Support vector machine incremental learning triggered by wrongly predicted samples*

TANG Ting-long (唐庭龙)[1,2], **GUAN Qiu** (管秋)[3]**, and **WU Yi-rong** (吴义熔)[2,4]

1. Collaborative Innovation Center of Yangtze River Delta Region Green Pharmaceuticals, Zhejiang University of Technology, Hangzhou 310023, China

2. College of Computer and Information Technology, China Three Gorges University, Yichang 443002, China

3. College of Computer Science and Technology, Zhejiang University of Technology, Hangzhou 310032, China

4. Department of Radiology, University of Wisconsin-Madison, Madison 53792, USA

(Received 29 November 2017; Revised 17 January 2018)

According to the classic Karush-Kuhn-Tucker (KKT) theorem, at every step of incremental support vector machine (SVM) learning, the newly adding sample which violates the KKT conditions will be a new support vector (SV) and migrate the old samples between SV set and non-support vector (NSV) set, and at the same time the learning model should be updated based on the SVs. However, it is not exactly clear at this moment that which of the old samples would change between SVs and NSVs. Additionally, the learning model will be unnecessarily updated, which will not greatly increase its accuracy but decrease the training speed. Therefore, how to choose the new SVs from old sets during the incremental stages and when to process incremental steps will greatly influence the accuracy and efficiency of incremental SVM learning. In this work, a new algorithm is proposed to select candidate SVs and use the wrongly predicted sample to trigger the incremental processing simultaneously. Experimental results show that the proposed algorithm can achieve good performance with high efficiency, high speed and good accuracy.

Document code: A **Article ID:** 1673-1905(2018)03-0232-4

DOI https://doi.org/10.1007/s11801-018-7254-3

Incremental learning methods are proposed to deal with the problems of data arriving sequentially. There are two approaches of incremental learning: one is batch incremental method, i.e. training a batch of samples at a time; the other is instance-incremental method (or called online learning method), i.e. learning from each sample as it arrives[1]. Through these approaches, incremental learning methods can reserve the historical results and avoid repeating study of old samples. Because of its fast speed and good efficiency, incremental support vector machine (ISVM) has been applied to anomaly identification[2], intrusion detection[3], defect detection[4], clinical medicine[5], finance[6], and so on.

Many basic theoretical works around solving the quadratic programming (QP) problem of ISVM have been done over the past decade[7-10]. For ISVM, there is an important issue that only the support vectors (SVs) will influence the classification hyperplane of SVM. Then, an important view has attacked many researchers' attention that which of the new samples will be new SVs and how the new samples will influence the migration of the vectors between error set, reserved set and support set[11-14]. Based on Karush-Kuhn-Tucker (KKT) conditions, when new samples arrive, part or all of the new samples will be new SVs and the old vectors will change between SVs and non-support vectors (NSVs), and the current learning model should be updated at the same time. However, it is not exactly clear that which samples would change between NSV set and SV set. Particularly, we believe that it is not very necessary to update the current model at every time of such a violating sample's arriving and propose a new algorithm to select candidate SVs and trigger the incremental update processing according to the wrongly predicted sample simultaneously. This procedure is very satisfied with the need of online tasks in the reality: if the current model is frequently updated, it could not catch up with the speed of samples' arriving; and if the current model can correctly classify all the current samples, we don't need to update it to find the optimal one.

Given the training set and their labels $\{(x, y)|x\in R^n, y_i=\pm 1, i=1,...,m\}$, the standard classification problem of SVM is to solve an optimization problem as

$$\min_{w,b,\xi} \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{n}\xi_i$$
$$s.t. \quad y_i(w^{\mathrm{T}}\Phi(x_i)+b) \geq 1-\xi_i, \quad \xi_i \geq 0, i=1,...l \tag{1}$$

where ξ_i is the vector of slack variables, and C is the regularization parameter. To solve such a problem, one can obtain the dual formulation as follows by introducing

* This work has been supported by the National Natural Science Foundation of China (Nos.U1509207 and 61325019).

** Email: gq@zjut.edu.cn

Lagrangian multipliers a:

$$\min -w(a)=\frac{1}{2}\sum_{i,j}a_iQ_{i,j}a_j-\sum_i a_i+b\sum_{j=1}^{N}y_ja_j\ , \tag{2}$$

where $Q_{i,j}=y_iy_jk(x_i, x_j)$ represents the kernel matrix. The KKT conditions are the necessary and sufficient conditions for an optimal point of a convex QP problem. For the dual-problem, KKT conditions are described as follows:

$$g_i=\frac{\partial w}{\partial a_i}=\sum_{j=1}^{N}Q_{i,j}a_j+y_ib-1, \tag{3}$$

$$h=\frac{\partial w}{\partial b}=\sum_{j=1}^{N}y_ja_j. \tag{4}$$

When a new sample x_l is added to the training set, the margin vector coefficients are updated,

$$\Delta\alpha_k=\beta_k\Delta p:\forall k\in S\ , \tag{5}$$

$$\Delta\alpha_l=\lambda_l\Delta p:\forall l\in U, \tag{6}$$

$$\Delta b=\beta\Delta p, \tag{7}$$

where Δp is the minimum change during the incremental learning. All the possible changes were listed in a category and the details were described in Ref.[8].

After that, the optimal hyperplane is obtained. The decision function of SVM is

$$f(x)=\mathrm{sgn}(\sum_{i=1}^{l}a_iy_ik(x_i\cdot x)+b)\ . \tag{8}$$

Because only the SVs will influence the classification hyperplane of SVM, many incremental SVM algorithms focus on improving SVM training process through collecting more useful data as SVs or deleting less useful data as NSVs[11,15]. For the KKT conditions, there are basic theorems[16,17]:

If there are samples in newly-increased samples which violate the KKT conditions, part or total of these samples will transfer to SVs and the NSVs in the original classification SVM can transfer to SVs.

In Fig.1, points A, B and C represent three different samples of violating KKT conditions, which can be expressed as $yf(x)<1$: A is in the margin with correctly predicted label, B is in the opposite margin with the wrongly predicted label, and C is out of the margin with the wrongly predicted label.

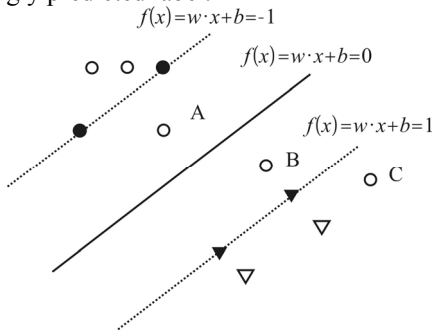

Fig.1 Violating KKT conditions

There are many algorithms relying on the idea of introducing a new vector, and then migrating specific vectors between the sets to have the satisfied KKT conditions again[7,12]. The KKT conditions guide the samples to migrate between reserved set, SV set and error set. However, it is not exactly clear that which samples of SV set would change into NSVs and which samples of NSV set would change into SVs when new samples are added. Particularly, the learning model will be immediately updated once a sample violating KKT conditions appears. However, we consider that it is not always necessary and will not greatly increase its accuracy but decrease its efficiency. Inspired by those ideas, we propose a new algorithm to select candidate SVs and only use the wrongly predicted sample to trigger the incremental processing simultaneously.

At the beginning, we only select one positive and one negative sample to train the initial SVM. When a newly arriving sample, like point A, B or C in Fig.2, is violating KKT conditions, in traditional methods, it will be a new SV and causes some of the old NSVs to change into SVs, and the retraining or updating processing must be acted immediately according to the new SVs. In our method, unless the current model meets a new sample with its predicted label not the same as its true label such as point B or C in Fig.2, we don't update it but only collect all the samples violating KKT conditions. Once we obtain a wrongly predicted sample like point B or point C, we immediately trigger the incremental updating process and only use the nearest sample(s) to current hyper-plane in the collected set above to update the current SVM model with bigger penalty coefficient C. Our algorithm is described in Tab.1.

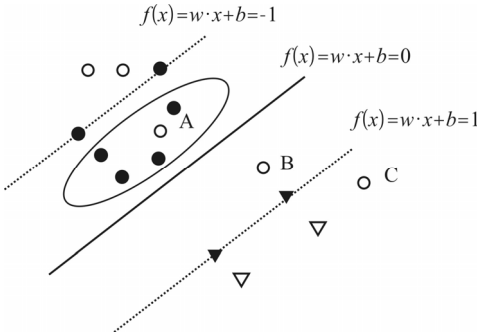

Fig.2 Collecting candidate SVs

Tab.1 The proposed algorithm

Input: Sequential data
Output: Predicted labels
Step 1 Initial training
1.1 select one positive and one negative sample from the labeled data set;
1.2 train the initial SVM;
Step 2 Incremental learning
if current sample violates KKT conditions
collect it to candidate set V;
if the predicted label is not equal to the actual label
find the nearest sample(s) to the hyperplane in the candidate set V;
use the nearest sample(s) to update current SVM with a bigger C;
end
end
Step 3 Testing
3.1 use the last 10% of samples for prediction and test;
3.2 evaluate the performance by 10-ford cross-validation.

Experiments are presented with seven data sets listed in Tab.2 from UCI machine learning repository[20]. All experiments were running on Intel(R) Core i7-6820 CPU @2.7 GHz, 16.0 GB RAM machine.

To make comparisons with recent methods, we firstly design experiments to compare our method with non-incremental batch learning methods (i.e. classic SVM implemented based on LibSVM[18]) and the batch-incremental method (i.e. ISVM[8]). Secondly, we design experiments to compare our method with recent online methods (i.e. mcpIncSVM (http://www-ti.informatik.uni-tuebingen.de/ ~spueler/mcpIncSVM/) and LASVM[9]).

Tab.2 Data sets description

Data set	Size
Breast	683×9
Car	1 728×6
Messidor	1 151×9
KrvsKp	3 196×36
Spambase	4 061×57
Ring	7 400×20
Coil	9 822×85
Credit	30 000×23

We use 10-ford cross-validation to get the average accuracy, training time and the SV number. The accuracy is defined as the proportion of correctly predicted samples in the test set. The training time is defined as the CPU time of training the model with different methods. After training, the number of SVs is calculated.

Firstly, comparisons between the proposed method and batch learning methods, such as classic SVM and classic ISVM, are presented in Figs.3—5.

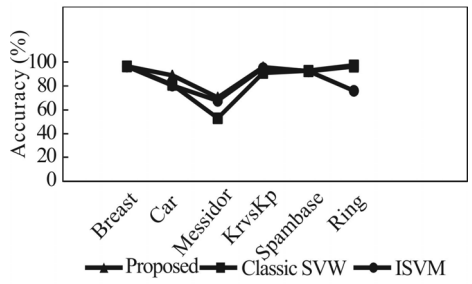

Fig.3 Accuracy comparison with batch learning methods

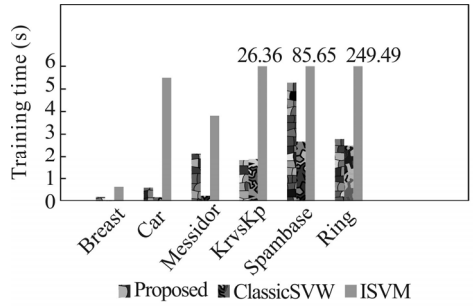

Fig.4 Training time comparison with batch learning methods

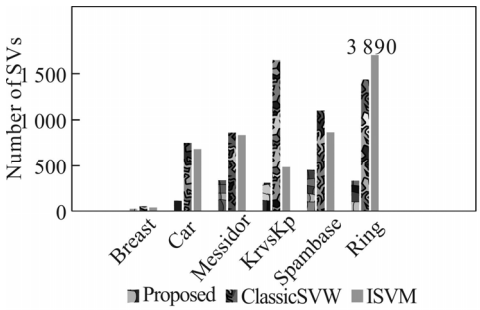

Fig.5 SV number comparison with batch learning methods

It is shown in Fig.3 that for all the data sets, our method has a better accuracy than ISVM. Though the accuracy is not better than LibSVM for the ring data set, it is very similar to it. As observed in Fig.4, the performance of training time of our method is much worse than that of LibSVM. This is because our method is under the particular suggestion of flow-based data which needs to check samples one by one, however, LibSVM is a pool-based method which processes all the training data at the same time. So when the number of instances is small, our method doesn't show a high speed under this condition. But when the number of instances is bigger, such as KRvsKP data set, the training time of ours is close to that of LibSVM. Especially when the number of instances is larger, the speed of ours is significantly faster than that of LibSVM, such as when classifying the ring data set. It's also very clear in Fig.4 that the speed of our method is significantly better than that of classic ISVM.

When we compare the SV numbers of different methods in Fig.5, we find the SV number of our method is obviously smaller than that of LibSVM and ISVM, which means our method has a better efficiency during the training phase.

As a result, our method can achieve good accuracy: though it's not very fast on little scale data sets, it still has a remarkable speed close to the classic non-incremental SVM on bigger scale data sets. When compared with the classic incremental SVM method, the performance of our method is much better.

Secondly, we compare the results of our method and two recent online learning methods in Tab.3: McpIncSVM and LASVM. Our method and mcpIncSVM are tested using Matlab R2014a under Windows OS, while LASVM is tested using C++ under Ubuntu OS. Though the implementations are in different operating systems and programming environments, the efficiency of different methods still can be analyzed according to the number of SVs. In Tab.3, we don't list the accuracy results and SV numbers when using mcpIncSVM method to test the coil and credit data set because it will cost hundreds of hours before getting the results. The speed of mcpIncSVM method is much slower than that of ours.

Tab.3 Comparison with online methods

Data set		Breast	Car	Messidor	KrvsKp	Spambase	Ring	Coil	Credit
Size		683×9	1 728×6	1 151×9	3 196×36	4 061×57	7 400×20	9 822×58	30 000×23
	Proposed	**96.48**	88.88	70.45	95.65	**92.33**	**96.22**	**93.64**	**91.37**
Accuracy	mcpIncSVM	91.95	93.63	**73.58**	95.77	92.15	89.82	-	-
(%)	LASVM	92.39	**95.53**	64.89	**98.53**	82.85	89.78	93.69	80.53
	Proposed	0.18	0.58	2.13	**1.81**	5.27	**2.78**	**8.92**	**344.89**
Training	mcpIncSVM	1.68	2.81	6.92	3.98	15.84	6 736.81	-	-
time (s)	LASVM	**0.02**	**0.01**	**0.09**	10.64	**4.56**	5.79	29.39	446.59
	Proposed	**29**	**117**	**339**	**314**	**454**	**335**	**554**	**5 211**
SVs	mcpIncSVM	372	4 260	679	380	918	5 773	-	-
	LASVM	354	611	791	1 760	3 352	5 413	7 941	17 425

It is shown in Tab.3 that our method has better accuracy, less training time and smaller SV numbers on large-scale data sets such as ring, coil and credit data sets. Though the accuracy is lower than other two methods on other data sets, it is very similar to them. It's very expressive that the speed of ours is significantly faster than that of mcpIncSVM for all the data sets. Our method is implemented using Matlab under Windows OS. LASVM is implemented using C++ under Ubuntu OS, which usually has a better programming efficiency. However, when testing on the data set with very large samples, for example, the coil data set with 9 822 samples and credit data set with 30 000 samples, our method can use a smaller number of SVs but achieve better classification accuracy and much faster speed than LASVM.

In this work, we consider that it is unnecessary to update the current SVM model every time when finding a new sample violating the KKT conditions. We collect the violating samples and only update the current SVM model according to the collected samples when finding the current arriving sample is wrongly predicted. Thus, the current SVM will not be so frequently updated and the accuracy is guaranteed together with efficiency. It is tested in different scale data sets that our method can achieve good accuracy and high speed. Most important of all, our method has the ability of beginning with only two samples and updating the current model according to the expert label with good performance.

References

[1] Ade R and Deshmukh P R, International Journal of Computer Applications **2**, 1039 (2014).

[2] Yuan Y, Fang J and Wang Q, IEEE Transactions on Cybernetics **45**, 548 (2015).

[3] Omar Abdel Wahab, Azzam Mourad, Hadi Otrok and Jamal Bentahar, Expert Systems with Applications **50**, 40 (2016).

[4] Maoxiang Chu, Jie Zhao, Xiaoping Liu and Rongfen Gong, Chemometrics and Intelligent Laboratory Systems **168**, 15 (2017).

[5] Nilashi M., Ibrahim O. B., Mardani A., Ahani A. and Jusoh A., A Soft Computing Approach for Diabetes Disease Classification, Health Informatics Journal, 2016.

[6] Ivana Marković, Miloš Stojanović, Miloš Božić and Jelena Stanković, ICT Innovations, 105 (2014).

[7] Masayuki Karasuyama and Ichiro Takeuchi, Advances in Neural Information Processing Systems, 907 (2009).

[8] Gert Cauwenberghs and Tomaso Poggio, Incremental and Decremental Support Vector Machine Learning, International Conference on Neural Information Processing Systems, MIT Press, 388 (2000).

[9] Antoine Bordes, Seyda Ertekin, Jason Weston and Léon Bottou, Journal of Machine Learning Research **6**, 1579 (2005).

[10] Seyda Ertekin, Leon Bottou and C Lee Giles, IEEE Transactions on Pattern Analysis and Machine Intelligence **33**, 368 (2011).

[11] Cunhe Li, Kangwei Liu and Hongxia Wang, Applied Intelligence **34**, 19 (2011).

[12] Bin Gu, Victor S. Sheng, Zhijie Wang, Derek Ho, Said Osman and Shuo Li, Neural Networks **67**, 140 (2015).

[13] Yang Yi, Jiansheng Wu and Wei Xu, Expert Systems with Applications **38**, 7698 (2011).

[14] Roshan Chitrakar and Chuanhe Huang, Computers & Security **45**, 231 (2014).

[15] Fei Gao, Jingyuan Mei, Jinping Sun, Jun Wang, Erfu Yang and Amir Hussain, Plos One **10**, e0135709 (2015).

[16] Wei-Yuan Cheng and Chia-Feng Juang, Fuzzy Sets and Systems **163**, 24 (2011).

[17] YouLong Yang, JinXing Che, YanYing Li, YanJun Zhao and SuLing Zhu, Energy **113**, 796 (2016).

[18] Chih-Chung Chang and Chih-Jen Lin, ACM Transactions on Intelligent Systems and Technology **2**, 1 (2001).

[19] Shaoning Pang, Lei Zhu, Gang Chen, Abdolhossein Sarrafzadeh, Tao Ban and Daisuke Inoue, Neural Networks **44**, 87 (2013).

[20] Bache K and Lichman M, UCI Machine Learning Repository, CA: University of California, School of Information and Computer Science, 2013.