The First International Symposium on Optimization and Systems Biology (OSB'07)
Beijing, China, August 8–10, 2007
Copyright © 2007 ORSC & APORC, pp. 358–366

Decremental Learning Algorithms for Nonlinear Langrangian and Least Squares Support Vector Machines

Hua Duan[1,2,*] Hua Li[2] Guoping He[2]
Qingtian Zeng[2]

[1] Department of Mathematics, Shanghai Jiaotong University, Shanghai 200240, P.R.China
[2] College of Information Science and Engineering,
Shandong University of Science and Technology, Qingdao 266510, P.R.China

Abstract Langrangian Support Vector Machine (LSVM) and Least Squares Support Vector Machine (LSSVM) are two quick and effective classification methods. In this paper, we first introduce the mathematical models for LSVM and LSSVM and analyze their properties. In the nonlinear case, Sherman-Morrison-Woodbury identity is not used to compute the inversion of a matrix. According to block computation of a matrix and properties of a symmetric and positive-definite matrix, an approach to compute the inversion of a matrix is obtained and applied in the decremental learning algorithms for nonlinear LSVM and LSSVM. The online and batch decremental learning algorithms for nonlinear LSVM and LSSVM are presented, respectively, in which it is not necessary to relearn since the inversion of matrix after decrement is solved based on the former information. Thus, the computation time can be reduced. Through experiments, it is shown that the algorithms proposed in this paper can reduce the computation time.

1 Introduction

Support vector machine (SVM) is a new machine learning technique based on optimization methods, which is introduced by Vapnik [1][8] in the last decades of the 20 century. Since SVM has good learning and generalization capacity, it has become one of the most useful tools to overcome the challenges in machine learning, for example, nonlinear, dimension disaster, over learning and so on.

With the increment of the scale of the training set, the computation of the Lagrangian multiplier and kernel function is also increased. The approaches to reduce the scale of the training set must be adopted otherwise the computation time will increase and the processor will be overloaded. To overcome this problem, the decremental learning of SVM is introduced. There are two kinds of decremental learning of SVM: (1) **online decremental learning**: there is one and only one sample to be discarded from the training set for each decrement; and (2) **batch decremental learning**: there are more than one samples to be discarded from the training set

*Email: hduan@sdust.edu.cn

for each decrement. Cauwenberghs and Poggio[6] presented the online incremental and decremental learning algorithms for SVM, where the decremental learning is regarded as an reversible procedure of the incremental learning, and offers an efficient method to exactly evaluate leave-one-out generalization performance. Tveit etc.[7] presented decremental algorithm for SVM based on decay coefficients, which is compared with an existing window-based decremental algorithms, and showed it provided significantly better computational performance.

Lagrangian support vector machine (LSVM) is a quick and simple classification method[3] which is an implicit Lagrangian[5] formulation for the dual of a simple reformulation of the standard linear SVM quadratic programming (QP) problem. Least squares support vector machine (LSSVM) was developed by Suykens [10] in which analytical solutions can be obtained by solving linear equations instead of a QP problem. For linear LSVM and LSSVM, the inversion of m-order matrix in the algorithm is transformed into the inversion of $(n+1)$-order matrix by using the Sherman-Morrison-Woodbury (SMW) identity $(n \ll m)$[2].

For nonlinear LSVM and LSSVM, it can not process large scale problems because SMW identity can not be used for the inversion of matrix. Especially, the computation in the nonlinear case is more complex than that in the linear case. Therefore, the decremental learning algorithms for nonlinear LSVM and LSSVM are necessary to be researched. In this paper, we propose the online and batch decremental learning algorithms for nonlinear LSVM and LSSVM to reduce the computation time, respectively. In the decremental learning algorithms, the inversion of matrix after decrement is solved by using the results before decrement, in which it is not necessary to re-learn the new classification problem. Experiments shown that the computation time can be reduced by our algorithms proposed in this paper.

This paper is organized as follows. Section 2 presents LSVM and LSSVM. Section 3 gives the decremental learning algorithms for LSVM and LSSVM. In section 4, the algorithms will be experimentally evaluated. Section 5 concludes the whole paper and gives the discussions.

2 LSVM and LSSVM

First, we describe our notations used for the discussions in this paper.

(1) Let $T=\{(x_i,y_i)|x_i \in R^n, i=1,\cdots,m\}$ be the training set of a classification problem, where x_i is the sample point of an n-dimensional space, represented by $A^T_{m\times n}=(x_1,\cdots,x_m)$, and $y_i \in \{\pm 1\}$ be the labels of the positive and negative class as to x_i where $i=1,\cdots,m$, represented by $D_{m\times m}=diag(y_1,\ldots,y_m)$.

(2) For any vector $x \in R^n$, K is a subset of the subscript set of x. $x\backslash K$ is obtained by discarding the elements $\{x_i|i \in K\}$ from x.

(3) The identity matrix of arbitrary dimension will be denoted by I. An elementary matrix $P(i,j)$ is obtained by transforming the ith row (or column) and jth row (or column) of I.

2.1 LSVM: Langrangian Support Vector Machines

Linear LSVM is defined by:

$$\begin{array}{ll} \min & \frac{1}{2}(\|w\|^2+b^2)+\frac{C}{2}\xi^T\xi \\ s.t. & y_i((w\cdot x_i)+b)+\xi_i \geq 1 \end{array} \tag{1}$$

where $C > 0$ is the penalty parameter. The dual problem is:

$$\min_{0\leq\alpha\in R^m} \frac{1}{2}\alpha^T Q\alpha - e^T\alpha \tag{2}$$

where $Q = \frac{I}{C} + HH^T$, $H = D[A\ \ -e]$, $e = (1,\ldots,1)^T$ is a vector of ones of the appropriate dimension.

The optimization KKT condition of its dual problem is: $0 \leq \alpha \perp Q\alpha - e \geq 0$.

By using the identity between any two real numbers (or two vectors) a and b:

$$0 \leq a \perp b \geq 0 \Longleftrightarrow a = (a-\lambda b)_+,\ \ \lambda > 0$$

where $(x)_+$ denotes the vector in R^n in which all of its negative components are set to zero.

The iteration formula given by LSVM algorithm is

$$\alpha^{i+1} = Q^{-1}(e+((Q\alpha^i - e)-\lambda\alpha^i)_+),\ \lambda > 0,\ \ i = 0,1,\ldots \tag{3}$$

While $0 < \lambda < \frac{2}{C}$, the algorithm is the global linear convergence from any starting point [3]. The inversion of m matrix Q changes to the inversion of $n+1(n \ll m)$ matrix by using SMW identity. This leads to process large data sets feasibly, and the computation time is reduced.

The SMW identity is:

$$\left(\frac{I}{C}+HH^T\right)^{-1} = C\left(I - H(\frac{I}{C}+H^TH)^{-1}H^T\right)$$

where $C > 0$ and H is an $m\times n$ matrix.

The above discussions are about linear LSVM. In the nonlinear case, we introduce kernel function $K(x,y) = \Phi(x)^T\Phi(y)$, and let $H = [A\ \ -e]$. $Q = \frac{I}{C} + DK(H,H^T)D$ in the nonlinear case. In the above discussions, we replace Q using $Q = \frac{I}{C} + DK(H,H^T)D$ and keep other contents invariant, the results in the nonlinear case can be obtained. However, the SMW identity is not applicable.

2.2 LSSVM: Least Squares Support Vector Machine

Linear LSSVM is defined by:

$$\begin{array}{ll} \min & \frac{1}{2}(\|w\|^2+\frac{C}{2}\sum_{i=1}^{m}\xi_i^2 \\ s.t. & y_i((w\cdot x_i)+b) = 1-\xi_i \Longrightarrow (w\cdot x_i)+b = y_i - \xi_i y_i \\ & i = 1,\ldots,m \end{array} \tag{4}$$

And, its Lagrangian function is:

$$L=\frac{1}{2}\|w\|^2+\frac{C}{2}\sum_{i=1}^{m}\xi_i^2-\sum_{i=1}^{m}\alpha_i((w\cdot x_i)+b-y_i+\xi_i y_i)$$

where α_i is the Lagrangian multiplier.

The KKT condition is:

$$\begin{cases} \frac{\partial L}{\partial w}=0\longrightarrow w=\sum\limits_{i=1}^{m}\alpha_i x_i \\ \frac{\partial L}{\partial b}=0\longrightarrow \sum\limits_{i=1}^{m}\alpha_i=0 \\ \frac{\partial L}{\partial \xi_i}=0\longrightarrow C\xi_i=\alpha_i y_i \\ \frac{\partial L}{\partial \alpha_i}=0\longrightarrow w\cdot x_i+b+\xi_i y_i-y_i=0 \end{cases} \tag{5}$$

After eliminating w and ξ from the above KKT condition (5),

$$\begin{pmatrix} 0 & e^T \\ e & Q \end{pmatrix}\begin{pmatrix} b \\ \alpha \end{pmatrix}=\begin{pmatrix} 0 \\ y \end{pmatrix} \tag{6}$$

where $\alpha=(\alpha_1,\ldots,\alpha_m)^T$, $y=(y_1,\ldots,y_m)^T$, $Q=AA^T+\frac{1}{C}I$.

Therefore, the classification problem (4) can be solved by solving the linear equations (6). Thus, the computation time can be reduced. For the nonlinear case, to solve LSSVM is similar to that in the linear case, and the only difference is that Q is replaced by $Q=K(A,A^T)+\frac{1}{C}I$ where $K(x,y)$ is the kernel function.

By solving the linear equations (6),

$$b=\frac{e^TQ^{-1}y}{e^TQ^{-1}e},\quad \alpha=Q^{-1}\left(y-\frac{ee^TQ^{-1}y}{e^TQ^{-1}e}\right) \tag{7}$$

According to (7), Q^{-1} is the key problem to solve LSSVM.

In the linear case, Q^{-1} can be solved using SMW identity to reduce computation time. However, in the nonlinear case, SMW identity can not be used for Q^{-1} to reduce computation time. In the following discussions, we mainly focus on decremental learning algorithms for the nonlinear LSVM and LSSVM to reduce computation time.

3 Decremental Learning Algorithms for LSVM and LSSVM

3.1 Primary Knowledge

First, a lemma used in the following discussions is introduced.

Lemma 1[9] Let matrix $T=\begin{pmatrix} A & B \\ C & D \end{pmatrix}$ and D are inverse, then

$$T^{-1}=\begin{pmatrix} (A-BD^{-1}C)^{-1} & -(A-BD^{-1}C)^{-1}BD^{-1} \\ -D^{-1}C(A-BD^{-1}C)^{-1} & D^{-1}C(A-BD^{-1}C)^{-1}BD^{-1}+D^{-1} \end{pmatrix}$$

For Q is a m-order symmetric and positive-definite matrix, Q^{-1} is known. First, we have elementary transformations on Q. Assume that there are m samples in the original training set, and there are k samples discarded in each decrement. The discarded sample set is $\{(x_{\imath(1)},y_{\imath(1)}),\dots,(x_{\imath(k)},y_{\imath(k)})\}$. We transform the corresponding rows and columns of the k samples to be discarded to the first k rows and k columns. Without loss of generalization, we assume that the subscript set of the k samples discarded is $\{\imath(1),\dots,\imath(k)\}$. Q_{m-k} is used to represent the matrix obtained by discarding the k samples from Q. Let $P(k,\imath(k))\cdots P(1,\imath(1))QP(1,\imath(1))$ $\cdots P(k,\imath(k))=\widehat{Q}$, $\widehat{Q}^{-1}=P(k,\imath(k))\cdots P(1,\imath(1))Q^{-1}P(1,\imath(1))\cdots P(k,\imath(k))=U$.

We have block decomposition on $\widehat{Q}$ and U as following:

$$\widehat{Q}=\begin{pmatrix} \widehat{Q}_{11} & \widehat{Q}_{12} \\ \widehat{Q}_{12}^T & Q_{m-k} \end{pmatrix}, \qquad U=\begin{pmatrix} U_{11} & U_{12} \\ U_{12}^T & U_{m-k} \end{pmatrix}$$

where $\widehat{Q}_{11}$ and U_{11} are $k\times k$ matrixes, and $\widehat{Q}_{12}$ and U_{12} are $k\times(m-k)$ matrixes.

According to Lemma 1, $\widehat{Q}^{-1}$ can be written as:

$$\widehat{Q}^{-1}=\begin{pmatrix} (\widehat{Q}_{11}-\widehat{Q}_{12}Q_{m-k}^{-1}\widehat{Q}_{12}^T)^{-1} & -(\widehat{Q}_{11}-\widehat{Q}_{12}Q_{m-k}^{-1}\widehat{Q}_{12}^T)^{-1}\widehat{Q}_{12}Q_{m-k}^{-1} \\ -Q_{m-k}^{-1}\widehat{Q}_{12}^T(\widehat{Q}_{11}-\widehat{Q}_{12}Q_{m-k}^{-1}\widehat{Q}_{12}^T)^{-1} & Q_{m-k}^{-1}\widehat{Q}_{12}^T(\widehat{Q}_{11}-\widehat{Q}_{12}Q_{m-k}^{-1}\widehat{Q}_{12}^T)^{-1}\widehat{Q}_{12}Q_{m-k}^{-1}+Q_{m-k}^{-1} \end{pmatrix}$$

By $\widehat{Q}^{-1}=U$,

$$\begin{array}{l} (\widehat{Q}_{11}-\widehat{Q}_{12}Q_{m-k}^{-1}\widehat{Q}_{12}^T)^{-1}=U_{11} \\ -(\widehat{Q}_{11}-\widehat{Q}_{12}Q_{m-k}^{-1}\widehat{Q}_{12}^T)^{-1}\widehat{Q}_{12}Q_{m-k}^{-1}=U_{12} \\ -Q_{m-k}^{-1}\widehat{Q}_{12}^T(\widehat{Q}_{11}-\widehat{Q}_{12}Q_{m-k}^{-1}\widehat{Q}_{12}^T)^{-1}=U_{12}^T \\ Q_{m-k}^{-1}\widehat{Q}_{12}^T(\widehat{Q}_{11}-\widehat{Q}_{12}Q_{m-k}^{-1}\widehat{Q}_{12}^T)^{-1}\widehat{Q}_{12}Q_{m-k}^{-1}+Q_{m-k}^{-1}=U_{m-k} \end{array} \tag{8}$$

According to (8),

$$Q_{m-k}^{-1}=U_{m-k}-U_{12}^TU_{11}^{-1}U_{12} \tag{9}$$

In (9), if $k>1$, the above discussions are about batch decremental learning. If $k=1$, U_{11} in (9) is a positive number, and this case is about online decremental learning. In the following, the online and batch decremental learning algorithms are presented together.

3.2 Decremental Learning Algorithm for LSVM

According to the above discussions, Q_{m-k}^{-1} can be obtained by Q^{-1}, in which it is not necessary to recompute Q_{m-k}^{-1}, thus the computation time can be reduced.

According to the iteration formula (3) of LSVM, the solution α_{new} after decrement,

$$\alpha_{new}^{i+1} = Q_{m-k}^{-1}\left(\left((Q_{m-k}\alpha_{new}^{i} - e) - \lambda\alpha_{new}^{i}\right)_{+} + e\right), i = 0,1,2,\ldots,\ \lambda > 0 \qquad (10)$$

where $\alpha_{new}^{0} = \alpha\backslash\{\imath(1),\cdots,\imath(k)\}$ and α is the optimal solution before decrement.

Therefore, we can present the decremental learning algorithm for LSVM as following.

Step 1 Select parameter λ such that $0 < \lambda < \frac{2}{C}$, parameter C, and precision $\varepsilon > 0$. Obtain the sample set to be discarded $\{(x_{\imath(1)}, y_{\imath(1)}), \ldots, (x_{\imath(k)}, y_{\imath(k)})\}$;

Step 2 Let $\alpha_{new}^{0} = \alpha\backslash\{\imath(1),\cdots,\imath(k)\}$, and let $i = 0$;

Step 3 The inversion of Q_{m-k} is solved using (9);

Step 4 Using the iteration formula (10) to compute α_{new}^{i+1};

Step 5 If $\|\alpha_{new}^{i+1} - \alpha_{new}^{i}\| \leq \varepsilon$, the new solution is obtained. Otherwise, let $i = i+1$ and go to Step 4.

3.3 Decremental Learning Algorithm for LSSVM

The linear equation after decrement,

$$\begin{pmatrix} 0 & e^T \\ e & Q_{m-k} \end{pmatrix} \begin{pmatrix} b^{new} \\ \alpha^{new} \end{pmatrix} = \begin{pmatrix} 0 \\ y^{new} \end{pmatrix} \qquad (11)$$

where $\alpha_{new} = \alpha\backslash\{\imath(1),\cdots,\imath(k)\}$ and α is the optimal solution before decrement.

Based on (11), the new solution after decrement is:

$$b^{new} = \frac{e^T Q_{m-k}^{-1} y^{new}}{e^T Q_{m-k}^{-1} e}, \quad \alpha^{new} = Q_{m-k}^{-1}\left(y^{new} - \frac{ee^T Q_{m-k}^{-1} y^{new}}{e^T Q_{m-k}^{-1} e}\right) \qquad (12)$$

The decremental learning algorithm for LSSVM can be presented as following.

Step 1 Obtain the sample set to be discarded $\{(x_{\imath(1)}, y_{\imath(1)}), \ldots, (x_{\imath(k)}, y_{\imath(k)})\}$;

Step 2 Formula (9) is used to compute the inversion of Q_{m-k} ;

Step 3 Formula (12) is used to solve the new solution.

4 Experiment

In this section, we present the experiments to evaluate the performance of the online and batch decremental learning algorithms for LSVM and LSSVM.

In the following discussions, in order to save space, we denote the online (batch) decremental learning algorithm proposed in this paper for LSVM as **DOLSVM** (**DBLSVM**) and online (batch) decremental learning algorithm for LSSVM as **DOLSSVM** (**DBLSSVM**), respectively. The approach to solve the inversion of a matrix directly for the online (batch) decremental learning algorithm for LSVM is denoted as **OLSVM** (**BLSVM**) and the online (batch) decremental learning algorithm for LSSVM is denoted as **OLSSVM** (**BLSSVM**), respectively.

The experiments are given based on UCI machine learning database (http://www.ics.uci.edu). The experiments are implemented by Matlab 7.0, and they run on PC environment. The main configurations of the PC are: (1) CPU: Pentium IV 2.0G, (2) Memory: 256M, and (3) OS: Windows XP. All the experiments are presented for the nonlinear cases, and the kernel function is Gaussian Radial Basis Kernel $K(x,y) = \exp(-\|x-y\|^2/2\sigma^2)$ and $\sigma = 2$. For LSVM, we take $\lambda = \frac{1.9}{C}$ where C is the penalty factor and $C = 1/m$ where m is the sample number of training set, and the precision is $\varepsilon = 10^{-5}$.

Since to solve the inversion of the matrix is the only difference between the algorithms proposed in this paper and the approaches to solve the inversion of a matrix directly, the training correctness and testing correctness will keep invariant. Therefore, in the experiments, we only compare the differences of CPU running time.

(1) Online case

The experimental results of online decremental learning for LSVM and LSSVM in the nonlinear case are shown in Table 1, in which we compare the difference between the computation time of OLSVM, DOLSVM, OLSSVM, and DOLSSVM. According to the results shown in Table 1, we can see that the CPU running time of DOLSVM and DOLSSVM is less than OLSVM and OLSSVM, respectively. When the scale of the training set is not large, their difference is not very clear. However, on the large-scale training set (such as Image1 and Image2), the difference of the CPU running time between DOLSVM and DOLSSVM and OLSVM and OLSSVM is obvious.

Table 1: The experimental results of online decremental learning for LSVM and LSSVM

Dataset	OLSVM	DOLSVM	OLSSVM	DOLSSVM
Thyoid (140×5)	1.40625	1.39065	1.54688	1.53128
Ringnorm (400×20)	11.3281	11.2812	11.9844	11.9375
German(720×20)	35.4688	35.1407	36.25	35.9219
Splice (1000×60)	76.0781	75.25	111.063	110.2349
Image1 (1300×18)	122.438	120.6411	125.547	123.7501
Image2 (2020×18)	289.326	282.4934	292.321	285.4884

(2) Batch case

The experimental results of batch decremental learning for LSVM and LSSVM in the nonlinear case are shown in Table 2. According to the results shown in Table 2, we can see:

- the CPU running time of DBLSVM and DBLSSVM is less than BLSVM and BLSSVM, respectively, especially on the large-scale training set.
- the value of k (k is the number of samples to be discarded in each decrement) is bigger, the more CPU running time will be saved.

Table 2: The experimental results of batch decremental learning LSVM and LSSVM

Dataset	k	BLSVM	DBLSVM	BLSSVM	DBLSSVM
Thyoid (140×5)	20	1.34375	1.26565	1.20313	1.12503
	50	1.375	1.3125	0.7343	0.6718
Ringnorm (400×20)	20	11.1563	11.0157	10.7344	10.5938
	50	11.1563	11.0625	9.25	9.1562
German(720×20)	20	34.3594	34.00	3.1563	33.7969
	50	34.9844	34.6719	31.3438	31.0313
Splice (1000×60)	20	75.2188	74.4063	104.094	103.2815
	50	75.0469	74.3438	98.6094	97.9063
Image1 (1300×18)	20	120.5	118.7344	120.156	118.3904
	50	122.125	120.5156	116.078	114.4686
Image2 (2020×18)	20	287.531	280.8591	273.766	267.0941
	50	281.109	275.0184	262.656	257.7654

5 Conclusion

LSVM and LSSVM are two fast and efficient algorithms for classification problems. With the increasing of the scale of the training set, the approaches to reduce the scale must be adopted otherwise the computation time will also increment and the processor will be overloaded. To overcome this problem, this paper proposes the online and batch decremental learning algorithms for LSVM and LSSVM. One of the main contributions in the proposed algorithms is that it is not necessary to re-learn the whole data set while part of samples are discarded. According to the experimental results, the algorithms proposed in this paper can reduce the CPU running time heavily and keep the training correctness and the testing correctness invariant.

In this paper, we only present the online and batch decremental learning algorithms for LSVM and LSSVM. In the proposed algorithms, how to select the samples to be discarded is not discussed. The standards to select the samples to be discarded for decremental learning algorithms of SVM are the future research work.

6 Acknowledgements

This work is supported partially by national science foundation of China (10571109 and 60603090).

References

[1] V.Vapnik. The nature of statistical learning theory, Springer-Verlag, New-York,1995.

[2] G.H. Golub and C.F. Van Loan. Matrix Computations. The John Hopkins University Press, Baltimore, Maryland, 3rd edition,1996.

[3] O.L.Mangasarian and D.R.Musicant. Lagrangian support vector machines. Journal of Machine Learning Research, 1:161-177, 2001.

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

[5] O.L.Mangasarian and M.V.Solodov. Nonlinear complementarity as unconstrained and constrained minimization. Mathematical Programming, Series B, 62:277-297, 1993.

[6] G.Cauwenberghs and T.Poggio. Incremental and decremental support vector machine learning. In Adv. Neural Information Processing, volume 13. MIT Press, 2001.

[7] A.Tveit,M.Lie and H.Engum. Incremental and decremental proximal support vector classification using decay coefficients. In Proc. 5th Int. Conf. on Data Warehousing and Knowledge Discovery, DaWak, Springer-verlag, 2003.

[8] N.Y.Deng and Y.J.TIAN. New Methods in Data Mining −− Support Vector Machines, Science Press, 2004.

[9] Peiking University. Advanced Algebra. China High Education Press, Beijing, 1987.

[10] J. A. K.Suykens and J.Vandewale. Least squares support vector machine classifiers. Neural Processing Letters, 9(3): 293-300, 1999.