Package ‘mvlearnR’

July 28, 2024

Type Package

Title Multiview Learning Methods in R

Version 0.0.9

Author Sandra E. Safo and Elise F. Palzer

Maintainer Sandra E. Safo <ssafo@umn.edu>

URL https://www.sandrasafo.com/software; https://multi-viewlearn.shinyapps.io/MultiView_Modeling/

Description The mvlearnR package and accompanying Shiny App is intended for integrating data from multiple sources
(e.g. genomics, proteomics, metabolomics). It is a compilation of various Multiview learning methods
including SIDA and SIDANet (Sparse Integrative Discriminant Analysis for Multiview Structured Data), SELPCCA (Sparse Estimation via Linear Programming for canonical correlation analysis [CCA]), and SELP-Predict.
The SIDA and SIDANet algorithms are for joint association and classification studies.
The algorithms consider the overall association between multiview data, and the separation within each view when choosing discriminant vectors
that are associated and optimally separate subjects. SIDANet incorporates prior structural information in joint association and classification studies.
It uses the normalized Laplacian of a graph to smooth coefficients of predictor variables, thus encouraging selection of predictors that are connected and
behave similarly. The SELPCCA method is an unsupervised method for associating two high dimensional data types.
The algorithm obtains linear combinations of subsets of variables for each data type that contribute to overall dependency structure between the data types. SELP-Predict is a two-stage method for associating two views and predicting
binary, continuous, poisson, or time-to-event data. Additional plotting and filtering functions such as variable importance plots,
volcano plots, discriminant and correlation plots, relevance network and biplots are also included.

License GPL (>= 3)

Encoding UTF-8

LazyData true

LazyDataCompression xz

RoxygenNote 7.3.1

Imports stats, graphics, doParallel, parallel, parallelly,
CVXR, foreach, igraph,

RSpectra, Matrix, dplyr, ggplot2, umap,ggthemes,methods, ROCit,
gridExtra, pscl, survival

Suggests knitr,
rmarkdown,
testthat (>= 3.0.0)

Depends R (>= 3.5.0)

VignetteBuilder knitr

NeedsCompilation no

Config/testthat/edition 3

R topics documented:

`BetweenViewBiplot`	*Biplots for Discriminant Scores or Canonical Correlation Variates between pairs of views*

Description

Biplots to visualize discriminant scores/ canonical variates between pairs of views. It shows how selected variables from the first and second discriminant (for SIDA and SIDANet) or canonical correlation (for SELPCCA) vectors in a view is related to selected variables in another view. Variables farther from the origin and close to first or second axis have higher impact on first or second discriminant/canonical vectors, respectively. Variables farther from the origin and between both first and second axes have similar higher contributions to the first and second discriminant/canonical correlation vectors. In both situations, for SIDA and SIDANet, this suggests that these variables contribute more to the separation of classes and association of views. For SELPCCA, this suggests that these variables contribute more to the association between the two views. This plot can only be generated for classification and association problems with 3 or more classes (SIDA and SIDANet), or for CCA problems with two or more canonical correlation vectors requested (i.e. ncancorr > 1 for SELPCCA). This plot shows the scores and loadings from pairs of views together. The scores are the sum of scores for each view. Solid and dashed lines represent vectors for Views 1 and 2, respectively.

Usage

```
BetweenViewBiplot(
  fit,
  Y,
  Xtest = NULL,
  color.palette = NULL,
  keep.loadings = NULL,
  plotIt = TRUE
)
```

Arguments

`fit`	the output from SIDA, SIDANet, and SELPCCA methods
`Y`	a vector of class membership for grouping canonical correlatoin variates and discriminant scores.
`Xtest`	list of D entries containing test data. If not null, scores for biplots will be constructed for testing data.
`color.palette`	character vector of length K (number of classes), specifying the colors to use for the classes, respectively. Defaults to shades of blue and orange (color.BlueOrange). Other option includes red and green combinations (color.GreenRed)
`keep.loadings`	numeric vector of length D (number of views), specifying how many variables to represent on loadings plot for each view. This is useful in situations where the number of variables selected is large, and could clutter the plot. If this number is more than the variables selected, it will be set to the maximum number of variables selected for each view. Default is plotting all selected variables.
`plotIt`	boolean, whether to print the result or just return it. default is TRUE.

Details

The function will return loading plots, one for each view.

Value

NULL

References

Sandra E. Safo, Eun Jeong Min, and Lillian Haine (2023), Sparse Linear Discriminant Analysis for Multi-view Structured Data, Biometrics. Sandra E. Safo, Jeongyoun Ahn, Yongho Jeon, and Sungkyu Jung (2018), Sparse Generalized Eigenvalue Problem with Application to Canonical Correlation Analysis for Integrative Analysis of Methylation and Gene Expression Data. Biometrics

See Also

cvSIDA DiscriminantPlots CorrelationPlots

Examples

```
## Not run:
data("sidanetData")
Xdata <- sidanetData[[1]]
Y <- sidanetData[[2]] #class membership already coded as 1,2,...
Xtestdata <- sidanetData[[3]]
Ytest <- sidanetData[[4]] #class membership already coded as 1,2,...
#edge information
myedges=sidanetData[[5]]
myedgeweight=sidanetData[[6]]
##---- call cross validation
mycvsidanet=cvSIDANet(Xdata,Y,myedges,myedgeweight,withCov=FALSE,plotIt=FALSE,Xtestdata=Xtestdata,
                     Ytest=Ytest, isParallel = FALSE)
BetweenViewBiplot(mycvsidanet, Y,keep.loadings=c(3,3) )

## End(Not run)
```

CorrelationPlots *Correlation Plots*

Description

Plots for visualizing correlation between estimated discriminant vectors for pairwise data.

Usage

```
CorrelationPlots(
  Xtestdata = Xtestdata,
  Ytest = Ytest,
  hatalpha = hatalpha,
  method.used = "SIDA",
  color.palette = NULL,
  plotIt = TRUE
)
```

Arguments

`Xtestdata`	A list with each entry containing views of size $ntest \times p_d$, where $d = 1, \ldots, D$.Rows are samples and columns are variables. Can use testing or training data
`Ytest`	$ntest \times 1$ vector of class membership.
`hatalpha`	A list of estimated sparse discriminant vectors for each view.
`method.used`	A character specifying the integration method used. These are used for appropriate labeling. Options are "SIDA" and "SELPCCA". Default is "SIDA"
`color.palette`	character vector of length K (number of classes), specifying the colors to use for the classes, respectively. Defaults to shades of blue and orange (color.BlueOrange). Other option includes red and green combinations (color.GreenRed)
`plotIt`	boolean; if TRUE, plots the plots. If FALSE, only returns the plots. This is useful to customize the ggplots.

Details

The function will return either a single ggplot (n views == 2), or a list of ggplots (n views > 2).

Value

`ggplot`

References

Sandra E. Safo, Eun Jeong Min, and Lillian Haine (2022), Sparse Linear Discriminant Analysis for Multi-view Structured Data, Biometrics.

See Also

`cvSIDA` `sidatunerange` `DiscriminantPlots`

Examples

```
## Not run:
 #call sida
data(sidaData)
##---- call sida algorithm to estimate discriminant vectors, and predict on testing data

Xdata=sidaData[[1]]
Y=sidaData[[2]]
Xtestdata=sidaData[[3]]
Ytest=sidaData[[4]]

#call sidatunerange to get range of tuning parameter
ngrid=10
mytunerange=sidatunerange(Xdata,Y,ngrid,standardize=TRUE,weight=0.5,withCov=FALSE)

# an example with Tau set as the lower bound
Tau=c(mytunerange$Tauvec[[1]][1], mytunerange$Tauvec[[2]][1])
mysida=sida(Xdata,Y,Tau,withCov=FALSE,Xtestdata=Xtestdata,Ytest=Ytest,AssignClassMethod='Joint',
            plotIt=FALSE, standardize=TRUE,maxiteration=20,weight=0.5,thresh= 1e-03)

test.error=mysida$sidaerror
test.correlation=mysida$sidacorrelation
```

```
#estimated discriminant vectors and predicted class
hatalpha=mysida$hatalpha

predictedClass=mysida$PredictedClass

##----plot discriminant and correlation plots
#---------Correlation plot
CorrelationPlots(Xtestdata,Ytest,mysida$hatalpha)

## End(Not run)
```

`COVIDData` *Multiomics data pertaining to COVID-19*

Description

RNA Sequencing (RNASeq) and Proteomics data pertaining to COVID-19. Clinical data are also available. Please refer to Overmyer et.al (2021) for a description of the data and Lipman et.al (2022) for how data were pre-processed.

Usage

```
COVIDData
```

Format

A list with 3 elements:

`Proteomic` Proteomics data. A data frame of size 120×264. Rows are samples and columns are variables.

`RNAseq` RNASeq data. A data frame of size 120×5800. Rows are samples and columns are variables.

`Clinical` Clinical and demographic data. A data frame of size 120×18. Rows are samples and columns are variables.

References

Multi-omic analysis reveals enriched pathways associated with COVID-19 and COVID-19 severity. PLOS ONE, 17(4) Overmyer, K.A., Shishkova, E., Miller, I.J., Balnis, J., Bernstein, M.N., Peters-Clarke, T.M., Meyer, J.G., Quan, Q., Muehlbauer, L.K., Trujillo, E.A., et al.: Large-scale multi-omic analysis of covid-19 severity. Cell systems 12(1), 23–40 (2021)

`cvselpscca` *Cross validation for Sparse Canonical Correlation Analysis*

Description

Performs n-fold cross validation to select optimal tuning parameters for SELPCCA based on training data. If you want to apply optimal tuning parameters to testing data, you may also use multiplescca.

Usage

```
cvselpscca(
  Xdata1 = Xdata1,
  Xdata2 = Xdata2,
  ncancorr = ncancorr,
  CovStructure = "Iden",
  isParallel = TRUE,
  ncores = NULL,
  nfolds = 5,
  ngrid = 10,
  standardize = TRUE,
  thresh = 1e-04,
  maxiteration = 20
)
```

Arguments

`Xdata1`	A matrix of size $n \times p$ for first dataset. Rows are samples and columns are variables.
`Xdata2`	A matrix of size $n \times q$ for second dataset. Rows are samples and columns are variables.
`ncancorr`	Number of canonical correlation vectors. Default is 1.
`CovStructure`	Covariance structure to use in estimating sparse canonical correlation vectors. Either "Iden" or "Ridge". Iden assumes the covariance matrix for each dataset is identity. Ridge uses the sample covariance for each dataset. See reference article for more details.
`isParallel`	TRUE or FALSE for parallel computing. Default is TRUE.
`ncores`	Number of cores to be used for parallel computing. Only used if isParallel=TRUE. If isParallel=TRUE and ncores=NULL, defaults to half the size of the number of system cores.
`nfolds`	Number of cross validation folds. Default is 5.
`ngrid`	Number of grid points for tuning parameters. Default is 10 for each dataset.
`standardize`	TRUE or FALSE. If TRUE, data will be normalized to have mean zero and variance one for each variable. Default is TRUE.
`thresh`	Threshold for convergence. Default is 0.0001.
`maxiteration`	Maximum iteration for the algorithm if not converged. Default is 20.

Details

The function will return several R objects, which can be assigned to a variable. To see the results, use the “$" operator.

Value

The output is a list containing the following components.

hatalpha	Estimated sparse canonical correlation vectors for first dataset.
hatbeta	Estimated sparse canonical correlation vectors for second dataset.
CovStructure	Covariance structure used in estimating sparse canonical correlation vectors. Either "Iden" or "Ridge".
optTau	Optimal tuning parameters for each dataset.
maxcorr	Estimated canonical correlation coefficient.
tunerange	Grid values for each dataset used for searching optimal tuning paramters.

References

Sandra E. Safo, Jeongyoun Ahn, Yongho Jeon, and Sungkyu Jung (2018), Sparse Generalized Eigenvalue Problem with Application to Canonical Correlation Analysis for Integrative Analysis of Methylation and Gene Expression Data. Biometrics

See Also

multiplescca

Examples

```
## Not run:
##---- read in data
data(selpData)

Xdata1=selpData[[1]]
Xdata2=selpData[[2]]

##---- call cross validation to estimate first canonical correlation vectors
ncancorr=1
mycv=cvselpscca(Xdata1=Xdata1,Xdata2=Xdata2,ncancorr=ncancorr,CovStructure="Iden",
                isParallel=TRUE,ncores=NULL,nfolds=5,ngrid=10,
                standardize=TRUE,thresh=0.0001,maxiteration=20)

#check output
train.correlation=mycv$maxcorr
optTau=mycv$optTau
hatalpha=mycv$hatalpha
hatbeta=mycv$hatbeta

#obtain correlation plot using training data
scoresX1=Xdata1%*% hatalpha
scoresX2=Xdata2%*% hatbeta
plot(scoresX1, scoresX2,lwd=3,
     xlab=paste(
       "First Canonical correlation variate for dataset", 1),
     ylab=paste("First Canonical correlation variate for dataset", 2),
    main=paste("Correlation plot for datasets",1, "and" ,2, ",", "\u03C1 =", mycv$maxcorr))

#obtain correlation plot using testing data
Xtestdata1=selpData[[3]]
Xtestdata2=selpData[[4]]
scoresX1=Xtestdata1%*%hatalpha
scoresX2=Xtestdata2%*%hatbeta
mytestcorr=round(abs(cor(Xtestdata1%*%hatalpha,Xtestdata2%*%hatbeta)),3)

plot(scoresX1, scoresX2,lwd=3,xlab=paste(
  "First Canonical correlation variate for dataset", 1),
  ylab=paste("First Canonical correlation variate for dataset", 2),
  main=paste("Correlation plot for datasets",1, "and" ,2, ",", "\u03C1 =", mytestcorr))

## End(Not run)
```

`cvSIDA` *Cross validation for Sparse Integrative Discriminant Analysis for Multi-View Data*

Description

Performs nfolds cross validation to select optimal tuning parameters for sida based on training data, which are then used with the training or testing data to predict class membership. Allows for inclusion of covariates which are not penalized. If you want to apply optimal tuning parameters to testing data, you may also use sida.

Usage

```
cvSIDA(
  Xdata = Xdata,
  Y = Y,
  withCov = FALSE,
  plotIt = FALSE,
  Xtestdata = NULL,
  Ytest = NULL,
  isParallel = TRUE,
  ncores = NULL,
  gridMethod = "RandomSearch",
  AssignClassMethod = "Joint",
  nfolds = 5,
  ngrid = 8,
  standardize = TRUE,
  maxiteration = 20,
  weight = 0.5,
  thresh = 0.001
)
```

Arguments

`Xdata`	A list with each entry containing training views of size $n \times p_d$, where $d = 1, \ldots, D$ views. Rows are samples and columns are variables. If covariates are available, they should be included as a separate view, and set as the last dataset.

	For binary or categorical covariates (assumes no ordering), we suggest the use of indicator variables.
`Y`	$n \times 1$ vector of class membership. Numeric, coded as 1, 2, ...
`withCov`	TRUE or FALSE if covariates are available. If TRUE, please set all covariates as one dataset and should be the last dataset. For binary and categorical variables, use indicator matrices/vectors. Default is FALSE.
`plotIt`	TRUE or FALSE. If TRUE, produces discriminants and correlation plots. Default is FALSE.
`Xtestdata`	A list with each entry containing testing views of size $ntest \times p_d$, where $d = 1, \ldots, D$. Rows are samples and columns are variables. The order of the list should be the same as the order for the training data, Xdata. Use if you want to predict on a testing dataset. If no Xtestdata, set to NULL.
`Ytest`	$ntest \times 1$ vector of test class membership. If no testing data provided, set to NULL.
`isParallel`	TRUE or FALSE for parallel computing. Default is TRUE
`ncores`	Number of cores to be used for parallel computing. Only used if isParallel=TRUE. If isParallel=TRUE and ncores=NULL, defaults to half the size of the number of system cores.
`gridMethod`	GridSearch or RandomSearch. Optimize tuning parameters over full grid or random grid. Default is RandomSearch.
`AssignClassMethod`	Classification method. Either Joint or Separate. Joint uses all discriminant vectors from D datasets to predict class membership. Separate predicts class membership separately for each dataset. Default is Joint.
`nfolds`	Number of cross validation folds. Default is 5.
`ngrid`	Number of grid points for tuning parameters. Default is 8 for each view if $D = 2$. If $D > 2$, default is 5.
`standardize`	TRUE or FALSE. If TRUE, data will be normalized to have mean zero and variance one for each variable. Default is TRUE.
`maxiteration`	Maximum iteration for the algorithm if not converged.Default is 20.
`weight`	Balances separation and association. Default is 0.5.
`thresh`	Threshold for convergence. Default is 0.001.

Details

The function will return several R objects, which can be assigned to a variable. To see the results, use the “$" operator.

Value

A list with the following components:

`sidaerror`	Estimated classification error. If testing data provided, this will be test classification error, otherwise, training error
`sidacorrelation`	Sum of pairwise RV coefficients. Normalized to be within 0 and 1, inclusive.
`hatalpha`	A list of estimated sparse discriminant vectors for each view.

`PredictedClass`	Predicted class. If AssignClassMethod='Separate', this will be a $ntest \times D$ matrix, with each column the predicted class for each data.
`PredictedClass.train`	Predicted class for train data. If AssignClassMethod='Separate', this will $ntrain \times D$ matrix, with each column the predicted class for each data.
`optTau`	Optimal tuning parameters for each view, not including covariates, if available.
`gridValues`	Grid values used for searching optimal tuning parameters.
`AssignClassMethod`	Classification method used. Joint or Separate.
`gridMethod`	Grid method used. Either GridSearch or RandomSearch

References

Sandra E. Safo, Eun Jeong Min, and Lillian Haine (2022) , Sparse Linear Discriminant Analysis for Multi-view Structured Data, Biometrics

See Also

`sida`

Examples

```
## Not run:
 #call sida
data(sidaData)
##---- call sida algorithm to estimate discriminant vectors, and predict on testing data

Xdata=sidaData[[1]]
Y=sidaData[[2]]
Xtestdata=sidaData[[3]]
Ytest=sidaData[[4]]

##---- call cross validation
mycv=cvSIDA(Xdata,Y,withCov=FALSE,plotIt=FALSE, Xtestdata=Xtestdata,Ytest=Ytest,
            isParallel=FALSE,gridMethod='RandomSearch',
            AssignClassMethod='Joint',nfolds=5,ngrid=8,standardize=TRUE,
            maxiteration=20, weight=0.5,thresh=1e-03)

#check output
test.error=mycv$sidaerror
test.correlation=mycv$sidacorrelation
optTau=mycv$optTau
hatalpha=mycv$hatalpha

#Obtain more performance metrics (applicable to two classes only)
 #train metrics
 Y.pred=mycv$PredictedClass.train-1 #to get this in 0 and 1
 Y.train=Y-1 #to get this in 0 and 1
 train.metrics=PerformanceMetrics(Y.pred,Y.train,family='binomial')

 print(train.metrics)
 #obtain predicted class
 Y.pred=mycv$PredictedClass-1 #to get this in 0 and 1
 Ytest.in=Ytest-1 #to get this in 0 and 1
 test.metrics=PerformanceMetrics(Y.pred,Ytest.in,family='binomial')
 print(test.metrics)

## End(Not run)
```

---

cvSIDANet *Cross validation for Sparse Integrative Discriminant Analysis for Multi-view Structured (Network) Data*

Description

Peforms nfolds cross validation to select optimal tuning parameters for sidanet based on training data, which are then used with the training or testing data to predict class membership. Allows for inclusion of covariates which are not penalized. If you want to apply optimal tuning parameters to testing data, you may also use sidanet.

Usage

```
cvSIDANet(
  Xdata = Xdata,
  Y = Y,
  myedges = myedges,
  myedgeweight = myedgeweight,
  withCov = FALSE,
  plotIt = FALSE,
  Xtestdata = NULL,
  Ytest = NULL,
  isParallel = TRUE,
  ncores = NULL,
  gridMethod = "RandomSearch",
  AssignClassMethod = "Joint",
  nfolds = 5,
  ngrid = 8,
  standardize = TRUE,
  maxiteration = 20,
  weight = 0.5,
  thresh = 0.001,
  eta = 0.5
)
```

Arguments

`Xdata`	A list with each entry containing training views of size $n \times p_d$, where $d = 1, \ldots, D$. Rows are samples and columns are variables. If covariates are available, they should be included as a separate view, and set as the last dataset. For binary or categorical covariates (assumes no ordering), we suggest the use of indicator variables.
`Y`	$n \times 1$ vector of class membership.
`myedges`	A list with each entry containing a $M_d \times 2$ matrix of edge information for each view. If a view has no edge information, set to 0; this will default to SIDA. If covariates are available as a view (Dth view), the edge information should be set to 0.

`myedgeweight`	A list with each entry containing a $M_d \times 1$ vector of weight information for each view. If a view has no weight information, set to 0; this will use the Laplacian of an unweighted graph. If covariates are available as a view (Dth view), the weight information should be set to 0.
`withCov`	TRUE or FALSE if covariates are available. If TRUE, please set all covariates as one dataset and should be the last dataset. For binary and categorical variables, use indicator matrices/vectors. Default is FALSE.
`plotIt`	TRUE or FALSE. If TRUE, produces discriminants and correlation plots. Default is FALSE.
`Xtestdata`	A list with each entry containing testing views of size $ntest \times p_d$, where $d = 1, \ldots, D$. Rows are samples and columns are variables. The order of the list should be the same as the order for the training data, Xdata. Use if you want to predict on a testing dataset. If no Xtestdata, set to NULL.
`Ytest`	$ntest \times 1$ vector of test class membership. Numeric, coded as 1, 2, ... If no testing data provided, set to NULL.
`isParallel`	TRUE or FALSE for parallel computing. Default is TRUE.
`ncores`	Number of cores to be used for parallel computing. Only used if isParallel=TRUE. If isParallel=TRUE and ncores=NULL, defaults to half the size of the number of system cores.
`gridMethod`	GridSearch or RandomSearch. Optimize tuning parameters over full grid or random grid. Default is RandomSearch.
`AssignClassMethod`	Classification method. Either Joint or Separate. Joint uses all discriminant vectors from D datasets to predict class membership. Separate predicts class membership separately for each dataset. Default is Joint
`nfolds`	Number of cross validation folds. Default is 5.
`ngrid`	Number of grid points for tuning parameters. Default is 8 for each view if $D = 2$. If $D > 2$, default is 5.
`standardize`	TRUE or FALSE. If TRUE, data will be normalized to have mean zero and variance one for each variable. Default is TRUE.
`maxiteration`	Maximum iteration for the algorithm if not converged. Default is 20.
`weight`	Balances separation and association. Default is 0.5.
`thresh`	Threshold for convergence. Default is 0.001.
`eta`	Balances the selection of network, and variables within network. Default is 0.5.

Details

The function will return several R objects, which can be assigned to a variable. To see the results, use the “$" operator.

Value

A list containing the following information:

`sidaerror`	Estimated classication error. If testing data provided, this will be test classification error, otherwise, training error
`sidacorrelation`	Sum of pairwise RV coefficients. Normalized to be within 0 and 1, inclusive.

`hatalpha`	A list of estimated sparse discriminant vectors for each view.
`PredictedClass`	Predicted class for test data. If AssignClassMethod='Separate', this will be a $ntest \times D$ matrix, with each column the predicted class for each data.
`PredictedClass.train`	Predicted class for train data. If AssignClassMethod='Separate', this will be a $ntrain \times D$ matrix, with each column the predicted class for each data.
`optTau`	Optimal tuning parameters for each view, not including covariates, if available.
`gridValues`	Grid values used for searching optimal tuning paramters.
`AssignClassMethod`	Classification method used. Joint or Separate.
`gridMethod`	Grid method used. Either GridSearch or RandomSearch

References

Sandra E. Safo, Eun Jeong Min, and Lillian Haine (2022), Sparse Linear Discriminant Analysis for Multi-view Structured Data, Biometrics.

See Also

sidanet

Examples

```
## Not run:
##---- read in data
data(sidanetData)

##---- call sidanet algorithm to estimate discriminant vectors, and predict on testing data
#call sidanettunerange to get range of tuning paramater
Xdata=sidanetData[[1]]
Y=sidanetData[[2]]
Xtestdata=sidanetData[[3]]
Ytest=sidanetData[[4]]
myedges=sidanetData[[5]]
myedgeweight=sidanetData[[6]]

 mycv=cvSIDANet(Xdata,Y,myedges,myedgeweight,withCov=FALSE,plotIt=FALSE,Xtestdata=Xtestdata,
    Ytest=Ytest,isParallel=FALSE,ncores=NULL,gridMethod='RandomSearch',
    AssignClassMethod='Joint',nfolds=5,ngrid=8,standardize=TRUE,
    maxiteration=20, weight=0.5,thresh=1e-03,eta=0.5)

#check output
test.error=mycv$sidaneterror
test.correlation=mycv$sidanetcorrelation
optTau=mycv$optTau
hatalpha=mycv$hatalpha

## End(Not run)
```

`cvtunerange` *Tuning parameter range*

Description

Obtain upper and lower bounds of tuning parameters for each canonical correlation vector. It is recommended to use cvselpscca to choose optimal tuning parameters for each dataset.

Usage

```
cvtunerange(
  Xdata1 = Xdata1,
  Xdata2 = Xdata2,
  ncancorr = ncancorr,
  CovStructure = "Iden",
  standardize = TRUE
)
```

Arguments

`Xdata1`	A matrix of size $n \times p$ for first dataset. Rows are samples and columns are variables.
`Xdata2`	A matrix of size $n \times q$ for second dataset. Rows are samples and columns are variables.
`ncancorr`	Number of canonical correlation vectors. Default is one.
`CovStructure`	Covariance structure to use in estimating sparse canonical correlation vectors. Either "Iden" or "Ridge". Iden assumes the covariance matrix for each dataset is identity. Ridge uses the sample covariance for each dataset. See reference article for more details.
`standardize`	TRUE or FALSE. If TRUE, data will be normalized to have mean zero and variance one for each variable. Default is TRUE.

Details

The function will return tuning ranges for sparse estimation of canonical correlation vectors. To see the results, use the “$" operator.

Value

The output is a list containing the following components.

`TauX1range`	A $ncancorr \times 2$ matrix of upper and lower bounds of tuning parameters for each canonical correlation vector for first dataset.
`TauX2range`	A $ncancorr \times 2$ matrix of upper and lower bounds of tuning parameters for each canonical correlation vector for second dataset.

References

Sandra E. Safo, Jeongyoun Ahn, Yongho Jeon, and Sungkyu Jung (2018), Sparse Generalized Eigenvalue Problem with Application to Canonical Correlation Analysis for Integrative Analysis of Methylation and Gene Expression Data. Biometrics

See Also

multiplescca cvselpscca

Examples

```
## Not run:
##---- read in data
data(selpData)

Xdata1=selpData[[1]]
Xdata2=selpData[[2]]

  ##---- estimate first canonical correlation vectors
ncancorr=1

#use cvtunerange for range of tuning parameters
mytunerange=cvtunerange(Xdata1=Xdata1,Xdata2=Xdata2,ncancorr=ncancorr,
                        CovStructure="Iden",standardize=TRUE)
print(mytunerange)

#Fix Tau for first and second datasets as 1.1 and 1.0 respectively
Tau=matrix(c(1,1.2,1),nrow=1)
mysparsevectors=multiplescca(Xdata1=Xdata1,Xdata2=Xdata2,ncancorr=ncancorr,
                             Tau=Tau, CovStructure="Iden",standardize=TRUE,
                             maxiteration=20, thresh=0.0001)

#example with two canonical correlation vectors
#use cvselpscca to obtain optimal tuning parameters
mycv=cvselpscca(Xdata1=Xdata1,Xdata2=Xdata2,ncancorr=ncancorr,
                CovStructure="Iden",isParallel=TRUE,ncores=NULL,nfolds=5,
                ngrid=10, standardize=TRUE,thresh=0.0001,maxiteration=20)

Tau=mycv$optTau
mysparsevectors=multiplescca(Xdata1=Xdata1,Xdata2=Xdata2,ncancorr=ncancorr,
                          Tau=Tau, CovStructure="Iden",standardize=TRUE,maxiteration=20,
                             thresh=0.0001)

## End(Not run)
```

---

devianceTable *Deviance Table for predicted, supervised data.*

Description

Produced a deviance table for supervised predictions.

Usage

```
devianceTable(fit, Xtestdata1, Xtestdata2, Ytest)
```

Arguments

`fit`	the output from the filter_supervised() function
`Xtestdata1`	a matrix or data.frame with the first view's test X data
`Xtestdata2`	a matrix or data.frame with the second view's test X data
`Ytest`	a vector of test response values

Value

A data.frame

`DiscriminantPlots` *Discriminant Plots*

Description

Plots discriminant scores (for SIDA) and canonical variates (for SELPCCA) for visualizing class separation

Usage

```
DiscriminantPlots(
  Xtestdata = Xtestdata,
  Ytest = Ytest,
  hatalpha = hatalpha,
  method.used = "SIDA",
  color.palette = NULL,
  plotIt = TRUE
)
```

Arguments

`Xtestdata`	A list with each entry containing views of size $ntest \times p_d$, where $d = 1, \ldots, D$. Rows are samples and columns are variables. Can use testing or training data.
`Ytest`	$ntest \times 1$ vector of class membership.
`hatalpha`	A list of estimated sparse discriminant vectors for each view.
`method.used`	A character specifying the integration method used. These are used for appropriate labeling. Options are "SIDA" and "SELPCCA". Default is "SIDA". For SELPCCA, ncancorr ≥ 2. If ncancorr > 2, plot will be generated for the first two canonical variates.
`color.palette`	character vector of length K (number of classes), specifying the colors to use for the classes, respectively. Defaults to shades of blue and orange (color.BlueOrange). Other option includes red and green combinations (color.GreenRed)
`plotIt`	boolean, if TRUE, prints the plots. If FALSE, returns the ggplots as an object or list. This is useful to customize the ggplots.

Details

The function will return discriminant plots.

Value

NULL

References

Sandra E. Safo, Eun Jeong Min, and Lillian Haine (2023), Sparse Linear Discriminant Analysis for Multi-view Structured Data, Biometrics.

See Also

cvSIDA sidatunerange CorrelationPlots

Examples

```
## Not run:
 #call sida
data(sidaData)
##---- call sida algorithm to estimate discriminant vectors, and predict on testing data

Xdata=sidaData[[1]]
Y=sidaData[[2]]
Xtestdata=sidaData[[3]]
Ytest=sidaData[[4]]

#call sidatunerange to get range of tuning parameter
ngrid=10
mytunerange=sidatunerange(Xdata,Y,ngrid,standardize=TRUE,weight=0.5,withCov=FALSE)

# an example with Tau set as the lower bound
Tau=c(mytunerange$Tauvec[[1]][1], mytunerange$Tauvec[[2]][1])
mysida=sida(Xdata,Y,Tau,withCov=FALSE,Xtestdata=Xtestdata,Ytest=Ytest,AssignClassMethod='Joint',
            plotIt=FALSE, standardize=TRUE,maxiteration=20,weight=0.5,thresh= 1e-03)

test.error=mysida$sidaerror
test.correlation=mysida$sidacorrelation

#estimated discriminant vectors and predicted class
hatalpha=mysida$hatalpha

predictedClass=mysida$PredictedClass

##----plot discriminant plots
#---------Discriminant plot
mydisplot=DiscriminantPlots(Xtestdata,Ytest,mysida$hatalpha)

## End(Not run)
```

filter_supervised *Supervised Filtering*

Description

Performs univariate supervised filtering on multi-source data. A separate model will be fit for each feature within each view of data and all features with p-values less than the specified threshold will be retained.

Usage

```
filter_supervised(
  X,
  Y,
  method = "linear",
  padjust = FALSE,
  adjmethod = "BH",
  thresh = 0.05,
  center = FALSE,
  scale = FALSE,
  standardize = FALSE,
  log2TransForm = FALSE,
  Xtest = NULL
)
```

Arguments

`X`	A list containing all data sources. Each row must represent a subject and each column represents a feature.
`Y`	An outcome vector of length equal to the number of rows in each view of X.
`method`	Options are "linear" for linear regression, "logistic" for logistic regression, "t.test" for a 2-sample unpaired T-test, or "kw" for a Kruskal-Wallis test. Default is "linear".
`padjust`	Boolean on whether or not to adjust pvalue for multiple testing. Default is "F".
`adjmethod`	Options are "holm", "hochberg", "hommel", "bonferroni", "BH" "BY","fdr","none". Default is "BH" if padjust is True.
`thresh`	P-value threshold to determine which features to keep after filtering. Default will keep all features with a p-value < 0.05.
`center`	Boolean on whether or not to center the features prior to filtering.
`scale`	Boolean on whether or not to scale the features to have variance 1 prior to filtering.
`standardize`	Boolean on whether or not to center and scale the features to have mean 0 and variance 1 prior to filtering.
`log2TransForm`	Boolean on whether or not to log2 transform the features prior to filtering. Will return an error if TRUE but data have negative values.
`Xtest`	Optional list containing test data. If included, filtering will be performed only on the training data, X, but Xtest will be subsetted to the same group of features.

Value

A list containing the following (and others):

`X`	List of the filtered X data
`Y`	Vector of the outcome
`Xtest`	List of the subsetted Xtest data
`method`	Method used for filtering
`pval.mat`	Dataset containing the calculated p-values for each feature, coefficients, and whether significant.

Examples

```
##---- read in data
data(sidaData)

Xdata=sidaData[[1]]
Y=sidaData[[2]]

data.red=filter_supervised(Xdata, Y, method="t.test", padjust=FALSE,adjmethod=NULL, thresh=0.05,
center=FALSE, scale=FALSE, standardize=FALSE, log2TransForm=FALSE, Xtest=NULL)

##-----Plot Result via UMAP
umapPlot(data.red)
```

---

filter_unsupervised *Unsupervised Filtering*

Description

Performs univariate unsupervised filtering on multi-source data. A separate model will be fit for each feature within each view of data and all features with p-values less than the specified threshold will be retained.

Usage

```
filter_unsupervised(
  X,
  method = "variance",
  pct.keep = 10,
  center = FALSE,
  scale = FALSE,
  standardize = FALSE,
  log2TransForm = FALSE,
  Xtest = NULL
)
```

Arguments

`X`	A list containing all data sources. Each row must represent a subject and each column represents a feature.
`method`	Options are "variance" which will keep the `pct.keep` percent of features with the highest variance, and "IQR", which will keep the features with the median amount of variance (+/- pct.keep/2). Default is "variance".
`pct.keep`	Percent of variables to keep in each view of data. Default is 10%.
`center`	Boolean on whether or not to center the features after filtering.
`scale`	Boolean on whether or not to scale the features after filtering.
`standardize`	Boolean on whether or not to center and scale the features to have mean 0 and variance 1 after filtering.
`log2TransForm`	Boolean on whether or not to log2 transform the features prior to filtering. Will return an error if TRUE but data have negative values.
`Xtest`	Optional list containing test data. If included, filtering will be performed only on the training data, X, but Xtest will be subsetted to the same group of features.

Value

A list containing the following

`X`	List of the filtered X data
`Xtest`	List of the subsetted Xtest data
`method`	Method used for filtering
`var.mat`	Dataset containing the calculated mean and variances for each feature.

Examples

```
##---- read in data
data(sidaData)

Xdata=sidaData[[1]]

data.red=filter_unsupervised(Xdata,  method="variance", pct.keep=10,
                    center=FALSE, scale=FALSE, standardize=FALSE,  log2TransForm=FALSE,
                     Xtest=NULL)
```

`Loadings` *Loadings*

Description

function to get loadings for variables selected by SIDA, SIDANet, and SELPCCA methods.

Usage

```
Loadings(fit)
```

Arguments

`fit`	the output from cvSIDA, cvSIDANet, or cvselpcca methods

Value

A data.frame with three columns. The first two columns are the non-zero loadings The last column is the View from which the loadings were obtained.

LoadingsPlots *Loadings Plots*

Description

Plots discriminant and canonical vectors to visualize how selected variables contribute to the first and second discriminant (for SIDA and SIDANet) or canonical correlation (for SELPCCA) vectors. Variables farther from the origin and close to first or second axis have higher impact on first or second discriminant/canonical vectors, respectively. Variables farther from the origin and between both first and second axes have similar higher contributions to the first and second discriminant/canonical correlation vectors. In both situations, for SIDA and SIDANet, this suggests that these variables contribute more to the separation of classes and association of views. For SELPCCA, this suggests that these variables contribute more to the association between the two views. This plot can only be generated for classification and association problems with 3 or more classes (SIDA and SIDANet), or for CCA problems with two or more canonical correlation vectors requested (i.e. ncancorr > 1 for SELPCCA).

Usage

```
LoadingsPlots(
  fit,
  color.line = "darkgray",
  keep.loadings = NULL,
  plotIt = TRUE
)
```

Arguments

`fit`	the output from SIDA, SIDANet, and SELPCCA methods
`color.line`	color to use for plotting direction vectors. Default is "darkgray".
`keep.loadings`	numeric, specifying how many variables to represent on loadings plot. This is useful in situations where the number of variables selected is large, and could clutter the plot. If this number is more than the variables selected, it will be set to the maximum number of variables selected for each view. Default is plotting all selected variables.
`plotIt`	boolean, if TRUE, prints the plots. If FALSE, returns the ggplots as an object or list. This is useful to customize the ggplots.

Details

The function will either return nothing (if plotIt == TRUE), or return loading plots, one for each view.

Value

`NULL`

References

Sandra E. Safo, Eun Jeong Min, and Lillian Haine (2023), Sparse Linear Discriminant Analysis for Multi-view Structured Data, Biometrics. Sandra E. Safo, Jeongyoun Ahn, Yongho Jeon, and Sungkyu Jung (2018), Sparse Generalized Eigenvalue Problem with Application to Canonical Correlation Analysis for Integrative Analysis of Methylation and Gene Expression Data. Biometrics

See Also

cvSIDA DiscriminantPlots CorrelationPlots

Examples

```
## Not run:
data("sidanetData")
Xdata <- sidanetData[[1]]
Y <- sidanetData[[2]] #class membership already coded as 1,2,...
Xtestdata <- sidanetData[[3]]
Ytest <- sidanetData[[4]] #class membership already coded as 1,2,...
#edge information
myedges=sidanetData[[5]]
myedgeweight=sidanetData[[6]]
##---- call cross validation
mycvsidanet=cvSIDANet(Xdata,Y,myedges,myedgeweight,withCov=FALSE,plotIt=FALSE,Xtestdata=Xtestdata,
                     Ytest=Ytest, isParallel = FALSE)
LoadingsPlots(mycvsidanet,keep.loadings=c(3,3))

## End(Not run)
```

---

multiplescca *Sparse canonical correlation vectors for fixed tuning parameters*

Description

Function for estimating canonical correlation vectors for a fixed tuning parameters for each dataset.

Usage

```
multiplescca(
  Xdata1 = Xdata1,
  Xdata2 = Xdata2,
  ncancorr = ncancorr,
  Tau = Tau,
  CovStructure = "Iden",
  standardize = TRUE,
  maxiteration = 20,
  thresh = 1e-04
)
```

Arguments

`Xdata1`	A matrix of size $n \times p$ for first dataset. Rows are samples and columns are variables.
`Xdata2`	A matrix of size $n \times q$ for second dataset. Rows are samples and columns are variables.
`ncancorr`	Number of canonical correlation vectors. Default is 1.
`Tau`	A vector of matrix of fixed tuning parameters for each dataset.
`CovStructure`	Covariance structure to use in estimating sparse canonical correlation vectors. Either "Iden" or "Ridge". Iden assumes the covariance matrix for each dataset is identity. Ridge uses the sample covariance for each dataset. See reference article for more details.
`standardize`	TRUE or FALSE. If TRUE, data will be normalized to have mean zero and variance one for each variable. Default is TRUE.
`maxiteration`	Maximum iteration for the algorithm if not converged. Default is 20.
`thresh`	Threshold for convergence. Default is 0.0001.

Details

The function will return several R objects, which can be assigned to a variable. To see the results, use the “$" operator.

Value

The output is a list containing the following components.

`hatalpha`	Estimated sparse canonical correlation vectors for first dataset.
`hatbeta`	Estimated sparse canonical correlation vectors for second dataset.
`maxcorr`	Estimated canonical correlation coefficient.

References

Sandra E. Safo, Jeongyoun Ahn, Yongho Jeon, and Sungkyu Jung (2018), Sparse Generalized Eigenvalue Problem with Application to Canonical Correlation Analysis for Integrative Analysis of Methylation and Gene Expression Data. Biometrics

See Also

`cvselpscca` `cvtunerange`

Examples

```
## Not run:
##---- read in data
data(selpData)

Xdata1=selpData[[1]]
Xdata2=selpData[[2]]

  ##---- estimate first canonical correlation vectors
ncancorr=1
```

```
#use cvtunerange for range of tuning parameters
mytunerange=cvtunerange(Xdata1=Xdata1,Xdata2=Xdata2,ncancorr=ncancorr,
                        CovStructure="Iden",standardize=TRUE)
print(mytunerange)

#Fix Tau for first and second datasets as 1.1 and 1.0 respectively
Tau=matrix(c(1,1.2,1),nrow=1)
mysparsevectors=multiplescca(Xdata1=Xdata1,Xdata2=Xdata2,ncancorr=ncancorr,
                             Tau=Tau, CovStructure="Iden",standardize=TRUE,
                             maxiteration=20, thresh=0.0001)

#example with two canonical correlation vectors
#use cvselpscca to obtain optimal tuning parameters
mycv=cvselpscca(Xdata1=Xdata1,Xdata2=Xdata2,ncancorr=ncancorr,
                CovStructure="Iden",isParallel=TRUE,ncores=NULL,nfolds=5,
                ngrid=10, standardize=TRUE,thresh=0.0001,maxiteration=20)

Tau=mycv$optTau
mysparsevectors=multiplescca(Xdata1=Xdata1,Xdata2=Xdata2,ncancorr=ncancorr,
                          Tau=Tau, CovStructure="Iden",standardize=TRUE,maxiteration=20,
                             thresh=0.0001)

## End(Not run)
```

`networkPlot` *Network visualization of selected variables from integrative analysis methods*

Description

Wrapper function to visualize graph of similarity matrix for selected variables. We estimate pairwise similarity matrix using low-dimensional representations of our sparse integrative analysis methods (selpcca, sida, sidanet). We follow ideas in González et al. 2012 to create bipartite graph (bigraph) where variables or nodes from one view are connected to variables or nodes from another view. We construct the bigraph from a pairwise similarity matrix obtained from the outputs of our integrative analysis methods. We estimate the similarity score between a pair of selected variables from two views by calculating the inner product of each selected variable and the sum of canonical variates (for SELPCCA) or discriminant vectors (for SIDA, SIDANet) for the pairs of views. As noted in González et al. 2012, the entries in the similarity matrix is a robust approximation of the Pearson correlation between pairs of variables and the two views under consideration. This network graph has potential to shed light on the complex associations between pairs of views.

Usage

```
networkPlot(
  object,
  cutoff = NULL,
  color.node = NULL,
  lty.edge = c("solid", "dashed"),
  show.edge.labels = FALSE,
  show.color.key = TRUE,
  vertex.frame.color = "red",
  layout.fun = NULL,
  save = NULL,
  name.save = NULL
)
```

Arguments

`object`	the output from SIDA, SIDANet, and SELPCCA methods
`cutoff`	a vector containing numeric values between 0 and 1 of similarity cutoff to use when generating graphs. Length of vector is number of pairwise data combinations. Variable pairs with high similarity measure may be of interest. The relevance of the associations can be explored by changing the cutoff. This can also be used to reduce the size of the graph, for dense network. Default is 0.5 meaning that graph will only be generated for variable pairs with similarity value greater than 0.5 for each data pair.
`color.node`	vector of length two, specifying the colors of nodes for pairs of views. Defaults to white and yellow.
`lty.edge`	character vector of length 2, specifying the line type for edges with positive and negative weights, respectively. Can be one of "solid", "dashed", "dotted", "dotdash", "longdash" and "twodash". See igraph package for more details. Defaults to c("solid", "dashed"), where positive weights are solid lines, and negative weights are dashed lines.
`show.edge.labels`	boolen indicating whether or not to show weights as edge labels.
`show.color.key`	boolen indicating whether or not to show color key on plot. Defaults to TRUE. Positive weights or similarity values (correlations) are indicated as red and negative values are indicated as green.
`vertex.frame.color`	a character string of color to use as frame for nodes. Defaults to "red".
`layout.fun`	a function, specifying how the vertices will be placed on the graph. Refer to igraph package using help(layout) for more details. Default is layout.fruchterman.reingold.
`save`	should the plot be saved? If so, choose one of these options: "jpeg", "tiff", "png" or "pdf"
`name.save`	character string for the name of the file to be saved.

Details

The function will return D R objects, where D is the number of views. To see the results, use the “$" operator.

Value

A network graph for variables selected. Each list will contain similarity matrix, cutoff used, and indices of pairings.

`networkGraph`	a graph object for each pair of views (if more than two views) that can be interrogated in cytoscape
`SimilarityMatrix`	the similarity matrix used for generating the network for each pair of views
`cutoff`	the cutoff used when generating network
`pairs`	The pairs of views for which network(s) were generated

References

Elise Palzer and Sandra E. Safo 2023. Submitted González I., Lê Cao K-A., Davis, M.J. and Déjean, S. (2012). Visualising associations between paired omics data sets. J. Data Mining 5:19.

Examples

```
## Not run:
##---- load SIDA data
data("sidaData")
Xdata <- sidaData[[1]]
Y <- sidaData[[2]]
Xtestdata <- sidaData[[3]]
Ytest <- sidaData[[4]]
##---- call cross validation
mycv=cvSIDA(Xdata,Y,withCov=FALSE,plotIt=FALSE, Xtestdata=Xtestdata,
Ytest=Ytest, isParallel = FALSE)
##----  Obtain relevance network
networkPlot(mycv,cutoff=0.7)

## End(Not run)
```

`PerformanceMetrics` *Performance Metrics*

Description

Estimates performance metrics for a predicted model. Currently works for binary and continuous outcomes

Usage

```
PerformanceMetrics(Y.pred, Y.test, family = "binomial")
```

Arguments

`Y.pred`	A vector of predicted values. For SELPCCA, this is a vector of predicted probabilities. For SIDA and SIDANet, this is a vector of predicted class.
`Y.test`	A vector of observed test values.
`family`	A string to denote the family for which metrics should be provided. Options are "gaussian", "binomial".

Details

For a binary outcome, we provide the following metrics: "Accuracy","Error rate","Sensitivity", "Specificity", "Matthews Correlation Coefficient (MCC)", "Balanced Accuracy","Balanced Error Rate", "F1 Score", "False.Discovery.Rate", and "Positive Predictive Value".

For a continuous outcome, we provide the following metrics: "Mean Squared Error","Root Mean.Squared Error","Relative Squared Error", . "Root Relative Squared.Error", "Root Absolute Error", "Mean Absolute Error".

Value

An output of performance metrics:

`Metrics`	A table of estimated metrics

See Also

`cvSIDA` `selpscca.pred` `predict.SELPCCA`

Examples

```
## Not run:
data(sidaData)
Xdata=sidaData[[1]]
Y=sidaData[[2]]
Xtestdata=sidaData[[3]]
Ytest=sidaData[[4]]
##---- call cross validation
 mycv=cvSIDA(Xdata,Y,withCov=FALSE,plotIt=FALSE, Xtestdata=Xtestdata,Ytest=Ytest,
             isParallel=FALSE,ncores=NULL,gridMethod='RandomSearch',
             AssignClassMethod='Joint',nfolds=5,ngrid=8,standardize=TRUE,
            maxiteration=20, weight=0.5,thresh=1e-03)
#check output
 test.error=mycv$sidaerror
 test.correlation=mycv$sidacorrelation
 optTau=mycv$optTau
 hatalpha=mycv$hatalpha
 #train metrics
 Y.pred=mycv$PredictedClass.train-1 #to get this in 0 and 1
 Y.train=Y-1 #to get this in 0 and 1
 train.metrics=PerformanceMetrics(Y.pred,Y.train,family='binomial')

 print(train.metrics)
 #obtain predicted class
 Y.pred=mycv$PredictedClass-1 #to get this in 0 and 1
 Ytest.in=Ytest-1 #to get this in 0 and 1
 test.metrics=PerformanceMetrics(Y.pred,Ytest.in,family='binomial')
 print(test.metrics)

## End(Not run)
```

`PerformanceMetricsPlot` *Performance Metrics Plot*

Description

Creates a ggplot showing either an AUC curve (family == "binomial") or a scatter of predicted vs. observed values (family == "gaussian")

Usage

```
PerformanceMetricsPlot(Y.pred, Y.test, family = "binomial")
```

Arguments

`Y.pred`	A vector of predicted values. For SELPCCA, this is a vector of predicted probabilities. For SIDA and SIDANet, this is a vector of predicted class.
`Y.test`	A vector of observed test values.
`family`	A string to denote the family for which metrics should be provided. Options are "gaussian", "binomial".

Details

For a binary outcome, plots an AUC curve. For a continuous (gaussian) outcome, plots a scatter of observed vs. predicted values.

Value

A ggplot object

See Also

`cvSIDA` `selpscca.pred` `predict.SELPCCAPerformanceMetrics`

Examples

```
## Not run:
data(sidaData)
Xdata=sidaData[[1]]
Y=sidaData[[2]]
Xtestdata=sidaData[[3]]
Ytest=sidaData[[4]]
##---- call cross validation
 mycv=cvSIDA(Xdata,Y,withCov=FALSE,plotIt=FALSE, Xtestdata=Xtestdata,Ytest=Ytest,
             isParallel=FALSE,ncores=NULL,gridMethod='RandomSearch',
             AssignClassMethod='Joint',nfolds=5,ngrid=8,standardize=TRUE,
            maxiteration=20, weight=0.5,thresh=1e-03)
#check output
 test.error=mycv$sidaerror
 test.correlation=mycv$sidacorrelation
 optTau=mycv$optTau
 hatalpha=mycv$hatalpha
 #train metrics
 Y.pred=mycv$PredictedClass.train-1 #to get this in 0 and 1
 Y.train=Y-1 #to get this in 0 and 1
 train.metrics=PerformanceMetrics(Y.pred,Y.train,family='binomial')

 print(train.metrics)
 #obtain predicted class
 Y.pred=mycv$PredictedClass-1 #to get this in 0 and 1
 Ytest.in=Ytest-1 #to get this in 0 and 1
 PerformanceMetricsPlot(Y.pred,Ytest.in,family='binomial')

## End(Not run)
```

predict.SELPCCA — *Prediction for out-of-sample data for SELPCCA predict*

Description

A wrapper function to obtain the canonical variates for an out-of-sample dataset based on a fitted SELPCCA model and then use that information to predict Y based on the fitted GLM or Cox model.

Usage

```
## S3 method for class 'SELPCCA'
predict(object, newdata, newdata2, type = "response", ...)
```

Arguments

`object`	A fitted model of class SELPCCA
`newdata`	A matrix of size $n \times p$ for the first dataset. Rows are samples and columns are variables.
`newdata2`	A matrix of size $n \times q$ for the second dataset. Rows are samples and columns are variables.
`type`	See predict.glm() and predict.coxph() for type options and defaults.
`...`	Additional arguments passed to predict.

Value

An object containing the output from predict.glm() or predict.coxph()

See Also

cvSIDA sidatunerange

Examples

```
## Not run:
##---- read in data
data(sidaData)

Xdata1=sidaData[[1]][[1]]
Xdata2=sidaData[[1]][[2]]
Xtestdata1=sidaData[[3]][[1]]
Xtestdata2=sidaData[[3]][[2]]
Y=sidaData[[2]]-1

myresult=selpscca.pred(Xdata1, Xdata2, Y,fitselpCCA=NULL, family="binomial",
                      event=NULL,model.separately=FALSE, ncancorr=1,
                      CovStructure="Iden", isParallel=FALSE, ncores=NULL,
                      nfolds=5, ngrid=10, standardize=TRUE,thresh=0.0001,
                      maxiteration=20, showProgress=T)

#check output
train.correlation=myresult$selp.fit$maxcorr
optTau=myresult$selp.fit$optTau
```

```
hatalpha=myresult$selp.fit$hatalpha
hatbeta=myresult$selp.fit$hatbeta
predictionModel=summary(myresult$mod.fit)

##Performance metrics
##Train Performance Metrics
newPredictions=predict(myresult, newdata=Xdata1, newdata2=Xdata2, type="response")
Y.pred=newPredictions$pred.mod #predicted probabilities
Y.train=Y
train.metrics=PerformanceMetrics(Y.pred,Y.train,family='binomial',isPlot=TRUE)
print(train.metrics)

##Test Performance Metrics
Y.test=sidaData[[4]]-1
newPredictions=predict(myresult, newdata=Xtestdata1, newdata2=Xtestdata2, type="response")
Y.pred=newPredictions$pred.mod #predicted probabilities
test.metrics=PerformanceMetrics(Y.pred,Y.test,family='binomial',isPlot=TRUE)
print(test.metrics)

## End(Not run)
```

`selpData` *Data example for SELPscca*

Description

Simulated data with one true canonical correlation vectors for first and second datasets. The first 20 and 15 variables are nonzero (i.e., signal variables) in the first canonical correlation vectors for the first and second datasets respectively.

Usage

```
selpData
```

Format

A list with 7 elements:

- `Xdata1` A matrix of size 80×200 for first dataset. Rows are samples and columns are variables.
- `Xdata2` A matrix of size 80×150 for second dataset. Rows are samples and columns are variables.
- `Xtestdata1` A matrix of size 400×200 for first dataset. Rows are samples and columns are variables.
- `Xtestdata2` A matrix of size 400×150 for second dataset. Rows are samples and columns are variables.
- `TrueAlpha` The first canonical correlation vector for Xdata1.
- `TrueBeta` The first canonical correlation vector for Xdata2.
- `TrueCorr` The first canonical correlation coefficient.

References

Sandra E. Safo, Jeongyoun Ahn, Yongho Jeon, and Sungkyu Jung (2018), Sparse Generalized Eigenvalue Problem with Application to Canonical Correlation Analysis for Integrative Analysis of Methylation and Gene Expression Data. Biometrics

`selpscca.pred` *2-step supervised SELPCCA*

Description

Performs n-fold cross validation to select optimal tuning parameters for SELPCCA based on training data. Then uses the results to build a GLM or survival model for a pre-specified outcome.

Usage

```
selpscca.pred(
  Xdata1,
  Xdata2,
  Y,
  fitselpCCA = NULL,
  family = "gaussian",
  event = NULL,
  model.separately = FALSE,
  ncancorr = 1,
  CovStructure = "Iden",
  isParallel = TRUE,
  ncores = NULL,
  nfolds = 5,
  ngrid = 10,
  standardize = TRUE,
  thresh = 1e-04,
  maxiteration = 20,
  showProgress = T
)
```

Arguments

`Xdata1`	A matrix of size n × p for first dataset. Rows are samples and columns are variables.
`Xdata2`	A matrix of size n × q for second dataset. Rows are samples and columns are variables.
`Y`	A vector of size n for the outcome. Continuous outcomes do not have to be centered or scaled. If family="survival", Y is a vector of size n indicating the time at which the event occurred or the observation was censored. See 'event' for more information on how to use function for a survival outcome.
`fitselpCCA`	The output of cvselpscca() function or multiplescca(). If NULL, the algorithm will fit a cvselpscca model.
`family`	A string to denote the type of prediction model to build. Options are "gaussian", "binomial", "poisson", or "survival". When family="survival", a proportional Cox model will be fitted. Otherwise a generalized linear model will be used.
`event`	A vector of size n needed when family="survival" to denote whether or not the event of interest occurred at timepoint Y. Let event=NULL when family does not equal "survival".

`model.separately`	A boolean to denote whether or not to use separate prediction models for Xdata1 and Xdata2. When model.separately=FALSE, a single model will be fit using the output for both datasets.
`ncancorr`	Number of canonical correlation vectors. Default is 1.
`CovStructure`	Covariance structure to use in estimating sparse canonical correlation vectors. Either "Iden" or "Ridge". Iden assumes the covariance matrix for each dataset is identity. Ridge uses the sample covariance for each dataset. See reference article for more details.
`isParallel`	TRUE or FALSE for parallel computing. Default is TRUE.
`ncores`	Number of cores to be used for parallel computing. Only used if isParallel=TRUE. If isParallel=TRUE and ncores=NULL, defaults to half the size of the number of system cores.
`nfolds`	Number of cross validation folds. Default is 5.
`ngrid`	Number of grid points for tuning parameters. Default is 10 for each dataset.
`standardize`	TRUE or FALSE. If TRUE, data will be normalized to have mean zero and variance one for each variable. Note that this only standardizes Xdata1 and Xdata2. Y will not be standardized. Default is TRUE.
`thresh`	Threshold for convergence. Default is 0.0001.
`maxiteration`	Maximum iteration for the algorithm if not converged. Default is 20.
`showProgress`	A boolean for whether or not the function should display text output at various stages in the function to indicate progress. Default is TRUE.

Details

The function will return several R objects, which can be assigned to a variable. To see the results, use the “$" operator.

Value

The output is a list containing the following components.

`selp.fit`	The output of the cvselpscca() function.
`mod.fit`	The output of the glm() or coxph() regression model.
`data.matrix`	The data matrix that was used to build the regression model.
`family`	The type of outcome specified.

References

Sandra E. Safo, Jeongyoun Ahn, Yongho Jeon, and Sungkyu Jung (2018), Sparse Generalized Eigenvalue Problem with Application to Canonical Correlation Analysis for Integrative Analysis of Methylation and Gene Expression Data. Biometrics

See Also

`cvselpscca`

Examples

```
## Not run:
##---- read in data
data(sidaData)

Xdata1=sidaData[[1]][[1]]
Xdata2=sidaData[[1]][[2]]
Y=sidaData[[2]]-1

myresult=selpscca.pred(Xdata1, Xdata2, Y,fitselpCCA=NULL, family="binomial",
            event=NULL,model.separately=FALSE, ncancorr=1,
            CovStructure="Iden", isParallel=FALSE, ncores=NULL,
            nfolds=5, ngrid=10, standardize=TRUE,thresh=0.0001,
            maxiteration=20, showProgress=T)

#check output
train.correlation=myresult$selp.fit$maxcorr
optTau=myresult$selp.fit$optTau
hatalpha=myresult$selp.fit$hatalpha
hatbeta=myresult$selp.fit$hatbeta
predictionModel=summary(myresult$mod.fit)

##Performance metrics
##Train Performance Metrics
newPredictions=predict(myresult, newdata=Xdata1, newdata2=Xdata2, type="response")
Y.pred=newPredictions$pred.mod #predicted probabilities
Y.train=Y
train.metrics=PerformanceMetrics(Y.pred,Y.train,family='binomial',isPlot=TRUE)
print(train.metrics)

##Test Performance Metrics
Y.test=sidaData[[4]]-1
newPredictions=predict(myresult, newdata=Xtestdata1, newdata2=Xtestdata2, type="response")
Y.pred=newPredictions$pred.mod #predicted probabilities
test.metrics=PerformanceMetrics(Y.pred,Y.test,family='binomial',isPlot=TRUE)
print(test.metrics)

## End(Not run)
```

---

sida *Sparse Integrative Discriminant Analysis for Multi-View Data*

Description

Performs sparse integrative discriminant analysis of multi-view data to 1) obtain discriminant vectors that are associated and optimally separate subjects into different classes 2) estimate misclassification rate, and total correlation coefficient. Allows for the inclusion of other covariates which are not penalized in the algorithm. It is recommended to use cvSIDA to choose best tuning parameter.

Usage

```
sida(
  Xdata = Xdata,
  Y = Y,
  Tau = Tau,
  withCov = FALSE,
  Xtestdata = NULL,
  Ytest = NULL,
  AssignClassMethod = "Joint",
  plotIt = FALSE,
  standardize = TRUE,
  maxiteration = 20,
  weight = 0.5,
  thresh = 0.001
)
```

Arguments

`Xdata`	A list with each entry containing training views of size $n \times p_d$, where $d = 1, \ldots, D$ views. Rows are samples and columns are variables. If covariates are available, they should be included as a separate view, and set as the last dataset. For binary or categorical covariates (assumes no ordering), we suggest the use of indicator variables.
`Y`	$n \times 1$ vector of class membership. Numeric, coded as 1, 2, ...
`Tau`	$d \times 1$ vector of tuning parameter. It is recommended to use sidatunerange to obtain lower and upper bounds for the tuning parameters since too large a tuning parameter will result in a trivial solution vector (all zeros) and too small may result in non-sparse vectors.
`withCov`	TRUE or FALSE if covariates are available. If TRUE, please set all covariates as one dataset and should be the last dataset. For binary and categorical variables, use indicator matrices/vectors. Default is FALSE.
`Xtestdata`	A list with each entry containing testing views of size $ntest \times p_d$, where $d = 1, \ldots, D$. Rows are samples and columns are variables. The order of the list should be the same as the order for the training data, Xdata. Use if you want to predict on a testing dataset. If no Xtestdata, set to NULL.
`Ytest`	$ntest \times 1$ vector of test class membership. If no testing data provided, set to NULL.
`AssignClassMethod`	Classification method. Either Joint or Separate. Joint uses all discriminant vectors from D datasets to predict class membership. Separate predicts class membership separately for each dataset. Default is Joint.
`plotIt`	TRUE or FALSE. If TRUE, produces discriminants and correlation plots. Default is FALSE.
`standardize`	TRUE or FALSE. If TRUE, data will be normalized to have mean zero and variance one for each variable. Default is TRUE.
`maxiteration`	Maximum iteration for the algorithm if not converged.Default is 20.
`weight`	Balances separation and association. Default is 0.5.
`thresh`	Threshold for convergence. Default is 0.001.

Details

The function will return several R objects, which can be assigned to a variable. To see the results, use the “$" operator.

Value

The output is a list containing the following components.

`sidaerror`	Estimated classication error. If testing data provided, this will be test classification error, otherwise, training error
`sidacorrelation`	Sum of pairwise RV coefficients. Normalized to be within 0 and 1, inclusive.
`hatalpha`	A list of estimated sparse discriminant vectors for each view.
`PredictedClass`	Predicted class. If AssignClassMethod='Separate', this will be a $ntest \times D$ matrix, with each column the predicted class for each data.

References

Sandra E. Safo, Eun Jeong Min, and Lillian Haine (2022), Sparse Linear Discriminant Analysis for Multi-view Structured Data, Biometrics.

See Also

`cvSIDA` `sidatunerange`

Examples

```
## Not run:
 #call sida
data(sidaData)
##---- call sida algorithm to estimate discriminant vectors, and predict on testing data

Xdata=sidaData[[1]]
Y=sidaData[[2]]
Xtestdata=sidaData[[3]]
Ytest=sidaData[[4]]

#call sidatunerange to get range of tuning parameter
ngrid=10
mytunerange=sidatunerange(Xdata,Y,ngrid,standardize=TRUE,weight=0.5,withCov=FALSE)

# an example with Tau set as the lower bound
Tau=c(mytunerange$Tauvec[[1]][1], mytunerange$Tauvec[[2]][1])
mysida=sida(Xdata,Y,Tau,withCov=FALSE,Xtestdata=Xtestdata,Ytest=Ytest,AssignClassMethod='Joint',
            plotIt=FALSE, standardize=TRUE,maxiteration=20,weight=0.5,thresh= 1e-03)

test.error=mysida$sidaerror
test.correlation=mysida$sidacorrelation

#estimated discriminant vectors and predicted class
hatalpha=mysida$hatalpha

predictedClass=mysida$PredictedClass

## End(Not run)
```

`sidaclassify` *Classification approach for SIDA and SIDANet*

Description

Performs classification using nearest centroid on separate or combined estimated discriminant vectors, and predicts class membership.

Usage

```
sidaclassify(
  hatalpha = hatalpha,
  Xtestdata = Xtestdata,
  Xdata = Xdata,
  Y = Y,
  AssignClassMethod = "Joint",
  standardize = TRUE
)
```

Arguments

`hatalpha`	A list of estimated sparse discriminant vectors for each view. This may be obtained from sida or cvSIDA.
`Xtestdata`	A list with each entry containing testing views of size $ntest \times p_d$, where $d = 1, \ldots, D$ views. Rows are samples and columns are variables. The order of the list should be the same as the order for the training data, Xdata. If covariates are available, they should be included as a separate view, and set as the last dataset. For binary or categorical covariates (assumes no ordering), we suggest the use of indicator variables. If you want to obtain training error, set as Xdata.
`Xdata`	A list with each entry containing training views of size $n \times p_d$, where $d = 1, \ldots, D$ views. Rows are samples and columns are variables. If covariates are available, they should be included as a separate view, and set as the last dataset. For binary or categorical covariates (assumes no ordering), we suggest the use of indicator variables.
`Y`	$n \times 1$ vector of class membership. Same size as the number of training samples. Numeric, coded as 1, 2, ...
`AssignClassMethod`	Classification method. Either Joint or Separate. Joint uses all discriminant vectors from D datasets to predict class membership. Separate predicts class membership separately for each dataset. Default is Joint.
`standardize`	TRUE or FALSE. If TRUE, data will be normalized to have mean zero and variance one for each variable. Default is TRUE.

Value

An R object containing the following information:

`PredictedClass`	Predicted class. If AssignClassMethod='Separate', this will be a $ntestD$ matrix, with each column the predicted class for each data.
`AssignClassMethod`	Classification method used. Either Joint or Separate.

References

Sandra E. Safo, Eun Jeong Min, and Lillian Haine (2022) , Sparse Linear Discriminant Analysis for Multi-view Structured Data, Biometrics.

See Also

`cvSIDA` `sida`

Examples

```
## Not run:
 #call sida
data(sidaData)
##---- call sida algorithm to estimate discriminant vectors, and predict on testing data

Xdata=sidaData[[1]]
Y=sidaData[[2]]
Xtestdata=sidaData[[3]]
Ytest=sidaData[[4]]

#call sidatunerange to get range of tuning paramater
ngrid=10

mytunerange=sidatunerange(Xdata,Y,ngrid,standardize=TRUE,weight=0.5,withCov=FALSE)
# an example with Tau set as the lower bound
Tau=c(mytunerange$Tauvec[[1]][1], mytunerange$Tauvec[[2]][1])
mysida=sida(Xdata,Y,Tau,withCov=FALSE,Xtestdata=Xtestdata,Ytest=Ytest)
#classification with combined estimated vectors
mysida.classify.Joint=sidaclassify(mysida$hatalpha,Xtestdata,Xdata,Y,
                                   AssignClassMethod='Joint')
mysida.PredClass.Joint=mysida.classify.Joint$PredictedClass
#classification with separate estimated vectors
mysida.classify.Separate=sidaclassify(mysida$hatalpha,Xtestdata,Xdata,Y,
                                     AssignClassMethod='Separate')
mysida.PredClass.Separate=mysida.classify.Separate$PredictedClass

## End(Not run)
```

`sidaData` *Data example for SIDA*

Description

Simulated data to demonstrate the use of SIDA.

Usage

```
sidaData
```

Format

A list with 4 elements:

`Xdata` A list with each entry containing two views of training data with dimension 160×2000 each. Rows are samples and columns are variables.

`Y` 160×1 vector of training class membership. There are two classes each with size 80.

`Xtestdata` A list with each entry containing two views of testing data with dimension 320×2000 each. Rows are samples and columns are variables.

`Ytest` 320×1 vector of testing class membership. There are two classes each with size 160.

References

Sandra E. Safo, Eun Jeong Min, and Lillian Haine (2019), Sparse Linear Discriminant Analysis for Multi-view Structured Data, submitted.

`sidanet` *Sparse Integrative Discriminant Analysis for Multi-view Structured (Network) Data*

Description

Performs sparse integrative disdcriminant analysis of multi-view structured (network) data to 1) obtain discriminant vectors that are associated and optimally separate subjects into different classes 2) estimate misclassification rate, and total correlation coefficient. The Laplacian of the underlying graph is used to smooth the discriminant vectors to encourage variables within a view that are connected to have a similar effect. Allows for the inclusion of other covariates which are not penalized in the algorithm. It is recommended to use cvSIDANet to choose best tuning parameter.

Usage

```
sidanet(
  Xdata = Xdata,
  Y = Y,
  myedges = myedges,
  myedgeweight = myedgeweight,
  Tau = Tau,
  withCov = FALSE,
  Xtestdata = NULL,
  Ytest = NULL,
  AssignClassMethod = "Joint",
  plotIt = FALSE,
  standardize = TRUE,
  maxiteration = 20,
  weight = 0.5,
  thresh = 0.001,
  eta = 0.5,
  mynormLaplacianG = NULL
)
```

Arguments

`Xdata`	A list with each entry containing training views of size $n \times p_d$, where $d = 1, \ldots, D$. Rows are samples and columns are variables. If covariates are available, they should be included as a separate view, and set as the last dataset. For binary or categorical covariates (assumes no ordering), we suggest the use of indicator variables.
`Y`	$n \times 1$ vector of class membership. Numeric, coded as 1, 2, ...
`myedges`	A list with each entry containing a $M_d \times 2$ matrix of edge information for each view. If a view has no edge information, set to 0; this will default to SIDA. If covariates are available as a view (Dth view), the edge information should be set to 0.
`myedgeweight`	A list with each entry containing a Md×1 vector of weight information for each view. If a view has no weight information,set to 0; this will use the Laplacian of an unweighted graph. If covariates are available as a view (Dth view), the weight information should be set to 0.
`Tau`	$d \times 1$ vector of tuning parameter. It is recommended to use sidatunerange to obtain lower and upper bounds for the tuning parameters since too large a tuning parameter will result in a trivial solution vector (all zeros) and too small may result in non-sparse vectors.
`withCov`	TRUE or FALSE if covariates are available. If TRUE, please set all covariates as one dataset and should be the last dataset. For binary and categorical variables, use indicator matrices/vectors. Default is FALSE.
`Xtestdata`	A list with each entry containing testing views of size $ntest \times p_d$, where $d = 1, ..., D$. Rows are samples and columns are variables. The order of the list should be the same as the order for the training data, Xdata. Use if you want to predict on a testing dataset. If no Xtestdata, set to NULL.
`Ytest`	$ntest \times 1$ vector of test class membership. Numeric, coded as 1, 2, ... If no testing data provided, set to NULL.
`AssignClassMethod`	Classification method. Either Joint or Separate. Joint uses all discriminant vectors from D datasets to predict class membership. Separate predicts class membership separately for each dataset. Default is Joint.
`plotIt`	TRUE or FALSE. If TRUE, produces discriminants and correlation plots. Default is FALSE.
`standardize`	TRUE or FALSE. If TRUE, data will be normalized to have mean zero and variance one for each variable. Default is TRUE.
`maxiteration`	Maximum iteration for the algorithm if not converged.Default is 20.
`weight`	Balances separation and association. Default is 0.5.
`thresh`	Threshold for convergence. Default is 0.001.
`eta`	Balances the selection of network, and variables within network. Default is 0.5.
`mynormLaplacianG`	The normalized Laplacian of a graph. Set to NULL and this would be estimated using edge matrix and edge weights.

Details

The function will return several R objects, which can be assigned to a variable. To see the results, use the “$" operator.

Value

A list containing the following information:

`sidaneterror`	Estimated classification error. If testing data provided, this will be test classification error, otherwise, training error
`sidanetcorrelation`	Sum of pairwise RV coefficients. Normalized to be within 0 and 1, inclusive.
`hatalpha`	A list of estimated sparse discriminant vectors for each view.
`PredictedClass`	Predicted class. If AssignClassMethod='Separate', this will be a $ntest \times D$ matrix, with each column the predicted class for each data.

References

Sandra E. Safo, Eun Jeong Min, and Lillian Haine (2022), Sparse Linear Discriminant Analysis for Multi-view Structured Data, Biometrics.

See Also

`cvSIDANet`

Examples

```
## Not run:
##---- read in data
data(sidanetData)

##---- call sidanet algorithm to estimate discriminant vectors, and predict on testing data
#call sidanettunerange to get range of tuning paramater
Xdata=sidanetData[[1]]
Y=sidanetData[[2]]
Xtestdata=sidanetData[[3]]
Ytest=sidanetData[[4]]
myedges=sidanetData[[5]]
myedgeweight=sidanetData[[6]]

ngrid=10
mytunerange=sidanettunerange(Xdata,Y,ngrid,standardize=TRUE,weight=0.5,eta=0.5,
                             myedges,myedgeweight)

# an example with Tau set as the lower bound
Tau=c(mytunerange$Tauvec[[1]][1], mytunerange$Tauvec[[2]][1])

#example with two views having edge weights
mysidanet=sidanet(Xdata,Y,myedges,myedgeweight,Tau,Xtestdata=Xtestdata,Ytest=Ytest)

test.error=mysidanet$sidaneterror
test.correlation=mysidanet$sidanetcorrelation
hatalpha=mysidanet$hatalpha
predictedClass=mysidanet$PredictedClass

## End(Not run)
```

`sidanetData` *Data example for SIDANet*

Description

Simulated data to demonstrate the use of SIDANet.

Usage

```
sidanetData
```

Format

A list with 6 elements:

`XdataNet` A list with each entry containing two views of training data with dimension 240 × 1000 each. Rows are samples and columns are variables.

`YNet` 240 × 1 vector of training class membership. There are three classes each with size 80.

`XtestdataNet` A list with each entry containing two views of testing data with dimension 480 × 1000 each. Rows are samples and columns are variables.

`YtestNet` 480 × 1 vector of testing class membership. There are three classes each with size 160.

`myedges` A list with each entry containing a 36 × 2 matrix of edge information for each view. Assumes variable 1 is connected to variables 2 to 10, variable 11 is connected to variables 12 to 20, variable 21 is connected to variables 22 to 30 and variable 31 is connected to variables 32 to 40. All remaining variables are singletons.

`myedgeweight` A list with each entry containing edgeweight. In this example, views 1 and 2 have edge weights so the Laplacian of a weighted graph will be used.

References

Sandra E. Safo, Eun Jeong Min, and Lillian Haine (2019), Sparse Linear Discriminant Analysis for Multi-view Structured Data, submitted.

`sidanettunerange` *Tuning paramter grid values for sidanet*

Description

Sidanet function to provide tuning parameter grid values for each view, not including covariates, if available. It is recommended to use this to get lower and upper bounds of tuning parameters for each view that can be used in sidanet. This function is called by cvSIDANet to select optimal tuning parameters.

Usage

```
sidanettunerange(
  Xdata = Xdata,
  Y = Y,
  ngrid = 8,
  standardize = TRUE,
  weight = 0.5,
  eta = 0.5,
  myedges = myedges,
  myedgeweight = myedgeweight,
  withCov = FALSE
)
```

Arguments

`Xdata`	A list with each entry containing training views of size $n \times p_d$, where $d = 1, \ldots, D$. Rows are samples and columns are variables. If covariates are available, they should be included as a separate view, and set as the last dataset. For binary or categorical covariates (assumes no ordering), we suggest the use of indicator variables.
`Y`	$n \times 1$ vector of class membership. Numeric, coded as 1, 2, ...
`ngrid`	Number of grid points for tuning parameters.
`standardize`	TRUE or FALSE. If TRUE, data will be normalized to have mean zero and variance one for each variable. Default is TRUE.
`weight`	Balances separation and association. Default is 0.5.
`eta`	Balances the selection of network, and variables within network. Default is 0.5.
`myedges`	A list with each entry containing a $M_d \times 2$ matrix of edge information for each view. If a view has no edge information, set to 0; this will default to SIDA. If covariates are available as a view (Dth view), the edge information should be set to 0.
`myedgeweight`	A list with each entry containing a $M_d \times 1$ vector of weight information for each view. If a view has no weight information, set to 0; this will use the Laplacian of an unweighted graph. If covariates are available as a view (Dth view), the weight information should be set to 0.
`withCov`	TRUE or FALSE if covariates are available. If TRUE, please set all covariates as one dataset and should be the last dataset. For binary and categorical variables, use indicator matrices/vectors. Default is FALSE.

Details

The function will return several R objects, which can be assigned to a variable. To see the results, use the “$" operator.

Value

`Tauvec`	Grid values for each data, not including covariates, if available.

References

Sandra E. Safo, Eun Jeong Min, and Lillian Haine (2022), Sparse Linear Discriminant Analysis for Multi-view Structured Data, Biometrics.

See Also

sidanet

Examples

```
## Not run:
##---- read in data
data(sidanetData)

##---- call sidanet algorithm to estimate discriminant vectors, and predict on testing data
#call sidanettunerange to get range of tuning paramater
Xdata=sidanetData[[1]]
Y=sidanetData[[2]]
Xtestdata=sidanetData[[3]]
Ytest=sidanetData[[4]]
myedges=sidanetData[[5]]
myedgeweight=sidanetData[[6]]

ngrid=10
mytunerange=sidanettunerange(Xdata,Y,ngrid,standardize=TRUE,weight=0.5,eta=0.5,
                             myedges,myedgeweight)

# an example with Tau set as the lower bound
Tau=c(mytunerange$Tauvec[[1]][1], mytunerange$Tauvec[[2]][1])

#example with two views having edge weights
mysidanet=sidanet(Xdata,Y,myedges,myedgeweight,Tau,Xtestdata=Xtestdata,Ytest=Ytest)

test.error=mysidanet$sidaneterror
test.correlation=mysidanet$sidanetcorrelation
hatalpha=mysidanet$hatalpha
predictedClass=mysidanet$PredictedClass

## End(Not run)
```

---

sidatunerange *Tuning parameter grid values for sida*

Description

Sida function to provide tuning parameter grid values for each view, not including covariates, if available. It is recommended to use this to get lower and upper bounds of tuning parameters for each view that can be used in sida. This function is called by cvSIDA to select optimal tuning parameters.

Usage

```
sidatunerange(
  Xdata,
  Y,
  ngrid = 10,
  standardize = TRUE,
  weight = 0.5,
  withCov = TRUE
)
```

Arguments

`Xdata`	A list with each entry containing training views of size $n \times p_d$, where $d = 1, \ldots, D$ views. Rows are samples and columns are variables. If covariates are available, they should be included as a separate view, and set as the last dataset. For binary or categorical covariates (assumes no ordering), we suggest the use of indicator variables.
`Y`	$n \times 1$ vector of class membership. Numeric, coded as 1, 2, ...
`ngrid`	Number of grid points for tuning parameters. Default is 10 for each view if $D = 2$. If $D > 2$, default is 5.
`standardize`	TRUE or FALSE. If TRUE, data will be normalized to have mean zero and variance one for each variable. Default is TRUE.
`weight`	Balances separation and association. Default is 0.5.
`withCov`	TRUE or FALSE if covariates are available. If TRUE, please set all covariates as one dataset and should be the last dataset. For binary and categorical variables, use indicator matrices/vectors. Default is FALSE.

Details

The function will return an R object with grid values for each data, not including covariates, if available. To see the results, use the “$" operator.

Value

An R object containing the following information:

`Tauvec`	grid values for each data, not including covariates, if available.

References

Sandra E. Safo, Eun Jeong Min, and Lillian Haine (2022) , Sparse Linear Discriminant Analysis for Multi-view Structured Data, Biometrics.

See Also

`sida`

Examples

```
## Not run:
 #call sida
data(sidaData)
##---- call sida algorithm to estimate discriminant vectors, and predict on testing data

Xdata=sidaData[[1]]
Y=sidaData[[2]]
Xtestdata=sidaData[[3]]
Ytest=sidaData[[4]]

#call sidatunerange to get range of tuning parameter
ngrid=10
```

```
mytunerange=sidatunerange(Xdata,Y,ngrid,standardize=TRUE,weight=0.5,withCov=FALSE)

# an example with Tau set as the lower bound
Tau=c(mytunerange$Tauvec[[1]][1], mytunerange$Tauvec[[2]][1])
mysida=sida(Xdata,Y,Tau,withCov=FALSE,Xtestdata=Xtestdata,Ytest=Ytest,AssignClassMethod='Joint',
            plotIt=FALSE, standardize=TRUE,maxiteration=20,weight=0.5,thresh= 1e-03)

test.error=mysida$sidaerror
test.correlation=mysida$sidacorrelation

#estimated discriminant vectors and predicted class
hatalpha=mysida$hatalpha

predictedClass=mysida$PredictedClass
#obtain more performance metrics (applicable to two classes)

 #train metrics
 Y.pred=mysida$PredictedClass.train-1 #to get this in 0 and 1
 Y.train=Y-1 #to get this in 0 and 1
 train.metrics=PerformanceMetrics(Y.pred,Y.train,family='binomial')
 print(train.metrics)

 #obtain test predicted class
 Y.pred=mysida$PredictedClass-1 #to get this in 0 and 1
 Ytest.in=Ytest-1 #to get this in 0 and 1
 test.metrics=PerformanceMetrics(Y.pred,Ytest.in,family='binomial')
 print(test.metrics)

## End(Not run)
```

umapPlot *UMAP Plot*

Description

Wrapper function to plot a UMAP of the results after supervised filtering. See "umap" R package for more details on the method.

Usage

```
umapPlot(
  fit,
  useFilteredData = TRUE,
  usePrincipleComponents = TRUE,
  plotIt = TRUE
)
```

Arguments

`fit`	the output from the filter_supervised() function
`useFilteredData`	Boolean on whether to plot UMAP on filtered or original data. Default is filtered data.

`usePrincipleComponents`	boolean on whether to apply PCA first
`plotIt`	boolean, whether to print the result or just return it (default = TRUE)

Value

A graph of the UMAP

Examples

```
##---- read in data
data(sidaData)

Xdata=sidaData[[1]]
Y=sidaData[[2]]

data.red=filter_supervised(Xdata, Y,  method="t.test", padjust=TRUE, thresh=0.05,
                 center=FALSE, scale=FALSE, Xtest=NULL)

##-----Plot Result via UMAP
umapPlot(data.red)
```

`VarImportance` *Variable Importance Table*

Description

returns Variable Importance tables for SIDA, SIDANet, or SELPCCA

Usage

```
VarImportance(fit)
```

Arguments

`fit`	the output from cvSIDA, cvSIDANet, or cvselpcca methods

Value

A dataframe of the absolute loadings for variables selected. The variables are normalized to the variable with the largest weight.

Examples

```
## Not run:
##---- load SIDA data
data("sidaData")
Xdata <- sidaData[[1]]
Y <- sidaData[[2]]
Xtestdata <- sidaData[[3]]
Ytest <- sidaData[[4]]
##---- call cross validation
mycv=cvSIDA(Xdata,Y,withCov=FALSE,plotIt=FALSE, Xtestdata=Xtestdata,Ytest=Ytest, isParallel = FALSE)
##----  Obtain variable importance plot
```

```
VarImportance(mycv)

## End(Not run)
```

VarImportancePlot *Variable Importance Plot*

Description

Wrapper function to visualize loadings for variables selected by SIDA, SIDANet, and SELPCCA methods.

Usage

```
VarImportancePlot(fit, max.loadings = 20, plotIt = TRUE)
```

Arguments

`fit`	the output from cvSIDA, cvSIDANet, or cvselpcca methods
`max.loadings`	the maximum number of variables to show per view (default = 20)
`plotIt`	boolean; if TRUE shows the plots, otherwise returns them as a list of ggplots

Value

A graph of the absolute loadings for variables selected. The variables are normalized to the variable with the largest weight.

Examples

```
## Not run:
#' ##---- load SIDA data
data("sidaData")
Xdata <- sidaData[[1]]
Y <- sidaData[[2]]
Xtestdata <- sidaData[[3]]
Ytest <- sidaData[[4]]
##---- call cross validation
mycv=cvSIDA(Xdata,Y,withCov=FALSE,plotIt=FALSE, Xtestdata=Xtestdata,Ytest=Ytest, isParallel = FALSE)
##----  Obtain variable importance plot
VarImportancePlot(mycv, max.loadings = 15)

## End(Not run)
```

volcanoPlot *Volcano Plot*

Description

Wrapper function for volcano plots of the results after supervised filtering.

Usage

```
volcanoPlot(fit, plotIt = TRUE)
```

Arguments

fit	the output from the filter_supervised() function
plotIt	boolean, whether to print the result (TRUE) or just return it

Value

A graph of the volcano plot

Examples

```
##---- read in data
data(COVIDData)

#make omics data numeric
Proteomics= apply(as.matrix(COVIDData[[1]]), 2, as.numeric)
RNASeq= apply(as.matrix(COVIDData[[2]]), 2, as.numeric)
Clinical= COVIDData[[3]]
X=list(Proteomics, RNASeq)
Y=Clinical$DiseaseStatus.Indicator

data.red=filter_supervised(X, Y,  method="t.test", padjust=TRUE,adjmethod="BH",
thresh=0.05,center=TRUE, scale=TRUE, Xtest=NULL)

##-----Volcano Plot of Result
volcanoPlot(data.red)
```

WithinViewBiplot *Biplots for Discriminant Scores or Canonical Correlation Variates for each View*

Description

Biplots to visualize discriminant scores/ canonical variates and how selected variables contribute to the first and second discriminant (for SIDA and SIDANet) or canonical correlation (for SELPCCA) vectors. Variables farther from the origin and close to first or second axis have higher impact on first or second discriminant/canonical vectors, respectively. Variables farther from the origin and between both first and second axes have similar higher contributions to the first and second discriminant/canonical correlation vectors. In both situations, for SIDA and SIDANet, this suggests that these variables contribute more to the separation of classes and association of views. For

SELPCCA, this suggests that these variables contribute more to the association between the two views. This plot can only be generated for classification and association problems with 3 or more classes (SIDA and SIDANet), or for CCA problems with two or more canonical correlation vectors requested (i.e. ncancorr > 1 for SELPCCA).

Usage

```
WithinViewBiplot(
  fit,
  Y,
  Xtest = NULL,
  color.palette = NULL,
  keep.loadings = NULL,
  plotIt = TRUE
)
```

Arguments

`fit`	the output from SIDA, SIDANet, and SELPCCA methods
`Y`	a vector of class membership for grouping canonical correlatoin variates and discriminant scores.
`Xtest`	list of D entries containing test data. If not null, scores for biplots will be constructed for testing data.
`color.palette`	character vector of length K (number of classes), specifying the colors to use for the classes, respectively. Defaults to shades of blue and orange (color.BlueOrange). Other option includes red and green combinations (color.GreenRed)
`keep.loadings`	numeric vector of length D (number of views), specifying how many variables to represent on loadings plot for each view. This is useful in situations where the number of variables selected is large, and could clutter the plot. If this number is more than the variables selected, it will be set to the maximum number of variables selected for each view. Default is plotting all selected variables.
`plotIt`	boolean, if TRUE, prints the plots. If FALSE, returns the ggplots as an object or list. This is useful to customize the ggplots.

Details

The function will either show the plots (if plotIt == TRUE) or return a list of loading plots, one for each view.

Value

`NULL`

References

Sandra E. Safo, Eun Jeong Min, and Lillian Haine (2023), Sparse Linear Discriminant Analysis for Multi-view Structured Data, Biometrics. Sandra E. Safo, Jeongyoun Ahn, Yongho Jeon, and Sungkyu Jung (2018), Sparse Generalized Eigenvalue Problem with Application to Canonical Correlation Analysis for Integrative Analysis of Methylation and Gene Expression Data. Biometrics

See Also

`cvSIDA` `DiscriminantPlots` `CorrelationPlots`

Examples

```
## Not run:
data("sidanetData")
Xdata <- sidanetData[[1]]
Y <- sidanetData[[2]] #class membership already coded as 1,2,...
Xtestdata <- sidanetData[[3]]
Ytest <- sidanetData[[4]] #class membership already coded as 1,2,...
#edge information
myedges=sidanetData[[5]]
myedgeweight=sidanetData[[6]]
##---- call cross validation
mycvsidanet=cvSIDANet(Xdata,Y,myedges,myedgeweight,withCov=FALSE,plotIt=FALSE,Xtestdata=Xtestdata,
                     Ytest=Ytest, isParallel = FALSE)
WithinViewBiplot(mycvsidanet,Y, color.palette=NULL,keep.loadings=c(3,3))

## End(Not run)
```