FIXED-RANK APPROXIMATION WITH CONSTRAINTS

JAN DE LEEUW AND IRINA KUKUYEVA

Abstract. We discuss fixed-rank weighted least squares approximation of a rectangular matrix with possibly singular Kronecker weights and with various types of constraints on the left and right components defining the approximation.

1. Introduction

Suppose $X(n \times m)$ is a *data matrix*, and $W(n \times n)$ and $V(m \times n)$ are known positive semi-definite matrices of *weights*. The problem we study in this paper is to minimize the sum-of-squares of the weighted residuals

$$\sigma(A, B) = \mathbf{tr}\ (X - AB')'W(X - AB')V, \tag{1}$$

over the $n \times r$ matrices A and the $m \times r$ matrices B.

The general weighted fixed-rank approximation problem (1), with possibly singular weight matrices, was discussed in Leeuw [1984]. See also Zha [1991]. Note that, at least if V and W are nonsingular,

Date: Sunday 13th September, 2009 — 15h 32min — Typeset in Kepler.

it can also be thought of as maximum likelihood estimation of A and B for a matrix-normally distributed $\underline{X}$ with $\mathbf{E}(\underline{X}) = AB'$ and $\mathbf{V}(\underline{X}) = \omega^2(V^{-1} \otimes W^{-1})$ [Gupta and Nagar, 2000].

In this paper we generalize previous results to include constraints on A and B. We write those constraints in the general form $A \in \mathcal{A} \subseteq \mathbb{R}^{n\times r}$ and $B \in \mathcal{B} \subseteq \mathbb{B}^{n\times r}$. In most cases the sets $\mathcal{A}$ and $\mathcal{B}$ will be defined by linear equality and inequality constraints, but we'll discuss the problem more generally.

2. Algorithm

The obvious algorithm to apply is of the alternating least squares (ALS) class [De Leeuw, 1994]. We alternate minimization over A and B. Thus we start, say, with, $B^{(0)} \in \mathcal{B}$ and then define for each $k = 1, 2, \cdots$

$$A^{(k)} = \underset{A\in\mathcal{A}}{\mathbf{argmin}}\ \sigma(A, B^{(k-1)}), \tag{2a}$$

$$B^{(k)} = \underset{B\in\mathcal{B}}{\mathbf{argmin}}\ \sigma(A^{(k)}, B). \tag{2b}$$

2.1. **Subproblems.** The two partial optimization problems in (2) can be handled by partitioning the sum-of-squares. To find unconstrained least squares estimates we set the partials equal to zero. This gives

$$WA(B'VB) = WXVB, \tag{3a}$$

$$VB(A'WA) = VX'WA. \tag{3b}$$

Expand the loss around any $\tilde{A}$. Then

$$\begin{aligned}\sigma(A,B) &= \mathbf{tr}\ (X - \{\tilde{A} + (A-\tilde{A})\}B')'W(X - \{\hat{A} + (A-\tilde{A})\}B')V = \\ &= \sigma(\tilde{A}, B) - 2\mathbf{tr}\ (X - \tilde{A}B')'W(A-\tilde{A})B'V + \mathbf{tr}\ (A-\hat{A})'W(A-\tilde{A})(B'VB).\end{aligned}$$

If[1] $\hat{A}(B) \in \mathop{\mathbf{Argmin}}_{A} \sigma(A,B)$ then (3a) gives $W(X-\hat{A}B')VB = 0$, and thus

$$\sigma(A,B) = \sigma(\hat{A}(B),B) + \mathbf{tr}\ (A-\hat{A}(B))'W(A-\hat{A}(B))(B'VB). \tag{4a}$$

In the same way if $\hat{B}(A) \in \mathop{\mathbf{Argmin}}_{B} \sigma(A,B)$ then

$$\sigma(A,B) = \sigma(A,\hat{B}(A)) + \mathbf{tr}\ (B-\hat{B}(A))'V(B-\hat{B}(A))(A'WA). \tag{4b}$$

Observe that this not require the weight matrices to be positive definite, it does not require the solutions of (3) to be unique, and it does not even requite that $p \leq \min(n,m)$.

It follows that

$$\mathop{\mathbf{Argmin}}_{A\in\mathcal{A}}\ \sigma(A,B^{(k-1)}) = \mathop{\mathbf{Argmin}}_{A\in\mathcal{A}} \mathbf{tr}\ (A-\hat{A}(B^{(k-1)}))'W(A-\hat{A}(B^{(k-1)}))C^{(k-1)}, \tag{5}$$

where $C^{(k-1)} = (B^{(k-1)})'VB^{(k-1)}$. Finding the optimal A for given $B = B^{(k-1)}$ can be done by first computing $\hat{A}(B^{(k-1)})$ and then projecting $\hat{A}(B^{(k-1)})$ on $\mathcal{A}$ in the metric $W \otimes C^{(k-1)}$, i.e. finding a minimizer for (5). Appendix A shows how to compute $\hat{A}(B^{(k-1)})$. The same procedure can be followed for updating B.

If $\mathcal{A}$ and $\mathcal{B}$ are defined by linear equality constraints the projection problems are weighted linear least squares problems. In that case both computing $\hat{A}(B^{(k-1)})$ and projecting $\hat{A}(B^{(k-1)})$ can be done using R functions `lsfit()` or `qr.solve()`.

If there are inequality constraints in the definition of $\mathcal{A}$ and $\mathcal{B}$ the subproblems become quadratic programming problems, which can also be handled efficiently by various R packages (`quadprog`, `nnls`, `isotone`, `pava` and `ipop` in `kernlab`). For non-linear constraints more complicated iterative projection methods will be needed.

[1]We define the set $\mathop{\mathbf{Argmin}}_{A} \sigma(A,B) = \{\hat{A} \in \mathbb{R}^{n\times p} \mid \sigma(\hat{A},B) = \min_A \sigma(A,B)\}$. If we know the minimum is unique, then we use $\mathop{\mathbf{argmin}}_{A} \sigma(A,B)$ for the unique minimizer.

3. Constraints

3.1. **Previous Work.**

Problems of this type have been studied in considerable detail by Yoshio Takane and a varying set of co-authors. In [Takane et al., 1980] an individual additive model is studied, in which the rows of the data matrix correspond with individuals and the columns with the cells of a factorial design. There are no constraints on A, but the columns of B must satisfy $B_s = H_s\gamma_s$, where H_s is the binary indicator [Kiers et al., 1996; Takane and Hunter, 2001; Hunter and Takane, 2002].

In Takane et al. [1995] the problem studied is minimization of $\sigma(A,B)$ under linearity constraints on the columns of A and B that can be written as $a_s = G_s\beta_s$ and $b_s = H_s\gamma_s$, where G_s and H_s are known matrices. Thus $AB' = \sum_{s=1}^p G_s\beta_s\gamma_s'H_s'$.

Suppose $\{1,2,\cdots,p\}$ is partitioned into r index sets $\mathcal{I}_q$, where $\mathcal{I}_q$ has p_q elements. Within each index set the matrices G_s and H_s are supposed to be equal. We then have $AB' = \sum_{q=1}^r G_qM_qH_q'$, where we require $\mathbf{rank}(M_q) \leq p_q$. If there is only one index set we have $AB' = GMH'$, with $\mathbf{rank}(M) \leq p$. This is the case treated in Takane and Shibayama [1991], in which we require the columns of A to be in the column space of G and the columns of B to be in the column space of H.

3.2. **Linear Constraints.**

Suppose the constraints are of the form

$$A = A_0 + \sum_{s=1}^{k_a} \theta_s A_s,$$

$$B = B_0 + \sum_{s=1}^{k_b} \xi_s B_s.$$

Here A_0 and B_0 are known matrices coding which elements are fixed to constants (and they contain those constants, the other elements are zero). The matrices A_s are B_s are binary indicators, and they code which elements of the matrices A and B must be equal. Each of them codes an equivalence class by using elements that are equal to one. Note that this implies that $\mathbf{tr}\ A_s'A_t = 0$ for all $s \neq t$, which includes as a special case that $\mathbf{tr}\ A_0'A_s = 0$ for all $s \neq 0$.

3.3. **Orthogonality.** In addition to the linear constraints, or more accurately instead of the linear constraints, we can also require orthonormality in the appropriate metric, i.e. $A'WA = I$ or $B'VB = I$. Or both, although there are few practical situations in which we will actually require both orthonormality constraints. The projection problems become Weighted Procrustus Problems, which are analyzed in detail in Appendix B

3.4. **Canonical and Correspondence Analysis.** Suppose Y and Z are two data matrices. Then *Canonical Correlation Analysis* can be formulated as

$$\min_{A,B} \sigma(A,B) = \mathbf{tr}\ (X'X)^{-1}(X'Y - AB')(Y'Y)^{-1}(X'Y - AB')'.$$

If X and Y are two indicator matrices, then the problem becomes *Correspondence Analysis*

$$\min_{A,B} \sigma(A,B) = \mathbf{tr}\ D^{-1}(F - AB')E^{-1}(F - AB')',$$

where $F = X'Y$ is the cross table, and D and F are the diagonal matrices of marginals.

An asymmetric form of Canonical Correlation Analysis, in which X are predictors and Y are outcomes, is known as *Redundancy Analysis*. It computes

$$\min_{A,B} \sigma(A,B) = \mathbf{tr}\ (X'Y - AB')'(X'X)^{-1}(X'Y - AB').$$

If X is a data matrix and Y is a matrix indicating group membership, then

$$\min_{A,B} \sigma(A,B) = \mathbf{tr}\ (X'X)^{-1}(X'Y - AB')E^{-1}(X'Y - AB')',$$

Now $M = E^{-1}Y'X$ are the group means, and thus $X'Y = M'E$. It follows that, if we let $\tilde{B} = E^{-1}B$,

$$\min_{A,B} \sigma(A,B) = \mathbf{tr}\ (X'X)^{-1}(M - \tilde{B}A')'E(M - \tilde{B}A').$$

Thus we make a fixed rank approximation of the between group variation. This is known as *Canonical Discriminant Analysis*.

Canonical Correspondence Analysis has a $n \times m$ matrix F of counts of m species in n enviroments. The n environments are also described in an $n \times r$ matrix Z of background variables. The row sums of F are in te diagonal matrix D, the column sums are in the diagonal matrix E. Solve

$$\min_{A,B} \sigma(A,B) = \mathbf{tr}\ (Z'DZ)^{-1}(Z'F - AB')E^{-1}(Z'F - AB')'.$$

Reduced Rank Regression Analysis computes

$$\min_{A,B} \sigma(A,B) = \mathbf{tr}\ W(X - AB')V(X - AB')',$$

where the constraint is that the $n \times p$ matrix A is of the form $A = ZU$, where Z is $n \times r$ and known, and where U is $r \times p$. Thus we require that each column of A is in the column space of Z. Because we fit $X \approx ZUB'$, we can also say that we regress X on Z, and require the $r \times m$ matrix of regression coefficients to have rank less than or equal to p.

3.5. **Factor Analysis.** Factor Analysis provides a nice example in which $m \leq p \leq n$. The approximation we are fitting is

$$\underset{n\times m}{X} \approx \begin{bmatrix} \underset{n\times p}{A_1} & \underset{n\times m}{A_2} \end{bmatrix} \begin{bmatrix} \underset{p\times m}{B_1'} \\ \underset{m\times m}{B_2'} \end{bmatrix},$$

where $A = (A_1 \mid A_2)$ is orthonormal, so $n \geq m + p$, and B_2 is diagonal. A_1 ate the common factor scores, A_2 the unique factor scores, B_1 the common factor loading, and B_2 the unique factor loadings. The unique variances are B_2^2. This can be combined with row and column weights.

4. R Program

5. Example

5.1. Artificial Example. X is a 10×4 matrix, and $p = 4$. We require A to be orthonormal, while the matrix B is constrained by

$$B = \begin{bmatrix} a & 0 & 1 & 0 \\ a & 0 & 0 & 1 \\ 0 & b & 1 & 0 \\ 0 & b & 0 & 1 \end{bmatrix}.$$

```
set.seed(12345)
x<-matrix(rnorm(40),10,4)
bEqual<-list(c(1,2),c(7,8))
bFix<-matrix(0,4,4)
bFix[c(9,11,14,16)]<-1
```

```
> z<-constrPCA(x,4,bFix=bFix,bEqual=bEqual,aOrth=TRUE)
Iteration:    1 fOld:   48.57686779 fNwA:   22.44550033 fNwB:   19
     .82379706
Iteration:    2 fOld:   19.82379706 fNwA:   18.55602105 fNwB:   18
     .15299803
Iteration:    3 fOld:   18.15299803 fNwA:   18.08857889 fNwB:   18
     .07918365
Iteration:    4 fOld:   18.07918365 fNwA:   18.07786073 fNwB:   18
     .07762363
Iteration:    5 fOld:   18.07762363 fNwA:   18.07756618 fNwB:   18
     .07754889
Iteration:    6 fOld:   18.07754889 fNwA:   18.07754314 fNwB:   18
     .07754115
Iteration:    7 fOld:   18.07754115 fNwA:   18.07754046 fNwB:   18
     .07754022
```

```
z$itel
[1] 7

z$loss
[1] 18.07754

z$a
              [,1]          [,2]          [,3]          [,4]
```

```
 [1,] -0.40357472  0.156894951  0.65526199  0.28321237
 [2,]  0.66820176  0.546563272  0.18545133  0.15109661
 [3,] -0.19577472  0.144700965 -0.12239218  0.45703993
 [4,]  0.01633959 -0.094524963 -0.48449469  0.50849630
 [5,]  0.14516202 -0.270383426 -0.20519814  0.09255575
 [6,] -0.46132799  0.632147943 -0.34564259 -0.33398327
 [7,]  0.22039113 -0.025501657 -0.15544173 -0.37671401
 [8,] -0.02687847 -0.073230366  0.05090725 -0.37585795
 [9,]  0.15122602  0.409929540 -0.17101921  0.13211762
[10,] -0.21442852 -0.002399574 -0.26257999  0.09358139

z$b
         [,1]    [,2] [,3] [,4]
[1,] 1.054598 0.00000    1    0
[2,] 1.054598 0.00000    0    1
[3,] 0.000000 2.66472    1    0
[4,] 0.000000 2.66472    0    1

z$rA.sq
      Y1       Y2       Y3       Y4       Y5       Y6       Y7       Y8
"0.9170" "0.9969" "0.7428" "0.7798" "0.2746" "0.4783" "0.5390" "0.6280"
      Y9      Y10
"0.9968" "0.7242"

z$rB.sq
      Y1       Y2       Y3       Y4
"0.8527" "0.8745" "0.9270" "0.9516"
```

5.2. **Real Example: Brand Rating.** This is an example in brand rating on multiple attributes, using brand and attribute effects. The theory is in Dillon et al. [2001]. X is an $I \times (J \times K)$ matrix, where $x_{i(jk)}$ is the rating of brand j on attribute k by consumer i. Thus X has I rows and JK columns. The matrix A of component scores is $I \times (J+K)$, and it is assumed to be orthonormal. The matrix B is $(JK) \times (J+K)$ and it is structured by linear equality constraints.

The resulting matrix of interest is B-squared component-wise, which shows how much variance is accounted for each brand separately. Furthermore, the first K columns of the matrix may be interpreted as "brand-specific associations (BSAs)" and the last J columns as the

”general brand impressions (GBIs)” [1]. BSAs show which attributes of a given brand set it apart from the competition [1]. GBIs illustrate the extent of brand-recognition [1].

In the case of our data set, X is 439×33: there were 439 respondents ($I = 439$), who rated three brands ($J = 3$) on 11 attributes ($K = 11$). Therefore, $p = 14$ (which is the number of brands plus the number of attributes). We require A to be orthonormal, while the matrix B can be constrained by:

$$\mathbf{B} = \left(\begin{array}{ccccc|ccc}
1 & 0 & 0 & \dots & 0 & a_1 & 0 & 0 \\
0 & 1 & 0 & \dots & 0 & a_1 & 0 & 0 \\
0 & 0 & 1 & \dots & 0 & a_2 & 0 & 0 \\
\vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots \\
0 & 0 & 0 & \dots & 1 & a_2 & 0 & 0 \\
\hline
1 & 0 & 0 & \dots & 0 & 0 & b_1 & 0 \\
0 & 1 & 0 & \dots & 0 & 0 & b_1 & 0 \\
0 & 0 & 1 & \dots & 0 & 0 & b_2 & 0 \\
\vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots \\
0 & 0 & 0 & \dots & 1 & 0 & b_2 & 0 \\
\hline
1 & 0 & 0 & \dots & 0 & 0 & 0 & c_1 \\
0 & 1 & 0 & \dots & 0 & 0 & 0 & c_1 \\
0 & 0 & 1 & \dots & 0 & 0 & 0 & c_2 \\
\vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots \\
0 & 0 & 0 & \dots & 1 & 0 & 0 & c_2
\end{array}\right),$$

where the first 2 of each a, b and c are equal to each other and then the next 9 of each.

Before we can execute the `constrPCA()` function, we need to standardize the data matrix X (and denote the result by $X.std$) to have mean zero for each of the variables and sum of squares equal to one with the aid of the `standard()` function. The corresponding R Code is shown below:

```
X=data[, -1]

# Save the number of attributes in a variable:
no.att=11
# Save the number of brands in a variable:
no.brands=3

# Run the get.bEqual() function:
bEqual.mat<-get.bEqual(no.att=11, no.brands=3, c1=2, c2=9, c3=0)

# Standardizing X:
X.std<-standard(X)

# Calling the main function:
out<-constrPCA(x=X.std, p=(no.att+no.brands), aFix=matrix(0, nrow=nrow(X.std),
ncol=(no.att+no.brands)), bFix=matrix(0, nrow=ncol(X.std),
ncol=(no.att+no.brands)), aEqual=NULL, bEqual=bEqual.mat, aOrth=TRUE, bOrth=FALSE)

# Matrix of interest:
out$b^2
```

The result is reproduced below:

6. Generalizations

The obvious generalizations are

- Missing Data;
- Weighted PCA;
- General Constraint Sets;
- Nonlinear PCA à la Gifi;
- More than two modes à la Tucker.

References

J. De Leeuw. Block Relaxation Methods in Statistics. In H.H. Bock, W. Lenski, and M.M. Richter, editors, *Information Systems and Data Analysis*, Berlin, 1994. Springer Verlag.

W.R. Dillon, T.J. Madden, A. Kirmani, and S. Mukherjee. Understanding What's in a Brand Rating: A Model for Assessing Brand and Attribute Effects and their Relationship to Brand Equity. *Journal of Marketing Research*, 38:415–429, 2001.

A.K. Gupta and D.K. Nagar. *Matrix Variate Distributions*. Chapman & Hall/CRC, Boca Raton, Florida, 2000.

M. Hunter and Y. Takane. Constrained Principal Component Analysis: Various Applications. *Journal of Educational and Behavioral Statistics*, 27:105–145, 2002.

H.A.L. Kiers, Y. Takane, and J.M.F. Ten Berge. The Analysis of Multitrait-Multimethod Matrices via Constrained Components Analysis. *Psychometrika*, 61:601–628, 1996.

Jan De Leeuw. Differentiability of Kruskal's Stress at a Local Minimum. *Psychometrika*, 49:111–113, 1984.

Y. Takane and M. Hunter. Constrained Principal Components Analysis: A Comprehensive Theory. *Applicable Algebra in Engineering, Communication and Computing*, 12:391–419, 2001.

Y. Takane and T. Shibayama. Principal Component Analysis with External Information on Both Subjects and Variables. *Psychometrika*, 56:97–120, 1991.

Y. Takane, F.W. Young, and J. De Leeuw. An Individual Differences Additive Model: an Alternating Least Squares Method with Optimal Scaling Features. *Psychometrika*, 45:183–209, 1980.

Y. Takane, H.A.L Kiers, and J. De Leeuw. Component Analysis with Different Sets of Constraints on Different Dimensions. *Psychometrika*, 60:259–280, 1995.

H. Zha. The Product-product Singular Value Decomposition of Matrix Triplets. *BIT*, 31:711–726, 1991.

Appendix A. The Equation $VXW = Y$

Suppose V and W are positive semi-definite matrices or orders n and m, respectively, and Y is $n \times m$. We want to find all $n \times m$ matrices X satisfying $VXW = Y$. This is most easily solved by using the eigen-decompositions of V and W. Suppose V is of rank r and W is of rank s. The eigen-decompositions are

$$V = \begin{bmatrix} \underset{n\times r}{K_1} & \underset{n\times(n-r)}{K_0} \end{bmatrix} \begin{bmatrix} \underset{r\times r}{\Lambda} & \underset{r\times(n-r)}{\emptyset} \\ \underset{(n-r)\times r}{\emptyset} & \underset{(n-r)\times(n-r)}{\emptyset} \end{bmatrix} \begin{bmatrix} \underset{r\times n}{K_1'} \\ \underset{(n-r)\times n}{K_0'} \end{bmatrix},$$

$$W = \begin{bmatrix} \underset{m\times s}{L_1} & \underset{m\times(m-s)}{L_0} \end{bmatrix} \begin{bmatrix} \underset{s\times s}{\Omega} & \underset{s\times(m-s)}{\emptyset} \\ \underset{(m-s)\times s}{\emptyset} & \underset{(m-s)\times(m-s)}{\emptyset} \end{bmatrix} \begin{bmatrix} \underset{s\times m}{L_1'} \\ \underset{(m-s)\times m}{L_0'} \end{bmatrix},$$

Theorem A.1. *The equation $VXW = Y$ is solvable if and only if both $YL_0 = 0$ and $Y'K_0 = 0$. If the equation is solvable, the solution space is of dimension $nm - rs$ and the general solution is*

$$X = V^+YW^+ + K_1PL_0' + K_0QL_1' + K_0RL_0'$$

where $\underset{r\times(m-s)}{P}$, $\underset{(n-r)\times s}{Q}$ and $\underset{(n-r)\times(m-s)}{R}$ are arbitrary, and V^+ and W^+ are Moore-Penrose inverses.

Proof. We can write Y in the form

$$Y = \begin{bmatrix} \underset{n\times r}{K_1} & \underset{n\times(n-r)}{K_0} \end{bmatrix} \begin{bmatrix} \underset{r\times s}{Y_{11}} & \underset{r\times(m-s)}{Y_{10}} \\ \underset{(n-r)\times s}{Y_{01}} & \underset{(n-r)\times(m-s)}{Y_{00}} \end{bmatrix} \begin{bmatrix} \underset{s\times m}{L_1'} \\ \underset{(m-s)\times m}{L_0'} \end{bmatrix}.$$

Now look for X in the form

$$X = \begin{bmatrix} \underset{n\times r}{K_1} & \underset{n\times(n-r)}{K_0} \end{bmatrix} \begin{bmatrix} \underset{r\times s}{X_{11}} & \underset{r\times(m-s)}{X_{10}} \\ \underset{(n-r)\times s}{X_{01}} & \underset{(n-r)\times(m-s)}{X_{00}} \end{bmatrix} \begin{bmatrix} \underset{s\times m}{L_1'} \\ \underset{(m-s)\times m}{L_0'} \end{bmatrix}.$$

It follows that we must have

$$\begin{bmatrix} \underset{r\times s}{\Lambda X_{11}\Omega} & \underset{r\times(m-s)}{\emptyset} \\ \underset{(n-r)\times s}{\emptyset} & \underset{(n-r)\times(m-s)}{\emptyset} \end{bmatrix} = \begin{bmatrix} \underset{r\times s}{Y_{11}} & \underset{r\times(m-s)}{Y_{10}} \\ \underset{(n-r)\times s}{Y_{01}} & \underset{(n-r)\times(m-s)}{Y_{00}} \end{bmatrix}.$$

Thus $VXW = Y$ is solvable if and only if Y_{10}, Y_{01}, and Y_{00} are all zero, which is equivalent to having both $YL_0 = 0$ and $Y'K_0 = 0$. We then have $X_{11} = \Lambda^{-1}Y_{11}\Omega^{-1} = \Lambda^{-1}K_1'YL_1\Omega^{-1}$, while X_{10}, X_{01}, and X_{00} can be chosen arbitrarily. This gives the expression for X in the theorem. □

If the equations are the stationary equations of a linear least squares problem, as they are in equations (3) in the body of the paper, then they are by definition solvable. A convenient solution in that case, which happens to be the solution of minimum norm, is $X = V^+YW^+$.

Appendix B. Weighted Procrustus

Suppose A is $n \times m$ and C and D are positive semi-definite of orders n and m, respectively. In the Weighted Procrustus Problem we want find an $n \times m$ matrix B such that $B'CB = I$ and $\mathbf{tr}\ (A-B)'C(A-B)D$ is minimized. This is clearly equivalent to maximizing $\mathbf{tr}\ B'H$ over $B'CB = I$, where $H = CAD$.

Suppose C has rank r. The positive eigenvalues are in the $r \times r$ diagonal matrix Λ, and the corresponding eigenvectors are in the $n \times r$ matrix K_1. Eigenvalues corresponding to the null space of C are in the $n \times (n-r)$ matrix K_0.

Lemma B.1. *$f(B) = \mathbf{tr}\ B'H$ has a maximum over $B'CB = I$ if and only if $r \geq m$.*

Proof. We can write $B = K_1\Lambda^{-\frac{1}{2}}S + K_0T$, where S is $r \times m$ and T is $(n-r)\times m$. In these new coordinates we have to maximize $\mathbf{tr}\ S'G$ over $S'S = I$. The equation $S'S = I$, and thus our maximization problem, has a solution if and only if $r \geq m$. □

Define the $r \times m$ matrix $G = \Lambda^{\frac{1}{2}}K_1'AD$, and suppose G has rank $s \leq m$. The singular value decomposition of G is

$$G = \begin{bmatrix} \underset{r\times s}{P_1} & \underset{r\times(r-s)}{P_0} \end{bmatrix} \begin{bmatrix} \underset{s\times s}{\Omega} & \underset{s\times(m-s)}{\emptyset} \\ \underset{(r-s)\times s}{\emptyset} & \underset{(r-s)\times(m-s)}{\emptyset} \end{bmatrix} \begin{bmatrix} \underset{s\times m}{Q_1'} \\ \underset{(m-s)\times m}{Q_0'} \end{bmatrix}.$$

Lemma B.2. *The maximum of $\mathbf{tr}\ S'G$ over $r \times m$ matrices S satisfying $S'S = I$ is $\mathbf{tr}\ \Omega$, and it is attained for any S of the form $S = P_1Q_1' + P_0RQ_0'$, where R is any $(r-s)\times(m-s)$ matrix satisfying $R'R = I$.*

Proof. Using a symmetric matrix of Lagrange multipliers M leads to the stationary equations $G = SM$, which implies $G'G = M^2$ or $M = \pm(G'G)^{1/2}$. It also implies that at a solution of the stationary equations $\mathbf{tr}\ S'G = \pm\mathbf{tr}\ \Omega$. The negative sign corresponds with the minimum, the positive sign with the maximum.

Thus

$$M = \begin{bmatrix} \underset{m\times s}{Q_1} & \underset{m\times(m-s)}{Q_0} \end{bmatrix} \begin{bmatrix} \underset{s\times s}{\Omega} & \underset{s\times(m-s)}{\emptyset} \\ \underset{(m-s)\times s}{\emptyset} & \underset{(m-s)\times(m-s)}{\emptyset} \end{bmatrix} \begin{bmatrix} \underset{s\times m}{Q_1'} \\ \underset{(m-s)\times m}{Q_0'} \end{bmatrix}.$$

If we write S in the form

$$S = \begin{bmatrix} \underset{r\times s}{P_1} & \underset{r\times(r-s)}{P_0} \end{bmatrix} \begin{bmatrix} \underset{s\times m}{S_1} \\ \underset{(r-s)\times m}{S_0} \end{bmatrix}$$

then $G = SM$ can be simplified to

$$S_1 Q_1 = I,$$

$$S_0 Q_1 = 0,$$

with in addition, of course, $S_1' S_1 + S_0' S_0 = I$. It follows that $S_1 = Q_1'$ and

$$\underset{(r-s)\times m}{S_0} = \underset{(r-s)\times(m-s)}{R} \underset{(m-s)\times m}{Q_0'},$$

with $R'R = I$. Thus $S = P_1 Q_1' + P_0 R Q_0'$. □

Theorem B.3. *Suppose $r \geq m$. $f(B) = \mathbf{tr}\ B'H$ attains a maximum equal to $\mathbf{tr}\ \Omega$ over $B'CB = I$ at all*

$$B = K_1 \Lambda^{-\frac{1}{2}} (P_1 Q_1' + P_0 R Q_0') + K_0 T,$$

where R is $(r-s)\times(m-s)$ and satisfies $R'R = I$, and where T is $(n-r)\times m$ and arbitrary.

Proof. From the lemmas. □

Appendix C. Code

```

#
#    constrPCA package
#    Copyright (C) 2009  Jan de Leeuw <deleeuw@stat.ucla.edu>
#    UCLA Department of Statistics, Box 951554, Los Angeles, CA
     90095-1554
#
#    This program is free software; you can redistribute it and/or modify
#    it under the terms of the GNU General Public License as published by
#    the Free Software Foundation; either version 2 of the License, or
#    (at your option) any later version.
#
#    This program is distributed in the hope that it will be useful,
#    but WITHOUT ANY WARRANTY; without even the implied warranty of
#    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
#    GNU General Public License for more details.
#
#    You should have received a copy of the GNU General Public License
#    along with this program; if not, write to the Free Software
#    Foundation, Inc., 675 Mass Ave, Cambridge, MA 02139, USA.
#
###################################################################
#
# version 0.0.1, 2009-02-21   Initial Alpha
# version 0.0.2, 2009-02-23   Orthogonality
# version 0.0.3, 2009-03-05   Bugfix, R^2
# version 0.0.4, 2009-03-07   Replace lsfit() by qr.solve()
# version 0.1.0, 2009-03-20   Weights
#

constrPCA<-function(x,p,wC=diag(nrow(x)),wR=diag(ncol(x)),aFix=matrix(0,
    nrow(x),p),bFix=matrix(0,ncol(x),p),aEqual=NULL,bEqual=NULL,aOrth=
    FALSE,bOrth=FALSE,eps=1e-6,itmax=100,verbose=TRUE) {
n<-nrow(x); m<-ncol(x); itel<-1
na<-length(aEqual); nb<-length(bEqual)
if (aOrth) aOld<-procrus(matrix(rnorm(n*p),n,p))
if (aOrth==FALSE) {
        aOld<-aFix
    if (na > 0) aOld<-aOld+vecList(n,p,rep(1,na),aEqual)
    }
if (bOrth) bOld<-procrus(matrix(rnorm(m*p),m,p))
```

```
if (bOrth==FALSE){
    bOld<-bFix
    if (nb > 0) bOld<-bOld+vecList(m,p,rep(1,nb),bEqual)
    }
fOld<-sum((x-tcrossprod(aOld,bOld))^2)
repeat {
    cc<-crossprod(bOld)
    aTilde<-t(qr.solve(bOld,t(x)))
    if (aOrth) aNew<-procrus(aTilde%*%cc)
    if (aOrth==FALSE){
        rTilde<-(aTilde-aFix)%*%cc
        aNew<-aFix
        if (na > 0) {
            ta<-rep(0,na); ca<-matrix(0,na,na)
            for (i in 1:na) {
                gi<-makeIndi(n,p,aEqual[[i]])
                ta[i]<-sum(gi*rTilde)
                for (j in 1:na) {
                    gj<-makeIndi(n,p,aEqual[[j]])
                    ca[i,j]<-sum(cc*crossprod(gi,gj))
                    }
                }
            v<-solve(ca,ta)(norw(x))
            aNew<-aNew+vecList(n,p,v,aEqual)
            }
    }
    fNwA<-sum((x-tcrossprod(aNew,bOld))^2)
    cc<-crossprod(aNew)
    bTilde<-t(qr.solve(aNew,x))
    if (bOrth) bNew<-procrus(bTilde%*%cc)
    if (bOrth==FALSE){
        rTilde<-(bTilde-bFix)%*%cc
        bNew<-bFix
        if (nb > 0) {
            tb<-rep(0,nb); cb<-matrix(0,nb,nb)
            for (i in 1:nb) {
                gi<-makeIndi(m,p,bEqual[[i]])
                tb[i]<-sum(gi*rTilde)
                for (j in 1:nb) {
                    gj<-makeIndi(m,p,bEqual[[j]])
                    cb[i,j]<-sum(cc*crossprod(gi,gj))
                    }
                }
```

```
            w<-solve(cb,tb)
            bNew<-bNew+vecList(m,p,w,bEqual)
            }
    }
    fNwB<-sum((x-tcrossprod(aNew,bNew))^2)
    if (verbose) cat(
        "Iteration: ",formatC(itel,width=3, format="d"),
        "fOld: ",formatC(fOld,digits=8,width=12,format="f"),
        "fNwA: ",formatC(fNwA,digits=8,width=12,format="f"),
        "fNwB: ",formatC(fNwB,digits=8,width=12,format="f"),
        "\n")
    if (((fOld - fNwB) < eps) || (itel == itmax)) break()
    aOld<-aNew; bOld<-bNew; fOld<-fNwB; itel<-itel+1
    }
return(list(itel=itel,loss=fNwB,a=aNew,b=bNew))
}

makeIndi<-function(n,p,k) {
x<-matrix(0,n,p)
x[k]<-1
return(x)
}

vecList<-function(n,p,v,x) {
y<-matrix(0,n,p); s<-length(v)
if (s == 0) stop("empty list")
if (length(x) != s) stop("length error")
for (i in 1:s) y<-y+v[i]*makeIndi(n,p,x[[i]])
return(y)
}

unconsRow<-function(){}

uncondCol<-function(){}

procrus<-function(x) {
    s<-svd(x)
    return(tcrossprod(s$u,s$v))
    }
```

DEPARTMENT OF STATISTICS, UNIVERSITY OF CALIFORNIA, LOS ANGELES, CA 90095-1554

E-mail address, Jan de Leeuw: deleeuw@stat.ucla.edu

URL, Jan de Leeuw: http://www.stat.ucla.edu/~deleeuw

E-mail address, Irina Kukuyeva: ikukuyeva@stat.ucla.edu

URL, Irina Kukuyeva: http://www.stat.ucla.edu/~ikukuyeva