Package ‘rifle’

May 9, 2026

Type Package

Title Sparse Generalized Eigenvalue Problem

Version 1.0

Date 2018-08-27

Author Kean Ming Tan

Maintainer Kean Ming Tan <ktan@umn.edu>

Depends MASS

Description Implements the algorithms for solving sparse generalized eigenvalue problem by Tan, et. al. (2018). Sparse Generalized Eigenvalue Problem: Optimal Statistical Rates via Truncated Rayleigh Flow. To appear in Journal of the Royal Statistical Society: Series B. <doi:10.48550/arXiv.1604.08697>.

NeedsCompilation no

Repository CRAN

Date/Publication 2018-08-31 20:00:03 UTC

Contents

`rifle-package` *Sparse Generalized Eigenvalue Problem*

Description

This package is called rifle. It implements algorithms for solving sparse generalized eigenvalue problem. The algorithms are described in the paper "Sparse Generalized Eigenvalue Problem: Optimal Statistical Rates via Truncated Rayleigh Flow", by Tan et al. (2018).

The main functions are as follows: (1) initial.convex (2) rifle

The first function, initial.convex, solves the sparse generalized eigenvalue problem using a convex relaxation. The second function, rifle, refines the initial estimates from initial.convex and gives a more accurate estimator of the leading generalized eigenvector.

Details

The package includes the following functions:

`initial.convex`:	Solve a convex relaxation of the sparse GEP
`rifle`:	Perform truncated rayleigh method to obtain the largest generalized eigenvector

Author(s)

Kean Ming Tan

Maintainer: Kean Ming Tan

References

Sparse Generalized Eigenvalue Problewm: Optimal Statistical Rates via Truncated Rayleigh Flow",
by Tan et al. (2018). To appear in Journal of the Royal Statistical Society: Series B. https://arxiv.org/pdf/1604.08697.pdf.

See Also

`initial.convex` `rifle`

Examples

```
# Example on Fisher's Discriminant Analysis on two class classification
# A small toy example
n <- 50
p <- 25

# Generate block diagonal covariance matrix with 5 blocks
Sigma <- matrix(0,p,p)
for(i in 1:p){
Sigma[i,] <- 1:(p)-i
}
Sigma <- 0.7^abs(Sigma)

# Generate mean vector for two classes
mu1 <- rep(0,p)
mu2 <- c(rep(c(0,1),5),rep(0,p-10))
```

```
# Generate data for two classes
X <- rbind(mvrnorm(n=n/2,mu1,Sigma),mvrnorm(n=n/2,mu2,Sigma))
y <- rep(1:2,each=n/2)

# Estimate the subspace spanned by the largest eigenvector using convex relaxation
# Estimates
 estmu1 <- apply(X[y==1,],2,mean)
 estmu2 <- apply(X[y==2,],2,mean)
 estwithin <- cov(X[y==1,])+cov(X[y==2,])
 estbetween <- outer(estmu1,estmu1)+outer(estmu2,estmu2)

# Running initialization using convex relaxation
 a <- initial.convex(A=estbetween,B=estwithin,lambda=2*sqrt(log(p)/n),K=1,nu=1,trace=FALSE)

# Use rifle to improve the leading generalized eigenvector
 init <- eigen(a$Pi+t(a$Pi))$vectors[,1]

# Pick k such that the generalized eigenvector is sparse
 k <- 10
#  Rifle 1
 final.estimator <- rifle(estbetween,estwithin,init,k,0.01,1e-3)

# True direction in this simulation setting
# truebetween <- mu1 %*% t(mu1)+ mu2 %*% t(mu2)
# truewithin <- Sigma+Sigma
# temp <- eigen(truewithin)
# sqrtwithin <- temp$vectors %*% diag(sqrt(temp$values)) %*% t(temp$vectors)

# vecres <-svd(solve(sqrtwithin)%*% truebetween%*% solve(sqrtwithin))$v[,1]

# oracledirection <- solve(sqrtwithin) %*% vecres

# oracledirection <- oracledirection/sqrt(sum(oracledirection^2))

# Comparing estimated vs true direction by computing the cosine angle
# 1-sum(abs(oracledirection*final.estimator))
```

`initial.convex` *Convex Relaxation for Sparse GEP*

Description

Estimate the K-dimensional subspace spanned by the largest K generalized eigenvector by solving a convex relaxation. The details are given in Tan et al. (2018).

Usage

```
initial.convex(A, B, lambda, K, nu = 1, epsilon = 0.005, maxiter = 1000, trace = FALSE)
```

Arguments

`A`	Input the matrix A for sparse generalized eigenvalue problem.
`B`	Input the matrix B for sparse generalized eigenvalue problem.
`lambda`	A positive tuning parameter that constraints the solution to be sparse
`K`	A positive integer tuning parameter that constraints the solution to be low rank.
`nu`	An ADMM tuning parameter that controls the convergence of the ADMM algorithm.
`epsilon`	Threshold for convergence. Default value is 0.005.
`maxiter`	Maximum number of iterations. Default is 1000 iterations.
`trace`	Default value of trace=FALSE. If trace=TRUE, each iteration of the ADMM algorithm is printed.

Value

`Pi`	Estimated subspace Pi

Author(s)

Kean Ming Tan

References

Sparse Generalized Eigenvalue Problewm: Optimal Statistical Rates via Truncated Rayleigh Flow",
by Tan et al. (2018). To appear in Journal of the Royal Statistical Society: Series B. https://arxiv.org/pdf/1604.08697.pdf.

`rifle` *Rifle - Truncated Rayleigh Flow Method*

Description

Estimate the largest sparse generalized eigenvector using truncated rayleigh flow method. The details are given in Tan et al. (2018).

Usage

```
rifle(A, B, init, k, eta = 0.01, convergence = 0.001, maxiter = 5000)
```

Arguments

`A`	Input the matrix A for sparse generalized eigenvalue problem.
`B`	Input the matrix B for sparse generalized eigenvalue problem.
`init`	Input an initial vector for the largest generalized eigenvector. This value can be obtained by taking the largest eigenvector of the results from initial.convex function.

`k`	A positive integer tuning parameter that controls the number of non-zero elements in the estimated leading generalized eigenvector.
`eta`	A tuning parameter that controls the convergence of the algorithm. Default value is 0.01. Theoretical results suggest that this value should be set such that eta*(largest eigenvalues of B) < 1.
`convergence`	Threshold for convergence. Default value is 0.001.
`maxiter`	Maximum number of iterations. Default is 5000 iterations.

Value

`xprime`	xprime is the estimated largest generalized eigenvector.

Author(s)

Kean Ming Tan

References

Sparse Generalized Eigenvalue Problewm: Optimal Statistical Rates via Truncated Rayleigh Flow",
by Tan et al. (2018). To appear in Journal of the Royal Statistical Society: Series B. https://arxiv.org/pdf/1604.08697.pdf.

Index