Generalizzazione dell'algoritmo di rotazione Chisquaremax di Knüsel

written by Silvia Testa | 21 Ottobre 2017

Nel 2008 Leo Knüsel ha introdotto un nuovo metodo con cui affrontare il problema della rotazione nell'analisi fattoriale, problema che consiste nel trovare una matrice di rotazione T da applicare ad una matrice iniziale di loading (A) al fine di produrre una matrice di loading ruotata (L= AT) in cui le relazioni tra variabili manifeste e fattori latenti siano più nitide e rendano così più agevole l'interpretazione dei fattori latenti. Con il metodo Chisquaremax, la matrice dei quadrati dei loading viene considerata alla stregua di una tabella di contingenza e la matrice di rotazione T viene scelta in modo da rendere massima l'associazione tra le righe (variabili manifeste) e le colonne (fattori latenti). Il metodo può essere applicato a rotazioni sia ortogonali sia oblique, qui viene considerato soltanto il caso di rotazioni ortogonali.

Chisquaremax possiede due caratteristiche in particolare che lo rendono una interessante alternativa ai numerosi criteri disponibili in letteratura:

- tratta in modo simmetrico le righe e le colonne, a differenza di altri metodi che tendono a semplificare unicamente il profilo delle righe (metodo Quartimax) o ad assegnare comunque un peso maggiore alla semplicità delle righe rispetto a quella delle colonne (come in generale i metodi della famiglia Orthomax);
- non tende a produrre un fattore generale, cioè una colonna su cui saturano molte variabili manifeste, aspetto che caratterizza il metodo Quartimax e neppure a produrre colonne di loadings con la stessa somma dei

quadrati, aspetto che caratterizza il metodo Varimax.

La versione dell'algoritmo resa disponibile dall'autore nel lavoro del 2008, tuttavia, non prevede la possibilità di operare la normalizzazione di Kaiser e prevede che la matrice di rotazione iniziale sia unicamente una matrice identità. In questo contributo viene proposta una versione "arricchita", più generale, che consente:

1. di applicare la normalizzazione di Kaiser
2. di usare una matrice di rotazione iniziale random o di ottenere la miglior soluzione all'interno di un certo numero di soluzioni prodotte variando la matrice random di rotazione iniziale.

In appendice viene riportato un esempio di applicazione della funzione "arricchita", gen.chi2max, e lo script in R.

Riferimenti bibliografici

Harman, H. H. (1976). Modern factor analysis. University of Chicago Press.

Knüsel, L. (2008a). Factor analysis: Chisquare as rotation criterion. Technical Report Number 040, Department of Statistics, University of Munich.

Knüsel, L. (2008b). Chisquare as a rotation criterion in factor analysis. Computational Statistics & Data Analysis, 52(9), 4243-4252.

APPENDICE

Esempio di applicazione
La matrice non ruotata A è tratta da Harman (1967, p.186).
Le soluzioni con e senza normalizzazione di Kaiser sono state prodotte con:
gen.chi2max(Mat, b=1000, c=1e-09, d=0, e=50, s=127, g=1)

gen.chi2max(Mat, b=1000, c=1e-09, d=1, e=50, s=127, g=1)

in cui:

Mat = la matrice A, non ruotata
b = numero di iterazioni
c = criterio di convergenza
d = normalizzazione di Kaiser (1=attiva; 0= non attiva)
e = numero di matrici random di rotazione iniziale (e= -1 una sola matrice random; 0= matrice identità; >0 più di una matrice random iniziale)
s = seme per la generazione delle matrici random
g = parametro gamma (vedi Knüsel, 2008a)

Come si osserva dalla tabella, il valore della funzione obiettivo (FOB) cresce passando dalla matrice non ruotata a quelle ruotate, inoltre la soluzione con normalizzazione di Kaiser è leggermente migliore rispetto a quella ottenuta senza la normalizzazione.

	Matrice non ruotata (A)			senza normalizzazione di Kaiser			con normalizzazione di Kaiser		
	F1	F2	F3	F1	F2	F3	F1	F2	F3
1	0.607	-0.060	-0.443	-0.233	0.684	0.214	-0.702	0.214	0.174
2	0.355	0.038	-0.266	-0.173	0.405	0.062	-0.418	0.062	0.139
3	0.418	0.148	-0.429	-0.214	0.577	-0.045	-0.592	-0.045	0.166
4	0.478	0.083	-0.287	-0.280	0.483	0.077	-0.505	0.078	0.238
5	0.729	0.257	0.244	-0.774	0.142	0.196	-0.205	0.204	0.757
6	0.707	0.354	0.167	-0.779	0.197	0.083	-0.260	0.091	0.760
7	0.721	0.367	0.257	-0.833	0.125	0.104	-0.193	0.112	0.819
8	0.705	0.197	0.062	-0.650	0.289	0.185	-0.342	0.191	0.622
9	0.698	0.409	0.252	-0.837	0.118	0.056	-0.186	0.065	0.824
10	0.455	-0.482	0.399	-0.232	-0.119	0.728	0.097	0.731	0.232
11	0.537	-0.390	0.145	-0.239	0.141	0.620	-0.162	0.622	0.220
12	0.487	-0.553	0.033	-0.068	0.216	0.702	-0.224	0.702	0.042
13	0.674	-0.368	-0.135	-0.238	0.452	0.589	-0.472	0.589	0.194

FOB	1.296	2.242	2.258

Con altre matrici tratte da esempi reali o generate casualmente l’applicazione della normalizzazione di Kaiser non sempre produce un miglioramento della FOB (talvolta produce i

medesimi risultati, altre volte un peggioramento della FOB), anche l'utilità di ricorrere a più matrici random di rotazione iniziale dipende dalla matrice di loading inziale.

Script in R

N.B.
La funzione di massimizzazione del Chi quadrato è tratta da Knüsel (2008b).
E' richiesto il package di R: GPArotation

```
chi2max.0 = function(x,TT,gamma, bb,cc,dd)
{
x=as.matrix(x)
nc=ncol(x)
 if(missing(bb)) bb=1000
 if(missing(cc)) cc=1e-9
 if(missing(dd)) dd=0
 if(missing(TT)) TT=diag(nc)
 if(missing(gamma)) gamma=1
eps=cc
if (nc<2)
 return(x)
p=nrow(x)
k=ncol(x)
c=0
d=0
z=x%*% TT
 if (dd==0){
H=drop(z^2%*%rep(1,k))
 # ss rows (comunalità)
 for (i in 1:bb) {
 D=drop(rep(1,p)%*% z^2 )
 E=drop((1/H) %*% z^4)
 C=diag(1/H) %*% z^3 %*% diag(1/D) - (1/2) * z %*% diag(E) %*%
diag(1/D^2)
 B=t(x) %*% C
 sB=La.svd(B)
 TT= sB$u %*% sB$vt
 dpast=d
 d=sum(sB$d)
```

```
 zpast=z
 z=x%*% TT
 D=drop(rep(1,p) %*% z^2 )
 cpast=c
 c=sum( diag(1/H) %*% z^4 %*% diag(1/D) )
 if (abs(d-dpast)/d < eps && abs(c-cpast) /c < eps && abs(d-
c/2)/d < eps)
 break
 z=gamma*z + (1-gamma)*zpast
 }
 }
 if (dd==1){
 uM=t(rep(1,p))
 uH=rep(1,k)
 # comunalità
 c.st=(x^2%*%uH)
 C.st=c.st%*%uH
 xk=x/(C.st^0.5)
 z=xk%*% TT
 H=drop(z^2%*%rep(1,k))
 for (i in 1:bb) {
 D=drop(rep(1,p)%*% z^2 )
 E=drop((1/H) %*% z^4)
 C=diag(1/H) %*% z^3 %*% diag(1/D) - (1/2) * z %*% diag(E) %*%
diag(1/D^2)
 B=t(xk) %*% C
 sB=La.svd(B)
 TT= sB$u %*% sB$vt
 dpast=d
 d=sum(sB$d)
 zpast=z
 z=xk%*% TT
 D=drop(rep(1,p) %*% z^2 )
 cpast=c
 c=sum( diag(1/H) %*% z^4 %*% diag(1/D) )
 if (abs(d-dpast)/d < eps && abs(c-cpast) /c < eps && abs(d-
c/2)/d < eps)
 break
 z=gamma*z + (1-gamma)*zpast
 }
 z=z*(C.st^0.5)
```

```
 }
list(iterations  =  i,Table=matrix(c(c,d),1,2),  Th=TT,
loadings=z, SS_columns=rep(1,p)%*%z^2)
}
gen.chi2max=function(X,b,c,d,e,s,g) {
X=as.matrix(X)
if(missing(b)) b=1000
if(missing(c)) c=1e-9
if(missing(d)) d=0
if(missing(e)) e=50
if(missing(s)) s=127
if(missing(g)) g=1
kais=list(FALSE,TRUE)
dd=d+1
P=nrow(X)
M=ncol(X)
M.it=diag(ncol(X))
if (e==-1){
set.seed(s)
M.it=Random.Start(M)
}
out0=chi2max.0(X,TT= M.it, gamma=g, bb=b,cc=c,dd=d)
 Qr=out0$Table[nrow(out0$Table),1]
 Mr=M.it
 Zr=out0$loadings
 Tr=out0$Th
if (e>0){
 set.seed(s)
 n.rand=e
 for (ii in 1:n.rand){
 M.it=Random.Start(M)
 out=chi2max.0(X,TT= M.it, gamma=g, bb=b,cc=c,dd=d)
 Qpast=Qr
 Mpast=Mr
 Zpast=Zr
 Tpast=Tr
 Qr=out$Table[nrow(out$Table),1]
 Mr=M.it
 Zr=out$loadings
 Tr=out$Th
 if (Qr<Qpast) {
```

```
 Qr=Qpast
 Mr=Mpast
 Zr=Zpast
 Tr=Tpast
 }
 }
}
list(fob=Qr,                                    rotmat=Tr,
loadings=Zr,SS_columns=rep(1,P)%*%Zr^2)
}
```