

Package ‘LOPART’

May 7, 2026

Type Package

Title Labeled Optimal Partitioning

Version 2024.6.19

Author Toby Dylan Hocking

Maintainer Toby Dylan Hocking <toby.hocking@r-project.org>

Description Change-point detection algorithm with label constraints and a penalty for each change outside of labels. Read TD Hocking, A Srivastava (2023) <[doi:10.1007/s00180-022-01238-z](https://doi.org/10.1007/s00180-022-01238-z)> for details.

License GPL-3

LinkingTo Rcpp

URL <https://github.com/tdhock/LOPART>

BugReports <https://github.com/tdhock/LOPART/issues>

Imports data.table, Rcpp

Suggests ggplot2, testthat

RoxygenNote 7.3.1

NeedsCompilation yes

Repository CRAN

Date/Publication 2024-06-20 21:30:07 UTC

Contents

LOPART	2
LOPART_interface	6
Index	8

 LOPART

Labeled Optimal PARTitioning

Description

Compute an optimal segmentation (change in Gaussian mean model, square loss), which is consistent with the given labels, and with a penalty for each changepoint outside of labeled regions.

Usage

```
LOPART(
  x,
  labels,
  penalty_unlabeled,
  n_updates = length(x),
  penalty_labeled = penalty_unlabeled
)
```

Arguments

<code>x</code>	numeric vector of data to fit a Gaussian mean model.
<code>labels</code>	data frame with at least three columns: start, end, changes. start/end should be indices of <code>x</code> , from 1 to <code>length(x)</code> . changes should be either 0 or 1. The predicted changepoints are guaranteed to be consistent with these labels.
<code>penalty_unlabeled</code>	non-negative penalty constant (larger for fewer changes, smaller for more changes). <code>penalty=0</code> means a change in every unlabeled region, <code>penalty=Inf</code> means no changes in unlabeled regions.
<code>n_updates</code>	how many dynamic programming updates to compute? Must be at least 1 and at most <code>length(x)</code> .
<code>penalty_labeled</code>	non-negative penalty constant to use for changes in positive labels.

Details

Provides a high-level interface to `LOPART_interface` R function and `LOPART` C code.

Value

list with named elements, all of which are data tables. `loss` has one row with loss/cost values. `cost` is the output from `LOPART_interface`. `changes` has one row for each predicted changepoint (e.g. `change=1.5` means a change between data points 1 and 2). `segments` has one row for each segment.

Author(s)

Toby Dylan Hocking

Examples

```

set.seed(2)
library(data.table)
signal <- c(
  rnorm(25, mean = 10),
  rnorm(25, mean = 7),
  rnorm(25, mean = 8),
  rnorm(25, mean = 5))
#outliers
signal[86] <- 10
labels.dt <- data.table(
  start = c(20, 45, 80),
  end = c(30, 55, 90),
  changes = c(1, 1, 0))
signal.dt <- data.table(
  signal,
  position=seq_along(signal))
label.colors <- c(
  "1"="#ff7d7d",
  "0"="#f6c48f")
sig.color <- "grey50"
if(require(ggplot2)){
  gg.data <- ggplot()+
    geom_rect(aes(
      xmin=start, xmax=end,
      fill=paste(changes),
      ymin=-Inf, ymax=Inf),
      alpha=0.5,
      data=labels.dt)+
    geom_point(aes(
      position, signal),
      color=sig.color,
      data=signal.dt)+
    scale_x_continuous(
      "position",
      breaks=seq(0, 100, by=10))+
    scale_fill_manual("label", values=label.colors)+
    theme_bw()+
    theme(panel.spacing=grid::unit(0, "lines"))
  print(gg.data)
}

label.list <- list(
  OPART=labels.dt[0],
  LOPART=labels.dt)
seg.dt.list <- list()
change.dt.list <- list()
cost.dt.list <- list()
for(model.name in names(label.list)){
  label.dt <- data.table(label.list[[model.name]])
  fit <- LOPART::LOPART(signal, label.dt, 10)
  Algorithm <- factor(model.name, names(label.list))
}

```

```

tau.dt <- fit$cost[, .(
  cost_candidates,
  tau=0:(.N-1),
  change=seq_along(cost_candidates)-0.5
)]
cost.dt.list[[model.name]] <- data.table(Algorithm, tau.dt)
seg.dt.list[[model.name]] <- data.table(Algorithm, fit$segments)
change.dt.list[[model.name]] <- data.table(Algorithm, fit$changes)
}
seg.dt <- do.call(rbind, seg.dt.list)
change.dt <- do.call(rbind, change.dt.list)
cost.dt <- do.call(rbind, cost.dt.list)

algo.sizes <- c(
  OPART=1,
  LOPART=0.5)
algo.colors <- c(
  OPART="deepskyblue",
  LOPART="black")
algo.shapes <- c(
  OPART=1,
  LOPART=2)
if(require(ggplot2)){
  gg.data+
  scale_size_manual(values=algo.sizes)+
  scale_color_manual(values=algo.colors)+
  geom_vline(aes(
    xintercept=change,
    size=Algorithm,
    color=Algorithm),
    data=change.dt)+
  geom_segment(aes(
    start=0.5, mean,
    size=Algorithm,
    color=Algorithm,
    xend=end+0.5, yend=mean),
    data=seg.dt)
}

if(require(ggplot2)){
  ggplot()+
  geom_rect(aes(
    xmin=start, xmax=end,
    fill=paste(changes),
    ymin=-Inf, ymax=Inf),
    alpha=0.5,
    data=labels.dt)+
  scale_fill_manual("label", values=label.colors)+
  theme_bw()+
  theme(panel.spacing=grid::unit(0, "lines"))+
  scale_x_continuous(
    "position",
    breaks=seq(0, 100, by=10))+

```

```

    geom_point(aes(
      change, cost_candidates,
      color=Algorithm, shape=Algorithm),
      data=cost.dt)+
    scale_color_manual(values=algo.colors)+
    scale_shape_manual(values=algo.shapes)
  }

abbrev.vec <- c(
  data="data and models",
  cost="cost of last change")
yfac <- function(l){
  factor(abbrev.vec[[l]], abbrev.vec)
}
COST <- function(dt){
  data.table(y.var=yfac("cost"), dt)
}
DATA <- function(dt){
  data.table(y.var=yfac("data"), dt)
}
if(require(ggplot2)){
  ggplot()+
    geom_rect(aes(
      xmin=start, xmax=end,
      fill=paste(changes),
      ymin=-Inf, ymax=Inf),
      alpha=0.5,
      data=labels.dt)+
    scale_fill_manual("label", values=label.colors)+
    theme_bw()+
    theme(panel.spacing=grid::unit(0, "lines"))+
    facet_grid(y.var ~ ., scales="free")+
    geom_vline(aes(
      xintercept=change,
      size=Algorithm,
      color=Algorithm),
      data=change.dt)+
    geom_segment(aes(
      start=0.5, mean,
      size=Algorithm,
      color=Algorithm,
      xend=end+0.5, yend=mean),
      data=DATA(seg.dt))+
    geom_point(aes(
      position, signal),
      color=sig.color,
      shape=1,
      data=DATA(signal.dt))+
    scale_size_manual(values=algo.sizes)+
    scale_color_manual(values=algo.colors)+
    scale_shape_manual(values=algo.shapes)+
    ylab("")+
    scale_x_continuous(

```

```

    "position",
    breaks=seq(0, 100, by=10))+
  geom_point(aes(
    change, cost_candidates,
    color=Algorithm, shape=Algorithm),
    data=COST(cost.dt))
}

```

LOPART_interface

Labeled Optimal Partitioning interface

Description

Low-level interface to LOPART C code

Usage

```

LOPART_interface(
  input_data,
  input_label_start,
  input_label_end,
  input_label_changes,
  n_updates,
  penalty_unlabeled,
  penalty_labeled = 0
)

```

Arguments

`input_data` numeric vector of N data to segment

`input_label_start` integer vector of label start positions in 0, ..., N-2

`input_label_end` integer vector of label end positions in 1, ..., N-1

`input_label_changes` integer vector of 0/1, number of labeled changes

`n_updates` number of dynamic programming updates to perform, usually should be number of `input_data` N, but can be less if you want to analyze/plot the cost/candidates at previous data.

`penalty_unlabeled` non-negative numeric scalar (bigger for fewer changes in unlabeled regions, smaller for more changes)

`penalty_labeled` non-negative numeric scalar (penalty for each change in a positive label).

Details

Avoid using this function and instead use the LOPART function.

Value

data frame with four columns: `cost_candidates` is the cost of each last segment start considered (from 1 to N) for the computation of the optimal cost up to the last data point (Inf means infeasible); `cost_optimal` is the optimal cost vector computed using dynamic programming; `mean` is the last segment mean of the optimal model ending at that data point; `last_change` is the optimal changepoints (negative numbers are not used).

Author(s)

Toby Dylan Hocking

Index

LOPART, [2](#)

LOPART_interface, [6](#)