Comments on "Efficient and Robust Feature Extraction by Maximum Margin Criterion"

Jun Liu, Songcan Chen, Xiaoyang Tan, and Daoqiang Zhang

***Abstract*—The goal of this comment is to first point out two loopholes in the paper by Li *et al.* (2006): 1) so-designed efficient maximal margin criterion (MMC) algorithm for small sample size (SSS) problem is problematic and 2) the discussion on the equivalence with the null-space-based methods in SSS problem does not hold. Then, we will present a really efficient MMC algorithm for SSS problem.**

***Index Terms*—Efficient algorithm, equivalence, maximal margin criterion (MMC), null space, small sample size (SSS) problem.**

I. Organization and Preparation

Organization: In this section, we will give some notations and a brief review of maximum margin criterion (MMC) [3] and point out the two loopholes. In Section II, we will propose a really efficient MMC, and then, conclude this comment in Section III.

Let the training set be composed of c classes $C_1, C_2, \ldots, C_c$, the ith class have n_i training samples, and x_j^i denote the jth D-dimensional sample from the ith class. In total, there will be $n = \sum_{i=1}^{c} n_i$ training samples. In applications such as face recognition, the small sample size (SSS) problem often takes place, namely, $D \gg n$. The within-class scatter matrix S_w and between-class scatter matrix S_b can be denoted as

$$S_w = \frac{1}{n}\sum_{i=1}^{c}\sum_{j=1}^{n_i}\left(x_j^i - m_i\right)\left(x_j^i - m_i\right)^T = H_w H_w^T \tag{1}$$

$$S_b = \frac{1}{n}\sum_{i=1}^{c} n_i (m_i - m)(m_i - m)^T = H_b H_b^T \tag{2}$$

where H_w and H_b are, respectively, defined as

$$H_w = \frac{1}{\sqrt{n}}\left[x_1^1 - m_1, \ldots, x_{n_1}^1 - m_1, \ldots, x_{n_c}^c - m_c\right] \tag{3}$$

$$H_b = \frac{1}{\sqrt{n}}\left[\sqrt{n_1}(m_1 - m), \ldots, \sqrt{n_c}(m_c - m)\right]. \tag{4}$$

The m_i is the centroid of the ith class and m is the centroid of the training set.

MMC [3] aims at maximizing the average margin between classes, where the interclass margin between the ith and the jth classes can be denoted as

$$d(C_i, C_j) = d(m_i, m_j) - (S(C_i) + S(C_j)) \tag{5}$$

where $d(m_i, m_j)$ is defined as the squared Euclidean distance between m_i and m_j and $S(C_i)$ and $S(C_j)$ are, respectively, defined as the traces of the scatter matrix of the ith and jth classes. After some deduction, the average margin between classes under transformation matrix W is [3]

$$J(W) = \mathrm{tr}(W^T(S_b - S_w)W). \tag{6}$$

Requiring the column vectors in W to be unit vectors, W can be calculated by solving the eigenequation

$$(S_b - S_w)w = \lambda w. \tag{7}$$

II. Loopholes and a Really Efficient MMC Algorithm

A. Loopholes in the Efficient MMC Algorithm

The time and storage complexities for directly solving the eigenequation (7) are, respectively, $O(D^3)$ and $O(D^2)$, which are very demanding for applications such as face recognition. For example, the face images in the ORL face database [3] have a resolution of 112×92 (namely, $D = 10304$), thus the time and storage costs are, respectively, in the order of 10^{12} and 10^8, which are very demanding, especially for personal computers.

In [3], Li *et al.* proposed to obtain a transformation matrix $P \in R^{D\times r}$ that simultaneously diagonalized S_b and S_t as

$$P^T S_b P = \Lambda \tag{8}$$

$$P^T S_t P = I_r \tag{9}$$

and P can be given by

$$P = \Phi\Theta^{-1/2}\Psi \tag{10}$$

where $\Theta \in R^{r\times r}$ is a diagonal matrix whose diagonal entries are the positive eigenvalues of the total scatter matrix $S_t = S_b + S_w, r \le \min\{n-1, D\}$ is the rank of $S_t, \Phi \in R^{D\times r}$ contains the r orthonormal eigenvectors of S_t corresponding to these positive eigenvalues, and Ψ is the eigenvector matrix of $\Theta^{-1/2}\Phi^T S_b \Phi\Theta^{-1/2}$.

Φ and Θ can be calculated in a time complexity of $O(Dn^2)$ by first solving the singular value decomposition of A^TA where $S_t = AA^T$; Ψ can be computed in a time complexity of $O(n^3)$, and thus, the time complexity for computing P in (10) is $O(Dn^2)$. Further, the largest matrix employed is $D \times n$, and thus, the space complexity is $O(Dn)$. As a result, the time and storage complexities for computing P are, respectively, $O(Dn^2)$ and $O(Dn)$. Taking the ORL database as an example, if the number of training samples n is 200, then r is at most 199, and the time and storage costs are now, respectively, in the order of 10^8 and 10^6, which are significant improvements over 10^{12} and 10^8 for directly solving the eigenequation (7).

Li *et al.* argued in [3] that the columns of P in (10) are the eigenvectors of $2S_b - S_t$ with the corresponding eigenvalues $2\Lambda - I_r$, based on which a fast MMC algorithm was proposed. However, such argument is problematic in that column vectors of P in (10) are not definitely the eigenvectors of $2S_b - S_t$. An analysis is given as follows.

From (8) and (9), we can get

$$P^T(2S_b - S_t)P = 2\Lambda - I_r \tag{11}$$

and

$$PP^T(2S_b - S_t)P = P(2\Lambda - I_r). \tag{12}$$

Manuscript received April 26, 2006; revised November 19, 2006; accepted April 9, 2007. This work was supported in part by the Natural Science Foundations of China and Jiangsu Province under Grants 60473035, 60505004, BK2005122, BK2006187, and BK2006521, and the Innovation Foundation of the Nanjing University of Aeronautics and Astronautics (NUAA) under Grand Y0603-042.

The authors are with the Department of Computer Science and Engineering, Nanjing University of Aeronautics and Astronautics, Nanjing 210016, P.R. China (e-mail: j.liu@nuaa.edu.cn; s.chen@nuaa.edu.cn; x.tan@nuaa.edu.cn; dqzhang@nuaa.edu.cn).

Digital Object Identifier 10.1109/TNN.2007.900813

However, from (8) and (9), we cannot definitely get

$$(2S_b - S_t)P = P(2\Lambda - I_r) \tag{13}$$

since we cannot assure

$$PP^T(2S_b - S_t)P = (2S_b - S_t)P \tag{14}$$

in real application. According to matrix theory [1], we can only assure that the columns of P are the eigenvectors of $2S_b - S_t$ with the corresponding eigenvalues $2\Lambda - I_r$ from (13), and thus, the column vectors of P in (10) are not definitely the eigenvectors of $2S_b - S_t$.

Now, let us consider an example of two classes ($c = 2$); each class has two training samples ($n_1 = n_2 = 2$). Samples from the first class are around $[1\ 2\ 4\ 4\ 2]^T$, while those from the second class are around $[4\ 4\ 8\ 1\ 5]^T$. More specifically, let $x_1^1 = [1.0\ 2.1\ 3.9\ 4.2\ 2.3]^T, x_2^1 = [1.1\ 1.7\ 4.3\ 4.0\ 1.9]^T, x_1^2 = [4.2\ 4.3\ 7.8\ 1.2\ 5.1]^T$, and $x_2^2 = [3.7\ 3.9\ 7.9\ 0.8\ 4.7]^T$. The rank of S_t, r, is equal to 3 and one can easily get $P, 2\Lambda - I_r$, $A = PP^T(2S_b - S_t)P$, and $B = (2S_b - S_t)P$, respectively, as

$$P = \begin{pmatrix} 0.0068253 & -4.5483 & -1.2639 \\ 0.11703 & 2.0388 & -1.1784 \\ 0.14688 & -1.9274 & 0.97184 \\ -0.2408 & -3.0248 & -1.8339 \\ 0.15195 & 2.3413 & -1.097 \end{pmatrix}$$

$$2\Lambda - I_r = \begin{pmatrix} 1.0000 & 0 & 0 \\ 0 & -1.0000 & 0 \\ 0 & 0 & -1.0000 \end{pmatrix}$$

$$A = \begin{pmatrix} 0.0068253 & 4.5483 & 1.2639 \\ 0.11703 & -2.0388 & 1.17840 \\ 0.14688 & 1.9274 & -0.97184 \\ -0.2408 & 3.0248 & 1.8339 \\ 0.15195 & -2.3413 & 1.0970 \end{pmatrix}$$

$$B = \begin{pmatrix} 1.4500 & 0.12587 & 0.12906 \\ 1.1000 & -0.041679 & 0.19561 \\ 1.8750 & 0.099403 & -0.10663 \\ -1.5500 & 0.017643 & 0.15713 \\ 1.4000 & -0.041679 & 0.19561 \end{pmatrix}. \tag{15}$$

From (15), one can easily observe that $A \neq B$ and, consequently, neither (14) nor (13) holds. Furthermore, by solving (7) directly, the three nonzero eigenvalues for $S_b - S_w$ are in fact $11.181, -1.2606$, and -0.022869, rather than $1.0000, -1.0000$, and -1.0000, as reported in (15). As a result, the calculated P and $2\Lambda - I_r$ are not the right solutions to MMC and the so-designed efficient MMC algorithm in [3] is problematic.

B. Loopholes in Relationship With the Null Space Method

Li *et al.* argued in [3] that "When the small samples size problem arises, MMC is actually equivalent to LDA + PCA [2] in general." However, this argument is incorrect. The null-space-based methods such as the LDA + PCA [2] in fact obtain a projection matrix in the null space of S_w as

$$W_{\text{opt}} = \arg \max_{W^T S_w W = 0} |W^T S_b W| \tag{16}$$

or, equivalently

$$W_{\text{opt}} = \arg \max_{W^T S_w W = 0} |W^T (S_b - S_w) W|. \tag{17}$$

For discussion convenience, we denote the projection matrix for MMC as W_{mmc} and that for LDA + PCA as W_{null}. Generally speaking, both W_{mmc} and W_{null} maximize $\text{tr}(W^T(S_b - S_w)W)$. However, MMC's projection matrix only has orthogonal constraint, i.e., $W_{\text{mmc}}^T W_{\text{mmc}}$ is an identity matrix, while LDA + PCA's projection has additional constraint, namely, $W_{\text{null}}^T W_{\text{null}}$ is an identity matrix and $W_{\text{null}}^T S_w W_{\text{null}} = 0$. As a result, they are distinct theoretically, and $\text{tr}(W_{\text{null}}^T(S_b - S_w)W_{\text{null}}) \leq \text{tr}(W_{\text{mmc}}^T(S_b - S_w)W_{\text{mmc}})$ since LDA + PCA has additional constraint compared to MMC.

To experimentally show this loophole, we perform experiment on the ORL face database [3], and utilize the first five images of each person to form the training set. The image resolution is 112×92 and the gray level values of the images are scaled to [0 1]. We set the number of projection vectors to 39 (or $c - 1$) and employ the subspace distance defined in [1, p. 76] to measure the distance between the subspaces, respectively, spanned by MMC[1] and LDA + PCA. The result is that the subspace distance between them is 0.9330 (note that two subspaces are identical only when the subspace distance is 0), which clearly shows that MMC is not identical to LDA + PCA. Furthermore, $\text{tr}(W_{\text{null}}^T(S_b - S_w)W_{\text{null}}) = 47.609$ and $\text{tr}(W_{\text{mmc}}^T(S_b - S_w)W_{\text{mmc}}) = 144.5$, from which we can clearly observe that LDA + PCA cannot achieve the same objective function value as MMC due to the additional constraint $W^T S_w W = 0$.

Finally, in the end of this section, it is worthwhile to note that, although the P computed by (10) is not the correct solution to MMC, it is in fact the solution to the generalized linear discriminant analysis (GLDA) method [4], which was proven equivalent to LDA + PCA [2] when $\text{rank}(S_t) = \text{rank}(S_w) + \text{rank}(S_b)$ in [4]. This partially explains the reasons why MMC was mistaken as equivalent to LDA + PCA in [3] and why the so-called fast algorithm proposed in [3] yielded identical classification performance to LDA + PCA in some experiments.

C. Really Efficient MMC Algorithm

Despite of the loopholes pointed out in this comment, MMC is a good method that is related to margin between classes. Thus, a really efficient algorithm for MMC in SSS problem is of great importance, and we will offer one in the following.

Let $\tilde{\Phi}$ contain the eigenvectors of S_t corresponding to the zero eigenvalues, and then, the column vectors in $[\Phi \quad \tilde{\Phi}]$ constitute a set of orthonormal bases for the space R^D. According to matrix knowledge [1], any w in R^D can be written as

$$w = \Phi p + \tilde{\Phi}\tilde{p} \tag{18}$$

where p and $\tilde{p}$ are $r \times 1$ and $(D - r) \times 1$ vectors, respectively.

Keeping in mind $S_t\tilde{\Phi} = 0$, $S_w\tilde{\Phi} = 0$ and $S_b\tilde{\Phi} = 0$, and substituting (18) into (7), we get

$$(S_b - S_w)\Phi p = \lambda(\Phi p + \tilde{\Phi}\tilde{p}). \tag{19}$$

Premultiplying Φ^T to both sides of (19), we get

$$(\Phi^T S_b \Phi - \Phi^T S_w \Phi)p = \lambda p. \tag{20}$$

Premultiplying $\tilde{\Phi}^T$ to both sides of (19), we get

$$0 = \lambda \tilde{p}. \tag{21}$$

We can set $\tilde{p} = 0$, since 1) λ should be as positive as possible in order to maximize MMC's objective function (6), and thus, it would be better than $\tilde{p} = 0$ by using (21); and 2) when the samples are projected by $\tilde{\Phi}\tilde{p}$, all the training samples concentrate to a common vector and the extracted features by $\tilde{\Phi}\tilde{p}$ contain no discriminant information, and thus, we can set $\tilde{p} = 0$. Now, substituting $\tilde{p} = 0$ into (18), we have

$$w = \Phi p. \tag{22}$$

[1]Due to typically large dimensionality, when implementing MMC, we employ the really efficient MMC algorithm to be given in Section II-C rather than directly solving (7).

It is easy to observe that the ws for MMC can be calculated in terms of the following three steps: 1) calculate Φ, 2) calculate ps by (20), and 3) obtain ws from (22). The time complexity of the newly proposed MMC algorithm is analyzed as follows: 1) Φ can be obtained in $O(Dn^2)$, 2) $\Phi^T S_b \Phi = (\Phi^T H_b)(\Phi^T H_b)^T$ and $\Phi^T S_w \Phi = (\Phi^T H_w)(\Phi^T H_w)^T$ can be computed in $O(Dnc)$ and $O(Dn^2)$, respectively, 3) ps can be obtained in $O(r^3)$, and 4) ws can be received from (22) in $O(Dnc)$. As a result, the time complexity is $O(Dn^2)$, which is much more efficient than $O(D^3)$ consumed by directly solving (7). Furthermore, since the largest matrix employed in the computation is $D \times n$, the space complexity is $O(Dn)$, which is far less than $O(D^2)$ needed by directly solving (7).

To experimentally verify the correctness of the newly proposed efficient algorithm, we employ it to calculate the projection matrix W on the two class samples given in Section II-A. Experimental results show that

$$W = \begin{pmatrix} -0.43224 & 0.24384 & -0.70655 \\ -0.32817 & 0.50625 & 0.28324 \\ -0.56102 & -0.42688 & -0.27138 \\ 0.46483 & 0.51061 & -0.48782 \\ -0.41793 & 0.49122 & 0.33006 \end{pmatrix} \tag{23}$$

$$\begin{aligned} WW^T(2S_b - S_t)W &= (2S_b - S_t)W \\ &= \begin{pmatrix} -4.8328 & -0.030739 & 0.016158 \\ -3.6692 & -0.063819 & -0.0064773 \\ -6.2726 & 0.053814 & 0.0062062 \\ 5.1972 & -0.064368 & 0.011156 \\ -4.6728 & -0.061925 & -0.0075481 \end{pmatrix} \end{aligned} \tag{24}$$

and the eigenvalues are 11.181, -1.2606, and -0.022869, equal to those directly computed by (7). Thus, the eigenvectors and eigenvalues calculated by the newly proposed efficient MMC are the solutions to (7), and the correctness of the proposed algorithm is experimentally proven.

III. Conclusion

In this comment, we point out two loopholes appearing in [3] both theoretically and experimentally, namely, 1) so-designed efficient MMC algorithm for SSS problem is problematic, and 2) the discussion on the equivalence with the null-space-based methods in SSS problem does not hold. Then, we present a really efficient MMC algorithm for the SSS problem.

Acknowledgment

The authors would like to thank the associate editor and the anonymous reviewers for the constructive comments.

References

[1] G. H. Golub and C. F. Van Loan, *Matrix Computations*, 3rd ed. Baltimore, MD: The Johns Hopkins University Press, 1996.

[2] R. Huang, Q. Liu, H. Lu, and S. Ma, "Solving the small sample size problem of lda," in *Proc. 16th Int. Conf. Pattern Recognit.*, Quebec City, QC, Canada, 2002, pp. 29–32.

[3] H. Li, T. Jiang, and K. Zhang, "Efficient and robust feature extraction by maximum margin criterion," *IEEE Trans. Neural Netw.*, vol. 17, no. 1, pp. 157–165, Jan. 2006.

[4] H. Li, K. Zhang, and T. Jiang, "Robust and accurate cancer classification with gene expression profiling," in *Proc. 4th IEEE Comput. Syst. Bioinf. Conf.*, Standford, CA, 2005, pp. 310–321.