Package ‘rocsvm.path’

October 14, 2022

Type Package

Title The Entire Solution Paths for ROC-SVM

Version 0.1.0

Description
We develop the entire solution paths for ROC-SVM presented by Rakotomamonjy. The ROC-SVM solution path algorithm greatly facilitates the tuning procedure for regularization parameter, lambda in ROC-SVM by avoiding grid search algorithm which may be computationally too intensive. For more information on the ROC-SVM, see the report in the ROC Analysis in AI workshop(ROCAI-2004) : Hernàndez-Orallo, José, et al. (2004) <doi:10.1145/1046456.1046489>.

Imports quadprog, svmpath

Depends R (>= 3.4.0)

License GPL-2

Encoding UTF-8

LazyData true

RoxygenNote 6.0.1

NeedsCompilation no

Maintainer Seung Jun Shin <sjshin@korea.ac.kr>

Author Seung Jun Shin [aut, cre],
Do Hyun Kim [aut]

Repository CRAN

Date/Publication 2018-10-14 17:30:03 UTC

R topics documented:

plot.rocsvm 2
poly.kernel 3
radial.kernel 3
rocsvm.get.solution 4
rocsvm.intercept 5
rocsvm.path 5
rocsvm.solve 7

Index **9**

plot.rocsvm *Plot the rocsvm.path, solution paths of ROC-SVM as a function of lambda*

Description

produces a plot of the ROC-SVM `lambda` path.

Usage

```
## S3 method for class 'rocsvm'
plot(x, ...)
```

Arguments

`x`	The rocsvm path object
`...`	Generic compatibility

Value

The entire solution path of ROC-SVM solution as a function of `lambda`.

Author(s)

Seung Jun Shin, Do Hyun Kim

See Also

`rocsvm.path`

Examples

```
# The 'obj' comes from an example description of rocsvm.path()
library(rocsvm.path)

n <- 30
p <- 2
delta <- 1
set.seed(309)
y <- c(rep(1, n/2), rep(-1, n/2))
x <- matrix(0, n, p)
for (i in 1:n){
 if (y[i] == 1) {
 x[i,] <- rnorm(p, -delta, 1)
 } else {
 x[i,] <- rnorm(p, delta, 1)
  }
 }
```

```
rho = 1
kernel = radial.kernel
param.kernel  = 1/ncol(x)
prop = 0.1

obj <- rocsvm.path(x, y, rho, kernel, param.kernel, prop)
plot(obj)
# or plot.rocsvm(obj, lty = 2, lwd = 2, col = 2)
```

`poly.kernel` *Compute the kernel matrix for ROC-SVM path*

Description

Compute the kernel matrix for ROC-SVM path. This function comes from `svmpath` package by Trevor Hastie. If you want to know details of this function, refer the `svmpath` package.

Usage

```
poly.kernel(x, y = x, param.kernel = 1, ...)
```

Arguments

`x`	An n x p matrix of features
`y`	An m x p matrix of features
`param.kernel`	The parameter(s) for the kernel. For the radial kernel, the parameter is known in the fields as "gamma". For the polynomial kernel, it is the "degree"
`...`	unused

`radial.kernel` *Compute the kernel matrix for ROC-SVM path*

Description

Compute the kernel matrix for ROC-SVM path. This function comes from `svmpath` package by Trevor Hastie. If you want to know details of this function, refer the `svmpath` package.

Usage

```
radial.kernel(x, y = x, param.kernel = 1/p, ...)
```

Arguments

x	An n x p matrix of features
y	An m x p matrix of features
param.kernel	The parameter(s) for the kernel. For this radial kernel, the parameter is known in the fields as "gamma". For the polynomial kernel, it is the "degree"
...	unused

---

rocsvm.get.solution — *Finding solutions fixed the regularization parameter of ROC-SVM.*

Description

Computes solution alpha values from a fixed regularization parameter, lambda value for ROC-SVM path object.

Usage

```
rocsvm.get.solution(obj, lambda)
```

Arguments

obj	The rocsvm.path object
lambda	The regularization parameter that users want in ROC-SVM model.

Author(s)

Seung Jun Shin, Do Hyun Kim

See Also

rocsvm.path

Examples

```
# library(rocsvm.path)
# The 'obj' comes from an example description of rocsvm.path()

rocsvm.get.solution(obj, lambda = 1)
```

rocsvm.intercept — *Finding an intercept fixed sensitivity or specificity for ROC-SVM*

Description

Computes an intercept at a specific sensitivity or specificity level from the ROC-SVM model.

Usage

```
rocsvm.intercept(obj, lambda = 1, sensitivity = 0.5, specificity = 0.5)
```

Arguments

`obj`	The rocsvm.path object
`lambda`	The regularization parameter that users want in ROC-SVM model.
`sensitivity`	Sensitivity in ROC curve, which means True Positive Rate (TPR).
`specificity`	Specificity in ROC curve, which means True Negative Rate (TNR) = 1-FPR.

Author(s)

Seung Jun Shin, Do Hyun Kim

See Also

`rocsvm.path`

Examples

```
# library(rocsvm.path)
# The 'obj' comes from an example description of rocsvm.path()

rocsvm.intercept(obj, lambda = 1, sensitivity = 0.9, specificity = 0.1)
```

rocsvm.path — *Fit the entire regularization path for ROC-Support Vector Machine (ROC-SVM)*

Description

This algorithm computes the entire regularization path for the ROC-Support Vector Machine with a relatively low cost compared to quadratic programming problem.

Usage

```
rocsvm.path(x, y, rho = 1, kernel = poly.kernel, param.kernel = 1,
  prop = 0.5, lambda.min = 1e-05, eps = 1e-05, Nmoves = 500)
```

Arguments

`x`	The data matrix (n x p) with n rows (observations) on p variables (columns)
`y`	The `{-1, 1}` valued response variable.
`rho`	A positive constant
`kernel`	This is a user-defined function. Provided options are polynomial kernel; `poly` (the default, with parameter set to default to a linear kernel) and radial kernel; `radial`.
`param.kernel`	The parameter(s) for the kernel. For this radial kernel, the parameter is known in the fields as "gamma". For the polynomial kernel, it is the "degree"
`prop`	The proportion of large class corresponding a point of small class by speed-up tricks (the default is `prop = 0.5`). If you don't want to use the "speed-up tricks", then set `prop` to 1.
`lambda.min`	The smallest value of lambda for termination of the algorithm (the default is `lambda.min = 1e-05`).
`eps`	An adjustment computing errors
`Nmoves`	The maximum number of iterations the rocsvm.path algorithm

Value

A 'rocsvm.path' object is returned, for which there are `lambda` values and corresponding values of `alpha` for each data point.

Author(s)

Seung Jun Shin, Do Hyun Kim

See Also

`rocsvm.get.solution`, `plot.rocsvm`, `rocsvm.intercept`

Examples

```
library(rocsvm.path)
n <- 30
p <- 2
delta <- 1
set.seed(309)
y <- c(rep(1, n/2), rep(-1, n/2))
x <- matrix(0, n, p)
for (i in 1:n){
 if (y[i] == 1) {
 x[i,] <- rnorm(p, -delta, 1)
 } else {
 x[i,] <- rnorm(p, delta, 1)
  }
 }

rho = 1
```

```
kernel = radial.kernel
param.kernel  = 1/ncol(x)
prop = 0.1
obj <- rocsvm.path(x, y, rho, kernel, param.kernel, prop)
```

---

rocsvm.solve *Finding Lagrangian multipliers of ROC-SVM by Qudratic Programming*

Description

Computes the Lagrangian multipliers(alpha), which are solutions of ROC-SVM using Quadratic Programming.

Usage

```
rocsvm.solve(K, lambda, rho = 1, eps = 1e-08)
```

Arguments

K	The kernelized matrix, i.e., K< .,. >.
lambda	The regularization parameter that users want in ROC-SVM model.
rho	A positive constant (default : 1)
eps	Adjustment computing errors (default : 1e-08)

Author(s)

Seung Jun Shin, Do Hyun Kim

See Also

rocsvm.path

Examples

```
n <- 30
p <- 2
delta <- 1
set.seed(309)
y <- c(rep(1, n/2), rep(-1, n/2))
x <- matrix(0, n, p)
for (i in 1:n){
 if (y[i] == 1) {
 x[i,] <- rnorm(p, -delta, 1)
 } else {
 x[i,] <- rnorm(p, delta, 1)
  }
```

```
 }

K <- radial.kernel(x,x)
rocsvm.solve(K, lambda = 1, rho = 1)
```

Index

plot.rocsvm, 2, *6*
poly.kernel, 3

radial.kernel, 3
rocsvm.get.solution, 4, *6*
rocsvm.intercept, 5, *6*
rocsvm.path, *2*, *4*, *5*, 5, *7*
rocsvm.solve, 7