Package 'PGICA'

February 19, 2015

Type Package

Title Parallel Group ICA Algorithm

Version 1.0

Date 2014-11-12

Author Shaojie Chen; Lei Huang; Huitong Qiu; Ani Eloyan; Brian Caffo; Ciprian Crainiceanu

Maintainer Shaojie Chen <schen89@jhu.edu>

Description A Group ICA Algorithm that can run in parallel on an SGE platform or multi-core PCs

License GPL-2

Imports parallel, fastICA

NeedsCompilation no

Repository CRAN

Date/Publication 2014-11-19 17:33:48

R topics documented:

PGICA-package 1
mica 3
PC 7
StVal 8

Index **10**

`PGICA-package` *Parallel Group ICA Algorithm*

Description

This package implements a Group ICA Alorithm which can run in parallel on clusters and multi-core personal computers. Unlike existing ICA algorithms, this parallel algorithm can analyze very big data. It can be used in many applications including signal processing and neuroimage data analysis.

Details

Package:	PGICA
Type:	Package
Version:	1.0
Date:	2014-11-12
License:	GPL-2

The two main functions in this package are StVal and mica. mica is the main ICA function and StVal calculates initial values for mica.

Author(s)

Ani Eloyan, Shaojie Chen, Lei Huang, Huitong Qiu

Maintainer: <schen89@jhu.edu> Shaojie Chen

References

Ani Eloyan, Ciprian M. Crainiceanu and Brian S. Caffo: Likelihood Based Population Independent Component Analysis

---

`mica` *This is the main function for Parallel ICA algorithm*

Description

This function is the implementation of a parallel likelihood-based ICA algorithm. It can run in parallel on clusters and multi-core personal computers.

Usage

```
mica(W0=0,n.b=1000,maxit=200,maxN=75,l.b=-10,u.b=10,N0=19,epsilon=10^(-4),
hc=0,alpha=1,ind=500,nproc=10)
```

Arguments

`W0`	W0 is the initial value for mixing matrix W. It is calculated from function StVal().
`n.b`	n.b is the number of bins when estimating densities of underlying independent components.
`maxit`	maxit is the maximum number of iterations for the algorithm.
`maxN`	maxN is the maximum number of Gaussian distributions used to estimate density.
`l.b`	l.b is a lower bound of the interval on which we will estimate density.
`u.b`	u.b is an upper bound of the interval on which we will estimate density.
`N0`	N0 is the starting number of Gaussian distributions for density estimation.

epsilon	epsilon is the tolerance for the difference of estimations between 2 successive iterations.
hc	hc is not used here.
alpha	alpha is the step size for Newton algorithm.
ind	ind is a tuning parameter to speed up big matrix multiplication.
nproc	nproc is the number of parallel processes to use.

Details

The Independent Component Analysis (ICA) model can be written as: X=AS, where X is T by V matrix, A is T by T square matrix and S is T by V matrix. We assume that the rows of S are independent. Denote the inverse of A as W, then with W and X we can calculate S. The goal of our parallel ICA algorithm is to calculate A (thus S) from X with independence assumption. In neuroimaging analysis, The rows of S can be intepreted as functional networks.

Value

mica returns a list of estimations. The first component of the list is the value of W.

Author(s)

Ani Eloyan, Shaojie Chen, Lei Huang and Huitong Qiu et.al.

References

Ani Eloyan, Ciprian M.Crainiceanu and Brian S.Caffo: Likelihood Based Population Independent Component Analysis

Examples

```
######################  Generating simulation data
n=2000
m=2
N.s=3

A.true1=matrix(c(0.75,0.25,0.5,-0.5), 2, 2, byrow=TRUE)
A.true2=matrix(c(1,0,0.5,-0.5), 2, 2, byrow=TRUE)
A.true3=matrix(c(1,0.5,0.75,1), 2, 2, byrow=TRUE)

W.true1=solve(A.true1)
W.true2=solve(A.true2)
W.true3=solve(A.true3)

S1=rgamma(n,shape=4,scale=0.25)
S2=rgamma(n,shape=2,scale=2)
S.true=cbind(S1,S2)
tr=PGICA:::trans(S.true,W.true1)
S.true=tr[[1]]
```

```
X.true1=S.true%*%A.true1
X.true2=S.true%*%A.true2
X.true3=S.true%*%A.true3

require(fastICA)
f1=fastICA(X.true1,n.comp=m)
f2=fastICA(X.true2,n.comp=m)
f3=fastICA(X.true3,n.comp=m)

X=c(list(t(f1$X%*%f1$K)),list(t(f2$X%*%f2$K)),list(t(f3$X%*%f3$K)))
K=c(list(f1$K),list(f2$K),list(f3$K))

X.full=c()
for(i in 1:N.s){
X.full=cbind(X.full,t(X[[i]]))
}

f=fastICA(X.full,n.comp=m)
tr=PGICA:::trans(f$S,f$W)
S.f=tr[[1]]
W1=solve(f$A[,1:m])
W2=solve(f$A[,(m+1):(2*m)])
W3=solve(f$A[,(2*m+1):(3*m)])

XtX=t(X.full)%*%X.full
sv=svd(XtX)
Sigma=diag(sqrt(sv$d))
SigmaInv=diag(1/sqrt(sv$d))
U=sv$u
V=X.full%*%U%*%SigmaInv

V.l=V[,1:m]
Sigma.l=Sigma[1:m,1:m]
U.l=t(U)[1:m,]
X.app=V.l%*%Sigma.l%*%U.l

X.a=c()
W0=c()
for(i in 1:N.s){
X.a[[i]]=t(X.app[,((i-1)*m+1):(i*m)])
W0=c(W0,list(t(solve(Sigma[1:m,1:m]%*%U.l[,((i-1)*m+1):(i*m)]))))
}

f=fastICA(X.full,n.comp=m)
S.f=f$S

dir.create('./Sim')
setwd("./Sim")

for(i in 1:N.s){
X=X.a[[i]]
save(X,file=paste(paste("app",i,sep=""),".rda",sep=""))
}
```

```
save(W0,A.true1,A.true2,A.true3,K,W1,W2,W3,file="W0.rda")
save(S.f,S.true,file="Sfiles.rda")
setwd("..")

####################  Use PGICA to analyze above generated data
maxit=100
maxN=50
N0=19
epsilon=10^-3
W0=0
hc=0
u.b=10
l.b=-10
alpha=0.5
require(fastICA)

m=2
n.b=1000
N.s=3
n=2000
ind=500
setwd("./Sim")
fileDir=getwd()
files = dir(fileDir, pattern = "*", full.names = TRUE)
files=files[c(3:5,1,2)]

load("./W0.rda")
load(files[1])
f=fastICA(t(X),n.comp=m)
tr=PGICA:::trans(f$S,t(f$A))
S.f=t(tr[[1]])
A.f=t(tr[[2]])

for(subj in 2:N.s){
W.temp=solve(W0[[1]]%*%t(A.f))%*%W0[[subj]]
W0[[subj]]=W.temp
}
W0[[1]]=solve(t(A.f))
require(parallel)
res=mica(W0,n.b,maxit,maxN,l.b,u.b,N0,epsilon,hc,alpha,ind,nproc=2)
tmp=((solve(res[[1]][[1]]))%*%S.f)
#hist(tmp[1,])
#hist(tmp[2,])
#hist(S.true[,1])
#hist(S.true[,2])


#This real data example is time-consuming
#m=20;
#N.s=1;
#alpha=0.5;
#setwd("..")
```

```
#dir.create('./data')
#data(PC,package="PGICA")
#save(PC,file="./data/sample.rda")
#StVal("./data/",m=20,N.s=1,V=30000)
#fileDir=getwd()
#files = dir(fileDir, pattern = "*.rda", full.names = TRUE)
#nfiles = length(files)
#outfile="m20.rda"
#setwd(fileDir)
#load("W0.rda")
#n=30000

#maxit=80
#maxN=50
#N0=19
#ind=100
#epsilon=10^-3
#hc=0
#u.b=10
#l.b=-10
#n.b=1000
#files=files[c(2,1)]
#res=mica(W0,n.b,maxit,maxN,l.b,u.b,N0,epsilon,hc,alpha)
```

PC *A sample fMRI data in .rda format.*

Description

This is a subsample of real fMRI data from the 1000 Connectome Project.

Usage

```
data(PC)
```

Format

PC is a 30000 by 20 matrix, where 30000 is the number of voxels in brain.

Details

This is not the original data, but the data after a dimension reduction(from 67749 by 100 to 67749 by 20) using PCA. Then a sample of 30000 voxels are picked out of 67749. The sample is achieved by setting the seed to be 0.

Source

http://www.nitrc.org/projects/fcon_1000/

References

Ani Eloyan, Ciprian M.Crainiceanu and Brian S. Caffo: Likelihood Based Population Independent Component Analysis

Examples

```
data(PC)
```

`StVal` *Computing Starting Values for PGICA*

Description

The main function uses a iterative algorithm. This function calculates the starting values for the algorithm.

Usage

```
StVal(fileDir, m = 20, N.s = 150, V = 67749)
```

Arguments

`fileDir`	fileDir is a file directory containing the original .rda files.
`m`	m is the number of independent components you want.
`N.s`	N.s is the number of subjects.
`V`	V is the number of observed signals. In the fMRI case, it is the number of voxels.

Details

We used the PVD algorithm by C.M.Crainiceanu et.al to compute starting values for our algorithm.

Value

This function has no returned values, instead it will write the output into .rda files. The W0.rda file contains starting values for W matrix and the appi.rda files are the starting values for S.

Author(s)

Ani Eloyan, Shaojie Chen, Lei Huang, Huitong Qiu

References

Ciprian M.Crainiceanu, Brian S.Caffo, Sheng Luo, Vadim Zipunikov Population Value Decomposition, A Framework for the Analysis of Image Populations http://biostats.bepress.com/jhubiostat/paper220/

Examples

```
dir.create('./data')
data(PC,package="PGICA")
save(PC,file="./data/sample.rda")
StVal("./data/",m=20,N.s=1,V=30000)
# You will find files W0.rda and app1.rda under your current working directory
```

Index

∗Topic **ICA**
mica, 3
PGICA-package, 1
∗Topic **Parallel Computing**
PGICA-package, 1
∗Topic **Starting Value**
StVal, 8
∗Topic **datasets**
PC, 7
∗Topic **parallel computing**
mica, 3

mica, 3

PC, 7
PGICA *(*PGICA-package*)*, 1
PGICA-package, 1

StVal, 8