Package 'veesa'

January 17, 2025

Type Package

Title Pipeline for Explainable Machine Learning with Functional Data

Version 0.1.6

Description Implements the Variable importance Explainable Elastic Shape Analysis pipeline for explainable machine learning with functional data inputs. Converts training and testing data functional inputs to elastic shape analysis principal components that account for vertical and/or horizontal variability. Computes feature importance to identify important principal components and visualizes variability captured by functional principal components. See Goode et al. (2025) <doi:10.48550/arXiv.2501.07602> for technical details about the methodology.

License MIT + file LICENSE

Encoding UTF-8

LazyData true

RoxygenNote 7.3.2

Depends R (>= 3.5.0)

Imports dplyr, fdasrvf, forcats, ggplot2, purrr, stats, stringr, tidyr

Suggests randomForest, testthat (>= 3.0.0)

Config/testthat/edition 3

NeedsCompilation no

Author Katherine Goode [cre, aut],
J. Derek Tucker [aut],
Sandia National Laboratories [cph, fnd]

Maintainer Katherine Goode <kjgoode@sandia.gov>

Repository CRAN

Date/Publication 2025-01-17 10:10:01 UTC

Contents

`align_pcdirs` *Obtain PC directions with centered warping functions*

Description

The function 'prep_training_data' does not center the warping functions, which leads to issues when visualizing joint and horizontal principal component directions. This function aligns the principal directions for improved interpretability of the principal directions. Currently, only alignment for jfPCA has been implemented.

The function 'prep_training_data' does not center the warping functions, which leads to issues when visualizing joint and horizontal principal component directions. This function aligns the principal directions for improved interpretability of the principal directions. Currently, only alignment for jfPCA has been implemented.

Usage

```
align_pcdirs(train_obj)

align_pcdirs(train_obj)
```

Arguments

`train_obj`	Output object from 'prep_training_data' (jfpca only)

Value

List with the same structure as 'prep_training_data', but the principal directions are replaced with the aligned version and gamI is included in the fpca_res object.

List with the same structure as 'prep_training_data', but the principal directions are replaced with the aligned version and gamI is included in the fpca_res object.

center_warping_funs *Center warping functions*

Description

The function 'prep_training_data' does not center the warping functions. For visualizing the aligned and warping functions, it can be easier to look at centered versions. This function centers the warping functions and corresponding aligned functions.

Usage

```
center_warping_funs(train_obj)
```

Arguments

train_obj	Output object from 'prep_training_data'

Value

Object with the same structure as 'train_obj' but qn, fn, and gam have been replaced by centered versions

compute_pfi *Compute permutation feature importance (PFI)*

Description

Function for computing PFI for a given model and dataset (training or testing)

Usage

```
compute_pfi(x, y, f, K, metric, eps = 1e-15)
```

Arguments

x	Dataset with n observations and p variables (training or testing)
y	Response variable (or matrix) associated with x
f	Model to explain
K	Number of repetitions to perform for PFI
metric	Metric used to compute PFI (choose from "accuracy", "logloss", and "nmse")
eps	Log loss is undefined for p = 0 or p = 1, so probabilities are

Value

List containing

- `pfi`: Vector of PFI values (averaged over replicates)
- `pfi_single_reps`: Matrix of containing the feature importance values from each replicate (rows associated with reps; columns associated with data observations)

Examples

```
# Load packages
library(dplyr)
library(tidyr)
library(randomForest)

# Select a subset of functions from shifted peaks data
sub_ids <-
  shifted_peaks$data |>
  select(data, group, id) |>
  distinct() |>
  group_by(data, group) |>
  slice(1:4) |>
  ungroup()

# Create a smaller version of shifted data
shifted_peaks_sub <-
  shifted_peaks$data |>
  filter(id %in% sub_ids$id)

# Extract times
shifted_peaks_times = unique(shifted_peaks_sub$t)

# Convert training data to matrix
shifted_peaks_train_matrix <-
  shifted_peaks_sub |>
  filter(data == "Training") |>
  select(-t) |>
  mutate(index = paste0("t", index)) |>
  pivot_wider(names_from = index, values_from = y) |>
  select(-data, -id, -group) |>
  as.matrix() |>
  t()

# Obtain veesa pipeline training data
veesa_train <-
  prep_training_data(
    f = shifted_peaks_train_matrix,
    time = shifted_peaks_times,
    fpca_method = "jfpca"
  )

# Obtain response variable values
model_output <-
```

```
  shifted_peaks_sub |>
  filter(data == "Training") |>
  select(id, group) |>
  distinct()

# Prepare data for model
model_data <-
  veesa_train$fpca_res$coef |>
  data.frame() |>
  mutate(group = factor(model_output$group))

# Train model
set.seed(20210301)
rf <-
  randomForest(
    formula = group ~ .,
    data = model_data
  )

# Compute feature importance values
pfi <-
  compute_pfi(
    x = model_data |> select(-group),
    y = model_data$group,
    f = rf,
    K = 1,
    metric = "accuracy"
 )
```

`plot_pc_directions` *Plot principal component directions*

Description

Function for plotting the functional PC directions

Usage

```
plot_pc_directions(
  fpcs,
  fdasrvf,
  fpca_method,
  times = NULL,
  digits = 0,
  alpha = 1,
  nrow = 1,
  linesizes = NULL,
  linetype = TRUE,
  freey = FALSE
)
```

Arguments

fpcs	Vector of numbers identifying the PCs to include in the plot
fdasrvf	Object output from jointFPCA, horizFPCA, or vertFPCA
fpca_method	Character string specifying the type of elastic fPCA method to use ('jfpca', 'hfpca', or 'vfpca')
times	Optional vector of times (if not included, times will be represented on the interval from 0 to 1)
digits	Number of digits to print in the title for the proportion of variability explained by a PC
alpha	Vector of alpha values associated with lines in plot (length must match number of lines in plot)
nrow	Number of rows to use when creating a grid of plots
linesizes	Vector of line widths associated with lines in plot (length must match number of lines in plot)
linetype	Vector of line types (e.g., "solid" or "dashed") associated with lines in plot (length must match number of lines in plot)
freey	Indicator for whether y-axis should be freed across facets

Value

ggplot2 plot of specified principal component directions

Examples

```
# Load packages
library(dplyr)
library(tidyr)

# Select a subset of functions from shifted peaks data
sub_ids <-
  shifted_peaks$data |>
  select(data, group, id) |>
  distinct() |>
  group_by(data, group) |>
  slice(1:4) |>
  ungroup()

# Create a smaller version of shifted data
shifted_peaks_sub <-
  shifted_peaks$data |>
  filter(id %in% sub_ids$id)

# Extract times
shifted_peaks_times = unique(shifted_peaks_sub$t)

# Convert training data to matrix
shifted_peaks_train_matrix <-
  shifted_peaks_sub |>
```

```
  filter(data == "Training") |>
  select(-t) |>
  mutate(index = paste0("t", index)) |>
  pivot_wider(names_from = index, values_from = y) |>
  select(-data, -id, -group) |>
  as.matrix() |>
  t()

# Obtain veesa pipeline training data
veesa_train <-
  prep_training_data(
    f = shifted_peaks_train_matrix,
    time = shifted_peaks_times,
    fpca_method = "jfpca"
  )

# Plot principal directions of PC1
plot_pc_directions(
  fpcs = 1,
  fdasrvf = veesa_train$fpca_res,
  fpca_method = "jfpca",
  times = -shifted_peaks_times,
  linesizes = rep(0.75,5),
  alpha = 0.9
 )
```

`prep_testing_data` *Align test data and apply fPCA using elastic method applied to training data*

Description

Applies steps 2 and 3 of the VEESA pipeline (alignment and elastic fPCA (jfpca, hfpca, or vfpca)) to the testing data based on the training data prepared using "prep_training_data".

Usage

```
prep_testing_data(f, time, train_prep, optim_method = "DP")
```

Arguments

`f`	Matrix (size M x N) of test data with N functions and M samples.
`time`	Vector of size M describing the sample points
`train_prep`	Object returned from applying "prep_training_data" to training data.
`optim_method`	Method used for optimization when computing the Karcher mean. "DP", "DPo", and "RBFGS".

Value

List containing (varies slightly based on fpca method used):

- time: vector of times when functions are observed (length of M)
- f0: original test data functions - matrix (M x N) of N functions with M samples
- fn: aligned test data functions - similar structure to f0
- q0: original test data SRSFs - similar structure to f0
- qn: aligned test data SRSFs - similar structure to f0
- mqn: training data SRSF mean (test data functions are aligned to this function)
- gam: test data warping functions - similar structure to f0
- coef: test data principal component coefficients
- psi: test data warping function SRVFs - similar structure to f0 (jfpca and hfpca only)
- nu: test data shooting functions - similar structure to f0 (jfpca and hfpca only)
- g: test data combination of aligned and shooting functions (jfpca only)

Examples

```
# Load packages
library(dplyr)
library(tidyr)

# Select a subset of functions from shifted peaks data
sub_ids <-
  shifted_peaks$data |>
  select(data, group, id) |>
  distinct() |>
  group_by(data, group) |>
  slice(1:4) |>
  ungroup()

# Create a smaller version of shifted data
shifted_peaks_sub <-
  shifted_peaks$data |>
  filter(id %in% sub_ids$id)

# Extract times
shifted_peaks_times = unique(shifted_peaks_sub$t)

# Convert training data to matrix
shifted_peaks_train_matrix <-
  shifted_peaks_sub |>
  filter(data == "Training") |>
  select(-t) |>
  mutate(index = paste0("t", index)) |>
  pivot_wider(names_from = index, values_from = y) |>
  select(-data, -id, -group) |>
  as.matrix() |>
  t()
```

```
# Obtain veesa pipeline training data
veesa_train <-
  prep_training_data(
    f = shifted_peaks_train_matrix,
    time = shifted_peaks_times,
    fpca_method = "jfpca"
  )

# Convert testing data to matrix
shifted_peaks_test_matrix <-
  shifted_peaks_sub |>
  filter(data == "Testing") |>
  select(-t) |>
  mutate(index = paste0("t", index)) |>
  pivot_wider(names_from = index, values_from = y) |>
  select(-data, -id, -group) |>
  as.matrix() |>
  t()

# Obtain veesa pipeline testing data
veesa_test <- prep_testing_data(
  f = shifted_peaks_test_matrix,
  time = shifted_peaks_times,
  train_prep = veesa_train,
  optim_method = "DP"
 )
```

`prep_training_data` *Align training data and apply a method of elastic fPCA*

Description

Applies steps 2 and 3 of the VEESA pipeline (alignment and elastic fPCA) to the training data in preparation for inputting the data to the model in step 4.

Usage

```
prep_training_data(
  f,
  time,
  fpca_method,
  lambda = 0,
  penalty_method = c("roughness", "geodesic", "norm"),
  centroid_type = c("mean", "median"),
  center_warpings = TRUE,
  parallel = FALSE,
  cores = -1,
  optim_method = c("DP", "DPo", "DP2", "RBFGS"),
```

```
  max_iter = 20L,
  id = NULL,
  C = NULL,
  ci = c(-2, -1, 0, 1, 2)
)
```

Arguments

f	Matrix (size M x N) of training data with N functions and M samples.
time	Vector of size M corresponding to the M sample points.
fpca_method	Character string specifying the type of elastic fPCA method to use. Options are 'jfpca', 'hfpca', or 'vfpca'.
lambda	Numeric value specifying the elasticity. Default is 0.
penalty_method	String specifying the penalty term used in the formulation of the cost function to minimize for alignment. Choices are "roughness" which uses the norm of the second derivative, "geodesic" which uses the geodesic distance to the identity and "norm" which uses the Euclidean distance to the identity. Defaults is "roughness".
centroid_type	String specifying the type of centroid to align to. Options are "mean" or "median". Defaults is "mean".
center_warpings	Boolean specifying whether to center the estimated warping functions. Defaults is TRUE.
parallel	Boolean specifying whether to run calculations in parallel. Defaults is FALSE.
cores	Integer specifying the number of cores in parallel. Default is -1, which uses all cores.
optim_method	Method used for optimization when computing the Karcher mean. Options are "DP", "DPo", and "RBFGS".
max_iter	An integer value specifying the maximum number of iterations. Defaults to 20L.
id	Integration point for f0. Default is midpoint.
C	Balance value. Default = NULL.
ci	Geodesic standard deviations to be computed. Default is c(-2, -1, 0, 1, 2).

Value

List with three objects:

- alignment: output from fdasrvf::time_warping
- fpca_type: type of elastic FPCA method applied
- fpca_res: output from fdasrvf::jointFPCA, fdasrvf::horizFPCA, or fdasrvf::vertFPCA (dependent on fpca_type)

Examples

```
# Load packages
library(dplyr)
library(tidyr)

# Select a subset of functions from shifted peaks data
sub_ids <-
  shifted_peaks$data |>
  select(data, group, id) |>
  distinct() |>
  group_by(data, group) |>
  slice(1:4) |>
  ungroup()

# Create a smaller version of shifted data
shifted_peaks_sub <-
  shifted_peaks$data |>
  filter(id %in% sub_ids$id)

# Extract times
shifted_peaks_times = unique(shifted_peaks_sub$t)

# Convert training data to matrix
shifted_peaks_train_matrix <-
  shifted_peaks_sub |>
  filter(data == "Training") |>
  select(-t) |>
  mutate(index = paste0("t", index)) |>
  pivot_wider(names_from = index, values_from = y) |>
  select(-data, -id, -group) |>
  as.matrix() |>
  t()

# Obtain veesa pipeline training data
veesa_train <-
  prep_training_data(
    f = shifted_peaks_train_matrix,
    time = shifted_peaks_times,
    fpca_method = "jfpca"
  )
```

`shifted_peaks` *"Shifted Peaks" Simulated Dataset*

Description

A simulated dataset generated for examples in the veesa pipeline manuscript. For the code used to prepare this dataset, see https://github.com/sandialabs/veesa/inst/data-shifted-peaks.md.

Usage

```
shifted_peaks
```

Format

A list.

Details

The objects in the list are:

`data`	Data frame containing simulated data
`params`	The parameters used to generate the data
`true_means`	The true functional means of the shifted peaks groups.

`simulate_functions` *Simulate example functional data*

Description

Function for simulating a set of functional data based on a deterministic function with covariates that affect the shape of the functions

Usage

```
simulate_functions(M, N, seed)
```

Arguments

`M`	Number of functions
`N`	Number of samples per function
`seed`	Seed for reproducibility

Details

The functions are generated using the following equation:

f(t) = (x_1*exp(-((t-0.3)^2)/0.005)) + (x_2(-((t-(0.7+x_3))^2/0.005)))

where the covariates are generated as follows:

- x_1 generated from Unif(0.1,1)
- x_2 generated from Unif(0.1,0.5)
- x_3 generated from Unif(-0.1,0.1)

Value

Data frame with the following columns (where f is the function):

- t: "time" associated with sample from function where t in [0,1]
- y: f(t) for the particular observation
- x1: covariate 1 for function f (constant across time)
- x2: covariate 2 for function f (constant across time)
- x3: covariate 3 for function f (constant across time)

Examples

```
# Simulate data
sim_data = simulate_functions(M = 100, N = 75, seed = 20211130)
```

Index