Title  Distance-Based Common Spatial Patterns
Version  0.0.2.0
Maintainer  Itsaso Rodríguez-Moreno <itsaso.rodriguez@ehu.eus>
Description  A way to apply Distance-Based Common Spatial Patterns (DB-CSP) techniques in different fields, both classical Common Spatial Patterns (CSP) as well as DB-CSP. The method is composed of two phases: applying the DB-CSP algorithm and performing a classification. The main idea behind the CSP is to use a linear transform to project data into low-dimensional subspace with a projection matrix, in such a way that each row consists of weights for signals. This transformation maximizes the variance of two-class signal matrices. The dbcsp object is created to compute the projection vectors. For exploratory and descriptive purpose, plot and boxplot functions can be used. Functions train, predict and selectQ are implemented for the classification step.
License  GPL (>= 2)
Depends  caret, R (>= 2.10), TSdist (>= 3.7)
Imports  geigen, ggplot2, MASS, Matrix, methods, parallelDist, plyr, stats, zoo
Suggests  testthat (>= 3.0.0)
Config/testthat/edition  3
Encoding  UTF-8
LazyData  true
RoxygenNote  7.1.2
NeedsCompilation  no
Author  Itziar Irigoien [aut], Concepción Arenas [aut], Itsaso Rodríguez-Moreno [cre, aut]
Repository  CRAN
Date/Publication  2022-02-17 16:32:03 UTC
**R topics documented:**

- `dbcsp-package` ................................................................. 2
- `AR.data` ........................................................................... 4
- `boxplot.dbcsp` ................................................................. 4
- `dbcsp-class` ..................................................................... 6
- `plot.dbcsp` ......................................................................... 8
- `predict.dbcsp` ................................................................. 9
- `print.dbcsp` ....................................................................... 10
- `selectQ` ............................................................................ 11
- `summary.dbcsp` ............................................................... 12
- `train.dbcsp` ....................................................................... 13

**Index**

```
dbcsp-package Distance-Based Common Spatial Patterns
```

**Description**

dbcsp is a package which offers a way to apply Distance-Based Common Spatial Patterns (DB-CSP) techniques in different fields, both classical Common Spatial Patterns (CSP) as well as DB-CSP.

**Author(s)**

Itsaso Rodriguez Moreno <itsaso.rodriguez@ehu.eus>
Itziar Irigoien <itziar.irigoien@ehu.eus>
Concepción Arenas <carenas@ub.edu>

**See Also**

dbcsp, print, summary, train, selectQ, predict, plot, boxplot

**Examples**

```
# There is an example dataset called AR.data shipped with the package.

# It contains the skeleton data extracted from videos of people performing six different actions,
# recorded by a humanoid robot. So, it has 6 different classes.

# As the whole process is performed pairwise, first two classes are selected (some instances are
# saved to use later as test)

handshake <- AR.data$handshake[1:15]  
ignore <- AR.data$ignore[1:15]  

# Now, the dbcsp object can be created, where q represents the number of vectors used in
# the projection, the obtained filter will have 2*q dimension.
```
# By default, euclidean distance is used. To change it, just select another distance type.
# A mixture with euclidean distance and another one can be performed too, changing the mixture
# parameter value.

mydbcsp <- new('dbcsp', handshake, ignore, type="minkowski", p=0.2)

# A user-defined custom distance can be used too.

fn <- function(x, y) mean(1 - cos(x - y))
mydbcsp <- new("dbcsp", X1 = handshake, X2 = ignore, type="fn")

# Regarding the classification, train, predict and selectQ functions can be used.
# An LDA model can be train with the features extracted after performing the CSP, both with the
# train function or using the training=TRUE parameter when creating the dbcsp object
mydbcsp <- new('dbcsp', handshake, ignore, q=10, training=TRUE, fold = 1)

# Once the object is created, print and summary functions can be used to see some details
print(mydbcsp)
summary(mydbcsp)

# The predict function returns the predicted labels for the selected test data.
# And if true_labels are indicated, the obtained accuracy is also printed

handshake_test <- AR.data$handshake[41:45]
ignore_test <- AR.data$ignore[41:45]
test_data <- c(handshake_test, ignore_test)
true_labels <- c(rep('handshake',length(handshake_test)),rep('ignore',length(ignore_test)))
predictions <- predict(mydbcsp, test_data,true_labels)

# To help us deciding which is the best dimension to use when performing the CSP algorithm, the
# selectQ function can be used. Instead of using train_size to validate with train/test split,
# cross validation can be performed too.
bestQ <- selectQ(mydbcsp, Q=c(2,3,5), train_size=0.8)

# A plot can also be obtained, which displays the signals before and after the CSP projection
# With the vectors parameter it can be decided which dimensions to show and if we just want to
# plot the transformed signals, the before parameter must be set to FALSE

plot(mydbcsp,class=2,index=1,before=FALSE,vectors = 1:5, legend=TRUE)

# A boxplot can also be obtained to show the features achieved after the CSP (the variances of
# the transformed signals) which are used to perform the classification
# In the CSP algorithm the vectors work in pairs (the first q vectors maximize the variance of
# one class and minimize the variance of the other, while the last q vectors do the opposite),
# by default the vectors are showed in pairs, but this can be changed.
# The logarithm of the variances are plotted, but the value of the variances are shown
# when changing show_log parameter to FALSE

boxplot(mydbcsp, vectors=c(2,4,8))
**Description**

A dataset containing the skeleton data extracted from videos of people performing six different actions, recorded by a humanoid robot. Each class has several instances with 50 values (25 XY body keypoints extracted using OpenPose [https://github.com/CMU-Perceptual-Computing-Lab/openpose](https://github.com/CMU-Perceptual-Computing-Lab/openpose)) per frame of the video (92 frames).

**Usage**

`AR.data`

**Format**

A list of 6 different actions:

- **come** Gesture for telling the robot to come to you. 46 instances of [50x92] matrices.
- **five** Gesture of 'high five'. 45 instances of [50x92] matrices.
- **handshake** Gesture of handshaking with the robot. 45 instances of [50x92] matrices.
- **hello** Gesture for telling hello to the robot. 44 instances of [50x92] matrices.
- **ignore** Ignore the robot, pass by. 46 instances of [50x92] matrices.
- **look_at** Stare at the robot in front of it. 46 instances of [50x92] matrices.

**Examples**

```r
X1 <- AR.data$come
X2 <- AR.data$five
mydbcsp <- new('dbcsp',X1,X2)
```

**Description**

This function plots the variance of the selected vectors in a boxplot.

**Usage**

```r
## S3 method for class 'dbcsp'
boxplot(x, vectors=1, pairs=TRUE, ordered_pairs=TRUE, show_log=TRUE,...)
```
Arguments

- **x**: object of class `dbcsp`.
- **vectors**: integer or array of integers, indicating the index of the projected vectors to plot, by default `vectors=1`.
- **pairs**: logical, if TRUE the pairs of the indicated vectors are also shown, by default `pairs=TRUE`.
- **ordered_pairs**: logical, if TRUE the pairs are plotted next to each other, else pairs are plotted at the end, by default `ordered_pairs=TRUE`.
- **show_log**: logical, if TRUE the logarithms of the variances are displayed, else the variances are displayed, by default `show_log=TRUE`.
- **...**: not currently used.

Details

A boxplot with the variances of the selected vectors. Vectors values must lie between 1 and 2*q, being q the number of dimensions used to perform the DB-CSP algorithm when creating the `dbcsp` object. The following should be taken into account when plotting:

- The first q values (1,...,q) are indicated as a1...aq
- The last q values (q+1,...,2*q) are indicated as b1...bq.

If `pairs=TRUE`, it is recommended that `vectors<q` for better understanding, since their pairs are plotted as well. In case that `vectors>q`, it should be noted that the values are displayed from b1 to bq, where b1 and bq represent q+1 vector and 2*q vector, respectively.

For example if q=15 and `boxplot(object,vectors=16,pairs=FALSE)`, b1 (16-q=1) vector is shown.

Among the selected boxplots, the largest whiskers are obtained and just the outliers within these whiskers are shown, the rest are not displayed. With the outliers which are outside the whiskers it is not possible to get a good visualization.

Value

Displays the boxplot of the variances of the selected vectors.

See Also

- `dbcsp`, `print`, `summary`, `train`, `selectQ`, `predict`, `plot`

Examples

```r
# Read data from 2 classes
x <- AR.data$come
y <- AR.data$five
mydbcsp <- new("dbcsp", X1 = x, X2 = y)
boxplot(mydbcsp)
boxplot(mydbcsp,vectors=1:4,pairs=FALSE)
boxplot(mydbcsp, vectors=c(1,4,7),ordered_pairs=FALSE)
```
Description

An object of class dbcsp. 'dbcsp' stands for Distance-Based Common Spatial Patterns. The object includes the Common Spatial Patterns filter obtained with the input lists and using the distance method indicated.

Details

If the lists of matrices $X_1$ or $X_2$ contain NA values, these are automatically interpolated by a linear interpolation using \texttt{na.approx} function. These new interpolated matrices are saved in the $X_1$ and $X_2$ slots of the object.

The supported distances for \texttt{type} are these ones:

- Included in \texttt{TSdist}: \texttt{infnorm}, \texttt{ccor}, \texttt{sts}, \texttt{lb.keogh}, \texttt{edr}, \texttt{lcss}, \texttt{fourier}, \texttt{tquest}, \texttt{dissim}, \texttt{acf}, \texttt{pacf}, \texttt{ar.lpc.ceps}, \texttt{ar.mah}, \texttt{ar.mah.statistic}, \texttt{ar.mah.pvalue}, \texttt{ar.pic}, \texttt{cdm}, \texttt{cid}, \texttt{cor}, \texttt{cort}, \texttt{int.per}, \texttt{per}, \texttt{mindist.sax}, \texttt{ncd}, \texttt{pred}, \texttt{spec.glk}, \texttt{spec.isd}, \texttt{spec.llr}, \texttt{pdc}, \texttt{frechet}, \texttt{tam}.

- Included in \texttt{parallelDist}: \texttt{bhattacharyya}, \texttt{bray}, \texttt{canberra}, \texttt{chord}, \texttt{divergence}, \texttt{dtw}, \texttt{euclidean}, \texttt{fJaccard}, \texttt{geodesic}, \texttt{hellinger}, \texttt{kullback}, \texttt{mahalanobis}, \texttt{manhattan}, \texttt{maximum}, \texttt{minkowski}, \texttt{poda}, \texttt{soergel}, \texttt{wave}, \texttt{whittaker}.

- It is possible to use a custom distance. The name of the custom distance function is passed as character to the \texttt{type} parameter. In order to use the \texttt{parallelDist} custom distance option, the custom function must be defined as explained in "Details: User-defined distance functions" part of \texttt{parallelDist} documentation. See Examples section below.

The additional parameters for the selected distance (see \texttt{TSdist}, \texttt{parallelDist}) can be passed as parameters when creating the object, which will be saved in \texttt{more} slot. See Examples section below.

The output is a list containing this information (\texttt{object@out}):

- \texttt{vectors} The projection vectors obtained after applying CSP.
- \texttt{eig} The eigenvalues obtained after applying CSP.
- \texttt{proy} The variance values of the projected signals obtained after applying CSP.

And if \texttt{training}=	exttt{TRUE} the following values are also saved:

- \texttt{acc} The mean accuracy value obtained for training data applying cross validation.
- \texttt{used_folds} List of the folds used in the cross validation.
- \texttt{folds_acc} Accuracy values for each of the folds of the cross validation.
- \texttt{model} The trained LDA classifier.
- \texttt{selected_q} The number of vectors used when training.
Slots

X1  list of matrices for data class 1.
X2  list of matrices for data class 2.
q  integer value indicating the number of vectors used in the projection, by default q=15.
labels  vector of two strings indicating labels names, by default names of variables X1 and X2.
type  string which sets the type of distance to be considered, by default type='EUCL'. See details section.
w  weight for the distances mixture D_mixture = w*D_euclidean + (1-w)*D_type, by default w=0.5.
mixture  logical value indicating whether to use distances mixture or not (EUCL + other), by default mixture=FALSE.
training  logical value indicating whether to perform the training or not.
fold  integer value, by default fold=10. It controls the number of partitions when training. If fold==1 a train/test split is performed, with p=0.2 for test indices.
seed  numeric value, by default seed=NULL. Set a seed to ensure reproducible results.
eig.tol  numeric value, by default eig.tol=1e-06, tolerance to convert distance matrix to be definite positive.
verbose logical
more  list, additional parameters to be passed to the distance methods. See details section.
out  list containing the output.

See Also
dbcsp, print, summary, train, selectQ, predict, plot, boxplot

Examples

# To create an instance of a class dbcsp given data from 2 classes
x <- AR.data$come[1:20]
y <- AR.data$five[1:20]
mydbcsp <- new("dbcsp", X1 = x, X2 = y)

# CUSTOM DISTANCE
x <- AR.data$come[1:10]
y <- AR.data$five[1:10]
fn <- function(x, y, eps=1) mean(1 - cos(x - y))*eps
mydbcsp <- new("dbcsp", X1 = x, X2 = y, type="fn", eps=0.9)
Plot function implemented by dbcsp class

Description
This function plots an instance before and/or after its DB-CSP projection.

Usage
```r
## S3 method for class 'dbcsp'
plot(x, class = 1, index = 1, vectors = 1:(x@q*2), pairs = TRUE,
     before = TRUE, after = TRUE, legend = FALSE, getsignals = FALSE, ...)
```

Arguments
- `x`: object of class `dbcsp`.
- `class`: integer, which of both classes to access (1 or 2), by default `class=1`.
- `index`: an integer, representing which instance of the class to plot, by default `index=1`.
- `vectors`: an integer or vector of integers, representing the vectors to plot after the projection, by default all the vectors used in the projection are plotted `vectors=1:(x@q*2)`.
- `pairs`: logical, if TRUE the pairs of the indicated vectors are also shown, by default `pairs=TRUE`.
- `before`: logical, if TRUE the original signals are plotted, by default `before=TRUE`.
- `after`: logical, if TRUE the signals after projection are plotted, by default `after=TRUE`.
- `legend`: logical, if true the legend of the transformed signals is shown, by default `legend=FALSE`. When plotting more than 15 pairs of signals (15*2=30 signals), the legend is not shown. If `before=TRUE` legends are not displayed.
- `getsignals`: logical, if TRUE the projected signals for the selected class, instance and vectors are returned, by default `getsignals=FALSE`.
- `...`: optional arguments inherited from the `matplot` method.

Details
It plots an instance before and/or after being projected with the DB-CSP filter. Vectors values must lie between 1 and 2*q, being q the number of dimensions used to perform the DB-CSP algorithm when creating the `dbcsp` object. The following should be taken into account when plotting:

- The first q values (1,...,q) are indicated as a1...aq, and are plotted with solid lines.
- The last q values (q+1,...,2q) are indicated as b1...bq, and are plotted with dashed lines.

If `pairs=TRUE`, it is recommended that vectors<q for better understanding, since their pairs are plotted as well. In case that `vectors>q`, it should be noted that the values are displayed from b1 to bq, where b1 and bq represent q+1 vector and 2q vector, respectively. The paired vectors (a1-b1, a2-b2, ...) are plotted with the same color, but different line type.

For example if `q=15` and `plot(object, vectors=16, pairs=FALSE), b1 (16-q=1) vector is shown.

The number of rows and columns of the layout (mfrow, mfcol) can not be modified, as the function select them according to before and after parameters.
Value

Displays a plot of the selected instance before and/or after the DB-CSP filter projection. The vectors shown after the projection are differentiated by the q first and q last vectors, since the former maximize the variance of one class and minimize the variance of the other, while the latter do the opposite. If `getsignals=TRUE`, a matrix with the projected signals shown in the plot is returned.

See Also

dbcsp, print, summary, train, selectQ, predict, boxplot

Examples

```r
# Read data from 2 classes
x <- AR.data$come
y <- AR.data$five
mydbcsp <- new("dbcsp", X1 = x, X2 = y)
plot(mydbcsp)
plot(mydbcsp, class=2, index=30, vectors=1:5, before=FALSE, legend=TRUE)
pSignals <- plot(mydbcsp, class=2, index=30, vectors=1:5, before=FALSE, legend=TRUE, getsignals=TRUE)
```

------

**predict.dbcsp**

*Predict function implemented by dbcsp class*

Description

This function returns the labels predicted for the input instances. If `true_targets` are passed as parameter, the accuracy obtained is printed too.

Usage

```r
## S3 method for class 'dbcsp'
predict(object, X_test, true_targets=NULL, ...)
```

Arguments

- **object**  
  object of class `dbcsp`.
- **X_test**  
  list of matrices for test data.
- **true_targets**  
  vector of true labels of the instances. Note that they must match the names of the labels used when training the model.
- **...**  
  not currently used.

Details

It gives the predictions for the test data using the model saved in the object, which has been previously trained with the `train.dbcsp` function. If the `true_targets` are indicated, the confusion matrix and obtained accuracy value are returned too.
Value

The values returned by the LDA `predict` function, a list with these components:

- **class** The MAP classification (a factor)
- **posterior** Posterior probabilities for the classes
- **x** The scores of test cases on up to dimen discriminant variables

If the `true_targets` are indicated, two more items are added to the output list:

- **confusion_matrix** The confusion matrix obtained with predicted labels and true labels.
- **acc** The accuracy value obtained for the test instances.

See Also

dbcsp, print, summary, train, selectQ, plot, boxplot

Examples

```r
# Read data from 2 classes
x <- AR.data$come[1:20]
y <- AR.data$five[1:20]
mydbcsp <- new("dbcsp", X1 = x, X2 = y)
mydbcsp <- train(mydbcsp, fold=3)
test_data <- c(AR.data$come[20:24], AR.data$five[20:24])
test_labels <- c(rep("x",5),rep("y",5))
predictions <- predict(mydbcsp, test_data, test_labels)
# Predicted classes
print(predictions$class)
# Confusion matrix
print(predictions$confusion_matrix)
# Accuracy
print(predictions$acc)
```
**Details**

It provides information about the object and the class.

**Value**

No return value, called for side effects.

**See Also**

dbcsp, summary, train, selectQ, predict, plot, boxplot

**Examples**

```r
# Read data from 2 classes
x <- AR.data$come[1:30]
y <- AR.data$five[1:30]
mydbcsp <- new("dbcsp", X1 = x, X2 = y)
print(mydbcsp)
```

**Description**

This function applies DB-CSP and classification with different dimensions to see which gets the best outcomes.

**Usage**

```r
selectQ(
  object,
  Q = c(1, 2, 3, 5, 10, 15),
  train.size = 0.75,
  CV = FALSE,
  folds = 10,
  seed = NULL
)
```

```r
# S4 method for signature 'dbcsp'
selectQ(
  object,
  Q = c(1, 2, 3, 5, 10, 15),
  train.size = 0.75,
  CV = FALSE,
  folds = 10,
  seed = NULL
)
```
Arguments

object  object of class `dbcsp`.
Q       list of integers which represents the dimensions to use, by default `Q=c(1, 2, 3, 5, 10, 15)`.
train_size float between 0.0 and 1.0 representing the proportion of the dataset to include in the train split, by default `train_size=0.75`.
CV      logical indicating if a cross validation must be performed or not (if TRUE, train_size is not used), by default `CV=FALSE`.
folds   integer, number of folds to use if CV is performed.
seed    numeric value, by default `seed=NULL`. Set a seed to ensure reproducible results.

Value

A data.frame including the dimensions and their corresponding accuracy values. If CV=TRUE, for each dimension, the standard deviation of the accuracy values of the folds is also included in the data frame.

See Also

dbcsp, print, summary, train, predict, plot, boxplot

Examples

```r
# Read data from 2 classes
x <- AR.data$come
y <- AR.data$five
mydbcsp <- new("dbcsp", X1 = x, X2 = y)
result <- selectQ(mydbcsp)
print(result)
```
train.dbcsp

Details

It prints the following information:

• Length and shape of the list of matrices of each class.
• The number of vectors (dimensions) used in the CSP projection.
• Distance used when performing the Common Spatial Patterns algorithm.
• If the training process has already been performed, the obtained training accuracy value.

Value

No return value, called for side effects.

See Also

dbcsp, print, train, selectQ, predict, plot, boxplot

Examples

# Read data from 2 classes
x <- AR.data$come[1:30]
y <- AR.data$five[1:30]
mydbcsp <- new("dbcsp", X1 = x, X2 = y)
summary(mydbcsp)

train.dbcsp

Training process of a dbcsp object, using LDA classifier.

Description

This function applies DB-CSP to the instances and perform the training of a Linear Discriminant Analysis (LDA) classifier using the object data.

Usage

## S3 method for class 'dbcsp'
train(x, selected_q=x@q, fold=x@fold, seed=x@seed, verbose=TRUE,...)

Arguments

x object of class dbcsp.
selected_q integer value indicating the number of vectors to use when training the model,
by default selected_q=x@q.
fold integer value, by default fold=x@fold. It controls the number of partitions. If fold==1
a train/test split is performed, with p=0.2 for test indices.
seed numeric value, by default fold=x@seed. Set a seed to ensure reproducible results.
verbose logical

... not currently used.
Value

The `dbcsp` object with the training results saved as list in `@out`:

- `vectors` The projection vectors obtained after applying CSP.
- `eig` The eigenvalues obtained after applying CSP.
- `proy` The variance values of the projected signals obtained after applying CSP.
- `acc` The mean accuracy value obtained for training data applying cross validation.
- `used_folds` List of the folds used in the cross validation.
- `folds_acc` Accuracy values for each of the folds of the cross validation.
- `model` The trained LDA classifier.
- `selected_q` The number of vectors used when training.

See Also

dbcsp, print, summary, selectQ, predict, plot, boxplot

Examples

```r
# Read data from 2 classes
x <- AR.data$come[1:20]
y <- AR.data$five[1:20]
mydbcsp <- new("dbcsp", X1 = x, X2 = y)
mydbcsp <- train(mydbcsp, fold=3)
print(mydbcsp@out$acc)
```
Index

* datasets
  AR.data, 4

AR.data, 4

boxplot, 2, 7, 9–14
boxplot.dbcsp, 4

dbcsp, 2, 5, 7–14
dbcsp-class, 6
dbcsp-package, 2

matplot, 8

na.approx, 6

parallelDist, 6
plot, 2, 5, 7, 10–14
plot.dbcsp, 8
predict, 2, 5, 7, 9–14
predict.dbcsp, 9
print, 2, 5, 7, 9, 10, 12–14
print.dbcsp, 10

selectQ, 2, 5, 7, 9–11, 11, 13, 14
selectQ.dbcsp-method (selectQ), 11
summary, 2, 5, 7, 9–12, 14
summary.dbcsp, 12

train, 2, 5, 7, 9–13
train.dbcsp, 9, 13
TSdist, 6