Package 'repsim'

May 9, 2026

Type Package

Title Measures of Representational Similarity Across Models

Version 0.1.0

Description Provides a collection of methods for quantifying representational similarity between learned features or multivariate data. The package offers an efficient 'C++' backend, designed for applications in machine learning, computational neuroscience, and multivariate statistics. See Klabunde et al. (2025) <doi:10.1145/3728458> for a comprehensive overview of the topic.

License MIT + file LICENSE

Encoding UTF-8

RoxygenNote 7.3.3

SystemRequirements C++17

Imports Rcpp (>= 1.1.0), Rdpack

LinkingTo Rcpp, RcppEigen

RdMacros Rdpack

NeedsCompilation yes

Author Kisung You [aut, cre]

Maintainer Kisung You <kisung.you@outlook.com>

Repository CRAN

Date/Publication 2025-11-07 14:00:14 UTC

Contents

cca 2
cka 3
dot_product 5
hsic 6
lin_reg 8
pwcca 9
repsim_hsic 11
repsim_kernels 12
svcca 12

Index **14**

`cca` *Canonical Correlation Analysis*

Description

Compute pairwise CCA-based similarities between multiple representations, summarized by either Yanai's GCD measure (Ramsay et al. 1984) or Pillai's trace statistic (Raghu et al. 2017).

Usage

```
cca(mats, summary_type = NULL)
```

Arguments

`mats`	A list of length M containing data matrices of size $(n_{\text{samples}}, p_k)$. All matrices must share the same number of rows for matching samples.
`summary_type`	Character scalar indicating the CCA summary statistic. One of `"yanai"` or `"pillai"`. Defaults to `"yanai"` if `NULL`.

Value

An $M \times M$ symmetric matrix of CCA summary similarities.

References

Golub GH, Zha H (1995). "The Canonical Correlations of Matrix Pairs and Their Numerical Computation." In Friedman A, Miller W, Bojanczyk A, Cybenko G (eds.), *Linear Algebra for Signal Processing*, volume 69, 27–49. Springer New York, New York, NY. ISBN 978-1-4612-8703-2 978-1-4612-4228-4.

Raghu M, Gilmer J, Yosinski J, Sohl-Dickstein J (2017). "SVCCA: Singular Vector Canonical Correlation Analysis for Deep Learning Dynamics and Interpretability." In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, 6078–6087. ISBN 978-1-5108-6096-4.

Ramsay JO, ten Berge J, Styan GPH (1984). "Matrix Correlation." *Psychometrika*, **49**(3), 403–423. ISSN 0033-3123, 1860-0980.

Examples

```
# ------------------------------------------------
# Use "iris" and "USArrests" datasets
#   1. apply scaling to reduce the effect of scales
#   2. add white noise to create multiple representations
#   3. generate 10 perturbations per each dataset
# ------------------------------------------------
# prepare the prototype
set.seed(1)
```

```
X = as.matrix(scale(as.matrix(iris[sample(1:150, 50, replace=FALSE),1:4])))
Y = as.matrix(scale(as.matrix(USArrests)))
n = nrow(X)
p_X = ncol(X)
p_Y = ncol(Y)

# generate 10 of each by perturbation
mats = vector("list", length=20L)
for (i in 1:10){
  mats[[i]] = X + matrix(rnorm(n*p_X, sd=1), nrow=n)
}
for (j in 11:20){
  mats[[j]] = Y + matrix(rnorm(n*p_Y, sd=1), nrow=n)
}

# compute two similarities
cca_gcd = cca(mats, summary_type="yanai")
cca_trace = cca(mats, summary_type="pillai")

# visualize
opar <- par(no.readonly=TRUE)
labs <- paste0("rep ",1:20)
par(pty="s", mfrow=c(1,2))

image(cca_gcd[,20:1], axes=FALSE, main="CCA:Yanai's GCD")
axis(1, seq(0, 1, length.out=20), labels = labs, las = 2)
axis(2, at = seq(0, 1, length.out=20), labels = labs[20:1], las = 2)

image(cca_trace[,20:1], axes=FALSE, main="CCA:Pillai's Trace")
axis(1, seq(0, 1, length.out=20), labels = labs, las = 2)
axis(2, at = seq(0, 1, length.out=20), labels = labs[20:1], las = 2)
par(opar)
```

---

cka *Centered Kernel Alignment*

Description

Compute pairwise CKA similarities between multiple representations using a chosen kernel and estimator.

Usage

```
cka(mats, kernel_type = NULL, estimator = NULL)
```

Arguments

mats	A list of length M containing data matrices of size $(n_{\text{samples}}, p_k)$. All matrices must share the same number of rows for matching samples.
kernel_type	Character scalar indicating the kernel. Defaults to "rbf" (if NULL). See repsim_kernels for a list of available kernels.
estimator	Character scalar indicating the HSIC estimator. Defaults to "gretton" (if NULL). See repsim_hsic for a list of available estimators.

Value

An $M \times M$ matrix of CKA values.

References

Cristianini N, Shawe-Taylor J, Elisseeff A, Kandola J (2001). "On Kernel-Target Alignment." In Dietterich T, Becker S, Ghahramani Z (eds.), *Advances in Neural Information Processing Systems*, volume 14.

Cortes C, Mohri M, Rostamizadeh A (2012). "Algorithms for Learning Kernels Based on Centered Alignment." *Journal of Machine Learning Research*, **13**(1), 795–828. ISSN 1532-4435.

Examples

```
# ------------------------------------------------
# Use "iris" and "USArrests" datasets
#   1. apply scaling to reduce the effect of scales
#   2. add white noise to create multiple representations
#   3. generate 10 perturbations per each dataset
# ------------------------------------------------
# prepare the prototype
set.seed(1)
X = as.matrix(scale(as.matrix(iris[sample(1:150, 50, replace=FALSE),1:4])))
Y = as.matrix(scale(as.matrix(USArrests)))
n = nrow(X)
p_X = ncol(X)
p_Y = ncol(Y)

# generate 10 of each by perturbation
mats = vector("list", length=20L)
for (i in 1:10){
  mats[[i]] = X + matrix(rnorm(n*p_X, sd=1), nrow=n)
}
for (j in 11:20){
  mats[[j]] = Y + matrix(rnorm(n*p_Y, sd=1), nrow=n)
}

# compute similarity with rbf kernel and different estimators
cka1 = cka(mats, estimator="gretton")
cka2 = cka(mats, estimator="song")
cka3 = cka(mats, estimator="lange")
```

```
# visualize
opar <- par(no.readonly=TRUE)
labs <- paste0("rep ",1:20)
par(mfrow=c(1,3), pty="s")

image(cka1[,20:1], axes=FALSE, main="CKA (Gretton)")
axis(1, seq(0, 1, length.out=20), labels = labs, las = 2)
axis(2, at = seq(0, 1, length.out=20), labels = labs[20:1], las = 2)

image(cka2[,20:1], axes=FALSE, main="CKA (Song)")
axis(1, seq(0, 1, length.out=20), labels = labs, las = 2)
axis(2, at = seq(0, 1, length.out=20), labels = labs[20:1], las = 2)

image(cka3[,20:1], axes=FALSE, main="CKA (Lange)")
axis(1, seq(0, 1, length.out=20), labels = labs, las = 2)
axis(2, at = seq(0, 1, length.out=20), labels = labs[20:1], las = 2)
par(opar)
```

`dot_product` *Dot product similarity*

Description

Compute pairwise dot-product similarities between multiple representations.

Usage

```
dot_product(mats)
```

Arguments

`mats`	A list of length M containing data matrices of size $(n_{\text{samples}}, p_k)$. All matrices must share the same number of rows for matching samples.

Value

An $M \times M$ matrix of symmetric dot-product similarities.

References

Kornblith S, Norouzi M, Lee H, Hinton G (2019). "Similarity of Neural Network Representations Revisited." In *International Conference on Machine Learning*, 3519–3529. PMLR.

Examples

```
# ------------------------------------------------
# Use "iris" and "USArrests" datasets
#   1. apply scaling to reduce the effect of scales
#   2. add white noise to create multiple representations
#   3. generate 10 perturbations per each dataset
# ------------------------------------------------
# prepare the prototype
set.seed(1)
X = as.matrix(scale(as.matrix(iris[sample(1:150, 50, replace=FALSE),1:4])))
Y = as.matrix(scale(as.matrix(USArrests)))
n = nrow(X)
p_X = ncol(X)
p_Y = ncol(Y)

# generate 10 of each by perturbation
mats = vector("list", length=20L)
for (i in 1:10){
  mats[[i]] = X + matrix(rnorm(n*p_X, sd=1), nrow=n)
}
for (j in 11:20){
  mats[[j]] = Y + matrix(rnorm(n*p_Y, sd=1), nrow=n)
}

# compute similarity
sim_mat = dot_product(mats)

# visualize
opar <- par(no.readonly=TRUE)
labs <- paste0("rep ",1:20)
par(pty="s")
image(sim_mat[,20:1], axes=FALSE, main="Dot product similarity")
axis(1, seq(0, 1, length.out=20), labels = labs, las = 2)
axis(2, at = seq(0, 1, length.out=20), labels = labs[20:1], las = 2)
par(opar)
```

`hsic` *Hilbert Schmidt Independence Criterion*

Description

Compute pairwise HSIC similarities between multiple representations using a chosen kernel and estimator.

Usage

```
hsic(mats, kernel_type = NULL, estimator = NULL)
```

Arguments

mats	A list of length M containing data matrices of size $(n_{\text{samples}}, p_k)$. All matrices must share the same number of rows for matching samples.
kernel_type	Character scalar indicating the kernel. Defaults to "rbf" (if NULL). See repsim_kernels for a list of available kernels.
estimator	Character scalar indicating the HSIC estimator. Defaults to "gretton" (if NULL). See repsim_hsic for a list of available estimators.

Value

An $M \times M$ matrix of HSIC values.

References

Gretton A, Bousquet O, Smola A, Schölkopf B (2005). "Measuring Statistical Dependence with Hilbert-Schmidt Norms." In Hutchison D, Kanade T, Kittler J, Kleinberg JM, Mattern F, Mitchell JC, Naor M, Nierstrasz O, Pandu Rangan C, Steffen B, Sudan M, Terzopoulos D, Tygar D, Vardi MY, Weikum G, Jain S, Simon HU, Tomita E (eds.), *Algorithmic Learning Theory*, volume 3734, 63–77. Springer Berlin Heidelberg, Berlin, Heidelberg.

Examples

```
# ------------------------------------------------
# Use "iris" and "USArrests" datasets
#   1. apply scaling to reduce the effect of scales
#   2. add white noise to create multiple representations
#   3. generate 10 perturbations per each dataset
# ------------------------------------------------
# prepare the prototype
set.seed(1)
X = as.matrix(scale(as.matrix(iris[sample(1:150, 50, replace=FALSE),1:4])))
Y = as.matrix(scale(as.matrix(USArrests)))
n = nrow(X)
p_X = ncol(X)
p_Y = ncol(Y)

# generate 10 of each by perturbation
mats = vector("list", length=20L)
for (i in 1:10){
  mats[[i]] = X + matrix(rnorm(n*p_X, sd=1), nrow=n)
}
for (j in 11:20){
  mats[[j]] = Y + matrix(rnorm(n*p_Y, sd=1), nrow=n)
}

# compute similarity with rbf kernel and different estimators
hsic1 = hsic(mats, estimator="gretton")
hsic2 = hsic(mats, estimator="song")
hsic3 = hsic(mats, estimator="lange")
```

```
# visualize
opar <- par(no.readonly=TRUE)
labs <- paste0("rep ",1:20)
par(mfrow=c(1,3), pty="s")

image(hsic1[,20:1], axes=FALSE, main="HSIC (Gretton)")
axis(1, seq(0, 1, length.out=20), labels = labs, las = 2)
axis(2, at = seq(0, 1, length.out=20), labels = labs[20:1], las = 2)

image(hsic2[,20:1], axes=FALSE, main="HSIC (Song)")
axis(1, seq(0, 1, length.out=20), labels = labs, las = 2)
axis(2, at = seq(0, 1, length.out=20), labels = labs[20:1], las = 2)

image(hsic3[,20:1], axes=FALSE, main="HSIC (Lange)")
axis(1, seq(0, 1, length.out=20), labels = labs, las = 2)
axis(2, at = seq(0, 1, length.out=20), labels = labs[20:1], las = 2)
par(opar)
```

`lin_reg` *Linear regression fit similarity*

Description

Compute pairwise linear–regression fit between multiple representations. For each pair, a least–squares map is used to assess how well one representation predicts the other, and the result is symmetrized.

Usage

```
lin_reg(mats)
```

Arguments

`mats`	A list of length M containing data matrices of size $(n_{\text{samples}}, p_k)$. All matrices must share the same number of rows for matching samples.

Value

An $M \times M$ matrix of symmetric dot-product similarities.

References

Romero A, Ballas N, Kahou SE, Chassang A, Gatta C, Bengio Y (2014). "FitNets: Hints for Thin Deep Nets." doi:10.48550/ARXIV.1412.6550.

Examples

```
# ------------------------------------------------
# Use "iris" and "USArrests" datasets
#   1. apply scaling to reduce the effect of scales
#   2. add white noise to create multiple representations
#   3. generate 10 perturbations per each dataset
# ------------------------------------------------
# prepare the prototype
set.seed(1)
X = as.matrix(scale(as.matrix(iris[sample(1:150, 50, replace=FALSE),1:4])))
Y = as.matrix(scale(as.matrix(USArrests)))
n = nrow(X)
p_X = ncol(X)
p_Y = ncol(Y)

# generate 10 of each by perturbation
mats = vector("list", length=20L)
for (i in 1:10){
  mats[[i]] = X + matrix(rnorm(n*p_X, sd=1), nrow=n)
}
for (j in 11:20){
  mats[[j]] = Y + matrix(rnorm(n*p_Y, sd=1), nrow=n)
}

# compute similarity
sim_mat = lin_reg(mats)

# visualize
opar <- par(no.readonly=TRUE)
labs <- paste0("rep ",1:20)
par(pty="s")
image(sim_mat[,20:1], axes=FALSE, main="Linear regression fit similarity")
axis(1, seq(0, 1, length.out=20), labels = labs, las = 2)
axis(2, at = seq(0, 1, length.out=20), labels = labs[20:1], las = 2)
par(opar)
```

`pwcca` *Projection-Weighted Canonical Correlation Analysis*

Description

Compute pairwise projection-weighted CCA (PWCCA) similarities between multiple representations. PWCCA reweights canonical directions by the magnitude of each representation's projection onto those directions, emphasizing components that are most used by the representation.

Usage

```
pwcca(mats)
```

Arguments

mats	A list of length M containing data matrices of size $(n_{\text{samples}}, p_k)$. All matrices must share the same number of rows for matching samples.

Value

An $M \times M$ symmetric matrix of PWCCA similarities.

References

Morcos AS, Raghu M, Bengio S (2018). "Insights on Representational Similarity in Neural Networks with Canonical Correlation." In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, NIPS'18, 5732–5741.

See Also

cca

Examples

```
# ------------------------------------------------
# Use "iris" and "USArrests" datasets
#   1. apply scaling to reduce the effect of scales
#   2. add white noise to create multiple representations
#   3. generate 10 perturbations per each dataset
# ------------------------------------------------
# prepare the prototype
set.seed(1)
X = as.matrix(scale(as.matrix(iris[sample(1:150, 50, replace=FALSE),1:4])))
Y = as.matrix(scale(as.matrix(USArrests)))
n = nrow(X)
p_X = ncol(X)
p_Y = ncol(Y)

# generate 10 of each by perturbation
mats = vector("list", length=20L)
for (i in 1:10){
  mats[[i]] = X + matrix(rnorm(n*p_X, sd=1), nrow=n)
}
for (j in 11:20){
  mats[[j]] = Y + matrix(rnorm(n*p_Y, sd=1), nrow=n)
}

# compute the similarity
sim_pwcca = pwcca(mats)

# visualize
opar <- par(no.readonly=TRUE)
labs <- paste0("rep ",1:20)
par(pty="s")
```

```
image(sim_pwcca[,20:1], axes=FALSE, main="PWCCA")
axis(1, seq(0, 1, length.out=20), labels = labs, las = 2)
axis(2, at = seq(0, 1, length.out=20), labels = labs[20:1], las = 2)

par(opar)
```

`repsim_hsic` *List of HSIC estimators*

Description

Returns a character vector of available HSIC estimators implemented in the package.

Usage

```
repsim_hsic()
```

Value

A character vector of estimator names.

References

Gretton A, Bousquet O, Smola A, Schölkopf B (2005). "Measuring Statistical Dependence with Hilbert-Schmidt Norms." In Hutchison D, Kanade T, Kittler J, Kleinberg JM, Mattern F, Mitchell JC, Naor M, Nierstrasz O, Pandu Rangan C, Steffen B, Sudan M, Terzopoulos D, Tygar D, Vardi MY, Weikum G, Jain S, Simon HU, Tomita E (eds.), *Algorithmic Learning Theory*, volume 3734, 63–77. Springer Berlin Heidelberg, Berlin, Heidelberg.

Song L, Smola A, Gretton A, Borgwardt KM, Bedo J (2007). "Supervised Feature Selection via Dependence Estimation." In *Proceedings of the 24th International Conference on Machine Learning*, 823–830.

Lange RD, Kwok D, Matelsky JK, Wang X, Rolnick D, Kording K (2023). "Deep Networks as Paths on the Manifold of Neural Representations." In Doster T, Emerson T, Kvinge H, Miolane N, Papillon M, Rieck B, Sanborn S (eds.), *Proceedings of 2nd Annual Workshop on Topology, Algebra, and Geometry in Machine Learning (TAG-ML)*, volume 221 of *Proceedings of Machine Learning Research*, 102–133.

repsim_kernels *List of kernel functions*

Description

Returns a character vector of available kernel methods implemented in the package.

Usage

```
repsim_kernels()
```

Value

A character vector of kernel names.

svcca *Singular Vector Canonical Correlation Analysis*

Description

Compute pairwise singular vector CCA (SVCCA) similarities between multiple representations. SVCCA first mean-centers and denoises each representation via SVD, retaining components explaining a high fraction of variance at 99 CCA is applied to the reduced representations, and the similarity is summarized with either Yanai's GCD or Pillai's trace.

Usage

```
svcca(mats, summary_type = NULL)
```

Arguments

`mats`	A list of length M containing data matrices of size $(n_{\text{samples}}, p_k)$. All matrices must share the same number of rows for matching samples.
`summary_type`	Character scalar indicating the CCA summary statistic. One of `"yanai"` or `"pillai"`. Defaults to `"yanai"` if `NULL`.

Value

An $M \times M$ symmetric matrix of SVCCA summary similarities.

References

Raghu M, Gilmer J, Yosinski J, Sohl-Dickstein J (2017). "SVCCA: Singular Vector Canonical Correlation Analysis for Deep Learning Dynamics and Interpretability." In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, 6078–6087. ISBN 978-1-5108-6096-4.

See Also

cca

Examples

```
# ------------------------------------------------
# Use "iris" and "USArrests" datasets
#   1. apply scaling to reduce the effect of scales
#   2. add white noise to create multiple representations
#   3. generate 10 perturbations per each dataset
# ------------------------------------------------
# prepare the prototype
set.seed(1)
X = as.matrix(scale(as.matrix(iris[sample(1:150, 50, replace=FALSE),1:4])))
Y = as.matrix(scale(as.matrix(USArrests)))
n = nrow(X)
p_X = ncol(X)
p_Y = ncol(Y)

# generate 10 of each by perturbation
mats = vector("list", length=20L)
for (i in 1:10){
  mats[[i]] = X + matrix(rnorm(n*p_X, sd=1), nrow=n)
}
for (j in 11:20){
  mats[[j]] = Y + matrix(rnorm(n*p_Y, sd=1), nrow=n)
}

# compute two similarities
svcca_gcd = svcca(mats, summary_type="yanai")
svcca_trace = svcca(mats, summary_type="pillai")

# visualize
opar <- par(no.readonly=TRUE)
labs <- paste0("rep ",1:20)
par(pty="s", mfrow=c(1,2))

image(svcca_gcd[,20:1], axes=FALSE, main="SVCCA:Yanai's GCD")
axis(1, seq(0, 1, length.out=20), labels = labs, las = 2)
axis(2, at = seq(0, 1, length.out=20), labels = labs[20:1], las = 2)

image(svcca_trace[,20:1], axes=FALSE, main="SVCCA:Pillai's Trace")
axis(1, seq(0, 1, length.out=20), labels = labs, las = 2)
axis(2, at = seq(0, 1, length.out=20), labels = labs[20:1], las = 2)
par(opar)
```

Index

cca, 2, *10*, *13*
cka, 3

dot_product, 5

hsic, 6

lin_reg, 8

pwcca, 9

repsim_hsic, *4*, *7*, 11
repsim_kernels, *4*, *7*, 12

svcca, 12