) + F_t(\alpha) > 0 \\ a_{i,t} = K(\mathrm{x}_{i,}\mathrm{x}_i) + K(\mathrm{x}_{t,}\mathrm{x}_t) - 2K(\mathrm{x}_i, \mathrm{x}_t) > 0 \end{cases}$$

If $\alpha_{i,t} \le 0$, then $\alpha_{i,t}$ is replaced with a small positive number. In our experiment, we also use second order information since it is faster than using first order information.

3 THE PROPOSED PARALLEL ALGORITHM

In this section, our improved parallel version of SMO (IPSMO) is proposed. From Eq. (3) we can see that the statuses of Lagrange multipliers are interacted on each other. Hush [12] has proved that SMO-type methods have strict decrease of Eq. (1) (i.e., $Q(\alpha^{k+1}) < Q(\alpha^k),\ \forall k$) if and only if B is a violating pair. PSMO can be regarded as a variation of SMO: denote (i_{up}^1, j_{low}^2) and (i_{up}^2, j_{low}^2) as the first and second violating pairs updated in the *kth* iteration, the *kth* iteration is kind of two single-pair update iterations:

$$k.1:\ \text{updates}\,(\alpha_{i_{up}^1}, \alpha_{j_{low}^1})$$

$$k.2:\ \text{updates}\,(\alpha_{i_{up}^2}, \alpha_{j_{low}^2})$$

The update of the whole set's gradients is delayed until $k.2$ iteration, which makes PSMO not obey Hush's theorem[12] because pair $(\alpha_{i_{up}^2}, \alpha_{j_{low}^2})$ may not be a violating pair if we perform iteration $k.1$ and $k.2$ in SMO scenario. IPSMO is proposed to guarantee the strict decrease of Eq. (1): it pre-computes the objective function value in both scenarios (single-pair update and multiple-pair

update), then chooses the way that makes the value decrease more. As Eq. (1) is convex quadratic and has feasible region, the value is limited, which means IPSMO will definitely converge. Note that when there is no multiple-pair update in the training period, IPSMO degenerates to SMO. The training algorithm of IPSMO is described as follows:

Algorithm 1. Constructing SVM model via IPSMO

Input: instances $\{x_i, y_i\}$ $i = 1,...,l$, constant C, RBF kernel g and stopping criteria τ.

1. Initialize: Set $k = 0,\ \alpha(k) = 0,\ G(k) = \nabla Q(\alpha(k))$ and n is the maximum number of violating pairs.
2. While τ-optimality conditions are not satisfied, do
3. Select n violating pairs:

(1) Get the top n indices $i_{up}^t \in I_{up}(\alpha),\ t = 1,...,n$,sorted by $-y_t G_t(k)$ in decreasing order.

(2) For each i_{up}^t , get n candidates $j_{low}^{t,m} \in I_{low}(\alpha),\ \ m = 1,...,n$, sorted by $-b_{t,j_{low}^{t,m}}^2 / a_{t,j_{low}^{t,m}}$ in increasing order.

(3) For $i = 1,...,n.\ m = 1,...,n$ if i_{up}^t and $j_{low}^{t,m}$ are not marked, mark them and pair $(i_{up}^t, j_{low}^{t,m})$ is chosen.

4. For each violating pair (i, j), calculate the corresponding modifications α_i^*, α_j^* and $G_z^*(k),\ z = 1,...,l$, l is the size of training set.
5. Calculate the decrement value of objective function $Q(\alpha)$ in both single-pair and multiple-pair scenario respectively, choose the one that $Q(\alpha)$ decreases more, then update $\alpha(k),\ G(k)$.
6. End while

Output: SVM model

We perform step 3.2 and step 4 in parallel. Note that in step 3.3, some candidate pairs may be discarded if i_{up}^t has already been in other violating pairs. In this case, the number of chosen pairs is smaller than n.

4 EXPERIMENTAL RESULTS

Similar to PSMO, IPSMO is also implemented on LIBSVM. In order to compare different algorithms' efficiency purely, tools (i.e., shrinking, caching) that can speed up training in LIBSVM are not adopted. Experiments are run on personal computers with Intel Core I5 processors and 8 GB RAMs. As different SVM parameters such as C in Eq. (1) and kernel parameters such as g in RBF kernel function $K(x_i, x_j) = e^{-g\|x_i - x_j\|^2}$ affect training performance, to simulate how one trains SVM model in practice, Experiments are conducted as the following procedure:

1. "Parameter selection" procedure: for each dataset, here we adopt RBF kernel as the kernel function and execute grid search strategy to find the proper (C, g) that result in the highest classification accuracies. C is selected from set $\{2^i \mid i = -5,-3,...,15\}$ and g is selected from $\{2^k \mid k = -15,-13,...,3\}$. For those datasets that do not have a specific testing set, we conduct 10-fold cross validation to determine the proper pair (C, g). To be more specific, we randomly divide the whole data set into 10 pieces, iteratively choose 9 of the 10 pieces of data for training and the rest one piece for testing, then average test accuracies, (C_{opt}, g_{opt}) that corresponds to the highest mean accuracy is chosen, shown in Table 2.

2. "Final training" procedure: Respectively construct SVM model via SMO, PSMO and IPSMO on each dataset with parameters (C_{opt}, g_{opt}), Note that the procedure is repeatedly several times to obtain the precise training time, results are shown in Table1.

Table 1. Results of SMO, PSMO, IPSMO on benchmark datasets

	SMO		PSMO(2-way)		IPSMO(2-way)		IPSMO(4-way)	
	iterations	time(s)	iterations	time(s)	iterations	time(s)	iterations	time(s)
australian*	58585	26.53	149585	70.55	37524	17.72	**31596**	**15.50**
breast-cancer*	1628	0.69	885	0.40	851	0.39	**521**	**0.25**
diabetes*	1443	0.66	744	0.37	744	0.37	**369**	**0.19**
german.numer*	358885	284.27	N/A	N/A	225209	188.32	**193641**	**167.10**
heart*	11087	1.84	42267	7.61	**7288**	**1.32**	7344	1.36
fourclass*	945	0.39	553	0.23	564	0.25	**329**	**0.15**
mushrooms*	5652	42.46	4780	38.66	3313	**26.58**	**3297**	27.42
letter(scale)	220651	168.87	N/A	N/A	115949	94.59	**71010**	**57.65**
splice	2664	3.43	1261	1.66	1309	1.73	**614**	**0.82**
a1a	3432	4.61	2053	2.95	1819	2.61	**1271**	**1.89**
a2a	4135	7.83	1961	3.95	1969	3.99	**1248**	**2.59**
a3a	2298	6.10	1183	3.31	1291	3.65	**696**	**2.01**
a4a	2612	10.31	1257	5.25	1291	5.42	**750**	**3.24**

From Table 1, we can see that compared with SMO, PSMO's training time reduces greatly in breast-cancer, diabetes, etc. In datasets heart and australian, PSMO deteriorates otherwise and in datasets german.numer and letter, PSMO even fails to converge. Generally IPSMO's training time per iteration is a bit more than PSMO's, which is mainly due to the cost in pre-computing the objective function value. But IPSMO performs more steadily: the training time is generally less than SMO, it also solves the convergence issues that exist in PSMO. IPSMO (4-way) is generally better than IPSMO (2-way) in both numbers of iterations and training time, because more violating pairs maybe updated in each step.

Table 2. Training parameters adopted on datasets

dataset	size	attrs.	$C_{opt,}$	g_{opt}
australian*	690	14	2048	3.05e-05
breast-cancer*	683	10	2	7.81e-03
diabetes*	768	8	2	7.81e-03
geman.numer*	1000	24	32768	3.05e-05
heart*	270	13	512	3.05e-05
fourclass*	862	2	0.5	7.81e-03
mushrooms*	8124	112	32	7.81e-03
letter(scale)	15000	16	32	7.81e-03
splice	1000	60	8	7.81e-03
a1a	1605	123	128	1.95e-03
a2a	2265	123	8	3.13e-03
a3a	3185	123	32	1.95e-03
a4a	4781	123	8	7.81e-03

these datasets are downloaded from http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/. Dataset denoted by '*' has no specific test set.

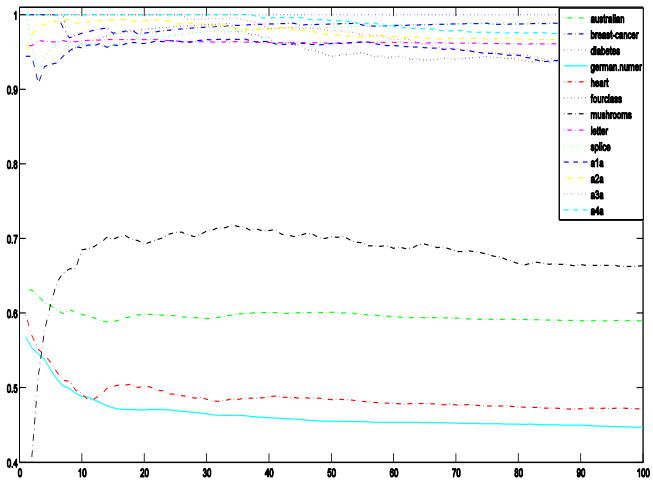

Figure 1. These datasets are trained with the usage of IPSMO (2-way). The training time is equally split to 100 pieces, any of which corresponds to a snapshot. The percent of multiple-pair updates tends to be stable as the training goes on.

5 CONCLUSION

In this paper, we propose an improved parallel version of SMO based on PSMO. Experiments show that IPSMO's performance is more stable than PSMO and convergence issues are also solved. In the future, there is much work to do. For example: what if IPSMO uses more parallels (8-way, 16-way, etc.)? Given a specific dataset, how to determine the optimal parameter in n-way that can speed up the training most? Since multiple violating pairs can be updated in each step, the advantages brought from advanced algorithms of selecting violating pairs can be more obvious. Developing a new selection algorithm is also an interesting research area.

REFERENCES

[1] Boser, B. E., Guyon, I. M., & Vapnik, V. N.: A training algorithm for optimal margin classifiers. In: Proceedings of the fifth annual workshop on Computational learning theory, ACM Press.pp.144-152.

[2] Cortes C, Vapnik V. 1995. Support-Vector Networks. Machine Learning.

[3] Vapnik, V., 1982. Estimation of Dependences Based on Empirical Data, Springer-Verlag.

[4] Osuna, E., Freund, R., & Girosi, F. 1997. An improved training algorithm for support vector machines. In: Neural Networks for Signal Processing [1997] VII. Proceedings of the 1997 IEEE Workshop, pp. 276-285.

[5] Platt, J. 1998. Sequential minimal optimization: A fast algorithm for training support vector machines.

[6] Cao, L. J., Keerthi, S. S., Ong, C. J., Zhang, J. Q., Periyathamby, U., Fu, X. J., & Lee, H. P. 2006. Parallel sequential minimal optimization for the training of support vector machines. In: Neural Networks, IEEE Transactions on. 17(4), pp.1039-1049.

[7] Graf, H. P., Cosatto, E., Bottou, L., Dourdanovic, I., & Vapnik, V. 2004. Parallel support vector machines: The cascade svm. In: Advances in neural information processing systems, pp. 521-528.

[8] Wang, X., & Guo, J. 2013. An Algorithm for Parallelizing Sequential Minimal Optimization. In: 20th International Conference, ICONIP, pp.657-664.

[9] Keerthi, S. S., Shevade, S. K., Bhattacharyya, C., & Murthy, K. R. K. 2001. Improvements to Platt's SMO algorithm for SVM classifier design. In: Neural Computation, 13(3), pp. 637-649.

[10] Fan, R. E., Chen, P. H., & Lin, C. J. 2005. Working set selection using second order information for training support vector machines. In: The Journal of Machine Learning Research, 6, pp.1889-1918.

[11] Chih, C. C., & Lin, C. J. 2011. LIBSVM: a library for support vector machines. In: ACM Transactions on Intelligent Systems and Technology (TIST), 2(3), 27.

[12] Hush,D., & Scovel,C. 2003. Polynomial-time decomposition algorithms for support vector machines. In: Machine Learning, 51:51–71.