International Journal of Computer and Electrical Engineering, Vol. 2, No. 2, April, 2010
1793-8163

Parallel Incremental Proximal Linear Support Vector Machine

Xin Zhang, Gengfeng Zhu

Abstract—**For accelerating the training speed of Support Vector Machine (SVM) when solving large scale unbalanced learning problem, a novel parallel incremental proximal linear SVM was proposed. This paper proposed a new strategy to obtain the weight matrix to overcome the unbalanced property, thus acquired higher prediction accuracy for unbalanced dataset. Moreover, a cascade architecture of parallel SVM methods was used to accelerate the training speed. Experiments show that this method could not only improve prediction accuracy, but also increase the training speed and save more training time.**

Index Terms—**Incremental algorithm, Parallel, Support Vector Machine, Unbalanced Dataset**

I. Introduction

Support Vector Machine (SVM) is a modern mechanism for binary classification, regression and clustering problems [1]. For large scale learning problems, the standard SVM is time consuming and needs large memory to store the kernel matrix. Recently, many scholars have proposed fast methods to solve large scale problems. On the other hand, unbalanced training dataset [2] is also a hot topic in machine learning. The unbalanced property of binary classification problem refers to the number of examples of positive (negative) class is much larger than the number of examples of negative (positive) class. This property results that the less examples class provides little classification information to classifier, thus reduces the prediction accuracy of classifier.

Fung and Mangasarian [3] proposed Proximal SVM to solve large scale learning problems. Proximal SVM leads to an extremely fast and simple algorithm for generating a linear or nonlinear classifier that merely requires the solution of a single system of linear equations. Next year, they proposed Incremental SVM [5] which is capable of modifying an existing linear classifier by both retiring old data and adding new data. For unbalanced training dataset, Liu [4] proposed class weighted SVM to improve prediction performance. In this paper, based on the class weighted SVM method and the cascade architecture of parallel SVM methods, we develop a parallel Incremental Proximal Linear SVM (PIPLSVM).

Manuscript received July 27, 2009. This work was supported in part by the Natural Science Foundation of China (10571109) and Natural Science Foundation of Shandong Province (Y2008A01).

Xin Zhang is with the Department of Electronic Engineering, City University of Hong Kong, Tat Chee Avenue, Kowloon, Hong Kong SAR. (corresponding author to provide phone: 0852-3442-3791; e-mail: marix4@gmail.com).

GengFeng Zhu was with College of Information Science and Engineering, Shandong University of Science and Technology, Shandong 266510, China. (e-mail: zwolfe@163.com).

The contributions of this paper are: (1) a new weight strategy is proposed to improve the prediction accuracy for unbalanced dataset; (2) a cascade architecture is used to compute the incremental dataset in parallel; on the other hand, it could also be considered as data distributed on processors. This case is useful for application in reality. We briefly review this important contribution in section 2, together with the proximal linear SVM algorithm. Section 3 presents the parallel incremental proximal linear SVM (PIPLSVM). Section 4 gives the experiments which have been conducted to investigate the properties of the proposed algorithm on large scale unbalanced dataset and summarize its generalization performance. Finally, we conclude this paper in section 5.

II. Improvement to Proximal Linear SVM

A. Proximal Linear SVM

Consider a binary classification problem as follows. Given dataset $S=\{(\boldsymbol{x}_1,y_1),\cdots,(\boldsymbol{x}_l,y_l)\}\subset R^m\times\{-1,1\}$, where $(\boldsymbol{x}_i,y_i)$ is called training example, $\boldsymbol{x}_i$ called input or pattern, y_i called output or label. The target is to construct a decision function from dataset S then predict the label of a new input. Proximal Linear SVM [3] is:

$$\min_{(w,b,\xi)\in R^{m+1+l}} \frac{1}{2}\left(\boldsymbol{w}^T\boldsymbol{w}+b^2\right)+\frac{C}{2}\boldsymbol{\xi}^T\boldsymbol{\xi} \qquad (1)$$
$$s.t. \quad \boldsymbol{Y}(\boldsymbol{Aw}+b\boldsymbol{e})+\boldsymbol{\xi}-\boldsymbol{e}=0$$

where $C>0$ is the weight parameter, $\boldsymbol{A}=(\boldsymbol{x}_1,\cdots,\boldsymbol{x}_l)^T$ is $l\times m$ matrix consisted of inputs, $\boldsymbol{Y}$ is the diagonal matrix of label y_i corresponding to input $\boldsymbol{x}_i$ in $\boldsymbol{A}$, $\boldsymbol{e}$ is a column vector of all ones. Thus, the decision function is:

$$h(\boldsymbol{x})=\mathrm{sgn}\left(\boldsymbol{w}^T\boldsymbol{x}+b\right) \qquad (2)$$

As in Fig. 1, the hyper-plane resulted from Proximal Linear SVM $\boldsymbol{w}^T\boldsymbol{x}+b=\pm1$ is not bounding planes like in standard SVM, but can be though of as proximal planes, around which the points of each class are clustered and which are pushed as far apart as possible by the term $\boldsymbol{w}^T\boldsymbol{w}+b^2$ in the objective function.

B. Class Weighted Proximal Linear SVM

For balanced dataset, the Proximal Linear SVM (PLSVM) performs well in classification; however, for unbalanced dataset, the plane biases to the class with more examples, thus the prediction accuracy decreases as depicted in Fig 2. To avoid this shortcoming, Liu proposed a class weighted PLSVM [4].

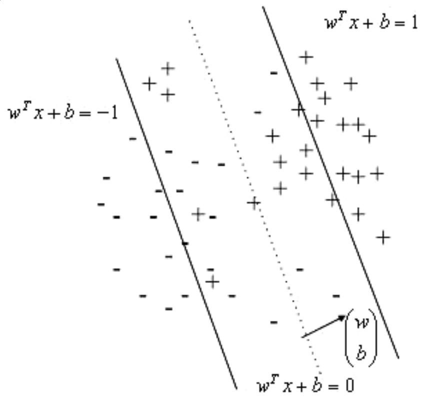

Fig. 1 PLSVM on balanced dataset

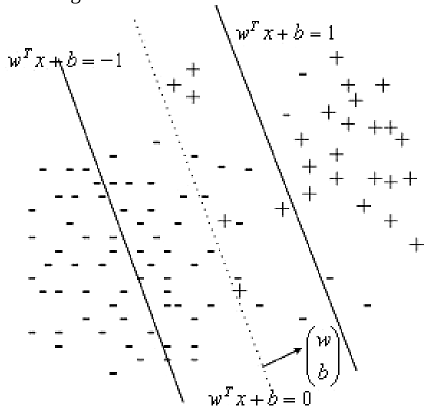

Fig. 2 PLSVM on unbalanced dataset

Suppose there are l training examples, l_1, l_2 are the number of examples of class plus and class minus, respectively. Liu defines $l \times l$ diagonal matrix $\boldsymbol{N}$ as the weight matrix to tune the training error. The element of $\boldsymbol{N}$ is defined as:

$$N_{ii} = \begin{cases} 1/l_+, & if\ y_i = 1 \\ 1/l_-, & if\ y_i = -1 \end{cases}$$

For very large dataset, $1/l_+$ and $1/l_-$ are approximate to zero, thus the prediction accuracy becomes worse than PLSVM. For instance, suppose l_+ is larger, this means that the weight of positive class is very small, thus losses more classification information and results bad predication performance. Here, we take the following strategy to define the weight matrix.

$$N_{ii} = \begin{cases} l_-/(l_+ + l_-), & if\ y_i = 1 \\ l_+/(l_+ + l_-), & if\ y_i = -1 \end{cases} \quad (3)$$

Then, the model (1) can be written as:

$$\min_{(\boldsymbol{w},b,\xi)\in R^{m+1+l}} \frac{1}{2}\left(\boldsymbol{w}^T\boldsymbol{w} + b^2\right) + \frac{C}{2}\xi^T \boldsymbol{N}\xi \quad (4)$$
$$s.t. \quad \boldsymbol{Y}(\boldsymbol{A}\boldsymbol{w} + b\boldsymbol{e}) + \xi - \boldsymbol{e} = 0$$

III. PIPLSVM

A. Incremental Proximal Linear SVM

To solve model (4), substituting for ξ from the equal constraints in terms of $\boldsymbol{w}, b$, we have the unconstrained optimization problem:

$$\min_{(\boldsymbol{w},b)} \frac{1}{2}\left(\boldsymbol{w}^T\boldsymbol{w} + b^2\right) +$$
$$\frac{C}{2}\left(\boldsymbol{Y}(\boldsymbol{A}\boldsymbol{w} + b\boldsymbol{e}) - \boldsymbol{e}\right)^T \boldsymbol{N}\left(\boldsymbol{Y}(\boldsymbol{A}\boldsymbol{w} + b\boldsymbol{e}) - \boldsymbol{e}\right) \quad (5)$$

Set the gradient to zero with respect to $\boldsymbol{w}, b$ gives:

$$\begin{pmatrix} \boldsymbol{w} \\ b \end{pmatrix} = \left(\frac{\boldsymbol{I}}{C} + \begin{pmatrix} \boldsymbol{A}^T \\ \boldsymbol{e}^T \end{pmatrix} \boldsymbol{YNY}(\boldsymbol{A} \quad \boldsymbol{e})\right)^{-1} \boldsymbol{YNe}$$

Defining $\boldsymbol{E} = (\boldsymbol{A} \quad \boldsymbol{e})$, noting that $\boldsymbol{YNY} = \boldsymbol{N}$, then:

$$\begin{pmatrix} \boldsymbol{w} \\ b \end{pmatrix} = \left(\frac{\boldsymbol{I}}{C} + \boldsymbol{ENE}\right)^{-1} \boldsymbol{E}^T\boldsymbol{YNe} \quad (6)$$

This expression only involves solving a $(m+1)\times\ (m+1)$ linear equations. In addition, due to $m << l$ in many cases, we can expect to obtain a fast and efficient algorithm.

Suppose $\boldsymbol{A} = \begin{pmatrix} \boldsymbol{A}_1 \\ \boldsymbol{A}_2 \end{pmatrix}$, then:

$$\boldsymbol{Y} = \begin{pmatrix} \boldsymbol{Y}_1 & 0 \\ 0 & \boldsymbol{Y}_2 \end{pmatrix}, \ \boldsymbol{N} = \begin{pmatrix} \boldsymbol{N}_1 & 0 \\ 0 & \boldsymbol{N}_2 \end{pmatrix},$$
$$\boldsymbol{E} = \begin{pmatrix} \boldsymbol{E}_1 \\ \boldsymbol{E}_2 \end{pmatrix} = \begin{pmatrix} \boldsymbol{A}_1 & \boldsymbol{e}_1 \\ \boldsymbol{A}_2 & \boldsymbol{e}_2 \end{pmatrix}.$$

It is easy to compute that:

$$\boldsymbol{E}^T\boldsymbol{NE} = \boldsymbol{E}_1^T\boldsymbol{N}_1\boldsymbol{E}_1 + \boldsymbol{E}_2^T\boldsymbol{N}_2\boldsymbol{E}_2 \quad (7)$$
$$\boldsymbol{E}^T\boldsymbol{YNe} = \boldsymbol{E}_1^T\boldsymbol{Y}_1\boldsymbol{N}_1 e_1 + \boldsymbol{E}_2^T\boldsymbol{Y}_2\boldsymbol{N}_2 e_2$$

By observing this, we can acquire an incremental algorithm, which could increase or decrease arbitrary number of examples. For simple, we denote the decreased example dataset as $A_1 \in R^{l_1\times m}$, the increased dataset as $A_2 \in R^{l_2\times m}$, and $\boldsymbol{E}_1 = (\boldsymbol{A}_1 \quad \boldsymbol{e}_1)$, $\boldsymbol{E}_2 = (\boldsymbol{A}_2 \quad \boldsymbol{e}_2)$, correspondingly. Then, the incremental proximal linear SVM algorithm is:

Algorithm 2.1 Incremental Proximal Linear SVM

Compute $\boldsymbol{w}, b$ as follows:

$$\begin{pmatrix} \boldsymbol{w} \\ b \end{pmatrix} = \left(\frac{\boldsymbol{I}}{C} + \boldsymbol{E}^T\boldsymbol{NE} - \boldsymbol{E}_1^T\boldsymbol{N}_1\boldsymbol{E}_1 + \boldsymbol{E}_2^T\boldsymbol{N}_2\boldsymbol{E}_2\right)^{-1} \times$$
$$\left(\boldsymbol{E}^T\boldsymbol{YNe} - \boldsymbol{E}_1^T\boldsymbol{Y}_1\boldsymbol{N}_1 e + \boldsymbol{E}_2^T\boldsymbol{Y}_2\boldsymbol{N}_2 e\right) \quad (8)$$

Get the classifier according to (2).

When needing to increase or decrease examples, return step 1.

Obviously, all we need to store is the relatively small

$(m+1)\times(m+1)$ matrix, and $(m+1)\times 1$ column vector. Thus, this method could reduce the storage requirement greatly. In addition, this algorithm could process examples in real time, either increasing new example or decreasing old example.

B. PIPLSVM

For large scale learning problem, parallel is a very good strategy. For instance, if the training time of a problem is t, and we use p processors to solve this problem in parallel, then in ideal, it takes t/p time for each processor to solve it.

Following the divide and conquer principle, Graf at al. proposed a cascade architecture to training SVM in parallel [6, 7]. To save the training time, we use this cascade architecture to compute the incremental data in parallel. Fig. 3 gives a cascade architecture example of four processors.

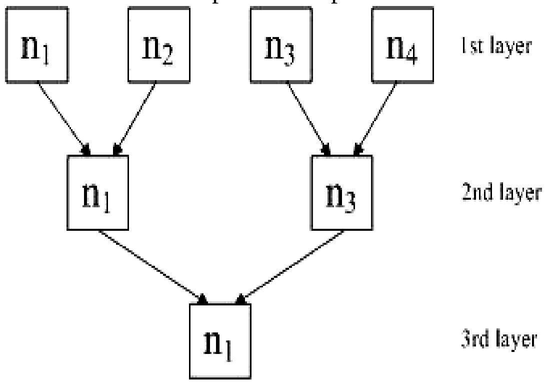

Fig. 3 Cascade architecture of four processors

In reality, the large amount data is usually distributed on different processors, if using this method, through training local dataset in parallel; it could greatly save training time. Thus we present the algorithm as follows.

Algorithm 3.1 PIPLSVM

Initialization: p is the number of processors, the index set $M=\{1,\cdots,P\}$, define processor 1 as the master, the dataset of each processor is $\boldsymbol{A}_i, i\in M$, the parameter $C=1$;

Processor $i\in M$ passes the number of examples of class plus and minus to master, then master compute $l_+=\sum_{i=1}^{p} l_+^i$, $l_-=\sum_{i=1}^{p} l_-^i$ to define the weight matrix $\boldsymbol{N}$, then broadcast $\boldsymbol{N}$ to each processor;

Processor $i\in M$ compute $\boldsymbol{E}_i^T\boldsymbol{N}_i\boldsymbol{E}_i$, $\boldsymbol{E}_i^T\boldsymbol{Y}_i\boldsymbol{N}_i\boldsymbol{e}$;

According to cascade architecture depicted in Figure 3, reduce the data computed in setp 3 to master, then computed $\boldsymbol{w}, b$ using (8) and get the decision function (2).

If needing to increase or decrease examples then update the example dataset of each processor and return step 2.

IV. Experiment

The simulation environment is a cluster system of 8 computer nodes, with CPU Pentium 4 2.40GHZ, 256M memory of each node. The operating system is Windows NT 5.1, with mpich2 and Microsoft Visual Studio C++ installed. The weight matrix $\boldsymbol{N}$ is computed according to (3).

To validate the proposed algorithm, we use adult [10] dataset for comparison experiment. We take 20,000 negative class examples, 120,000 positive class examples to simulate the large scale unbalanced learning problem. After training, the test is conducted by using 5,000 positive class examples and 5,000 negative class examples.

During experiment, the number of negative class examples keeps 20,000 without change. Each time we only increase the number of positive class examples, so as to better simulate the effect of unbalanced property to prediction accuracy. The parameter $C\equiv 1$ and all experiment results are averaged by running the algorithm 5 times.

As Table 1 shows, the weight strategy (3) is better than class weighted strategy and the original PLSVM; it is stable when the problem becomes more unbalanced according to the increasing of positive class examples. Obviously, the prediction accuracy of our weight matrix strategy is much bigger than that of the other two weight strategy. This indicates the better efficiency of our method.

Table 1 Prediction accuracy comparison of different weight strategies

Number of examples	PLSVM	Class weighted PLSVM	Weight (3)'s PLSVM
100, 000	87.92%	82.91%	89.77%
120, 000	87.89%	82.56%	88.96%
140, 000	87.80%	82.89%	89.54%

To validate algorithm 3.1, we construct an eight-node cluster system, use mpich2[1] to implement the algorithm and introduce two indexes--speedup rate and parallel efficiency [7, 8]-- to test the efficiency. The experiment performs on 100,000 and 140,000 training examples, respectively.

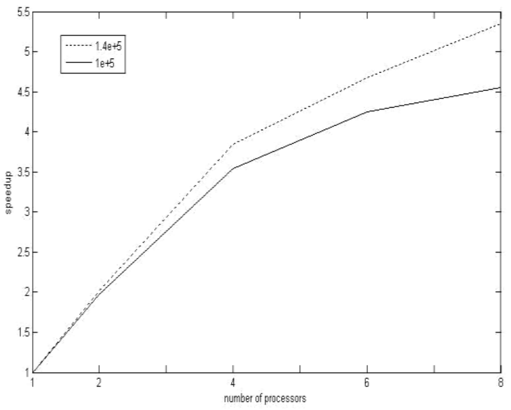

Fig. 4 Change of speedup rate

As Fig.4 shows, with the increase of the number of processors p, the speedup rate increases nearly linear; then up to the communication time taking large amount of the whole training time, the increasing of speedup rate slow. The real curve line of 100,000 examples shows this more obvious. Therefore, it is reasonable to expect that our algorithm will

1 http://www.mcs.anl.gov/research/projects/mpich2/

perform well for larger number of examples.

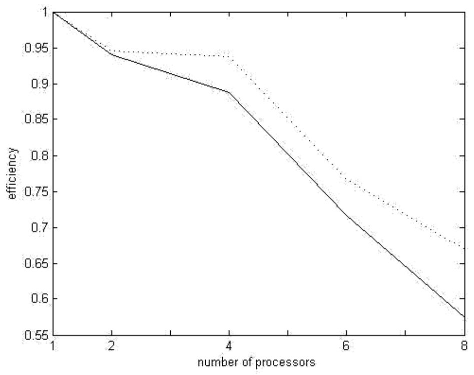

Fig. 5 Variation curve of parallel efficiency

Fig. 5 presents the parallel efficiency curve of 100,000 and 140,000 training examples when the number of processors increases. The dashed curve stands for 140,000 training examples' case. The greatest parallel efficiency reaches at $p = 2$, and the value is 0.945. Then, with the increasing of processors, this curve descends slightly. The main reason is: the communication time taking large amount of the whole training time, the increasing of parallel efficiency slow.

V. Conclusion

Both PLSVM and class weighted PLSVM are efficient algorithm for large scale learning problem; however, they confronts difficult for large scale unbalanced dataset. This paper proposed a new strategy for defining the weight matrix. This strategy performs better than Liu's method. Moreover, for saving training time, we use cascade architecture to computing the training data in parallel, thus greatly saving training time. The experiments show that: this algorithm could not only improve prediction accuracy, but also increase the training speed, save more training time.

References

[1] Naiyang Deng, Yingjie Tian, Optimization methods of data mining --support vector machine, Bei Jing, China: Science Press, 2004, pp. 30-190.

[2] R. Akbani, S. Kwek, N. Japkowicz. Applying support vector machines to imbalanced datasets. European Conference on Machine Learning, 2004, pp. 39-50.

[3] G. Fung, O. L. Mangasarian. Proximal support vector machine classifier, In F. Provost, R. Srikant, editors, Proceedings KDD-2001: Knowledge Discovery and Data Mining, New York: ACM, 2001. pp. 77-86.

[4] Quanchang Liu. Support vector machine and its application in intrusion detection, Master's thesis, College of Information Science and Engineering, Shandong University of Science and Technology, China, 2008.

[5] G. Fung, O. L. Mangasarian, Incremental support vector machine classification, In R. Grossman, H. Mannila, R. Motwani, editors, Proceedings of the Second SIAM International Conference on Data Mining, Virginia: SIAM, Philadelphia, 2002, pp. 247-260.

[6] Hans Peter Graf, Eric Cosatto, Leon Bottou, Igor Durdanovic, Vladimir Vapnik. Parallel support vector machines: the cascade SVM, In L. Saul, Y. Weiss, L. Bottou, editors, Advances in NIPS 17, Cambridge, MA: MIT Press, 2005, pp. 521-528.

[7] Y. M. Wen, B. L. Lu. A cascade method for reducing training time and the number of support vectors. Advances in Neural Networks-ISNN2004, LNCS 3173(1), 2004, pp. 480-485.

[8] L. J. Cao, S. S. Keerthi, C. J. Ong, P. Uvaraj, X. J. Fu, H. P. Lee. Developing parallel sequential minimal optimization for fast training support vector machine, Neurocomputing, Elsevier, Amsterdam, Netherlands, 2006, 70(1), pp. 93-104.

[9] Michael J. Quinn. Parallel programming in C with API and Open MP, New York, USA: McGraw-Hill, 2004, pp. 138-180.

[10] A. Asuncion, D. J. Newman. UCI machine learning repository, http://www.ics.uci.edu/~mlearn/MLRepository.html. Irvine, CA: University of California, School of Information and Computer Science. 2007.

Xin Zhang received the master's degree in Mathematics from the Institute of Operations Research, Shandong University of Science and Technology in 2009. Currently, he is pursuing PhD of Electronic Engineering at City University of Hong Kong. His research interests include nonlinear optimization, support vector machine and computer vision.

GengFeng Zhu is pursuing the master's degree in Mathematics at Institute of Operations Research, Shandong University of Science and Technology. His research interests include nonlinear optimization and machine learning.