2DFPCA (Two-Dimensional Functional Principal Component Analysis for Image Feature Extraction)

R codes for implementing the two-dimensional functional principal component analysis for image feature extraction.

Description

Methodologies for functional principal component analysis are well established in the one-dimensional setting. However, for two-dimensional surfaces, e.g., images, conducting functional principal component analysis is complicated and challenging, because the conventional eigendecomposition approach would require the estimation of a four-dimensional covariance function, which may incur high cost in terms of time and machine memory. To circumvent such computational difficulties, we propose a novel two-dimensional functional principal component analysis for extracting functional principal components and achieving dimensionality reduction for images. Different from the conventional eigendecomposition approach, our proposed method is based on the direct estimation of the optimal two-dimensional functional principal components via tensor product B-spline, which opens up a new avenue for estimating functional principal components.

The repository includes two functions:

- func_2DFPCA.R: The R source code for fitting the two-dimensional FPCA.

```
first_FPC_2d_image(beta1, observed, timepoints1, timepoints2,
basis1, basis2, threshold, minit)
second_FPC_conditional_2d_image(beta1, pc_index, observed, timepoints1, timepoints2,
basis1, basis2, betalist, threshold)
```

- demo.R: A demo script for fitting two-dimensional FPCA on the MNIST data of digit 0.
- demo_simu.R: The MNIST handwritten digits data set.
- Mnist.RData: A demo script for a simulation study of the two-dimensional FPCA.
- fpc_0and1x.RData: The first three FPCs for the simulation study.

Inputs

- `beta1` : The initialization matrix of spline coefficients,e.g., a matrix of 0.
- `observed` : A list of observed grayscale values for all the subjects.

- `timepoints1` : A list of time points (or indices for pixels in an image) on the first dimension for all the subjects.
- `timepoints2` : A list of time points (or indices for pixels in an image) on the second dimension for all the subjects..
- `basis1` : The basis object for the first dimension in the tensor product B-spline.
- `basis2` : The basis object for the first dimension in the tensor product B-spline.
- `threshold` : The stopping boundary for the absolute maximum change in value of the spline coefficients between iterations. The default value is 0.0001.
- `minit` : The minimum number of iterations. The default value is 1.
- `pc_index` : The index of the FPC. Used for the function second_FPC_conditional_2d_image() only.
- `betalist` : The list of the fitted spline coefficients for the lower degree FPCs computed from the previous steps. Used for the function second_FPC_conditional_2d_image() only.

Example

We apply the two-dimensional FPCA method to all the images with digit 0 in the MNIST data set. The code depends on R packages `fda` , `dplyr` and `lsei` .

```
load("Mnist.RData")
source("funs_2DFPCA.R")

library(dplyr)
library(fda)
library(lsei)

fulldata <- observed
selectindex <- which(label == 0)
observed <- fulldata[selectindex]
```

The data set contains the list of vectorized observed pixel values of the images (28 by 28) in the MNIST data set. We first centralize the observed pixel value by removing the mean of the pixels in all the subjects.

```
meanobserve <- Reduce("+", observed)/length(selectindex)
observed <- lapply(observed, function(x) {
  x-meanobserve
})
timepoints1 <- timepoints1[selectindex]
timepoints2 <- timepoints2[selectindex]
```

We use a tensor product B spline basis (12 by 12), and set the initialization matrix with values 0.01 for the spline coefficients. We then create the two underlying cubic splines in the tensor product B spline using function `create.bspline.basis` .

```
maxpixelsize <- 28
nbasis <- 12
library(fda)
beta1 <- matrix(1, nrow = nbasis, ncol = nbasis)
for (i in 1:nbasis) {
  beta1[, i] <- seq(0.01, 0.12, by = 0.01)
}

basis1 <- create.bspline.basis(rangeval = c(1, maxpixelsize), nbasis = nbasis, norder = 4)
basis2 <- create.bspline.basis(rangeval = c(1, maxpixelsize), nbasis = nbasis, norder = 4)

initializeGlobalXmat(timepoints1, timepoints2, basis1, basis2)
```

To fit the first FPC, we use the function `first_FPC_2d_image` and store the fitted spline coefficients in a list named `previous_beta` (to be used as input the fitting the next FPC).

```
previous_beta <- list()

res_first <- first_FPC_2d_image(beta1, observed, timepoints1, timepoints2,
basis1, basis2, threshold = 1e-4)

previous_beta[[1]] <- res_first$beta
```

After fitting the first FPC, we fit the second and third FPC using the function `second_FPC_conditional_2d_image` and store the fitted spline coefficients in the list named `previous_beta` (to be used as input the fitting the next FPC).

```
for (i in i:3) {
  res_second <- second_FPC_conditional_2d_image(beta1, pc_index = i, observed,
  timepoints1, timepoints2, basis1, basis2, betalist = previous_beta, threshold = 1e-4)

  previous_beta[[i]] <- res_second$beta
}
```

Authors and Reference

- Haolun Shi, Yuping Yang, Liangliang Wang, Da Ma, Mirza Faisal Beg, Jian Pei and Jiguo Cao
- Shi, H., Yang, Y., Wang, L., Ma, D., Beg, M. F., Pei, J., Cao, J. (2021) Two-Dimensional Functional Principal Component Analysis for Image Feature Extraction.