

UCLA

Department of Statistics Papers

Title

The Multiway Package

Permalink

<https://escholarship.org/uc/item/54q399d6>

Author

de Leeuw, Jan

Publication Date

2008-01-09

THE MULTIWAY PACKAGE

JAN DE LEEUW

ABSTRACT. This note documents R code for the CANDECOMP and TUCKER generalizations of the Singular Value Decomposition to multiway arrays. Alternating Least Squares algorithms are used to generate a convergent sequence of low-rank approximations.

1. CANDECOMP

```
1 candecomp<-function(a, x, ortho=rep(FALSE, length(x)),
2   itmax=1000, eps=1e-6, verbose=FALSE)
```

The two leading arguments of the function `candecomp()` are an $n_1 \times n_2 \times \cdots \times n_m$ array `a` and a list of m matrices `x`, where `x[[i]]` has $n_i \times p$ elements. The function minimizes the least squares loss function over the `x[[i]]`, using the original values as starting values.

$$\sigma(x) = \sum_{i_1=1}^{n_1} \sum_{i_2=1}^{n_2} \cdots \sum_{i_m=1}^{n_m} (a_{i_1 i_2 \cdots i_m} - \sum_{s=1}^p x_{i_1 s}^1 x_{i_2 s}^2 \cdots x_{i_m s}^m)^2.$$

There is an additional parameter `ortho`, a logical vector of length m . If `ortho[i]` is TRUE then `x[[i]]` is required to be columnwise orthonormal. The additional arguments `itmax`, `eps` and `verbose` set the iteration parameters.

2. TUCKER

```
1 tucker<-function(a, x, ident=rep(FALSE, length(x)),
2   itmax=1000, eps=1e-6, verbose=FALSE)
```

Date: March 29, 2008 — 12h 24min — Typeset in TIMES ROMAN.

2000 Mathematics Subject Classification. 00A00.

Key words and phrases. Multidimensional Scaling, Array Approximation, Generalized SVD.

The leading arguments a and x are the same as before, except that now the $x[[i]]$ are $n_i \times p_i$ columnwise orthonormal matrices. The loss function is

$$\sigma(x, b) = \sum_{i_1=1}^{n_1} \sum_{i_2=1}^{n_2} \cdots \sum_{i_m=1}^{n_m} (a_{i_1 i_2 \cdots i_m} - \sum_{s_1=1}^{p_1} \sum_{s_2=1}^{p_2} \cdots \sum_{s_m=1}^{p_m} b_{s_1 s_2 \cdots s_m} x_{i_1 s_1}^1 x_{i_2 s_2}^2 \cdots x_{i_m s_m}^m)^2.$$

The $p_1 \times p_2 \times \cdots \times p_m$ array b is called the *core array*. There is an additional parameter `ident`, a logical vector of length m . If `ident[i]` is TRUE then matrix $x[[i]]$ is restricted to be the $n_i \times n_i$ identity matrix.

Note that we can also write

$$\sigma(x, b) = \|a - (X_1 \otimes X_2 \otimes \cdots \otimes X_m)b\|^2,$$

where a and b are the $\text{vec}(\bullet)$ of the arrays, \otimes is the Kronecker product, and $\|\bullet\|$ is the Frobenius norm.

3. UTILITIES

The package includes functions that compute generalized Hadamard products, generalized outer products and generalized Kronecker products on lists of matrices. A Procrustus approximation routine is also included.

4. THREE-WAY

In the important three-way case `tucker()` with `ident[3]=TRUE` fits the approximation $A_k = XB_kY'$, while `candecom()` fits the same approximation $A_k = XB_kY'$, with the B_k restricted to be diagonal.

5. CODE

```

1 #
2 # multiway package
3 # Copyright (C) 2008 Jan de Leeuw <deleeuw@stat.ucla.edu>
4 # UCLA Department of Statistics, Box 951554, Los Angeles, CA 90095-1554
5 #
6 # This program is free software; you can redistribute it and/or modify
7 # it under the terms of the GNU General Public License as published by
8 # the Free Software Foundation; either version 2 of the License, or
9 # (at your option) any later version.
10 #
11 # This program is distributed in the hope that it will be useful,
12 # but WITHOUT ANY WARRANTY; without even the implied warranty of
13 # MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
14 # GNU General Public License for more details.

```

```

15 #
16 # You should have received a copy of the GNU General Public License
17 # along with this program; if not, write to the Free Software
18 # Foundation, Inc., 675 Mass Ave, Cambridge, MA 02139, USA.
19 #
20 #####
21 #
22 # version 0.1.0, 2008-03-27,          first release
23 # version 0.2.0, 2008-03-28,          added tucker and constraints
24 #
25
26 candecomp<-function(a,x,ortho=rep(FALSE,length(x)),itmax=1000,eps=1e-6,verbose=
  FALSE) {
27   ndam<-dim(a)
28   nard<-length(ndam)
29   ndim<-ncol(x[[1]])
30   for (k in 1:nard) if (ortho[k]) x[[k]]<-procrustus(x[[k]])
31   c<-lapply(x,crossprod)
32   oloss<-candecomp(a,x)$loss
33   itel<-1
34   repeat {
35     for (k in 1:nard) {
36       cc<-arrHadamard(c[-k])
37       for (p in 1:ndim) {
38         y<-lapply(x[-k],function(z) z[,p])
39         b<-arrOuter(y)
40         x[[k]][,p]<-apply(a,k,function(z) sum(z*b))
41       }
42       if (ortho[k]) x[[k]]<-procrustus(x[[k]])
43       else x[[k]]<-t(solve(cc,t(x[[k]])))
44       c[[k]]<-crossprod(x[[k]])
45     }
46     cval<-candecomp(a,x)
47     nloss<-cval$loss; ahat<-cval$ahat
48     if (verbose)
49       cat("Iteration: ",formatC(itel,digits=3,width=3),
50         "Old Loss: ",formatC(oloss,digits=6,width=10,
51           format="f"),
52         "New Loss: ",formatC(nloss,digits=6,width=10,
53           format="f"),
54         "\n")
55     if ((itel == itmax) | ((oloss - nloss) < eps)) break()
56     itel<-itel+1; oloss<-nloss
57   }
58   return(list(x=x,ahat=ahat,loss=nloss))
59 }
60
61 candecomp<-function(a,x) {
62   ndim<-ncol(x[[1]])
63   ahat<-array(0,dim(a))
64   for (p in 1:ndim) {
65     y<-sapply(x,function(z) z[,p])
66     b<-arrOuter(y)

```

```

65         ahat<-ahat+b
66     }
67     loss<-sum((a-ahat)^2)
68     return(list(ahat=ahat,loss=loss))
69 }
70
71 tucker<-function(a,x,ident=rep(FALSE,length(x)),itmax=1000,eps=1e-6,verbose=FALSE)
    {
72     ndam<-dim(a)
73     for (k in 1:nard) if (ident[k]) x[[k]]<-diag(ndam[k])
74     ndbm<-sapply(x,function(z) ncol(z))
75     nard<-length(ndam)
76     rard<-rev(1:nard)
77     x<-lapply(x,procrustus)
78     xx<-arrKronecker(x)
79     aa<-as.vector(aperm(a,rard))
80     bb<-colSums(aa*xx)
81     b<-aperm(array(bb,rev(ndbm)),rard)
82     ahat<-aperm(array(drop(xx%*%bb),rev(ndam)),rard)
83     oloss<-sum((a-ahat)^2)
84     itel<-1
85     repeat {
86         for (k in 1:nard) {
87             if (!ident[k]) {
88                 z<-crossprod(flatten(a,k),arrKronecker(x[-k]))
89                 %*%flatten(b,k)
90                 x[[k]]<-procrustus(z)
91             }
92             xx<-arrKronecker(x)
93             aa<-as.vector(aperm(a,rard))
94             bb<-colSums(aa*xx)
95             b<-aperm(array(bb,rev(ndbm)),rard)
96             ahat<-aperm(array(drop(xx%*%bb),rev(ndam)),rev(1:nard))
97             nloss<-sum((a-ahat)^2)
98             if (verbose)
99                 cat("Iteration: ",formatC(itel,digits=3,width=3),
100                    "Old Loss: ",formatC(oloss,digits=6,width=10,
101                    format="f"),
102                    "New Loss: ",formatC(nloss,digits=6,width=10,
103                    format="f"),
104                    "\n")
105             if ((itel == itmax) | ((oloss - nloss) < eps)) break()
106             itel<-itel+1; oloss<-nloss
107         }
108     }
109     return(list(x=x,b=b,ahat=ahat,loss=nloss))
110 }
111
112 arrHadamard<-function(c,fun=function(x,y) x*y) {
113     nmat<-length(c)
114     if (nmat == 0) stop("empty argument in arrHadamard")
115     res<-c[[1]]
116     if (nmat == 1) return(res)

```

```

114     for (i in 2:nmat) res<-fun(res,c[[i]])
115     return(res)
116   }
117
118   arrOuter<-function(x,fun="*") {
119     nmat<-length(x)
120     if (nmat == 0) stop("empty argument in arrOuter")
121     res<-x[[1]]
122     if (length(x) == 1) return(res)
123     for (i in 2:nmat) res<-outer(res,x[[i]],fun)
124     return(res)
125   }
126
127   arrKronecker<-function(x,fun="*") {
128     nmat<-length(x)
129     if (nmat == 0) stop("empty argument in arrKronecker")
130     res<-x[[1]]
131     if (length(x) == 1) return(res)
132     for (i in 2:nmat) res<-kronecker(res,x[[i]],fun)
133     return(res)
134   }
135
136   flatten<-function(a,k) {
137     nard<-rev(1:(length(dim(a))-1))
138     apply(a,k,function(z) as.vector(aperm(z,nard)))
139   }
140
141   procrustus<-function(x) {
142     res<-svd(x)
143     return(tcrossprod(res$u,res$v))
144   }

```

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://gifi.stat.ucla.edu>