Package FPOPapprox2D

Description

FPOPapprox2D is an R package developed in Rcpp/C++ performing parametric changepoint detection in p-variate time series with the Gaussian cost function. Changepoint detection uses the Functional Pruning Optimal Partitioning method (FPOP) based on an exact dynamic programming algorithm.

Package structure

- Part R
  - FPOPapprox2D.R
    The file contains the implementation of the following function:
    data_genDp - generation of data of dimension p with a given values of means and changepoints
  - RcppExports.R
    The file contains the implementation of the function **FPOPDp** that calls the function **FPOPDp** implemented in C++.
- Part C++
  - GausseCostDp.h, GausseCostDp.cpp
    The files contain the implementation code for the class**GausseCostDp**.
  - DiskDp.h, DiskDp.cpp
    The files contain the implementation code for the class **DiskDp**.
  - RectDp.h, RectDp.cpp
    The files contain the implementation code for the class**RectDp**.
  - Geom1Dp.h, Geom1Dp.cpp
    The files contain the implementation code for the class **Geom1Dp**.
  - Geom2Dp.h, Geom2Dp.cpp
    The files contain the implementation code for the class **Geom2Dp**.
  - Geom3Dp.h, Geom3Dp.cpp
    The files contain the implementation code for the class **Geom3Dp**.
  - OPDp.h, OPDp.cpp
    The files contain the implementation code for the class **OPDp**.
  - main.cpp
    The file contains the code of the function **FPOPDp** that implements the change-point detection in p-variate time-series using the Functional Pruning Optimal Partitioning algorithm.
  - RcppExports.cpp
    The file contains the code that exports data R/C ++ .

Class GausseCostDp

We consider $(x^0, .., x^p)$ - p-variate time-series when $x^i = (x^i_0, .., x^i_{n-1}), \quad i = 0 : (p-1)$ - the vectors of univariate data size n.
We use the Gaussian cost of the segmented p-variate data when m_t is the value of the optimal cost, $m_0 = -\beta$. We introduce the notations:

$$\begin{aligned} mu_k &= E(x^k_{i:t}), \\ coef &= (t - i + 1), \\ mi\_1\_p &= m_{i-1} + \beta, \\ coef\_Var &= (t - i + 1) \cdot \sum_{k=0}^{p-1} Var(x^k_{i:t}). \end{aligned} \tag{1}$$

Then the Gaussian cost function takes the form:

$$q^i_t(\theta) = mi\_1\_p + coef \cdot (sum^{p-1}_{k=0}(\theta_k - mu - k)^2 + coef\_Var, \quad i = 1 : t. \tag{2}$$

We define the class characteristics (1):

- **coef**, **coef_Var**, **mi_1_p**, **mu**

The class implements constructor:

- **GausseCostDp(unsigned int dim, unsigned int i, unsigned int t, double* si_1, double* st, double mi_1pen)** is the Gaussian cost function at the time $[i, t]$.

We define the class methods:

- **get_coef()**, **get_coef_Var()**, **get_mi_1_p()**, **get_mu()**

  We use these methods to access the characteristics of the class.

- **get_min()**

  We use this method to get the minimum value of the cost function.

Class DiskDp

A class element is a circle in p-dimension that is defined by the center coordinates and the radius.

- We define the class characteristics:
  - **center** is the center of the circle.
  - **radius** is the radius of the circle.
- The class implements two constructors:
  - **DiskDp(unsigned int dim)**
  - **DiskDp(unsigned int dim, double* c, double r)**
    Vector c is the center coordinates and r is the radius of the circle.
- We define the class methods:
  - **get_center()**, **get_radius()**
    We use these methods to access the class characteristics.

Class RectDp

A class element is a rectangle in p-dimension. The values for each axis are set by two constraints

- We define the class characteristics:
  - **p, coordinates**
- The class implements two constructors:
  - **RectDp(unsigned int dim)**:
    - $p = dim$;
    - $coordinates_{i,0} = -\inf, \quad i = 0 : p-1$;
    - $coordinates_{i,1} = \inf, \quad i = 0 : p-1$.
  - **Rect(unsigned int dim,double** coords)**
- We define the class methods:
  - **get_p()**, **get_coordinates()**
    We use these methods to access the class characteristics.
  - **min_ab(double a, double b)**, **max_ab(double a, double b)**
    We use these methods to find the minimum and maximum of two numbers.
  - **point_max_min(double* pnt_max, double* pnt_min, DiskDp disk)**
    The function changes the values of *pnt_max*, *pnt_min* to the values of the vertices of the rectangle that are maximally and minimally distant from the center of the *disk*.
  - **bool IsEmpty_rect()**
    The function checks the correctness of the coordinates of the rectangle and returns a value *true* if the coordinates are not correct.
  - **Intersection_disk(DiskDp disk)**
    The function approximates a rectangle and a circle intersection area by horizontal and vertical lines. Basing on the intersection points of these lines, we construct a rectangle with a minimum area, which contains the intersection area of the rectangle and the circle.
  - **Exclusion_disk(DiskDp disk)**
    The function approximates a rectangle and a circle difference area by horizontal and vertical lines. Basing on the intersection points of these lines, we construct a rectangle with a minimum area, which contains the difference area of the rectangle and the circle.

Class Geom1Dp

- We define the class characteristics:
  - **p** is the value of dimension.
  - **label_t** is the time moment.
  - **rect_t** is the rectangle, the element of class **RectDp**.
- The class implements constructor:
  - **Geom1Dp(unsigned int dim, unsigned int t)**
    - $p = dim$;
    - $label\_t = t$;
    - $rect\_t = RectDp(dim)$.

- We define the class methods:
  - **get_p()**, **get_label_t()**, **get_rect_t()**
  - **InitialGeometry(std::list<DiskDp> disks)** is empty function body.
  - **UpdateGeometry(DiskDp disk)**
    The function calls the function **Intersection_disk(DiskDp disk)** implemented in the RectDp class.
  - **bool EmptyGeometry()**
    The function checks the parameters of the rectangle **rect_t**. If the parameters are not correct, this rectangle is empty.

Class Geom2Dp

- We define the class characteristics:
  - **p** is the value of dimension.
  - **label_t** is the time moment.
  - **rect_t** is the rectangle, the element of class **RectDp**.
  - **disks_t_1** is the list of active circles for the moment $t-1$.
- The class implements constructor:
  - **Geom2Dp(unsigned int dim, unsigned int t)**
    * $p = dim$;
    * $label\_t = t$;
    * $rect\_t = RectDp(dim)$;
    * $disks\_t\_1.clear()$.
- We define the class methods:
  - **get_p()**, **get_label_t()**, **get_rect_t()**, **get_disks_t_1()**
  - **InitialGeometry(std::list<DiskDp> disks)**
    We define $disks\_t\_1$ as $disks$.
  - **UpdateGeometry(DiskDp disk_t)**
    The function calls the function **Intersection_disk(DiskDp diskt)** implemented in the **RectDp** class. If the result rectangle exists it removes from the list $disks\_t\_1$ the disks that are far from the $disk\_t$. Then it calls a function **Exclusion_disk(DiskDp disk)** for each disk from the list $disks\_t\_1$.
  - **bool EmptyGeometry()**
    The function checks the parameters of the rectangle **rect_t**. If the parameters are not correct, this rectangle is empty.

Class Geom3Dp

- We define the class characteristics:
  - **p** is the value of dimension.
  - **label_t** is the time moment.
  - **fl_empty** is $false$ if geometry exists, otherwise $true$.

- **disks_t_1** is the list of active circles for the moment $t-1$.

- The class implements constructor:
  - **Geom3Dp(unsigned int dim, unsigned int t)**
    - $p = dim$;
    - $label\_t = t$;
    - $fl\_empty = false$;
    - $disks\_t\_1.clear()$.

- We define the class methods:
  - **get_p()**, **get_fl_empty()**, **get_label_t()**, **get_disks_t_1()**
  - **InitialGeometry(std::list<DiskDp> disks)** We define $disks\_t\_1$ as $disks$.
  - **UpdateGeometry(DiskDp disk_t)**
    The function removes from the list $disks\_t\_1$ the disks that are far from the $disk\_t$. Then it calls a function **Exclusion_disk(DiskDp disk)** for each disk from the list $disks\_t\_1$.
  - **bool EmptyGeometry()**
    The function checks the parameter **fl_empty**. If the parameter is *true* this geometry is empty.

Class OPDp

- We define the class characteristics:
  - **p** is the value of dimension.
  - **n** is the number of data points.
  - **penalty** is a value of penalty (a non-negative real number).
  - **geom** is the element of **GeomX** class.
  - **list_geom** is the list of **GeomX** class element.
  - **sx12** are sum vectors $\sum_{i=0}^{t-1} x_i^k$, $\sum_{i=0}^{t-1}(x_i^k)^2$, $\quad t = 0 : n-1, \quad k = 0 : p-1$.
  - **chpts** is the vector of changepoints.
  - **means** is the matrrix of successive means for data x.
  - **globalCost** is the global cost.

- The class implements the constructor:
  - **OPDp<GeomX> (Rcpp::NumericMatrix x, double beta)**
    - $penalty = beta$;
    - $p = (unsignedint)x.nrow()$;
    - $n = (unsignedint)x.ncol()$;
    - we allocate memory for $sx12$.

- We define the class methods:
  - **get_chpts()**, **get_means()**, **get_p()**, **get_globalCost()**, **get_n()**, **get_penalty()**
    We use these methods to access the class characteristics.

– **vect_sx12(Rcpp::NumericMatrix x)**
We use this method to find the sum vectors:
$\sum_{i=0}^{t-1} x_i^k, \quad \sum_{i=0}^{t-1}(x_i^k)^2, \quad t = 0 : n-1, \quad k = 0 : p-1.$
– **algoFPOP(Rcpp::NumericMatrix x, int type, bool test_mode)**
The function implements the Functional Pruning Optimal Partitioning algorithm with 3 types of geometries:
$type = 1 : Geom1Dp$,
$type = 2 : Geom2Dp$,
$type = 3 : Geom3Dp$.

void RectDp::Intersection_disk(DiskDp disk)

Description

The function approximates a rectangle and a circle intersection area by horizontal and vertical lines. Basing on the intersection points of these lines, we construct a rectangle with a minimum area, which contains the intersection area of the rectangle and the circle.
If there is no intersection, the function makes at least one of the rectangle parameters satisfies the condition:

$$coordinates_{i,0} \geq coordinates_{i,1}, \quad i = 0 : p-1. \tag{3}$$

Input parameters:

The input of this function:

- **disk** is the circle, the element of class **DiskDp**

Algorithm:

Preprocessing

Using the methods **get_center()**, **get_radius()** implemented in the class **DiskDp** we define the parameters of the circle **disk**:

$$\begin{aligned} c &= disk.get\_center(), \\ r &= disk.get\_radius(). \end{aligned} \tag{4}$$

We define the points pnt_max and pnt_min and we update them using the function $point\_max\_min(pnt\_max, pnt\_min, disk$

$$\begin{aligned} pnt\_max &= newdouble[p]; \\ pnt\_min &= newdouble[p]; \\ point\_max\_min(pnt\_max, pnt\_min, disk); \end{aligned} \tag{5}$$

Approximation

For each axis we update two constraints: $coordinates_{k,0}$, $coordinates_{k,1}, \quad k = 0, \quad p-1.$

- If for all $j = 0 : p-1$ and $j \neq k$ $coordinates_{j,0} \leq c_j \leq coordinates_{j,1}$ then:

$$\begin{aligned} coordinates_{k,0} &= \max\{coordinates_{k,0}, c_k - r\}, \\ coordinates_{k,1} &= \min\{coordinates_{k,1}, c_k + r\}. \end{aligned} \tag{6}$$

Otherwise, we consider the points of intersection of the circle with $p-1$-planes, when we fixe coordinates $x_i = pnt\_min_i$, and $x_i = pnt\_max_i, \quad i \neq k$. The circle has two intersection points if the discriminant is positive. We define the values db_4, dt_4 (9) as the value of a discriminant devided by 4 of each system (7, 8):

$$\begin{cases} (x_k - c_k)^2 + \sum_{i=0, i\neq k}^{p-1} (x_i - c_i)^2 = r^2, \\ x_i = pnt\_min_i, \quad i \neq k. \end{cases} \tag{7}$$

$$\begin{cases} (x_k - c_k)^2 + \sum_{i=0, i\neq k}^{p-1} (x_i - c_i)^2 = r^2, \\ x_i = pnt\_max_i, \quad i \neq k. \end{cases} \tag{8}$$

$$\begin{aligned} db\_4 &= r^2 - \sum_{i=0, i\neq k}^{p-1} (pnt\_min_i - c_i)^2, \\ dt\_4 &= r^2 - \sum_{i=0, i\neq k}^{p-1} (pnt\_max_i - c_i)^2, \end{aligned} \tag{9}$$

Note: we define the default intersection points for the algorithm to work correctly as:

$$\begin{aligned} b1 = t1 &= \infty, \\ b2 = t2 &= -\infty. \end{aligned} \tag{10}$$

We check the sign of db_4, dt_4 and find the intersection points:

$$\begin{cases} db\_4 > 0, \\ b1 = c_k - \sqrt{db\_4}, \\ b2 = c_k + \sqrt{db\_4}. \end{cases} \tag{11}$$

$$\begin{cases} dt\_4 > 0, \\ t1 = c_k - \sqrt{dt\_4}, \\ t2 = c_k + \sqrt{dt\_4}. \end{cases} \tag{12}$$

We define the characteristics of rectangle as:

$$\begin{aligned} coordinates_{k,0} &= \max\{coordinates_{k,0}, \min\{b1, t1\}\}, \\ coordinates_{k,1} &= \min\{coordinates_{k,1}, \max\{b2, t2\}\}. \end{aligned} \tag{13}$$

- If all the values db_4, dt_4 for all axis are non-positive, the rectangle has no intersections with the circle and we define the characteristics of rectangle as:

$$coordinates_{0,0} = coordinates_{0,1}. \tag{14}$$

bool RectDp::IsEmpty_rect()

Description

The function checks the parameters of the rectangle. If the parameters are not correct, this rectangle is empty.

Output parameters:

The function returns a boolean value **true** if the rectangle is empty, and **false** if it is not empty.

Algorithm:

If at least one of the rectangle parameters satisfies the condition (3) this rectangle is empty and the function returns a boolean value **true**, else **false**.

void RectDp::Exclusion_disk(DiskDp disk)

Description

The function approximates a rectangle and a circle difference area by horizontal and vertical lines. Basing on the intersection points of these lines, we construct a rectangle with a minimum area, which contains the difference area of the rectangle and the circle.
If the difference is the empty set, the function makes the rectangle with parameters that correspond to the condition (3).

Input parameters:

The input of this function consists:

- **disk** is the circle, the element of class **DiskDp**

Algorithm:

Preprocessing

We define:

- the parameters of the circle **disk** (4),
- We define the points pnt_max and pnt_min and we update them using the function $point\_max\_min(pnt\_max, pnt\_min$ (5)

Approximation

For each axis we update two constraints: $coordinates_{k,0}$, $coordinates_{k,1}$, $k = 0, \quad p - 1$.
We consider two directions and update the characteristics of rectangle:

- We consider the points of intersection of the circle with $p-1$-planes, when we fixe coordinates $x_i = pnt\_min_i$ and $x_i = pnt\_max_i$, $i \neq k$. The circle has two intersection points if the discriminant is positive. We define the values db_4, dt_4 (9) as the value of a discriminant devided by 4 of each system (7, 8)

  We update the characteristics $coordinates_{k,0}, coordinates_{k,1}$ if db_4, dt_4 (9) are positive.

  Note: we define the default intersection points for the algorithm to work correctly as (10).

We check the sign of db_4, dt_4 and find the intersection points (11) and (12).

We define the characteristics of rectangle as:

$$\begin{aligned} coordinates_{k,0} &= \max\{coordinates_{k,0}, \min\{b2, t2\}\}, \\ coordinates_{k,1} &= \min\{coordinates_{k,1}, \max\{b1, t1\}\}. \end{aligned} \tag{15}$$

void OPDp<X>::algoFPOP(Rcpp::NumericMatrix x, int type, bool test_mode)

Description

The function implements FPOP method.

Input parameters:

- **x** is the matrix of data.
- **type** is the value defined the type of geometry.

Output parameters:

The function forms the vectors **chpts**, list of **means** and the value of **globalCost**.

Algorithm:

Preprocessing

- We define:
  - $sx12 = vect\_sx12(x)$.
  - $m[0] = 0$.
- We allocate memory for:
  - the vector $last\_chpt$ of best last changepoints.
  - the matrix $last\_mean$.
  - the vector m is the value of the sum optimal cost and penalty at the moment t, $t = 0 : n-1$.
  - the vector mus is the values of temporary means.

Processing

For each $t = 0 : n - 1$ we do:

- We define:
  - $list\_disk$ is a list of active disks for $t - 1$ as $NULL$.
  - $cost$ is the cost function for the interval (t, t).
  - $min\_val = cost.get\_min()$ is a minimum value for the $cost$.
  - $lbl = t$ is a best last position for t.
  - $mus = cost.get\_mu()$ vector temporary means of the interval (lbl, t).

- The first run: Searching of $m[t+1]$

  For each element of the list $list\_geom$ we do:

  - We define u is the $label\_t$ of the current element.
  - We define the list of active disks $list\_disk$ for $t-1$ using the cost function $cost\_t\_1$ for the interval $(u, t-1)$.
  - We create $cost$ is the cost function for the interval (u, t).
  - We find the minimum value min_val of the cost function using the method **get_min()**.

- We choose the minimum among all found values min_val and we define the value lbl that correspond this minimum.
- We put the value $min\_val + penalty$ to the vector m by the position $t+1$.
- We put the values lbl, mus that corresponds $m[t+1]$ to the vector $last\_chpt$ and matrix $last\_mean$.
- New geometry

  We create $geom$ for the point $(x_t^0, .., x_t^{p-1})$ of the p-variate time series and do the initialization of geometry parameters as (16) and we add this element to the list $list\_geom$.

$$\begin{aligned} geom &= GeomX(p, t), \\ geom.InitialGeometry(list\_disk); \end{aligned} \tag{16}$$

- The second run: Pruning

  For each element of the list $list\_geom$ we do:

  - We define $label\_t$ of the current element as lbl .
  - We calculate the cost function for the interval (lbl, t) as 17 and $r2$ is the radius to the second power of the new disk as 18.

$$cost = GausseCostDp(p, lbl, t, sy12[lbl], sy12[t+1], m[lbl]); \tag{17}$$

$$r2 = \frac{m[t+1] - m[lbl] - cost.get\_coef\_Var()}{cost.get\_coef()}. \tag{18}$$

  - PELT-pruning:

    If $r2 \leq 0$ we remove this element of $list\_geom$, else we define $disk\_lblt = DiskDp(cost.get\_mu(), \sqrt{r2})$.
  - FPOP-pruning:

    We update the geometry $geom$ using the function $UpdateGeometry(disk\_lblt)$. If $geom$ is empty we remove this element o $list\_geom$.

Output:

Knowing the values of vector $last\_chpt$, the matrix $last\_mean$ and vector m we forme the vector of $chpts$, the list of $means$ and the value $globalCost$.