Constrained Principal Components

Jan de Leeuw

First created on March 12, 2019. Last update on April 15, 2019

Abstract

In the usual forms of least squares nonlinear principal component analysis observed variables are quantified or transformed to optimize low-rank approximations. Thus NLPCA is linear PCA on optimally scaled variables. In this note we extend the approach by allowing for optimally scaled components.

Contents

Note: This is a working paper which will be expanded/updated frequently. All suggestions for improvement are welcome. The directory deleeuwpdx.net/pubfolders/ordered has a pdf version, the bib file, the complete Rmd file with the code chunks, and the R and C source code. Thanks to Yoshio for some useful comments.

1 Loss Function

The problem studied in this paper is minimization of the least squares loss function

$$\sigma(X,B)=\mathbf{SSQ}(Y-X\underline{B}) \tag{1}$$

over X and B. Here $\mathbf{SSQ}()$ stands for the sum of squares, i.e. the square of the Frobenius norm. We use $\underline{B}$ for the matrix transpose of B. The (partial) unknowns are X, an $n\times p$

matrix of *components*, and B, an $m \times p$ matrix of *loadings*. This is just least squares low-rank approximation. It is typically solved by directly using the singular value decomposition of Y, or by using some version of alternating least squares (iterate finding the optimal X for given B and the optimal B for given X).

In order to at least partially identify X and B we require, without loss of generality, that $\underline{X}X = I$, i.e. the components are orthonormal. This identifies X and B up to a rotation. All of this is completely standard. In this paper we introduce additional non-trivial constraints on the components. As an example, we could require for the first component

$$x_{11} \leq x_{21} \leq \cdots \leq x_{n1}, \tag{2}$$

but general partial orders and other cone or subspace constraints may also be useful. The important thing is that the constraints are imposed on each component seperately.

We write Ω_s for the set of all matrices satisfying the constraint for component s, which means we require $X \in \Omega$, with

$$\Omega = \Omega_1 \otimes \cdots \otimes \Omega_p.$$

This form of nonlinear principal component analysis is different from the more familiar form in which the columns of Y are transformed nonlinearly to optimize (1). See De Leeuw (2006) or De Leeuw (2014) for discussion and references. It is more closely related to the forms of constrained principal component analysis discussed in great detail by Takane (2014), and specifically to the form that allows separate constraints for separate dimensions (Takane, Kiers, and De Leeuw (1995)). In constrained principal component analysis, however, the emphasis is primarily on linear or subspace constraints on the components.

2 Alternating Least Squares

If we apply block relaxation to a least squares loss function we obtain an alternating least squares algorithm (De Leeuw (1994), De Leeuw (2016a)). Alternating least squares algorithms as a general class of algorithms useful in data analysis were introduced by De Leeuw (1968), and then applied systematically in the ALSOS project, starting with De Leeuw, Young, and Takane (1976).

0: Start with $k = 0$ and $X^{(0)} \in \Omega$.

1: $B^{(k)} \in \mathbf{argmin}_{B \in \mathbb{R}^{m \times p}}\ \mathbf{SSQ}\ (Y - X^{(k)}\underline{B})$

2: $X^{(k+1)} \in \mathbf{argmin}_{X \in \Omega}\ \mathbf{SSQ}\ (Y - X\underline{B}^{k})$

3: If there is convergence, stop, otherwise $k \leftarrow k + 1$ and go back to step k1.

Step 1 is a straightforward linear least squares problem, since we do not impose any constraints on B. Step 2 is more complicated, and we'll discuss it in detail in the next section. Convergence can be defined in terms of the decreasing sequence of loss function values, or in terms of changes in X and B from one iteration to the next.

3 Majorization

Consider the problem of minimizing $\mathbf{SSQ}(Y - X\underline{B})$ over $X \in \mathcal{K}$. Although the constraints are defined for each column of X separately, the matrix B combines columns and thus complicates the overall minimization problem. In order to solve the problem we use the majorization method, introduced by De Leeuw (1994), Heiser (1995), and Lange, Hunter, and Yang (2000), to separate the columns again. Majorization methods are discussed systematically in Lange (2016) and De Leeuw (2016a). In our overall algorithm this means we need an iterative majorization algorithm in step 2 of each cycle of the alternating least squares algorithm.

For our majorization we choose a positive semi-definite diagonal matrix W such that $W \gtrsim \underline{B}B$, i.e. such that $W - \underline{B}B$ is positive semi-definite. We can use, for example, $W = \lambda I$, with λ the largest eigenvalue of $\underline{B}B$, or we can choose $W = p\ \mathbf{diag}(\underline{B}B)$, with p the number of columns of X and B. Of course if $\underline{B}B$ already is diagonal we can choose $W = \underline{B}B$.

Suppose $Z \in \mathcal{K}$, and in the majorization iteration we want improve Z. By defining

$$U = Z + (Y - Z\underline{B})BW^{-1}$$

and completing the square, we find the majorization inequality

$$\mathbf{SSQ}(Y - X\underline{B}) \leq \mathbf{SSQ}(Y - Z\underline{B}) + \mathbf{tr}\ (X - U)W\underline{(X - U)} - \mathbf{tr}\ UW\underline{U},$$

with equality if $Z = X$.

In the majorization algorithm we iteratively minimize

$$\mathbf{tr}\ (X - U)W\underline{(X - U)} = \sum_{s=1}^{p} w_s \mathbf{SSQ}(x_{\star s} - u_{\star s})$$

over $x_{\star s} \in \mathcal{K}_s$, which can obviously be done for each s separately. The update of column $x_{\star s}$ of X is the metric projection of column $u_{\star s}$ of U on Ω_s. For constraint (2), for example, we apply isotone regression to $u_{\star s}$.

4 Example: Linear Constraints

Our first example is in the spirit of the DCDD method in Takane (2014). Each column of X is constrained linearly by $x_{\star s} = G_s \alpha_s$. Thus $x_{\star s}$ must be in the subspace spanned by the columns of G_s. The G_s can be design matrices, indicator matrices, bases of polynomials or splines, and whatever else.

Here is a simple example.

```
g1 <- matrix (0, 16, 4)
g1[1:4, 1] <- 1
g1[5:8, 2] <- 1
g1[9:12, 3] <- 1
g1[13:16, 4] <- 1
```

```
g1 <- standardize (center (g1))
g2 <- rbind (diag (4), diag (4), diag (4), diag (4))
g2 <- standardize (center (g2))
g <- list (g1, g2)
set.seed (12345)
y <- standardize (center (mnorm(16, 5)))
```

The `linRes` function approximates Y using the two constrained dimensions code in the list with matrices G_1 and G_2. Note that in the current implementation we only do a single majorization iteration to update the columns of X before we update B. A more flexible strategy would be to allow for more than one inner majorization iterations.

```
h <- linRes (y, g)
```

```
## itel     1 fold     4.6627879883 fnew     4.6085187514
## itel     2 fold     4.6085187514 fnew     4.5490565074
## itel     3 fold     4.5490565074 fnew     4.4865061504
## itel     4 fold     4.4865061504 fnew     4.4189832984
## itel     5 fold     4.4189832984 fnew     4.3676298613
## itel     6 fold     4.3676298613 fnew     4.3396567038
## itel     7 fold     4.3396567038 fnew     4.3279401130
## itel     8 fold     4.3279401130 fnew     4.3238626172
## itel     9 fold     4.3238626172 fnew     4.3226023894
## itel    10 fold     4.3226023894 fnew     4.3222239136
## itel    11 fold     4.3222239136 fnew     4.3220998929
## itel    12 fold     4.3220998929 fnew     4.3220505942
## itel    13 fold     4.3220505942 fnew     4.3220267104
## itel    14 fold     4.3220267104 fnew     4.3220135396
## itel    15 fold     4.3220135396 fnew     4.3220058435
## itel    16 fold     4.3220058435 fnew     4.3220012301
## itel    17 fold     4.3220012301 fnew     4.3219984440
## itel    18 fold     4.3219984440 fnew     4.3219967506
## itel    19 fold     4.3219967506 fnew     4.3219957194
## itel    20 fold     4.3219957194 fnew     4.3219950873
## itel    21 fold     4.3219950873 fnew     4.3219946980
## itel    22 fold     4.3219946980 fnew     4.3219944556
## itel    23 fold     4.3219944556 fnew     4.3219943028
## itel    24 fold     4.3219943028 fnew     4.3219942048
## itel    25 fold     4.3219942048 fnew     4.3219941406
## itel    26 fold     4.3219941406 fnew     4.3219940973
## itel    27 fold     4.3219940973 fnew     4.3219940671
## itel    28 fold     4.3219940671 fnew     4.3219940454
## itel    29 fold     4.3219940454 fnew     4.3219940292
## itel    30 fold     4.3219940292 fnew     4.3219940166
## itel    31 fold     4.3219940166 fnew     4.3219940066
```

```
## itel    32 fold     4.3219940066 fnew     4.3219939985
## itel    33 fold     4.3219939985 fnew     4.3219939917
## itel    34 fold     4.3219939917 fnew     4.3219939860
## itel    35 fold     4.3219939860 fnew     4.3219939810
## itel    36 fold     4.3219939810 fnew     4.3219939768
## itel    37 fold     4.3219939768 fnew     4.3219939731
## itel    38 fold     4.3219939731 fnew     4.3219939699
## itel    39 fold     4.3219939699 fnew     4.3219939671
## itel    40 fold     4.3219939671 fnew     4.3219939646
## itel    41 fold     4.3219939646 fnew     4.3219939625
## itel    42 fold     4.3219939625 fnew     4.3219939606
## itel    43 fold     4.3219939606 fnew     4.3219939589
## itel    44 fold     4.3219939589 fnew     4.3219939575
## itel    45 fold     4.3219939575 fnew     4.3219939562
## itel    46 fold     4.3219939562 fnew     4.3219939550
## itel    47 fold     4.3219939550 fnew     4.3219939540
## itel    48 fold     4.3219939540 fnew     4.3219939532
## itel    49 fold     4.3219939532 fnew     4.3219939524
## itel    50 fold     4.3219939524 fnew     4.3219939517
## itel    51 fold     4.3219939517 fnew     4.3219939511
## itel    52 fold     4.3219939511 fnew     4.3219939506
## itel    53 fold     4.3219939506 fnew     4.3219939501
## itel    54 fold     4.3219939501 fnew     4.3219939497
## itel    55 fold     4.3219939497 fnew     4.3219939494
## itel    56 fold     4.3219939494 fnew     4.3219939490
## itel    57 fold     4.3219939490 fnew     4.3219939488
## itel    58 fold     4.3219939488 fnew     4.3219939485
## itel    59 fold     4.3219939485 fnew     4.3219939483
## itel    60 fold     4.3219939483 fnew     4.3219939481
## itel    61 fold     4.3219939481 fnew     4.3219939479
## itel    62 fold     4.3219939479 fnew     4.3219939478
## itel    63 fold     4.3219939478 fnew     4.3219939477
## itel    64 fold     4.3219939477 fnew     4.3219939475
## itel    65 fold     4.3219939475 fnew     4.3219939474
## itel    66 fold     4.3219939474 fnew     4.3219939474
```

5 Example: Multiple Ordinal Variables

In the nonlinear multivariate analysis system of Gifi (1990) (see also Michailidis and De Leeuw (1998)) transformations can be nominal, ordinal, or numerical, and quantifications can be single or multiple. Multiple ordinal transformations have not really been implemented because their definition has never been entirely obvious.

A definition that has been used before is to require that the first column $x_{\star 1}$ is isotone, while the remaining $p-1$ columns are arbitrary, but orthogonal to the first. Thus we have $\underline{X}X = I$

with an isotone first column. It is easy to see that the orthonormality constraint is actually just for identification purposes, it does not enter into the majorization iterations. In fact, the algorithm in the funcion `multOrd` does not impose orthonormality. It uses the modified Gram-Schmidt method to transform to orthonormality after convergence, using the fact that modified Gram-Schmidt (without pivoting) merely normalizes the first column, which does not disturb the order relations. Clearly the same trick can be used for other cone constraints.

It is interesting to observe that requiring isotonicity of the first column of X is actually equivalent to requiring that an isotone vector exists in the column space of X. Because if such a vector exists we can use the indeterminacy in the product $X\underline{B}$ to move that vector to the first column of X.

For our numerical example we use the same Y as before.

```r
set.seed (12345)
y <- standardize (center (mnorm(16, 5)))
```

The program `multOrd` require the first column of X to be increasing. It uses Gram-Schmidt from De Leeuw (2015) and isotone regression from De Leeuw (2016b).

```r
h <- multOrd (y, 2)
```

```
## itel     1 fold     2.9238552791 fnew     2.3439684622
## itel     2 fold     2.3439684622 fnew     2.0718818606
## itel     3 fold     2.0718818606 fnew     2.0413535991
## itel     4 fold     2.0413535991 fnew     2.0263303595
## itel     5 fold     2.0263303595 fnew     2.0212415190
## itel     6 fold     2.0212415190 fnew     2.0177619980
## itel     7 fold     2.0177619980 fnew     2.0149648432
## itel     8 fold     2.0149648432 fnew     2.0127152559
## itel     9 fold     2.0127152559 fnew     2.0108485613
## itel    10 fold     2.0108485613 fnew     2.0093161023
## itel    11 fold     2.0093161023 fnew     2.0080300445
## itel    12 fold     2.0080300445 fnew     2.0069592899
## itel    13 fold     2.0069592899 fnew     2.0060533653
## itel    14 fold     2.0060533653 fnew     2.0052907927
## itel    15 fold     2.0052907927 fnew     2.0046414247
## itel    16 fold     2.0046414247 fnew     2.0040901592
## itel    17 fold     2.0040901592 fnew     2.0036182740
## itel    18 fold     2.0036182740 fnew     2.0032150474
## itel    19 fold     2.0032150474 fnew     2.0028684164
## itel    20 fold     2.0028684164 fnew     2.0025707084
## itel    21 fold     2.0025707084 fnew     2.0023138952
## itel    22 fold     2.0023138952 fnew     2.0020924447
## itel    23 fold     2.0020924447 fnew     2.0019008666
## itel    24 fold     2.0019008666 fnew     2.0017351430
## itel    25 fold     2.0017351430 fnew     2.0015914345
```

```
## itel    26 fold     2.0015914345 fnew     2.0014668028
## itel    27 fold     2.0014668028 fnew     2.0013585141
## itel    28 fold     2.0013585141 fnew     2.0012644056
## itel    29 fold     2.0012644056 fnew     2.0011825024
## itel    30 fold     2.0011825024 fnew     2.0011112032
## itel    31 fold     2.0011112032 fnew     2.0010490649
## itel    32 fold     2.0010490649 fnew     2.0009948954
## itel    33 fold     2.0009948954 fnew     2.0009476303
## itel    34 fold     2.0009476303 fnew     2.0009063783
## itel    35 fold     2.0009063783 fnew     2.0008703482
## itel    36 fold     2.0008703482 fnew     2.0008388706
## itel    37 fold     2.0008388706 fnew     2.0008113543
## itel    38 fold     2.0008113543 fnew     2.0007872946
## itel    39 fold     2.0007872946 fnew     2.0007662472
## itel    40 fold     2.0007662472 fnew     2.0007478306
## itel    41 fold     2.0007478306 fnew     2.0007317096
## itel    42 fold     2.0007317096 fnew     2.0007175951
## itel    43 fold     2.0007175951 fnew     2.0007052332
## itel    44 fold     2.0007052332 fnew     2.0006944042
## itel    45 fold     2.0006944042 fnew     2.0006849153
## itel    46 fold     2.0006849153 fnew     2.0006765992
## itel    47 fold     2.0006765992 fnew     2.0006693093
## itel    48 fold     2.0006693093 fnew     2.0006629180
## itel    49 fold     2.0006629180 fnew     2.0006573133
## itel    50 fold     2.0006573133 fnew     2.0006523977
## itel    51 fold     2.0006523977 fnew     2.0006480858
## itel    52 fold     2.0006480858 fnew     2.0006443030
## itel    53 fold     2.0006443030 fnew     2.0006409838
## itel    54 fold     2.0006409838 fnew     2.0006380711
## itel    55 fold     2.0006380711 fnew     2.0006355148
## itel    56 fold     2.0006355148 fnew     2.0006332710
## itel    57 fold     2.0006332710 fnew     2.0006313014
## itel    58 fold     2.0006313014 fnew     2.0006295723
## itel    59 fold     2.0006295723 fnew     2.0006280541
## itel    60 fold     2.0006280541 fnew     2.0006267211
## itel    61 fold     2.0006267211 fnew     2.0006255505
## itel    62 fold     2.0006255505 fnew     2.0006245225
## itel    63 fold     2.0006245225 fnew     2.0006236196
## itel    64 fold     2.0006236196 fnew     2.0006228267
## itel    65 fold     2.0006228267 fnew     2.0006221302
## itel    66 fold     2.0006221302 fnew     2.0006215183
## itel    67 fold     2.0006215183 fnew     2.0006209809
## itel    68 fold     2.0006209809 fnew     2.0006205087
## itel    69 fold     2.0006205087 fnew     2.0006200939
## itel    70 fold     2.0006200939 fnew     2.0006197295
```

```
## itel    71 fold     2.0006197295 fnew     2.0006194093
## itel    72 fold     2.0006194093 fnew     2.0006191279
## itel    73 fold     2.0006191279 fnew     2.0006188807
## itel    74 fold     2.0006188807 fnew     2.0006186635
## itel    75 fold     2.0006186635 fnew     2.0006184726
## itel    76 fold     2.0006184726 fnew     2.0006183048
## itel    77 fold     2.0006183048 fnew     2.0006181574
## itel    78 fold     2.0006181574 fnew     2.0006180279
## itel    79 fold     2.0006180279 fnew     2.0006179140
## itel    80 fold     2.0006179140 fnew     2.0006178140
## itel    81 fold     2.0006178140 fnew     2.0006177260
## itel    82 fold     2.0006177260 fnew     2.0006176487
## itel    83 fold     2.0006176487 fnew     2.0006175808
## itel    84 fold     2.0006175808 fnew     2.0006175211
## itel    85 fold     2.0006175211 fnew     2.0006174686
## itel    86 fold     2.0006174686 fnew     2.0006174225
## itel    87 fold     2.0006174225 fnew     2.0006173820
## itel    88 fold     2.0006173820 fnew     2.0006173463
## itel    89 fold     2.0006173463 fnew     2.0006173150
## itel    90 fold     2.0006173150 fnew     2.0006172875
## itel    91 fold     2.0006172875 fnew     2.0006172633
## itel    92 fold     2.0006172633 fnew     2.0006172420
## itel    93 fold     2.0006172420 fnew     2.0006172233
## itel    94 fold     2.0006172233 fnew     2.0006172069
## itel    95 fold     2.0006172069 fnew     2.0006171924
## itel    96 fold     2.0006171924 fnew     2.0006171797
## itel    97 fold     2.0006171797 fnew     2.0006171686
## itel    98 fold     2.0006171686 fnew     2.0006171588
## itel    99 fold     2.0006171588 fnew     2.0006171501
## itel   100 fold     2.0006171501 fnew     2.0006171425
## itel   101 fold     2.0006171425 fnew     2.0006171359
## itel   102 fold     2.0006171359 fnew     2.0006171300
## itel   103 fold     2.0006171300 fnew     2.0006171249
## itel   104 fold     2.0006171249 fnew     2.0006171203
## itel   105 fold     2.0006171203 fnew     2.0006171164
## itel   106 fold     2.0006171164 fnew     2.0006171129
## itel   107 fold     2.0006171129 fnew     2.0006171098
## itel   108 fold     2.0006171098 fnew     2.0006171071
## itel   109 fold     2.0006171071 fnew     2.0006171047
## itel   110 fold     2.0006171047 fnew     2.0006171026
## itel   111 fold     2.0006171026 fnew     2.0006171008
## itel   112 fold     2.0006171008 fnew     2.0006170992
## itel   113 fold     2.0006170992 fnew     2.0006170977
## itel   114 fold     2.0006170977 fnew     2.0006170965
## itel   115 fold     2.0006170965 fnew     2.0006170954
```

```
## itel   116 fold     2.0006170954 fnew     2.0006170944
## itel   117 fold     2.0006170944 fnew     2.0006170936
## itel   118 fold     2.0006170936 fnew     2.0006170928
## itel   119 fold     2.0006170928 fnew     2.0006170922
## itel   120 fold     2.0006170922 fnew     2.0006170916
## itel   121 fold     2.0006170916 fnew     2.0006170911
## itel   122 fold     2.0006170911 fnew     2.0006170906
## itel   123 fold     2.0006170906 fnew     2.0006170903
## itel   124 fold     2.0006170903 fnew     2.0006170899
## itel   125 fold     2.0006170899 fnew     2.0006170896
## itel   126 fold     2.0006170896 fnew     2.0006170893
## itel   127 fold     2.0006170893 fnew     2.0006170891
## itel   128 fold     2.0006170891 fnew     2.0006170889
## itel   129 fold     2.0006170889 fnew     2.0006170887
## itel   130 fold     2.0006170887 fnew     2.0006170886
## itel   131 fold     2.0006170886 fnew     2.0006170884
## itel   132 fold     2.0006170884 fnew     2.0006170883
## itel   133 fold     2.0006170883 fnew     2.0006170882
## itel   134 fold     2.0006170882 fnew     2.0006170881
```

6 Appendix: Weights

In least squares majorization with non-diagonal positive definite weight matrix C we often want to find a diagonal matrix such that $D \gtrsim C$ and D is, in some sense, as small as possible (cf Groenen, Giaquinto, and Kiers (2003)). This problem is similar, but not identical, to minimum trace factor analysis (MTFA). In MTFA we find a diagonal $D \gtrsim 0$ such that $D \lesssim C$ and D is as large as possible. Typically "large" is defined as maximizing the trace or some other linear function of the diagonal elements. The majorization problem is different, not so much because "as large as possible" is replaced by "as small as possible", but more so because $D \gtrsim C$ implies that $D \gtrsim 0$.

For our majorization we could also use the trace, so we want to minimize $\mathbf{tr}\ D$ over $D \gtrsim C$. As in MTFA, this is a convex programming problem. It can be solved in many ways (see, for example, Jamshidian and Bentler (1998)), but each of them involves a non-trivial computational effort. In the body of the paper we have mentioned two more simple, and more easily computable, choices for D. The first is $D = \lambda(C)I$, where $\lambda(C)$ is the largest eigenvalue of C. Of course if C is large computing the largest eigenvalue is not entirely trivial either. The second choice, which is actually used in the `R` programs in the appendix, is $D = p\ \mathbf{diag}(C)$, where p is the order of C and D. This is trivial to compute, but the corresponding D may not be very good (too large). It uses the trace as an upper bound for the largest eigenvalue of the correlation matrix corresponding with C, and there are much better bounds.

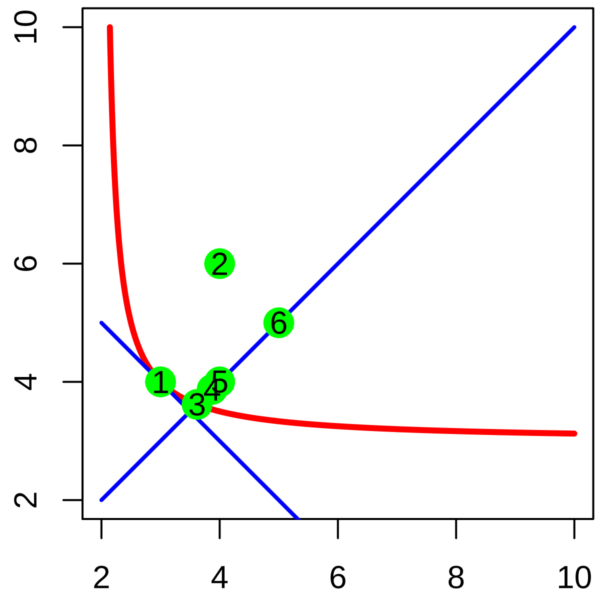
This suggests using better bounds for the maximum eigenvalue, which are relatively easy to compute. We use the result that $\lambda(C) \leq \|C\|$, where $\|C\|$ is any matrix norm. We could use the Frobenius norm $\sqrt{\mathbf{tr}\ C^2}$ or the norm $\max_{s=1}^p \sum_{t=1}^p |c_{st}|$.

Let's illustrate this with the 2×2 matrix

$$\begin{bmatrix} 2 & -1 \\ -1 & 3 \end{bmatrix}$$

In the figure below all points in the convex area above the red branch from the hyperbola $(x-2)(y-3)=1$ are D for which $D \gtrsim C$. The trace of D is minimized at $(3,4)$, where the blue line $x+y=7$ is tangent to the hyperbola. This is the point labeled 1. Point labeled 2 is $p\ \mathbf{diag}(C)$, which is $(4,6)$. The other four points are on the other blue line $x=y$. Point 3 is at $x=y=(5+\sqrt{5})/2$, which corresponds with the largest eigenvalue of C. Point 4 has $x=y=\sqrt{15}$, which is the Frobenius bound, and point 5 has $x=y=4$, which is the maximum row absolute sum norm. Finally point 6 uses the trace as the eigenvalue bound, which gives $x=y=5$.

```
par(pty="s")
f<-function (x,y) (x-2)*(y-3)-1
y<-seq(2,10,length=100)
x<-seq(2,10,length=100)
z<-outer(x,y,f)
contour(x=x,y=y,z=z,level=0,drawlabels=FALSE,lwd=3,col="RED")
lines(x,7-x,col="BLUE",lwd=2)
lines(x, x, col = "BLUE", lwd = 2)
lbd <- (5+sqrt(5))/2
sbd = sqrt(15)
points (lbd, lbd, col ="GREEN", cex = 2, pch = 19)
points (4, 6, col ="GREEN", cex = 2, pch = 19)
points (3, 4, col ="GREEN", cex = 2, pch = 19)
points (4, 4, col = "GREEN", cex = 2, pch = 19)
points (sbd, sbd, col = "GREEN", cex = 2, pch = 19)
points (5, 5, col = "GREEN", cex = 2, pch = 19)
text(3,4, labels = "1")
text(4,6, labels = "2")
text(lbd,lbd, labels = "3")
text(sbd,sbd, labels = "4")
text(4,4, labels = "5")
text(5,5, labels = "6")
```



7 Appendix: Code

7.1 linRes.R

```
linRes <-
  function (y,
            g,
            bound = "M",
            itmax = 1000,
            eps = 1e-10,
            verbose = TRUE,
            center = FALSE,
            standardize = FALSE) {
    if (center) {
      y <- center (y)
    }
    if (standardize) {
      y <- standardize (y)
    }
    p <- length (g)
    s <- 1
    xold <- NULL
    while (s <= p) {
      xold <- cbind (xold, g[[s]] %*% 1:ncol (g[[s]]))
      s <- s + 1
    }
    bold <- t (lm.fit (xold, y)$coefficients)
```

```
  rold <- y - tcrossprod (xold, bold)
  fold <- ssq (rold)
  itel <- 1
  repeat {
    bnew <- t (lm.fit (xold, y)$coefficients)
    cnew <- crossprod (bnew)
    if (bound == "M") {
      e <- max (rowSums (abs (cnew)))
      w <- diag (p) / e
    }
    if (bound == "E") {
      e <- max (eigen (cnew)$values)
      w <- diag (p) / e
    }
    if (bound == "F") {
      e <- sqrt (sum (cnew ^ 2))
      w <- diag (p) / e
    }
    if (bound == "D") {
      w <- diag (1 / (p * diag (cnew)))
    }
    u <- xold + rold %*% bnew %*% w
    xnew <- NULL
    s <- 1
    while (s <= p) {
      xnew <- cbind (xnew, lm.fit (g[[s]], u[, s])$fitted.values)
      s <- s + 1
    }
    rnew <- y - tcrossprod (xnew, bnew)
    fnew <- ssq (rnew)
    if (verbose) {
      cat (
        "itel ",
        formatC (itel, width = 4, format = "d"),
        "fold ",
        formatC (
          fold,
          width = 15,
          digits = 10,
          format = "f"
        ),
        "fnew ",
        formatC (
          fnew,
```

```
            width = 15,
            digits = 10,
            format = "f"
          ),
          "\n"
        )
      }
      if ((itel == itmax) || ((fold - fnew) < eps))
        break
      itel <- itel + 1
      fold <- fnew
      xold <- xnew
      bold <- bnew
      rold <- rnew
    }
    return (list (
      x = xnew,
      b = bnew,
      r = rnew,
      f = fnew
    ))
  }
```

7.2 multOrd.R

```
source ("jbkPava.R")
source ("gs.R")

multOrd <-
  function (y,
            p,
            bound ="M",
            itmax = 1000,
            eps = 1e-10,
            verbose = TRUE,
            center = FALSE,
            standardize = FALSE) {
    set.seed (12345)
    if (center) {
      y <- center (y)
    }
    if (standardize) {
      y <- standardize (y)
    }
```

```
n <- nrow (y)
xold <- standardize (center (cbind (1:n, mnorm (n, p - 1))))
bold <- t (lm.fit (xold, y)$coefficients)
rold <- y - tcrossprod (xold, bold)
fold <- ssq (rold)
itel <- 1
repeat {
  bnew <- t (lm.fit (xold, y)$coefficients)
  cnew <- crossprod (bnew)
  if (bound == "M") {
    e <- max (rowSums (abs (cnew)))
    w <- diag (p) / e
  }
  if (bound == "E") {
    e <- max (eigen (cnew)$values)
    w <- diag (p) / e
  }
  if (bound == "F") {
    e <- sqrt (sum (cnew ^ 2))
    w <- diag (p) / e
  }
  if (bound == "D") {
    w <- diag (1 / (p * diag (cnew)))
  }
  xnew <- xold + rold %*% bnew %*% w
  xnew[, 1] <- jbkPava (xnew[, 1])
  rnew <- y - tcrossprod (xnew, bnew)
  fnew <- ssq (rnew)
  if (verbose) {
    cat (
      "itel ",
      formatC (itel, width = 4, format = "d"),
      "fold ",
      formatC (
        fold,
        width = 15,
        digits = 10,
        format = "f"
      ),
      "fnew ",
      formatC (
        fnew,
        width = 15,
        digits = 10,
```

```
            format = "f"
          ),
          "\n"
        )
      }
      if ((itel == itmax) || ((fold - fnew) < eps))
        break
      itel <- itel + 1
      fold <- fnew
      xold <- xnew
      bold <- bnew
      rold <- rnew
    }
    z <- gsrc(xnew)
    return (list (x = z$q, b = tcrossprod (bnew, z$r), r = rnew, f = fnew))
  }
```

7.3 auxilary.R

```
mnorm <- function (n, p) {
  return (matrix (rnorm (n * p), n, p))
}

center <- function (y) {
  return (apply (y, 2, function (x)
    x - mean (x)))
}

standardize <- function (y) {
  return (apply (y, 2, function (x)
    x / norm (x)))
}

ssq <- function (x) {
  return (sum (x ^ 2))
}

norm <- function (x) {
  return (sqrt (ssq (x)))
}
```

References

De Leeuw, J. 1968. "Nonmetric Discriminant Analysis." Research Note 06-68. Department of Data Theory, University of Leiden. http://deleeuwpdx.net/janspubs/1968/reports/deleeuw_R_68d.pdf.

———. 1994. "Block Relaxation Algorithms in Statistics." In *Information Systems and Data Analysis*, edited by H.H. Bock, W. Lenski, and M.M. Richter, 308–24. Berlin: Springer Verlag. http://deleeuwpdx.net/janspubs/1994/chapters/deleeuw_C_94c.pdf.

———. 2006. "Nonlinear Principal Component Analysis and Related Techniques." In *Multiple Correspondence Analysis and Related Methods*, edited by M. Greenacre and J. Blasius, 107–33. Boca Raton, FA: Chapman; Hall. http://deleeuwpdx.net/janspubs/2006/chapters/deleeuw_C_06b.pdf.

———. 2014. "History of Nonlinear Principal Component Analysis." In *The Visualization and Verbalization of Data*, edited by J. Blasius and M. Greenacre. Chapman; Hall. http://deleeuwpdx.net/janspubs/2014/chapters/deleeuw_C_14.pdf.

———. 2015. "Exceedingly Simple Gram-Schmidt Code." http://deleeuwpdx.net/pubfolders/gs/gs.pdf.

———. 2016a. *Block Relaxation Methods in Statistics*. Bookdown. http://deleeuwpdx.net/bookdown/bras/_book.

———. 2016b. "Exceedingly Simple Isotone Regression with Ties." 2016. http://deleeuwpdx.net/pubfolders/isotone/isotone.pdf.

De Leeuw, J., F. W. Young, and Y. Takane. 1976. "Additive Structure in Qualitative Data: An Alternating Least Squares Method with Optimal Scaling Features." *Psychometrika* 41: 471–504. http://deleeuwpdx.net/janspubs/1976/articles/deleeuw_young_takane_A_76.pdf.

Gifi, A. 1990. *Nonlinear Multivariate Analysis*. New York, N.Y.: Wiley.

Groenen, P.J.F., P. Giaquinto, and H.A.L. Kiers. 2003. "Weighted Majorization Algorithms for Weighted Least Squares Decomposition Models." EI 2003-09. Rotterdam, Netherlands: Econometric Institute, Erasmus University.

Heiser, W.J. 1995. "Convergent Computing by Iterative Majorization: Theory and Applications in Multidimensional Data Analysis." In *Recent Advances in Descriptive Multivariate Analysis*, edited by W.J. Krzasnowski. Oxford, England: Clarendon Press.

Jamshidian, M., and P. M. Bentler. 1998. "A Quasi-Newton Method for Minimum Trace Factor Analysis." *Journal of Statistical Computation and Simulation* 62 (1-2): 73–89.

Lange, K. 2016. *MM Optimization Algorithms*. SIAM.

Lange, K., D.R. Hunter, and I. Yang. 2000. "Optimization Transfer Using Surrogate Objective Functions." *Journal of Computational and Graphical Statistics* 9: 1–20.

Michailidis, G., and J. De Leeuw. 1998. "The Gifi System for Descriptive Multivariate Analysis." *Statistical Science* 13: 307–36. http://deleeuwpdx.net/janspubs/1998/articles/

michailidis_deleeuw_A_98.pdf.

Takane, Y. 2014. *Constrained Principal Component Analysis and Related Techniques*. Monographs on Statistics and Applied Probability 129. CRC Press.

Takane, Y., H.A.L. Kiers, and J. De Leeuw. 1995. "Component Analysis with Different Sets of Constraints on Different Dimensions." *Psychometrika* 60: 259–80. http://deleeuwpdx.net/janspubs/1995/articles/takane_kiers_deleeuw_A_95.pdf.