Online Updating the Generalized Inverse of Centered Matrices

Qing Wang and **Liang Zhang**
School of Computer Science, Fudan University, 200433, Shanghai, China
{wangqing,lzhang}@fudan.edu.cn

Abstract

In this paper, we present the exact online updating formulae for the generalized inverse of centered matrices. The computational cost is $O(mn)$ for matrices of size $m \times n$. Experimental results validate the proposed method's accuracy and efficiency.

Introduction

The generalized inverse of an arbitrary matrix, also called Moore-Penrose inverse or Pseudoinverse, is an generalization of the inverse of full rank square matrix (Israel and Greville 2003). It has many applications in machine learning (Gutman and Xiao 2004), computer vision (Ng, Bharath, and Kin 2007), data mining (Korn et al. 2000), etc. For example, it allows for solving least square systems, even with rank deficiency, and the solution has the minimum norm which is the desired property under regularization.

Online learning algorithm often needs to update the trained model when some new observations arrive and/or some observations become obsolete. Online updating for the generalized inverse of original data matrix when a row or column vector is inserted or deleted, is given by the well-known Greville algorithm and Cline algorithm, respectively. The computational cost for one updation is $O(mn)$ on matrix with size $m \times n$.

However, in many machine learning algorithms, computation for the generalized inverse of **centered** data matrix other than the original data matrix is needed. For example, computing the generalized inverse of the laplacian matrix (Gutman and Xiao 2004) for some graph-based learning, computing the least square formulation for a class of generalized eigenvalue problems (Liu, Jiang, and Zhou 2009; Sun, Ji, and Ye 2009) which include LDA, CCA, OPLS, etc.

In this paper, we present the exact updating formulae for generalized inverse of centered matrix, when a row or column vector is inserted or deleted. The computational cost is also $O(mn)$ on matrix with size $m \times n$. Experimental results show that it could achieve high accuracy with low time cost.

Notations: Let $C^{m\times n}$ denotes the set of all $m \times n$ matrices over the field of complex numbers. The symbols $A^\dagger$ and A^* stand for the generalized inverse and the conjugate transpose of matrix $A \in C^{m\times n}$, respectively. I is the identity matrix and **1** is a vector of all ones.

Updating for Original Data Matrices

In this section, we briefly introduce Greville and Cline algorithm for updating the generalized inverse of the original data matrix when a column vector is appended or the first column vector is deleted. Extension to insertion or deletion of **any** column can be easily transformed based on equation $(AQ)^\dagger = Q^* A^\dagger$ where Q is unitary matrix. To the case of the row, it can be transformed into column case based on equation $A^\dagger = [(A^*)^\dagger]^*$.

The incremental computation for the generalized inverse of matrix is given by the well-known Greville algorithm (Israel and Greville 2003), which is shown in Lemma 1 below.

Lemma 1 (Greville Algorithm) *Let $\widehat{A} = [A, a] \in C^{m\times n}$, where $A \in C^{m\times(n-1)}$ and $a \in C^{m\times 1}$. Define $c = (I - AA^\dagger)a$, then the generalized inverse of matrix $\widehat{A}$ is given by*

$$\widehat{A}^\dagger = \begin{bmatrix} A^\dagger - A^\dagger a b^* \\ b^* \end{bmatrix} \tag{1}$$

where b^* is defined as:

$$b^* = \begin{cases} c^\dagger & \text{if } c \neq 0 \\ (1 + a^* A^{\dagger *} A^\dagger a)^{-1} a^* A^{\dagger *} A^\dagger & \text{if } c = 0 \end{cases} \tag{2}$$

The decremental computation for the generalized inverse of matrix is given by Cline Algorithm (Israel and Greville 2003), which is shown in Lemma 2 below.

Lemma 2 (Cline Algorithm) *Let $\widehat{A} = [a, A] \in C^{m\times n}$, where $a \in C^{m\times 1}$ and $A \in C^{m\times(n-1)}$. And $\widehat{A}^\dagger = \begin{bmatrix} d^* \\ G \end{bmatrix} \in C^{n\times m}$, where $d \in C^{m\times 1}$ and $G \in C^{(n-1)\times m}$. Define $\lambda = 1 - d^* a$, then the generalized inverse of matrix A is given by*

$$A^\dagger = \begin{cases} G + \frac{1}{\lambda} G a d^* & \text{if } \lambda \neq 0 \\ G - \frac{1}{d^* d} G d d^* & \text{if } \lambda = 0 \end{cases} \tag{3}$$

Updating for Centered Data Matrices

In this section, we will present the updating formulae for generalized inverse of centered matrix when a column vector is appended or the first column is deleted. Extension to insertion or deletion of **any** column, and row case is similar to original data matrix. Due to space limitation, we just present the updating formulae. The detailed proof of these updating formulae can be found at [1].

[1]http://homepage.fudan.edu.cn/~wangqing/onlinegeninv.html

Appending a new column Let $A \in C^{m\times(n-1)}$ be the the original data matrix and $m \in C^{m\times 1}$ be the the column mean of A. And $X \in C^{m\times(n-1)}$ be the column centered matrix of A and $X^\dagger$ be the generalized inverse of X. In the updating process, the mean of original data matrix m, the column centered data matrix X and its generalized inverse $X^\dagger$ are kept and updated during the process.

When a column vector a is appended, the mean m is first updated according to $\widetilde{m} = m + \frac{1}{n}(a-m)$. Then the centered data matrix is updated. After the append of a, X should be re-centered according to

$$\widetilde{X} = [X - \frac{1}{n}(a-m)\mathbf{1}^*, \frac{n-1}{n}(a-m)] \tag{4}$$

where $\mathbf{1}$ denotes the column vector of all ones with size $n-1$. Then $\widetilde{X}^\dagger$ is calculated according to Theorem 3.

Theorem 3 *Let $\widetilde{X}$, X, $X^\dagger$, a and m are defined above. Define $e = (I - XX^\dagger)(a-m)$, then*

$$\widetilde{X}^\dagger = \begin{bmatrix} X^\dagger - X^\dagger(a-m)h^* - \frac{1}{n-1}\mathbf{1}h^* \\ h^* \end{bmatrix} \tag{5}$$

where the h is defined as :

$$h^* = \begin{cases} e^\dagger & \text{if } e \neq 0 \\ \frac{(n-1)(a-m)^* X^{\dagger *} X^\dagger}{n+(n-1)(a-m)^* X^{\dagger *} X^\dagger (a-m)} & \text{if } e = 0 \end{cases} \tag{6}$$

Deleting the first column Let $\widetilde{X} = [x, \widehat{X}] \in C^{m\times n}$, $\widetilde{X}^\dagger = \begin{bmatrix} l^* \\ U \end{bmatrix} \in C^{n\times m}$ and $\widetilde{m}$ be the column mean of the original matrix corresponding to $\widetilde{X}$. When the first column x is deleted, the mean vector m is first updated according to $m = \widetilde{m} - \frac{1}{n-1}x$. Then the centered data matrix is updated. After the deletion of x, $\widehat{X}$ should be re-centered according to

$$X = \widehat{X} + \frac{1}{n-1}x\mathbf{1}^* \tag{7}$$

Then $X^\dagger$ is calculated according to Theorem 4.

Theorem 4 *Let X, U, x and l are defined above. Define $\theta = 1 - \frac{n}{n-1}l^*x$, then*

$$X^\dagger = \begin{cases} U + \frac{1}{\theta}(\frac{n}{n-1}Uxl^* + \frac{1}{n-1}\mathbf{1}l^*) & \text{if } \theta \neq 0 \\ U - \frac{1}{l^*l}Ull^* & \text{if } \theta = 0 \end{cases} \tag{8}$$

Experiments and Conclusions

In this experiment, we compare the accuracy and efficiency of our method to SVD method (Golub and Loan 1996) for the computation of generalized inverse of centered matrix. The results are obtained by running the matlab (version R2008a) codes on a PC with Intel Core 2 Duo P8600 2.4G CPU and 2G RAM. We generate synthetic matrix of size m = 1000 and n = 800 which entry is random number in $[-1, 1]$. And the rank deficiency is produced by randomly choosing 10% columns be replaced by other random columns in the matrix.

We start with a matrix X composed of the first column of the generated matrix A, then sequentially insert each column of A into X. After all the columns of A are inserted into X,

Method	n	time	$\|XX^\dagger X - X\|_F$	$\|X^\dagger XX^\dagger - X^\dagger\|_F$	$\|(XX^\dagger)^* - XX^\dagger\|_F$	$\|(X^\dagger X)^* - X^\dagger X\|_F$
SVD method	50	0.0406	3.13E-13	9.02E-16	1.94E-14	1.78E-14
our method		0.0065	5.00E-14	1.86E-16	4.52E-15	1.50E-15
SVD method	100	0.1154	6.04E-13	1.76E-15	3.73E-14	3.46E-14
our method		0.0156	9.01E-14	3.48E-16	9.04E-15	3.89E-15
SVD method	200	0.3697	1.07E-12	3.25E-15	6.65E-14	6.28E-14
our method		0.0343	1.95E-13	7.21E-16	1.74E-14	1.17E-14
SVD method	400	1.5412	1.96E-12	6.78E-15	1.27E-13	1.22E-13
our method		0.0871	4.93E-13	2.15E-15	3.69E-14	4.20E-14
SVD method	800	10.0037	3.43E-12	1.89E-14	2.83E-13	2.60E-13
our method		0.1841	1.86E-12	2.02E-14	9.57E-14	4.21E-13
SVD method	800	9.9716	3.42E-12	1.89E-14	2.83E-13	2.59E-13
our method		0.1571	1.86E-12	2.01E-14	1.07E-13	4.19E-13
SVD method	400	1.5303	1.93E-12	6.69E-15	1.26E-13	1.20E-13
our method		0.0791	1.12E-12	6.10E-15	8.27E-13	9.14E-14
SVD method	200	0.3712	1.06E-12	3.28E-15	6.56E-14	6.22E-14
our method		0.0309	6.69E-13	3.64E-15	7.10E-13	5.16E-14
SVD method	100	0.1139	5.78E-13	1.72E-15	3.67E-14	3.38E-14
our method		0.0147	3.76E-13	1.56E-15	5.42E-13	2.76E-14
SVD method	50	0.0359	3.32E-13	9.33E-16	2.08E-14	1.96E-14
our method		0.0061	2.16E-13	7.71E-16	3.95E-13	1.41E-14

Figure 1: Computational time and error of SVD and our method for column centered data matrix when the *n*-th column is inserted, then randomly chosen n-th column is deleted inversely.

we turn to inversely delete one randomly chosen column in X each time until X is null . At each step, the accuracy of algorithms is examined in the error matrices corresponding to the four properties characterizing the generalized inverse: $XX^\dagger X - X$, $X^\dagger XX^\dagger - X^\dagger$, $(XX^\dagger)^* - XX^\dagger$ and $(X^\dagger X)^* - X^\dagger X$. The process is repeated ten times and the averaged value is reported. Fig. 1 shows the running time (second) and the four errors of certain steps.

From Fig.1, we can see that the computational error of our method is lower than 10^{-12} in all cases and is often very closed to SVD method. Moreover the computational time of our method is significantly lower than SVD method, especially when the matrix is large. So, we can conclude that our method is a robust and efficient tool for online computing the generalized inverse of centered matrix.

In addition, our method can make the least squares formulation for a class of generalized eigenvalue problems (Sun, Ji, and Ye 2009) be suitable for online learning, since these problems require the data matrix to be centered.

Acknowledgements This work was supported in part by NSFC (60873115) and AHEDUSF (2009SQRZ075).

References

Golub, G. H., and Loan, C. F. V. 1996. *Matrix Computations*. Baltimore, MD: The Johns Hopkins University Press, 3rd edition.

Gutman, I., and Xiao, W. 2004. Generalized inverse of the laplacian matrix and some applications. *Bulletin de Academie Serbe des Sciences at des Arts (Cl. Math. Natur.)* 129:15–23.

Israel, A. B., and Greville, T. N. E. 2003. *Generalized inverses: Theory and applications*. New York, NY: Springer, 2nd edition.

Korn, F.; Labrinidis, A.; Kotidis, Y.; and Faloutsos, C. 2000. Quantifiable data mining using ratio rules. *The VLDB Journal* 8:254–266.

Liu, L. P.; Jiang, Y.; and Zhou, Z. H. 2009. Least square incremental linear discriminant analysis. In *Proceedings of the 9th International Conference on Data Mining (ICDM 2009)*, 298–306.

Ng, J.; Bharath, A.; and Kin, P. 2007. Extrapolative spatial models for detecting perceptual boundaries in colour images. *International Journal of Computer Vision* 73:179–194.

Sun, L.; Ji, S. W.; and Ye, J. 2009. A least squares formulation for a class of generalized eigenvalue problems in machine learning. In *Proceedings of the 26th International Conference on Machine Learning (ICML 2009)*, 1207–1216. Morgan Kaufmann.