example_univar

Haotian Xu

9/2/2021

This is a simple guide for offline changepoint detection on univariate mean.

There are 3 methods implemented for univariate mean changepoint detection:

1. *DP.univar*: perform dynamic programming for univariate mean changepoint detection through l0 penalty.
    - *CV.search.DP.univar*: perform grid search to select the tuning parameter through Cross-Validation.
2. *BS.univar*: perform standard binary segmentation for univariate mean changepoint detection.
3. *WBS.univar*: perform wild binary segmentation for univariate mean changepoint detection.

In addition, function *local.refine.univar* performs local refinement for an initial changepoint estimation.

Simulate data

```
library(changepoints)
```

```
## Loading required package: gglasso

## Loading required package: glmnet

## Loading required package: Matrix

## Loaded glmnet 4.1-2

## Loading required package: penalized

## Loading required package: survival

##
## Attaching package: 'survival'

## The following object is masked from 'package:gglasso':
##
##     colon

## Welcome to penalized. For extended examples, see vignette("penalized").

## Loading required package: ks

## Loading required package: MASS
```

```
delta = 10 # 2*delta represents the minimum gap between boundaries
sigma2 = 1 # error variance

set.seed(0)
y = c(rep(0, 50), rep(1, 50), rep(0, 50), rep(1, 50)) + rnorm(200, mean = 0, sd = sqrt(sigma2)) # univa
```

```r
cpt_true = c(50, 100, 150)
n = length(y) # sample size
```

Perform dynamic programming

```r
gamma.set = c(0.01, 0.5, 1, 5, 10, 50) # a set of tuning parameters for DP
DP_result = CV.search.DP.univar(y, gamma.set, delta = 5) # grid search through cross-validation
```

```
##             [,1]      [,2]      [,3]      [,4]      [,5]      [,6]
## cpt_hat     numeric,7 numeric,6 numeric,6 numeric,3 numeric,3 numeric,0
## K_hat       7         6         6         3         3         0
## test_error  97.19456  97.13118  97.13118  88.23253  88.23253  107.8672
## train_error 81.46384  81.54316  81.54316  87.80528  87.80528  128.2893
```

```r
min_idx = which.min(DP_result$test_error) # select gamma achieves the minimum validation error
cpt_DP_hat = unlist(DP_result$cpt_hat[[min_idx]]) # estimated changepoints by DP
cpt_DP_hat
```

```
## [1]  47 101 147
```

```r
Hausdorff.dist(cpt_DP_hat, cpt_true)
```

```
## [1] 3
```

```r
cpt_DPlr_hat = local.refine.univar(cpt_DP_hat, y, w = 1/3) # perform local refinement
cpt_DPlr_hat
```

```
## [1]  47 101 150
```

```r
Hausdorff.dist(cpt_DPlr_hat, cpt_true)
```

```
## [1] 3
```

Perform standard binary segmentation

```r
tau_BS = 3 # threshold parameter for BS
BS_result = threshold.BS(BS.univar(y, 1, n, delta), tau_BS)
BS_result$BS_tree_trimmed # trace BS
```

```
## [[1]]
##   current parent location    value
## 1       1      1      150 5.190086
##
## [[2]]
##   current parent location    value
## 1       1      1      101 4.152929
##
## [[3]]
##   current parent location    value
## 1       1      1       47 5.143882
##
## [[4]]
## [1] current  parent   location value
## <0 rows> (or 0-length row.names)
##
```

```
## [[5]]
## [1] current  parent   location value
## <0 rows> (or 0-length row.names)
##
## [[6]]
## [1] current  parent   location value
## <0 rows> (or 0-length row.names)
```

```
BS_result$change_points
```

```
##   location    value level
## 1      150 5.190086     1
## 2      101 4.152929     2
## 3       47 5.143882     3
```

```
cpt_BS_hat = sort(BS_result$change_points[,1]) # estimated changepoints by BS
cpt_BS_hat
```

```
## [1]  47 101 150
```

```
Hausdorff.dist(cpt_BS_hat, cpt_true)
```

```
## [1] 3
```

```
cpt_BSlr_hat = local.refine.univar(cpt_BS_hat, y, w = 1/3) # perform local refinement
cpt_BSlr_hat
```

```
## [1]  47 101 150
```

```
Hausdorff.dist(cpt_BSlr_hat, cpt_true)
```

```
## [1] 3
```

Perform wild binary segmentation

```
tau_WBS = 3 # threshold parameter for WBS
intervals = WBS.intervals(M = 300, lower = 1, upper = n) # generate random intervals for WBS
WBS_result = threshold.BS(WBS.univar(y, 1, n, intervals$Alpha, intervals$Beta, delta), tau_WBS)
WBS_result$BS_tree_trimmed # trace BS
```

```
## [[1]]
##   current parent location    value
## 1       1      1      150 6.405863
##
## [[2]]
##   current parent location    value
## 1       1      1      101 6.106824
##
## [[3]]
##   current parent location    value
## 1       1      1       47 5.345979
##
## [[4]]
## [1] current  parent   location value
## <0 rows> (or 0-length row.names)
##
## [[5]]
```

```
## [1] current  parent   location value
## <0 rows> (or 0-length row.names)
##
## [[6]]
## [1] current  parent   location value
## <0 rows> (or 0-length row.names)
```

```r
WBS_result$change_points
```

```
##   location    value level
## 1      150 6.405863     1
## 2      101 6.106824     2
## 3       47 5.345979     3
```

```r
cpt_WBS_hat = sort(WBS_result$change_points[,1]) # estimated changepoints by WBS
cpt_WBS_hat
```

```
## [1]  47 101 150
```

```r
Hausdorff.dist(cpt_WBS_hat, cpt_true)
```

```
## [1] 3
```

```r
cpt_WBSlr_hat = local.refine.univar(cpt_WBS_hat, y, w = 1/3) # perform local refinement
cpt_WBSlr_hat
```

```
## [1]  47 101 150
```

```r
Hausdorff.dist(cpt_WBSlr_hat, cpt_true)
```

```
## [1] 3
```