Package ‘SVHM’

February 28, 2022

Type Package

Title Support-Vector-Hazard-Machine

Version 0.1.0

Author Luca Herglotz

Maintainer Luca Herglotz <st163109@stud.uni-stuttgart.de>

Description
Implements the Support-Vector-Hazard machine approach presented in the paper 'Support Vector Hazards Machine: A Counting Process Framework for Learning Risk Scores for Censored Outcomes' by Wang, Yuanjia and Chen, Tianle and Zeng, Donglin. To solve the quadratic optimization problem for SVHM either the package osqp or the package Rmosek has to be installed.

Imports Matrix, Rmosek, osqp, distances, matchingR, dplyr

Encoding UTF-8

LazyData true

RoxygenNote 7.1.2

R topics documented:

condition_mat 2
createDataPartition 2
createListPartition 3
create_risk_and_event_matrix 4
create_svhm 5
create_time_svhm 6
data_at_time 8
normalize 8
optimization_data 9
optimization_time_data 10
opt_sol_mosek 10
opt_sol_osqp 11
opt_time_sol_mosek 11
opt_time_sol_osqp 12
predict_event_time 12
radial_kernel 13
radial_kernel_mat 14
risk_score 14
risk_score_training 15
risk_time_score 16

risk_time_score_training 17
train_svhm 17
train_time_svhm 19
weight_mat 20

Index **21**

`condition_mat` *Constraint matrix in quadratic optimization problem*

Description

calculates the matrix which defines the constraints in the SVHM algorithm

Usage

```
condition_mat(event_vec, num_event_time)
```

Arguments

`event_vec` vector containing information if a subject is at risk or if an event happens. If n are the number of subjects and m the number of event times, then event_vec has length n*m

`num_event_time` number of event

Value

matrix

`createDataPartition` *createDataPartition*

Description

partitions a dataset into a test set and cross validation sets. createDataPartition() is not randomized, therefore df should be randomized before creating the partition!

Usage

```
createDataPartition(df, cross_validation_val, test_size = 0.2)
```

Arguments

`df` data frame

`cross_validation_val` number of sets for k-fold cross validation

`test_size` size of test set (default=.8)

Value

partitioned dataset

Examples

```
{
# Example with the preloaded mtcars dataset
df<-mtcars
partition <- SVHM:::createDataPartition(mtcars, 4, .2)
}
```

---

createListPartition *createListPartition*

Description

partitions a List into a test set and cross validation sets. createDataPartition() is not randomized, therefore the list should be randomized before creating the partition!

Usage

```
createListPartition(l, cross_validation_val, test_size = 0.2)
```

Arguments

`l`	list of data
`cross_validation_val`	number of sets for k-fold cross validation
`test_size`	size of test set (default=.8)

Value

partitioned list

Examples

```
{
# Example with the preloaded mtcars dataset
l<-list("A", "B", "C", "D", "E", "F", "G", "H", "I", "J")
partition <- SVHM:::createListPartition(l, 3, .1)
}
```

create_risk_and_event_matrix

Risk and Event Matrix

Description

calculates two matrices of length n*m, if n are the number of subjects and m the number of event times. The Risk Matrix indicates for every subject in a dataset if the subject is still at risk at every event time. The Event Matrix is equal to the Risk Matrix but if a subject experiences an event at an event time the entrie is set to -1

Usage

```
create_risk_and_event_matrix(training_dataset, ordered_event_times)
```

Arguments

`training_dataset`
data frame representing the training data

`ordered_event_times`
data frame of all event times ordered in ascending order

Value

List `$r_mat` matrix indicating at risk at every event time for subjects `$e_mat` matrix indicating if subjects are at risk and if they are experiencing an event at any event time

Examples

```
{
# Create random data
train <- data.frame(futime = sample.int(10,6),
                    death = sample(c(TRUE,FALSE), 6, replace=TRUE),
                    training_id=1:6)
ordered_event_times <- with(train,
                           data.frame(
                              futime = sort(train$futime[train$death == TRUE]),
                              training_id = train$training_id[train$death == TRUE])
)

SVHM:::create_risk_and_evemt_matrix(train, ordered_event_times)
}
```

create_svhm *Train SVHM*

Description

predicts the event times of a given dataset using cross validation for the cost parameter. Final model includes predicted event times as well as parameters to predict new event times from subject who are not in df. All values of covariates are first normalized to the intervall [0,1] before the SVHM algorithm is applied. The cost parameter for the final model is chosen with the best pearson correlation.

Usage

```
create_svhm(
  df,
  covariates,
  cross_validation_val,
  cost_grid,
  varName_cencored,
  varName_futime,
  k = 3,
  test_size = 0.2,
  opt = "osqp",
  gamma_squared = 0.5,
  choose = "c"
)
```

Arguments

`df` data frame

`covariates` vector of name of covariates

`cross_validation_val` number of subset to use for cost optimization

`cost_grid` grid of all cost parameter to be optoimzed uponl

`varName_cencored` name of variable in df that indicates cencoring

`varName_futime` name of variable in df that indicates event time

`k` integer of how many nearest event times are used to predict the event time (default is 3)

`test_size` size of final test set in precent

`opt` which quadratic optimization is used (`opt='mosek'` or `opt='osqp'`)

`gamma_squared` width of gaussian kernel

`choose` optional parameter which decides if the C-index or the pearson correlation is used to determine the optimal cost parameter. Values are either `'c'` for the C-Index or `'p'` for the pearson correlation

Value

trained model with `$e_vec` vector indicating vector containing information if a subject is at risk or if an event happens. If n are the number of subjects and m the number of event times, then event_vec has length n*m, `$k_mat` kernel matrix, `$sol` calculated optimal solution, `$t_predict` test dataset with risk scores `risk` and `t_predict`, `$p_corr` pearson correlation of the predicted times `$C_indes` C-Index

Note

The mosek package requires a license

Examples

```
{

library(KMsurv)
library(SVHM)

##############
# Parameters #
##############

gamma_squared <- 100
k <- 1
cross_validation_val <- 3
test_size=.3
cost_grid <- 2^c(-6:6)

covariates <- c('z7')

######################
#  Model prediction  #
######################

data(bmt)

model <- create_svhm(bmt, covariates, cross_validation_val, cost_grid, varName_cencored="d3", varName_futime
}
```

create_time_svhm *Train Time Dependent SVHM*

Description

Calculates the Risk score and the value of the prediction function for each individual in the data set.

Usage

```
create_time_svhm(
  df,
  covariates,
  cost,
  varName_cencored,
  varName_futime,
  start_interval,
  end_interval,
  test_size = 0.3,
  opt = "osqp",
  gamma_squared = 0.5
)
```

Arguments

`df`	data frame
`covariates`	vector of name of covariates
`cost`	cost parameter to be used
`varName_cencored`	name of variable in df that indicates cencoring
`varName_futime`	name of variable in df that indicates event time
`start_interval`	name of variable that indicates when the interval starts
`end_interval`	name of variable that indicates when the interval ends
`test_size`	size of final test set in precent
`opt`	which quadratic optimization is used (`opt='mosek'` or `opt='osqp'`)
`gamma_squared`	width of gaussian kernel

Value

trained model with `$e_vec` vector indicating if an event happens at each event time `$sol` calculated optimal solution for each event time `$train` train dataset with risk scores `$test` test dataset with risk scores `cost` cost parameter

Note

In contrast to the `create_svhm()` function this function does not predict event times!

Examples

```
{

library(timereg)
library(SVHM)

##############
# Parameters #
##############

opt <- "osqp"
gamma_squared <- 200
test_size=.3
cost <- 16

######################
#  Model prediction  #
######################
```

```
data(csl)

time_model <- create_time_svhm(csl, c("sex", "age"), cost, varName_cencored='dc', varName_futime='eventT', st
}
```

`data_at_time` *data_at_time*

Description

Retrieves the relevant data for each individual at every event time.

Usage

```
data_at_time(i, j, df, times)
```

Arguments

`i`	index of individuals
`j`	index of event time
`df`	list of dataframes of the individuals
`times`	dataframe of event times

Value

row of data for i-th individual if the individual is still under risk, otherwise return row of NA.

`normalize` *Normalize*

Description

normalizes a vector

Usage

```
normalize(df, covariates)
```

Arguments

`df`	dataframe
`col`	columns to be normalized

Value

normalized columns of dataframe

Examples

```
{
Example with the preloaded mtcars dataset
SVHM:::normalize(mtcars, c('disp',  'hp'))
}
```

---

optimization_data *Optimization Data*

Description

calculates all needed values to execute quadratic optimization in SVHM

Usage

```
optimization_data(
  covariates,
  training_dataset,
  ordered_event_times,
  gamma_squared = 0.5,
  d = 1
)
```

Arguments

`covariates` dataset of covariates of the subjects in a dataset

`training_dataset` data frame representing the trainings dataset

`ordered_event_times` data frame of all event times ordered in ascending order

`gamma_squared` width of gaussian kernel

`d` degree of polynomial kernel

`type` Type of kernel, either 'gauss' or 'poly' for gaussian or polynomial kernel

Value

List `$r_vec` vector representing at which event times the subjects are under risk `$adap_kernel_mat` matrix on which quadratic optimization will be performed `$c_mat` matrix representing the constraints of the optimization problem `$w_vec` vector of weights at any event time for all subjects `$kernel_mat` Gram matrix of covariates `$e_vec` vector indicating vector containing information if a subject is at risk or if an event happens. If n are the number of subjects and m the number of event times, then event_vec has length n*m,

optimization_time_data *Time Dependent Optimization Data*

Description

calculates all needed values to execute quadratic optimization for the time dependent SVHM at the given event time.

Usage

```
optimization_time_data(covariates, mat_train, event_time, gamma_squared = 0.5)
```

Arguments

`covariates`	dataset of covariates of the subjects in a dataset
`mat_train`	matrix of all individuals under risk at the event time
`event_time`	event time for which data is calculated
`gamma_squared`	width of gaussian kernel

Value

List `$adap_k_mat` matrix on which quadratic optimization will be performed `$w_vec` vector of weights at any event time for all subjects `$k_mat` Gram matrix of covariates `$e_vec` vector indicating vector containing information if a subject experiences an event.

opt_sol_mosek *Optimal solution of SVHM*

Description

Uses the Rmosek package to solve the quadratic optimization problem defined by SVHM.

Usage

```
opt_sol_mosek(optimizazion_data, num_event_times, cost)
```

Arguments

`num_event_times`	number of event times in the training dataset
`cost`	cost parameter of the support vector machine of type numeric
`optimization_data`	all values needed for optimization in a list with order (risk_vector, adapted_kernel_matrix, cond_mat, weight_vec)

Value

optimal solution for the SVHM

Note

Rmosek requires a license to use!

`opt_sol_osqp` *Optimal solution of SVHM*

Description

Uses the osqp package to solve the quadratic optimization problem defined by SVHM.

Usage

```
opt_sol_osqp(optimizazion_data, num_event_times, cost)
```

Arguments

`num_event_times`
number of event times in the training dataset

`cost` cost parameter of the support vector machine of type numeric

`optimization_data`
all values needed for optimization in a list with order (risk_vector, adapted_kernel_matrix, cond_mat, weight_vec)

Value

optimal solution for the SVHM

`opt_time_sol_mosek` *Optimal solution of time dependent SVHM*

Description

Uses the Rmosek package to solve the quadratic optimization problem defined by SVHM.

Usage

```
opt_time_sol_mosek(e_vec, k_mat, w_vec, cost)
```

Arguments

`e_vec` vector indicating if a subject experienced an event at an event time

`k_mat` matrix

`w_vec` weight vector

`cost` cost parameter of the support vector machine of type numeric

Value

optimal solution for the time dependent SVHM

Note

Rmodek package requires a licence!

`opt_time_sol_osqp` *Optimal solution of time dependent SVHM*

Description

Uses the osqp package to solve the quadratic optimization problem defined by the time dependent SVHM.

Usage

```
opt_time_sol_osqp(e_vec, k_mat, w_vec, cost)
```

Arguments

`e_vec`	vector indicating if a subject experienced an event at an event time
`k_mat`	matrix
`w_vec`	weight vector
`cost`	cost parameter of the support vector machine of type numeric

Value

optimal solution for the time dependent SVHM

`predict_event_time` *Predict Event Time of a subject*

Description

calculate the predicted event time of an individual

Usage

```
predict_event_time(df, x, k = 3, rounding = "ceil")
```

Arguments

`df`	dataframe of non censored subjects in the training set
`x`	Risk score of the individual which will be predicted upon
`k`	integer of how many nearest event times are used to predict the event time (default is 3)
`rounding`	Options are 'ceil', 'floor' and 'no'. (default is 'ceil')

Details

This function predicts the event time of a subject based on the k closest risks subjects in the training dataset of non cencored individuals df. The risks in df are ranked and the predicted event time is the average of the k event times that coincide with the rank of the k closest risks. The predicted event time is rounded up to integers by default. A vectorized version predict_event_time_vec() for the parameter x exists.

Value

predicted event time

References

Wang, Y., Chen, T., and Zeng, D. Support vector hazards machine: A counting process framework for learning risk scores for censored outcomes. Journal of Machine Learning Research, 17(167):1-37, 2016

`radial_kernel` *Gaussian Kernel*

Description

calculates the Gaussian Kernel value of two inputs

Usage

```
radial_kernel(x, y, gamma_squared)
```

Arguments

`x`	first input vector
`y`	second input vector
`gamma_squared`	width of the kernel

Value

gaussian kernel value

Examples

```
{
x <- runif(n=10)
y <- runif(n=10)
SVHM:::radial_kernel(x,y,.5)
}
```

radial_kernel_mat *Gaussian Kernel Matrix*

Description

calculates the gaussian kernel value of the covariates with each other. calculated matrix will be symmetric

Usage

```
radial_kernel_mat(covariates, gamma_squared)
```

Arguments

covariates	dataset of covariates of the subjects in a dataset
gamma_squared	width of the kernel

Value

gaussian kernel matrix

Examples

```
{
# Example with the preloaded mtcars dataset
covariates <- subset( mtcars, select = c('drat', 'wt') )
SVHM:::radial_kernel_mat(covariates,.5)
}
```

risk_score *risk scores*

Description

calculates the risk scores for one individual with the help of the calculated optimal solution to the quadratic programming problem of SVHM and the kernel matrix of the covariates of the test dataset.

Usage

```
risk_score(
  gamma_sol,
  event_vec,
  v,
  covariates_train,
  num_event_times,
  gamma_squared = 0.5,
  d = 1
)
```

Arguments

`gamma_sol`	optimal solution of the SVHM
`event_vec`	vector containing information of the training if a subject in the training dataset is at risk or if an event happens. If n are the number of subjects in the training dataset and m the number of event times in the training dataset, then event_vec has length n*m
`v`	covariates of the individual for which the risk is to be calculated
`covariates_train`	dataset of covariates of the subjects in the training dataset
`num_event_times`	number of event times that occour in the training data set
`gamma_squared`	width of gaussian kernel
`d`	degree of polynomial kernel
`type`	Type of kernel, either 'gauss' or 'poly' for gaussian or polynomial kernel

Value

risk score of the individual

Note

The calculated risk score is not the actual risk scores defined by the Risk function but it induce an ordering of the risk scores. For detailed information see reference

References

Wang, Y., Chen, T., and Zeng, D. Support vector hazards machine: A counting process framework for learning risk scores for censored outcomes. Journal of Machine Learning Research, 17(167):1-37, 2016

`risk_score_training` *Training risk scores*

Description

calculates the risk scores for all individuals in the training dataset.

Usage

```
risk_score_training(
  gamma_sol,
  kernel_mat,
  event_vec,
  num_event_times,
  training_set_size
)
```

Arguments

`gamma_sol`	optimal solution of the SVHM
`kernel_mat`	Gram matrix of the covariates
`num_event_times`	number of event times that occour in the training data set

Value

vector of risk scores for all training subjects

Note

The calculated risk scores are not the actual risk scores defined by the Risk function but the induce an ordering of the risk scores. For detailed information see reference

References

Wang, Y., Chen, T., and Zeng, D. Support vector hazards machine: A counting process framework for learning risk scores for censored outcomes. Journal of Machine Learning Research, 17(167):1-37, 2016

`risk_time_score` *risk time scores*

Description

calculates the risk scores for one individual with the help of the calculated optimal solution to the quadratic programming problem of time dependent SVHM at time j.

Usage

```
risk_time_score(
  gamma_sol,
  event_vec,
  weight_vec,
  v,
  covariates_train,
  n,
  gamma_squared = 0.5
)
```

Arguments

`gamma_sol`	optimal solution of the SVHM
`event_vec`	vector containing information of the training if a subject experiences an event happens.
`weight_vec`	vector containing weigths
`v`	covariates of the individual for which the risk is to be calculated
`covariates_train`	dataset of covariates of the subjects in the training dataset
`n`	number of individuals in the training dataset
`gamma_squared`	width of gaussian kernel

Value

`$f_at_j` decision function at j `$risk_at_j` risk score of the individualat time j

---

`risk_time_score_training` *Training risk time scores*

Description

calculates the risk scores for all individuals in the training dataset for the time dependent SVHM.

Usage

```
risk_time_score_training(kernel_mat, event_vec, weight_vec, f_vec, n)
```

Arguments

`kernel_mat`	Gram matrix of the covariates
`event_vec`	vector containing information of the training if a subject experiences an event happens.
`weight_vec`	vector containing weigths
`f_vec`	optimal decision function
`n`	number of individuals in the training dataset

Value

vector of risk scores for all training subjects

`train_svhm` *Train SVHM*

Description

Uses the Rmosek or osqp package to train the SVHM on a given training and test set. Names of the cencoring variable and event variable mus be `death` and `futime`

Usage

```
train_svhm(
  train,
  test,
  covariates,
  cost,
  k = 3,
  opt = "osqp",
  gamma_squared = 0.5
)
```

Arguments

`train`	training dataset
`test`	test dataset
`covariates`	vector of name of covariates
`cost`	cost parameter of the support vector machine of type numeric
`k`	integer of how many nearest event times are used to predict the event time (default is 3)
`opt`	which quadratic optimization is used (`opt='mosek'` or `opt='osqp'`)
`gamma_squared`	width of gaussian kernel

Value

trained model with `$e_vec` vector indicating vector containing information if a subject is at risk or if an event happens. If n are the number of subjects and m the number of event times, then event_vec has length n*m, `$k_mat` kernel matrix, `$sol` calculated optimal solution, `$t_predict` test dataset with risk scores `risk` and `t_predict`, `$p_corr` pearson correlation of the predicted times `$C_indes` C-Index

Note

The mosek package requires a license

Examples

```
{

library(KMsurv)
library(SVHM)

data(bmt)
df<-bmt[1:40,]

# shuffle data
rows <- sample(nrow(df))
df <- df[rows, ]

covariates <- c('z3', 'z4')

# censoring variable and event variable need to have names "death" and "futime"
names(df)[names(df) == "d3"] <- "death"
names(df)[names(df) == "t2"] <- "futime"

n<-floor(nrow(df)/2)
train<- df[(1:n), ]
test<- df[-(1:n), ]

train_svhm(train, test, covariates, 10, .5, k=1, opt='osqp')
}
```

`train_time_svhm` *Train time dependent SVHM*

Description

Uses the Rmosek or osqp package to train the time dependent SVHM on a given training and test set. The training calculates the risk scores and the optimal decision function values for each individual at every event time of the training set. The columns of the dataset must contain `id,futime,death,covariates,lt,rt` where `lt` and `rt` are the start and end times of each time interval. the `death` column must also be logical values.

Usage

```
train_time_svhm(
  train,
  test,
  covariates,
  cost,
  opt = "osqp",
  gamma_squared = 0.5
)
```

Arguments

`train`	training dataset
`test`	test dataset
`cost`	cost parameter of the support vector machine of type numeric
`opt`	which quadratic optimization is used (`opt='mosek'` or `opt='osqp'`)
`gamma_squared`	width of gaussian kernel

Value

trained model with `$e_vec` vector indicating if an event happens at each event time `$event_times` ordered event times of the training dataset `$sol` calculated optimal solution for each event time `$train` train dataset with risk scores `$test` test dataset with risk scores

Examples

```
{
library(timereg)
library(SVHM)

data(csl)

df <- csl

names(df)[names(df) == "dc"] <- "death"

names(df)[names(df) == "eventT"] <- "futime"

df <- transform(df,
                death = as.logical(death))


df<-split(df, df$id)

df[sample(1:length(df))]

partition <- SVHM:::createListPartition(df, 1, test_size=.3)

df_test <- partition$"test"

df_train <- partition[["1"]]

trained_model <- train_time_svhm(df_train, df_test, c("sex"), 10, opt="osqp", gamma_squared=100)
}
```

weight_mat *Weight matrix*

Description

calculates the weights for every individual in the training dataset at every event time. If an individual experiences an event the weight is given by the ratio of at risk subjects with no event to all at risk subjects. If no event is experienced the weight is given by the ratio of one over all at risk subjects.

Usage

```
weight_mat(training_dataset, ordered_event_times)
```

Arguments

`training_dataset`
: data frame representing the training data

`ordered_event_times`
: data frame of all event times ordered in ascending order

Value

matrix storing all weights for every individual

Examples

```
{
training <- data.frame(id =c(1:5), futime= sort(runif(5)), death = c(TRUE, FALSE, FALSE, TRUE, TRUE), Y=c(5,4,3
times <- subset(training[training$death==TRUE,], select=sort(futime))
SVHM:::weight_mat(training, times)
}
```

Index

condition_mat, 2
create_risk_and_event_matrix, 4
create_svhm, 5
create_time_svhm, 6
createDataPartition, 2
createListPartition, 3

data_at_time, 8

normalize, 8

opt_sol_mosek, 10
opt_sol_osqp, 11
opt_time_sol_mosek, 11
opt_time_sol_osqp, 12
optimization_data, 9
optimization_time_data, 10

predict_event_time, 12

radial_kernel, 13
radial_kernel_mat, 14
risk_score, 14
risk_score_training, 15
risk_time_score, 16
risk_time_score_training, 17

train_svhm, 17
train_time_svhm, 19

weight_mat, 20