Package ‘mildsvm’

October 13, 2022

Type  Package
Title  Multiple-Instance Learning with Support Vector Machines
Version  0.4.0


License  MIT + file LICENSE
URL  https://github.com/skent259/mildsvm

BugReports  https://github.com/skent259/mildsvm/issues
Depends  R (>= 3.5.0)
Imports  dplyr, e1071, kernlab, magrittr, mvtnorm, pillar, pROC, purrr, rlang, stats, tibble, tidyr, utils
Suggests  covr, gurobi, Matrix, testthat

Config/testthat/edition  3
Encoding  UTF-8
LazyData  true
R topics documented:

as_mild_df ................................................................. 3
as_mi_df ................................................................. 4
bag_instance_sampling ............................................. 5
build_fm ................................................................. 6
build_instance_feature .............................. 7
classify_bags ......................................................... 8
cv_misvm .............................................................. 9
formatting ........................................................... 12
generate_mild_df ........................................... 13
kfm_exact ........................................................... 16
kfm_nystrom ......................................................... 17
kme ................................................................. 19
mi ................................................................. 20
mild ............................................................... 21
mild_df ............................................................ 22
mior ............................................................... 23
mismm ............................................................ 26
misvm ............................................................ 30
misvm_orova .................................................... 33
mi_df .............................................................. 36
omisvm ......................................................... 37
ordmvnorm ...................................................... 40
predict.cv_misvm ........................................... 41
predict.mior ..................................................... 42
predict.mismm .................................................. 44
predict.misvm .................................................... 46
predict.misvm_orova ........................................ 47
predict.omisvm ................................................. 49
predict.smm ..................................................... 50
predict.svor_exc ............................................... 52
smm ............................................................ 53
summarize_samples ............................................... 56
svor_exc ......................................................... 57

Index 60
as_mild_df

Coerce to MILD data frame

Description

as_mild_df() turns an existing object, such as a data frame, into a MILD data frame, a data frame with 'mild_df'. This is in contrast with mild_df(), which builds a MILD data frame from individual columns.

Usage

as_mild_df(
  x,
  bag_label = "bag_label",
  bag_name = "bag_name",
  instance_name = "instance_name",
  instance_label = "instance_label",
  ...
)

Arguments

  x          A data-frame or similar to convert.
  bag_label  A character (default 'bag_label') describing which column refers to the bag label.
  bag_name   A character (default 'bag_name') describing which column refers to the bag name.
  instance_name A character (default 'instance_name') describing which column refers to the instance name.
  instance_label A character (default 'instance_label') describing which column refers to the instance labels. If NULL, no instance_labels will be used.
  ...        Arguments reserved for other methods.

Value

A 'mild_df' object. This data.frame-like has columns bag_label, bag_name, instance_name, and potentially others. It also inherits from the 'tbl_df' and 'tbl' classes.

Author(s)

Sean Kent

See Also

mild_df() to build a mild_df object.
Examples

```r
x <- data.frame('bag_LABEL' = factor(c(1, 1, 0)),
                 'bag_name' = c(rep('bag_1', 2), 'bag_2'),
                 'instance_name' = c('bag_1_inst_1', 'bag_1_inst_2', 'bag_2_inst_1'),
                 'X1' = c(-0.4, 0.5, 2),
                 'instance_label' = c(0, 1, 0))

df <- as_mild_df(x)
```

---

### Description

`as_mi_df()` turns an existing object, such as a data frame, into a MI data frame, a data frame with 'mi_df'. This is in contrast with `mi_df()`, which builds a MI data frame from individual columns.

### Usage

```r
as_mi_df(
  x,
  bag_label = "bag_label",
  bag_name = "bag_name",
  instance_label = "instance_label",
  ...
)
```

### Arguments

- `x` A data-frame or similar to convert.
- `bag_label` A character (default `bag_label`) describing which column refers to the bag label.
- `bag_name` A character (default `bag_name`) describing which column refers to the bag name.
- `instance_label` A character (default `instance_label`) describing which column refers to the instance labels. If NULL, no instance_labels will be used.
- `...` Arguments reserved for other methods.

### Value

A `mi_df` object. This data.frame-like has columns `bag_label`, `bag_name`, and potentially others. It also inherits from the `tbl_df` and `tbl` classes.

### Author(s)

Sean Kent
See Also

`mi_df()` to build a `mi_df` object.

Examples

```r
x = data.frame('bag_LABEL' = factor(c(1, 1, 0)),
                'bag_name' = c(rep('bag_1', 2), 'bag_2'),
                'X1' = c(-0.4, 0.5, 2),
                'instance_label' = c(0, 1, 0))

df <- as_mi_df(x)
```

---

**bag_instance_sampling**  
Sample `mild_df` object by bags and instances

**Description**

From a `mild_df` object, return a sample that evenly pulls from the unique bags and unique instances from each bag as much as possible. This is a form of stratified sampling to avoid randomly sampling many rows from a few bags.

**Usage**

```r
bag_instance_sampling(data, size)
```

**Arguments**

- `data`: A `mild_df` object containing the data.
- `size`: A non-negative integer giving the number of rows to choose from `data`.

**Value**

A numeric vector of length `size` indicating which rows were sampled.

**Author(s)**

Sean Kent

**Examples**

```r
mil_data <- generate_mild_df(positive_dist = "mvnormal",
                              nbag = 2,
                              ninst = 2,
                              nsample = 2)

rows <- bag_instance_sampling(mil_data, 6)
table(mil_data$bag_name[rows])
```
**build_fm**

```r
table(mil_data$instance_name[rows])
rows <- bag_instance_sampling(mil_data, 4)
table(mil_data$bag_name[rows])
table(mil_data$instance_name[rows])
```

---

**build_fm**

*Build a feature map on new data*

**Description**

Feature maps provide a set of covariates in a transformed space. The `build_fm()` function creates these covariates based on an object that specifies the feature map and a provided dataset.

**Usage**

```r
build_fm(kfm_fit, new_data, ...)
```

---

**Arguments**

- `kfm_fit`  
  An object from a function in the `kfm_*` family, such as `kfm_nystrom()`.
- `new_data`  
  The data to generate features from.
- `...`  
  Additional arguments for methods.

**Value**

A matrix of covariates in the feature space, with the same number of rows as `new_data`. If `new_data` is a `mild_df` object, `build_fm()` will also return the columns containing 'bag_label', 'bag_name', 'instance_name'.

**Methods (by class)**

- `kfm_exact`: Method for `kfm_exact` class.
- `kfm_nystrom`: Method for `kfm_nystrom` class.

**Author(s)**

Sean Kent
**build_instance_feature**

See Also

- `kfm_nystrom()` fit a Nystrom kernel feature map approximation.
- `kfm_exact()` create an exact kernel feature map.

Examples

```r
df <- data.frame(
  X1 = c(2, 3, 4, 5, 6, 7, 8),
  X2 = c(1, 1.2, 1.3, 1.4, 1.1, 7, 1),
  X3 = rnorm(7)
)

fit <- kfm_nystrom(df, m = 7, r = 6, kernel = "radial", sigma = 0.05)
fm <- build_fm(fit, df)

fit <- kfm_exact(kernel = "polynomial", degree = 2, const = 1)
fm <- build_fm(fit, df)
```

---

**build_instance_feature**

*Flatten mild_df data to the instance level*

Description

Flatten mild_df type of data to regular multiple instance data where each instance is a vector by extracting distribution sample quantiles, mean and sd.

Usage

```r
build_instance_feature(
  data,
  qtls = seq(0.05, 0.95, length.out = 10),
  mean = TRUE,
  sd = TRUE
)
```

Arguments

- `data` A mild_df object.
- `qtls` Quantiles to be extracted from each instance empirical distribution.
- `mean` A logical for whether or not to extract mean.
- `sd` A logical for whether or not to extract standard deviation.

Value

A summarized data.frame at the instance level.
classify_bags

Author(s)
Yifei Liu

See Also
summarize_samples() for a more general way to make a similar data frame.

Examples

mild_df1 <- generate_mild_df(positive_degree = 3, nbag = 3)
df1 <- build_instance_feature(mild_df1, seq(0.05, 0.95, length.out = 10))

classify_bags

Classify y from bags

Description

Formally, this function applies max() on y for each level of bags.

Usage

classify_bags(y, bags, condense = TRUE)

Arguments

y
A numeric, character, or factor vector of bag labels for each instance. Must satisfy length(y) == nrow(x). Suggest that one of the levels is 1, ‘1’, or TRUE, which becomes the positive class; otherwise, a positive class is chosen and a message will be supplied.

bags
A vector specifying which instance belongs to each bag. Can be a string, numeric, or factor.

condense
A logical (default TRUE) for whether to return classification at the level of unique bags or not.

Value

A named vector of length length(unique(b)) which gives the classification for each bag. Names come from bags.

Author(s)
Sean Kent
Examples

```r
y <- c(1, 0, 0, 1, 1, 1, 0, 0, 0)
bags <- rep(1:3, each = 3)

classify_bags(y, bags)
classify_bags(y, bags, condense = FALSE)

# works with regular vector too
scores <- 1:9
classify_bags(scores, bags)
```

**cv_misvm**

*Fit MI-SVM model to the data using cross-validation*

**Description**

Cross-validation wrapper on the `misvm()` function to fit the MI-SVM model over a variety of specified cost parameters. The optimal cost parameter is chosen by the best AUC of the cross-fit models. See `?misvm` for more details on the fitting function.

**Usage**

```r
## Default S3 method:
cv_misvm(
x,
y,
bags,
cost_seq,
n_fold,
fold_id,
method = c("heuristic", "mip", "qp-heuristic"),
weights = TRUE,
control = list(kernel = "linear", sigma = 1, nystrom_args = list(m = nrow(x), r = nrow(x), sampling = "random"), max_step = 500, type = "C-classification", scale = TRUE, verbose = FALSE, time_limit = 60, start = FALSE),
...
)

## S3 method for class 'formula'
cv_misvm(formula, data, cost_seq, n_fold, fold_id, ...)

## S3 method for class 'mi_df'
cv_misvm(x, ...)
```
Arguments

- **x**: A data.frame, matrix, or similar object of covariates, where each row represents a sample.

- **y**: A numeric, character, or factor vector of bag labels for each instance. Must satisfy `length(y) == nrow(x)`. Suggest that one of the levels is 1, '1', or TRUE, which becomes the positive class; otherwise, a positive class is chosen and a message will be supplied.

- **bags**: A vector specifying which instance belongs to each bag. Can be a string, numeric, or factor.

- **cost_seq**: A sequence of cost arguments (default $2^{(-2:2)}$) in `misvm()`.

- **n_fold**: The number of folds (default 5). If this is specified, `fold_id` need not be specified.

- **fold_id**: The ids for the specific the fold for each instance. Care must be taken to ensure that ids respect the bag structure to avoid information leakage. If `n_fold` is specified, `fold_id` will be computed automatically.

- **method**: The algorithm to use in fitting (default 'heuristic'). When `method` = 'heuristic', which employs an algorithm similar to Andrews et al. (2003). When `method` = 'mip', the novel MIP method will be used. When `method` = 'qp-heuristic', the heuristic algorithm is computed using the dual SVM. See details.

- **weights**: named vector, or TRUE, to control the weight of the cost parameter for each possible y value. Weights multiply against the cost vector. If TRUE, weights are calculated based on inverse counts of instances with given label, where we only count one positive instance per bag. Otherwise, names must match the levels of y.

- **control**: list of additional parameters passed to the method that control computation with the following components:
  - **kernel**: either a character that describes the kernel ('linear' or 'radial') or a kernel matrix at the instance level.
  - **sigma**: argument needed for radial basis kernel.
  - **nystrom_args**: a list of parameters to pass to `kfm_nystrom()`. This is used when `method` = 'mip' and `kernel` = 'radial' to generate a Nystrom approximation of the kernel features.
  - **max_step**: argument used when `method` = 'heuristic'. Maximum steps of iteration for the heuristic algorithm.
  - **type**: argument used when `method` = 'heuristic'. The type argument is passed to `e1071::svm()`.
  - **scale**: argument used for all methods. A logical for whether to rescale the input before fitting.
  - **verbose**: argument used when `method` = 'mip'. Whether to message output to the console.
  - **time_limit**: argument used when `method` = 'mip'. FALSE, or a time limit (in seconds) passed to `gurobi()` parameters. If FALSE, no time limit is given.
• start argument used when method = 'mip'. If TRUE, the mip program will be warm_started with the solution from method = 'qp-heuristic' to potentially improve speed.

... Arguments passed to or from other methods.

formula a formula with specification mi(y, bags) ~ x which uses the mi function to create the bag-instance structure. This argument is an alternative to the x, y, bags arguments, but requires the data argument. See examples.

data If formula is provided, a data.frame or similar from which formula elements will be extracted.

Value

An object of class cv_misvm. The object contains the following components:

• misvm_fit: A fit object of class misvm trained on the full data with the cross-validated choice of cost parameter. See misvm() for details.
• cost_seq: the input sequence of cost arguments
• cost_aucs: estimated AUC for the models trained for each cost_seq parameter. These are the average of the fold models for that cost, excluding any folds that don’t have both levels of y in the validation set.
• best_cost: The optimal choice of cost parameter, chosen as that which has the maximum AUC. If there are ties, this will pick the smallest cost with maximum AUC.

Methods (by class)

• default: Method for data.frame-like objects
• formula: Method for passing formula
• mi_df: Method for mi_df objects, automatically handling bag names, labels, and all covariates.

Author(s)

Sean Kent, Yifei Liu

See Also

misvm() for fitting without cross-validation.

Examples

set.seed(8)
mil_data <- generate_mild_df(nbag = 20,
  positive_prob = 0.15,
  dist = rep("mvnormal", 3),
  mean = list(rep(1, 10), rep(2, 10)),
  sd_of_mean = rep(0.1, 3))
df <- build_instance_feature(mil_data, seq(0.05, 0.95, length.out = 10))
cost_seq <- 2^seq(-5, 7, length.out = 3)
# Heuristic method

```r
mdl1 <- cv_misvm(x = df[, 4:123], y = df$bag_label,
                 bags = df$bag_name, cost_seq = cost_seq,
                 n_fold = 3, method = "heuristic")
mdl2 <- cv_misvm(mi(bag_label, bag_name) ~ X1_mean + X2_mean + X3_mean, data = df,
                 cost_seq = cost_seq, n_fold = 3)

if (require(gurobi)) {
  # solve using the MIP method
  mdl3 <- cv_misvm(x = df[, 4:123], y = df$bag_label,
                   bags = df$bag_name, cost_seq = cost_seq,
                   n_fold = 3, method = "mip")
}

predict(mdl1, new_data = df, type = "raw", layer = "bag")

# summarize predictions at the bag layer
suppressWarnings(library(dplyr))
   df %>%
     bind_cols(predict(mdl2, df, type = "class")) %>%
     bind_cols(predict(mdl2, df, type = "raw")) %>%
     distinct(bag_name, bag_label, .pred_class, .pred)
```

---

**Printing multiple instance data frames**

**Description**

Specialized print methods for the `mi_df`, `mild_df` classes. These return helpful information such as the number of rows, columns, bags, and instances (for `mild_df` objects).

These methods print the data frame based on the underlying subclass. This allows for additional arguments that can be passed to `print.tbl()` when the subclass is a tibble (`tbl_df`, `tbl`), documented below.

**Usage**

```r
## S3 method for class 'mi_df'
print(x, ...)

## S3 method for class 'mild_df'
print(x, ...)
```

**Arguments**

- `x`  
  Object to format or print.

- `...`  
  Passed to other methods. See `print.tbl()` or details for more information.
Details

The following extra arguments are available when x has subclass tbl:

- **n**: Number of rows to show. If NULL, the default, will print all rows if less than the print_max option. Otherwise, will print as many rows as specified by the print_min option.
- **width**: Width of text output to generate. This defaults to NULL, which means use the width option.
- **max_extra_cols**: Number of extra columns to print abbreviated information for, if the width is too small for the entire tibble. If NULL, the max_extra_cols option is used. The previously defined n_extra argument is soft-deprecated.
- **max_footer_lines**: Maximum number of footer lines. If NULL, the max_footer_lines option is used.

Value

The object passed in x, invisibly. Primarily called to print the object to the console.

Examples

```r
data("ordmvnorm")
print(as_mi_df(ordmvnorm, instance_label = "inst_label"))

print(as_mi_df(ordmvnorm, instance_label = "inst_label"), n = 2)
```

---

**generate_mild_df**

Generate mild_df using multivariate t and normal distributions.

Description

This function samples multiple instance distributional data (a mild_df object) where each row corresponds to a sample from a given instance distribution. Instance distributions can be multivariate t and normal, with mean and variance parameters that can be fixed or sampled based on prior parameters. These instances are grouped into bags and the bag labels follow the standard MI assumption.

Usage

```r
generate_mild_df(
    nbag = 50,
    ninst = 4,
    nsample = 50,
    ncov = 10,
    nimp_pos = 1:ncov,
    nimp_neg = 1:ncov,
    positive_prob = 0.2,
    dist = c("mvt", "mvnormal", "mvnormal"),
    mean = list(rep(0, length(nimp_pos)), rep(0, length(nimp_neg))), 0),
```
generate_mild_df

sd_of_mean = c(0.5, 0.5, 0.5),
cov = list(diag(1, nrow = length(nimp_pos)), diag(1, nrow = length(nimp_neg)), 1),
sample_cov = FALSE,
df_wishart_cov = c(length(nimp_pos), length(nimp_neg), ncov - length(nimp_pos)),
degree = c(3, NA, NA),
positive_bag_prob = NULL,
n_noise_inst = NULL,
...)

Arguments

nbag The number of bags (default 50).
ninst The number of instances for each bag (default 4).
nsample The number of samples for each instance (default 50).
ncov The number of total covariates (default 10).
nimp_pos An index of important covariates for positive instances (default 1:ncov).
nimp_neg An index of important covariates for negative instances (default 1:ncov). (default 1:ncov).
positive_prob A numeric value between 0 and 1 indicating the probability of an instance being positive (default 0.2).
dist A vector (length 3) of distributions for the positive, negative, and remaining instances, respectively. Distributions can be one of 'mvnormal' for multivariate normal or 'mvt' for multivariate student's t.
mean A list (length 3) of mean vectors for the positive, negative, and remaining distributions. mean[[1]] should match nimp_pos in length; mean[[2]] should match nimp_neg in length.
sd_of_mean A vector (length 3) of standard deviations in sampling the mean for positive, negative, and remaining distributions, where the prior is given by mean. Use sd_of_mean = c(0, 0, 0) to keep the mean consistent across all instances.
cov A list (length 3) of covariance matrices for the positive, negative, and remaining distributions. cov[[3]] should be an integer since the dimension of remaining features can vary depending on if the important distribution is positive or negative.
sample_cov A logical value for whether to sample the covariance for each distribution. If FALSE (the default), each covariance is fixed at cov. If TRUE, the prior is given by cov and sampled from a Wishart distribution with df_wishart_cov degrees of freedom to have an expectation of cov.
df_wishart_cov A vector (length 3) of degrees-of-freedom to use in the Wishart covariance matrix sampling.
degree A vector (length 3) of degrees-of-freedom used when any of dist is 'mvt'. This parameter is ignored when dist[[1]] == 'mvnormal', in which case NA can be specified.
positive_bag_prob  
A numeric value between 0 and 1 indicating the probability of a bag being positive. Must be specified jointly with n_noise_inst, in which case positive_bag_prob is ignored. If NULL (the default), instance labels are sampled first according to positive_bag_prob.

n_noise_inst  
An integer indicating the number of negative instances in a positive bag. Must be specified jointly with positive_bag_prob. n_noise_inst should be less than ninst.

...  
Arguments passed to or from other methods.

Details

The first consideration to use this function is to determine the number of bags, instances per bag, and samples per instance using the nbag, ninst, and nsample arguments. Next, one must consider the number of covariates ncov, and how those covariates will differ between instances with positive and negative labels. Some covariates can be common between the positive and negative instances, which we call the remainder distribution. Use nimp_pos and nimp_neg to specify the index of the important (non-remainder) covariates in the distributions with positive and negative instance labels.

The structure of how many instances/bags are positive and negative is determined by positive_bag_prob or the joint specification of positive_bag_prob and n_noise_inst. In the first case, instances labels have independent Bernoulli draws based on positive_prob and bag labels are determined by the standard MI assumption (i.e. positive if any instance in the bag is positive). In the second case, bag labels are drawn independently as Bernoulli with positive_bag_prob chance of success. Each positive bag will be given n_noise_inst values with instance label of 0, and the remaining with instance label of 1.

The remaining arguments are used to determine the distributions used for the positive, negative, and remaining features. Each argument will be a vector of list of length 3 corresponding to these 3 different groups. To create different distributions, the strategy is to first draw the mean parameter from Normal(mean, sd_of_mean * I) and the covariance parameter from Wishart(df_wishart_cov, cov), with expectation equal to cov. Then we can sample i.i.d. draws from the specified distribution (either multivariate normal or student’s t). To ensure that each instance distribution has the same mean, set sd_of_mean to 0. To ensure that each instance distribution has the same covariance, set sample_cov = FALSE.

The final data.frame will have nsample * nbag * ninst rows and ncov + 3 columns including the bag_label, bag_name, instance_name, and ncov sampled covariates.

Value

A mild_df object.

Author(s)

Yifei Liu, Sean Kent

Examples

```R
set.seed(8)
mild_data <- generate_mild_df(nbag = 7, ninst = 3, nsample = 20,
```
ncov = 2,
nimp_pos = 1,
dist = rep("mvnormal", 3),
mean = list(
    rep(5, 1),
    rep(15, 2),
    0
)

library(dplyr)
distinct(mild_data, bag_label, bag_name, instance_name)

split(mild_data[, 4:5], mild_data$instance_name) %>%
sapply(colMeans) %>%
round(2) %>%
t()

kfm_exact

Create an exact kernel feature map

Description
For some kernels, it is possible to create the exact features from given data. This function stores the information needed to build those exact features.

Usage
kfm_exact(kernel = "polynomial", degree = 2, const = 1)

Arguments
kernel A character determining the kernel to use. Currently, only 'radial' is implemented.
degree A numeric value (default 2) that provides the degree for kernel = 'polynomial'
const A numeric value (default 1) for the constant term when kernel = 'polynomial'.

Details
Currently, the following kernels are supported:
• 'polynomial', with degree = d and const = c

Value
An object of class kfm_exact with the following components, returned from the inputs:
• kernel
• degree
• const
kfm_nystrom

Author(s)
Sean Kent

See Also
Other kernel feature map functions: kfm_nystrom()

Examples

```r
df <- data.frame(
  X1 = c(2, 3, 4, 5, 6, 7, 8),
  X2 = c(1, 1.2, 1.3, 1.4, 1.1, 7, 1),
  X3 = rnorm(7)
)

fit <- kfm_exact(kernel = "polynomial", degree = 2, const = 1)
fmc <- build_fm(fit, df)
```

---

**kfm_nystrom**

*Fit a Nyström kernel feature map approximation*

Description

Use the Nyström method to fit a feature map that approximates a given kernel.

Usage

```r
kfm_nystrom(df, m, r, kernel, sampling, ...)
```

## Default S3 method:
```r
kfm_nystrom(
  df,
  m = nrow(df),
  r = m,
  kernel = "radial",
  sampling = "random",
  ...
)
```

## S3 method for class 'mild_df'
```r
kfm_nystrom(
  df,
  m = nrow(df),
  r = m,
  kernel = "radial",
  sampling = "random",
  ...
)
```
Arguments

- **df**: An object containing covariates for training. Usually a data.frame or matrix.
- **m**: The number of examples from df to sample in fitting.
- **r**: The rank of matrix approximation to use. Must be less than or equal to m, the default.
- **kernel**: A character determining the kernel to use. Currently, only 'radial' is implemented.
- **sampling**: A character determining how to sample instances. Default is 'random'. For kfm_nystrom.mild_df(), one can specify sampling = 'stratified' to ensure that samples are chosen evenly from bags and instances. sampling can also be a numeric vector of length m of pre-determined samples.
- ... additional parameters needed for the kernels. See details.

Details

For the ... argument, the additional parameters depend on which kernel is used:

- For kernel = 'radial', specify sigma to define kernel bandwidth.

Value

An object of class kfm_nystrom with the following components:

- df_sub: the sub-sampled version of df
- dv: pre-multiplication matrix which contains information on the eigenvalues and eigenvectors of df_sub
- method: 'nystrom'
- kernel: the input parameter kernel
- kernel_params: parameters passed to ...

Methods (by class)

- default: For use on objects of class data.frame or matrix.
- mild_df: Ignore the information columns 'bag_label', 'bag_name', and 'instance_name' when calculating kernel approximation.

Author(s)

Sean Kent

References


**kme**

*Calculate the kernel mean embedding matrix*

**See Also**

Other kernel feature map functions: `kfm_exact()`

**Examples**

```r
df <- data.frame(
  X1 = c(2, 3, 4, 5, 6, 7, 8),
  X2 = c(1, 1.2, 1.3, 1.4, 1.1, 7, 1),
  X3 = rnorm(7)
)

fit <- kfm_nystrom(df, m = 7, r = 6, kernel = "radial", sigma = 0.05)
fm <- build_fm(fit, df)
```

**Description**

Function to calculate the kernel mean embedding for to distributional data sets. It uses the empirical approximation for the integral

\[
\int_X \int_Y K(x, y) dP_X dQ_Y
\]

for a given kernel \(K(\cdot, \cdot)\). Currently only supports radial basis function kernel for fast computation.

**Usage**

```r
## Default S3 method:
kme(df, df2 = NULL, sigma = 0.05, ...)

## S3 method for class 'mild_df'
kme(df, df2 = NULL, sigma = 0.05, ...)
```

**Arguments**

- **df**: A data.frame of `mild_df` object, must have column 'instance_name' which defines the instances.
- **df2**: A data.frame, `mild_df` object, or NULL (default NULL).
- **sigma**: The parameter for 'radial' kernel (default 0.05).
- **...**: Additional arguments passed to methods.

**Details**

If `df2 = NULL`, calculate the kernel mean embedding matrix of `(df, df)` otherwise calculate `(df, df2)`
Value

A matrix of kernel mean embedding at the instance level.

Methods (by class)

- default: Default S3 method
- mild_df: S3 method for class mild_df

Author(s)

Yifei Liu, Sean Kent

Examples

```r
x = data.frame('instance_name' = c('inst_1', 'inst_2', 'inst_1'), 'X1' = c(-0.4, 0.5, 2))
kme(x)
mild_df1 <- generate_mild_df(nbag = 10, positive_degree = 3)
kme(mild_df1)
```

---

### mi

Create an mi object

Description

Create an mi object, usually used as a response variable in a model formula.

Usage

```r
mi(bag_label, bag_name)
```

Arguments

- `bag_label`: The bag label or response, recorded as 0 = negative, 1 = positive.
- `bag_name`: A unique bag identifier for each instance.

Value

An object of class mi. Currently, no methods are implemented for this.

Author(s)

Sean Kent
See Also

Other multiple instance formula helper functions: \texttt{mild()}

Examples

mil_data <- generate_mild_df(positive_degree = 3, nbag = 10)
with(mil_data, head(mi(bag_label, bag_name))))
df <- get_all_vars(mi(bag_label, bag_name) ~ X1 + X2, data = mil_data)
head(df)

mild \hspace{1cm} \textit{Create a mild object}

Description

Create a mild object, usually used as a response variable in a model formula.

Usage

\texttt{mild(bag_label, bag_name, instance_name)}

Arguments

- \texttt{bag_label} \hspace{1cm} The bag label or response, recorded as 0 = negative, 1 = positive.
- \texttt{bag_name} \hspace{1cm} A unique bag identifier for each instance.
- \texttt{instance_name} \hspace{1cm} A unique instance identifier for each sample.

Value

An object of class mild. Currently, no methods are implemented for this.

Author(s)

Sean Kent

See Also

Other multiple instance formula helper functions: \texttt{mi()}

Examples

mil_data <- generate_mild_df(positive_degree = 3, nbag = 10)
with(mil_data, head(mild(bag_label, bag_name, instance_name))))
df <- get_all_vars(mild(bag_label, bag_name) ~ X1 + X2, data = mil_data)
head(df)
mild_df

Build a MILD data frame

Description

mild_df() constructs a data frame that corresponds to Multiple Instance Learning with Distributional Instances (MILD) data. A mild_df object must have three special columns:

- bag_label, determines the label of each bag, typically from c(0, 1)
- bag_name, character or factor that specifies the bag that each sample belongs to.
- instance_name, character or factor that specifies the instance that each sample belongs to.

Usage

mild_df(
  bag_label = character(),
  bag_name = character(),
  instance_name = character(),
  ..., 
  instance_label = NULL
)

Arguments

- bag_label A character, factor, or numeric vector.
- bag_name A character or factor vector.
- instance_name A character or factor vector.
- ... A set of name-value pairs. These construct the covariates for a mild_df.
- instance_label A character, factor, or numeric vector, or NULL.

Details

We refer to the rows of a mild_df as samples, since they are thought of as draws from the distribution that determines each instance. Each instance is contained in a bag, with a corresponding label. Instance labels can be provided, but they will be pulled in as an attribute.

Value

A 'mild_df' object. This data.frame-like has columns bag_label, bag_name, instance_name, and those specified in .... It also inherits from the 'tbl_df' and 'tbl' classes.

Author(s)

Yifei Liu, Sean Kent
See Also

- `as_mild_df()` to convert data.frames to milddfs.
- `generate_mild_df()` for simulating a mild_df object.
- `summarize_samples()` for summarizing the mild_df into a multiple instance learning data set.

Examples

```r
mild_df('bag_label' = factor(c(1, 1, 0)),
     'bag_name' = c(rep('bag_1', 2), 'bag_2'),
     'instance_name' = c('bag_1_inst_1', 'bag_1_inst_2', 'bag_2_inst_1'),
     'X1' = c(-0.4, 0.5, 2),
     'instance_label' = c(0, 1, 0))
```

### mior

Fit MIOR model to the data

#### Description

This function fits the MIOR model, proposed by Xiao Y, Liu B, and Hao Z (2018) in "Multiple-instance Ordinal Regression". MIOR is a modified SVM framework with parallel, ordered hyper-planes where the error terms are based only on the instance closest to a midpoint between hyper-planes.

#### Usage

```r
## Default S3 method:
mior(
  x,
  y,
  bags,
  cost = 1,
  cost_eta = 1,
  method = "qp-heuristic",
  weights = NULL,
  control = list(kernel = "linear", sigma = if (is.vector(x)) 1 else 1/ncol(x),
                 max_step = 500, scale = TRUE, verbose = FALSE, time_limit = 60,
                 option = c("corrected", "xiao")),
  ...
)
## S3 method for class 'formula'
mior(formula, data, ...)
## S3 method for class 'mi_df'
mior(x, ...)
```
Arguments

x A data.frame, matrix, or similar object of covariates, where each row represents an instance. If a mi_df object is passed, y, bags are automatically extracted, and all other columns will be used as predictors.

y A numeric, character, or factor vector of bag labels for each instance. Must satisfy length(y) == nrow(x). Suggest that one of the levels is 1, '1', or TRUE, which becomes the positive class; otherwise, a positive class is chosen and a message will be supplied.

bags A vector specifying which instance belongs to each bag. Can be a string, numeric, of factor.

cost The cost parameter in SVM. If method = 'heuristic', this will be fed to kernlab::ksvm(), otherwise it is similarly in internal functions.

cost_eta The additional cost parameter in MIOR which controls how far away the first and last separating hyperplanes are relative to other costs.

method The algorithm to use in fitting (default 'heuristic'). When method = 'heuristic', which employs an algorithm similar to Andrews et al. (2003). When method = 'mip', the novel MIP method will be used. When method = 'qp-heuristic', the heuristic algorithm is computed using the dual SVM. See details.

weights named vector, or TRUE, to control the weight of the cost parameter for each possible y value. Weights multiply against the cost vector. If TRUE, weights are calculated based on inverse counts of instances with given label, where we only count one positive instance per bag. Otherwise, names must match the levels of y.

control list of additional parameters passed to the method that control computation with the following components:

• kernel either a character the describes the kernel ('linear' or 'radial') or a kernel matrix at the instance level.
• sigma argument needed for radial basis kernel.
• max_step argument used when method = 'heuristic'. Maximum steps of iteration for the heuristic algorithm.
• scale argument used for all methods. A logical for whether to rescale the input before fitting.
• verbose argument used when method = 'mip'. Whether to message output to the console.
• time_limit argument used when method = 'mip'. FALSE, or a time limit (in seconds) passed to gurobi() parameters. If FALSE, no time limit is given.
• option argument the controls the constraint calculation. See details.

... Arguments passed to or from other methods.

formula a formula with specification mi(y, bags) ~ x which uses the mi function to create the bag-instance structure. This argument is an alternative to the x, y, bags arguments, but requires the data argument. See examples.

data If formula is provided, a data.frame or similar from which formula elements will be extracted
Details

Predictions (see `predict.mior()`) are determined by considering the smallest distance from each point to the midpoint hyperplanes across all instances in the bag. The prediction corresponds to the hyperplane having such a minimal distance.

It appears as though an error in Equation (12) persists to the dual form in (21). A corrected version of this dual formulation can be used with `control$option = 'corrected'`, or the formulation as written can be used with `control$option = 'xiao'`.

Value

An object of class `mior` The object contains at least the following components:

- `gurobi_fit`: A fit from model optimization that includes relevant components.
- `call_type`: A character indicating which method `misvm()` was called with.
- `features`: The names of features used in training.
- `levels`: The levels of y that are recorded for future prediction.
- `cost`: The cost parameter from function inputs.
- `weights`: The calculated weights on the cost parameter.
- `repr_inst`: The instances from positive bags that are selected to be most representative of the positive instances.
- `n_step`: If method `%in% c('heuristic', 'qp-heuristic'), the total steps used in the heuristic algorithm.
- `x_scale`: If `scale = TRUE`, the scaling parameters for new predictions.

Methods (by class)

- `default`: Method for data.frame-like objects
- `formula`: Method for passing formula
- `mi_df`: Method for `mi_df` objects, automatically handling bag names, labels, and all covariates.

Author(s)

Sean Kent

References


See Also

`predict.misvm()` for prediction on new data.
Examples

```r
if (require(gurobi)) {
  set.seed(8)
  # make some data
  n <- 15
  X <- rbind(
    mvtnorm::rmvnorm(n/3, mean = c(4, -2, 0)),
    mvtnorm::rmvnorm(n/3, mean = c(0, 0, 0)),
    mvtnorm::rmvnorm(n/3, mean = c(-2, 1, 0))
  )
  score <- X %*% c(2, -1, 0)
  y <- as.numeric(cut(score, c(-Inf, quantile(score, probs = 1:2 / 3), Inf)))
  bags <- 1:length(y)

  # add in points outside boundaries
  X <- rbind(
    X,
    mvtnorm::rmvnorm(n, mean = c(6, -3, 0)),
    mvtnorm::rmvnorm(n, mean = c(-6, 3, 0))
  )
  y <- c(y, rep(-1, 2*n))
  bags <- rep(bags, 3)
  repr <- c(rep(1, n), rep(0, 2*n))

  y_bag <- classify_bags(y, bags, condense = FALSE)
  mdl1 <- mior(X, y_bag, bags)
  predict(mdl1, X, new_bags = bags)
}
```

mismm

**Fit MILD-SVM model to the data**

Description

This function fits the MILD-SVM model, which takes a multiple-instance learning with distributions (MILD) data set and fits a modified SVM to it. The MILD-SVM methodology is based on research in progress.

Usage

```r
# Default S3 method:
mismm(
  x,
  y,
  bags,
  instances,
  cost = 1,
)```
mismm

method = c("heuristic", "mip", "qp-heuristic"),
weights = TRUE,
control = list(kernel = "radial", sigma = if (is.vector(x)) 1 else 1/ncol(x),
nystrom_args = list(m = nrow(x), r = nrow(x), sampling = "random"), max_step = 500,
    scale = TRUE, verbose = FALSE, time_limit = 60, start = FALSE),
...

## S3 method for class 'formula'
mismm(formula, data, ...)

## S3 method for class 'mild_df'
mismm(x, ...)

Arguments

- **x**: A data.frame, matrix, or similar object of covariates, where each row represents a sample. If a mild_df object is passed, y, bags, instances are automatically extracted, and all other columns will be used as predictors.

- **y**: A numeric, character, or factor vector of bag labels for each instance. Must satisfy `length(y) == nrow(x)`. Suggest that one of the levels is 1, '1', or TRUE, which becomes the positive class; otherwise, a positive class is chosen and a message will be supplied.

- **bags**: A vector specifying which instance belongs to each bag. Can be a string, numeric, or factor.

- **instances**: A vector specifying which samples belong to each instance. Can be a string, numeric, or factor.

- **cost**: The cost parameter in SVM. If method = 'heuristic', this will be fed to kernlab::ksvm(), otherwise it is similarly in internal functions.

- **method**: The algorithm to use in fitting (default 'heuristic'). When method = 'heuristic', the algorithm iterates between selecting positive witnesses and solving an underlying smm() problem. When method = 'mip', the novel MIP method will be used. When method = 'qp-heuristic', the heuristic algorithm is computed using a slightly modified dual SMM. See details.

- **weights**: named vector, or TRUE, to control the weight of the cost parameter for each possible y value. Weights multiply against the cost vector. If TRUE, weights are calculated based on inverse counts of instances with given label, where we only count one positive instance per bag. Otherwise, names must match the levels of y.

- **control**: list of additional parameters passed to the method that control computation with the following components:
  - **kernel**: either a character the describes the kernel ('linear' or 'radial') or a kernel matrix at the instance level.
  - **sigma**: argument needed for radial basis kernel.
  - **nystrom_args**: a list of parameters to pass to kfm_nystrom(). This is used when method = 'mip' and kernel = 'radial' to generate a Nystrom approximation of the kernel features.
• `max_step` argument used when `method = 'heuristic'`. Maximum steps of iteration for the heuristic algorithm.
• `scale` argument used for all methods. A logical for whether to rescale the input before fitting.
• `verbose` argument used when `method = 'mip'`. Whether to message output to the console.
• `time_limit` argument used when `method = 'mip'`. FALSE, or a time limit (in seconds) passed to gurobi() parameters. If FALSE, no time limit is given.
• `start` argument used when `method = 'mip'`. If TRUE, the mip program will be warm_started with the solution from `method = 'qp-heuristic'` to potentially improve speed.

... Arguments passed to or from other methods.

`formula` A formula with specification `mild(y, bags, instances) ~ x` which uses the `mild` function to create the bag-instance structure. This argument is an alternative to the `x, y, bags, instances` arguments, but requires the `data` argument. See examples.

`data` If `formula` is provided, a data.frame or similar from which `formula` elements will be extracted.

### Details

Several choices of fitting algorithm are available, including a version of the heuristic algorithm proposed by Andrews et al. (2003) and a novel algorithm that explicitly solves the mixed-integer programming (MIP) problem using the gurobi package optimization back-end.

### Value

An object of class `mismm` The object contains at least the following components:

• `_fit`: A fit object depending on the `method` parameter. If `method = 'heuristic'`, this will be a `ksvm` fit from the kernlab package. If `method = 'mip'` this will be `gurobi_fit` from a model optimization.
• `call_type`: A character indicating which method `mism()` was called with.
• `x`: The training data needed for computing the kernel matrix in prediction.
• `features`: The names of features used in training.
• `levels`: The levels of `y` that are recorded for future prediction.
• `cost`: The cost parameter from function inputs.
• `weights`: The calculated weights on the `cost` parameter.
• `sigma`: The radial basis function kernel parameter.
• `repr_inst`: The instances from positive bags that are selected to be most representative of the positive instances.
• `n_step`: If `method %in% c('heuristic', 'qp-heuristic')`, the total steps used in the heuristic algorithm.
• `useful_inst_idx`: The instances that were selected to represent the bags in the heuristic fitting.
• `inst_order`: A character vector that is used to modify the ordering of input data.
• `x_scale`: If `scale = TRUE`, the scaling parameters for new predictions.

**Methods (by class)**

- `default`: Method for `data.frame`-like objects
- `formula`: Method for passing formula
- `mild_df`: Method for `mild_df` objects

**Author(s)**

Sean Kent, Yifei Liu

**References**


**See Also**

`predict.mismm()` for prediction on new data.

**Examples**

```r
set.seed(8)
mil_data <- generate_mild_df(nbag = 15, nsample = 20, positive_prob = 0.15, 
                             sd_of_mean = rep(0.1, 3))

# Heuristic method
mdl1 <- mismm(mil_data)
mil2 <- mismm(mild(bag_label, bag_name, instance_name) ~ X1 + X2 + X3, data = mil_data)

# MIP method
if (require(gurobi)) {
m3 <- mismm(mil_data, method = "mip", control = list(nystrom_args = list(m = 10, r = 10)))
predict(m3, mil_data)
}

predict(mdl1, new_data = mil_data, type = "raw", layer = "bag")

# summarize predictions at the bag layer
library(dplyr)
mil_data %>%
  bind_cols(predict(mdl2, mil_data, type = "class")) %>%
  bind_cols(predict(mdl2, mil_data, type = "raw")) %>%
  distinct(bag_name, bag_label, .pred_class, .pred)
```
misvm

Fit MI-SVM model to the data

Description

This function fits the MI-SVM model, first proposed by Andrews et al. (2003). It is a variation on the traditional SVM framework that carefully treats data from the multiple instance learning paradigm, where instances are grouped into bags, and a label is only available for each bag.

Usage

## Default S3 method:
misvm(
x,  
y,  
bags,  
cost = 1,  
method = c("heuristic", "mip", "qp-heuristic"),  
weights = TRUE,  
control = list(kernel = "linear", sigma = if (is.vector(x)) 1 else 1/ncol(x),  
nystrom_args = list(m = nrow(x), r = nrow(x), sampling = "random"), max_step = 500,  
type = "C-classification", scale = TRUE, verbose = FALSE, time_limit = 60, start =  
FALSE),  
...)

## S3 method for class 'formula'
misvm(formula, data, ...)

## S3 method for class 'mi_df'
misvm(x, ...)

## S3 method for class 'mild_df'
misvm(x, .fns = list(mean = mean, sd = stats::sd), cor = FALSE, ...)

Arguments

- **x**: A data frame, matrix, or similar object of covariates, where each row represents an instance. If a mi_df object is passed, y, bags are automatically extracted, and all other columns will be used as predictors. If a mild_df object is passed, y, bags, instances are automatically extracted, and all other columns will be used as predictors.

- **y**: A numeric, character, or factor vector of bag labels for each instance. Must satisfy length(y) == nrow(x). Suggest that one of the levels is 1, '1', or TRUE, which becomes the positive class; otherwise, a positive class is chosen and a message will be supplied.
misvm

bags A vector specifying which instance belongs to each bag. Can be a string, numeric, or factor.
cost The cost parameter in SVM. If method = 'heuristic', this will be fed to kernlab::ksvm(), otherwise it is similarly in internal functions.
method The algorithm to use in fitting (default 'heuristic'). When method = 'heuristic', which employs an algorithm similar to Andrews et al. (2003). When method = 'mip', the novel MIP method will be used. When method = 'qp-heuristic', the heuristic algorithm is computed using the dual SVM. See details.
weights named vector, or TRUE, to control the weight of the cost parameter for each possible y value. Weights multiply against the cost vector. If TRUE, weights are calculated based on inverse counts of instances with given label, where we only count one positive instance per bag. Otherwise, names must match the levels of y.
control list of additional parameters passed to the method that control computation with the following components:
  • kernel either a character the describes the kernel ('linear' or 'radial') or a kernel matrix at the instance level.
  • sigma argument needed for radial basis kernel.
  • nystrom_args a list of parameters to pass to kfm_nystrom(). This is used when method = 'mip' and kernel = 'radial' to generate a Nystrom approximation of the kernel features.
  • max_step argument used when method = 'heuristic'. Maximum steps of iteration for the heuristic algorithm.
  • type: argument used when method = 'heuristic'. The type argument is passed to e1071::svm().
  • scale argument used for all methods. A logical for whether to rescale the input before fitting.
  • verbose argument used when method = 'mip'. Whether to message output to the console.
  • time_limit argument used when method = 'mip'. FALSE, or a time limit (in seconds) passed to gurobi() parameters. If FALSE, no time limit is given.
  • start argument used when method = 'mip'. If TRUE, the mip program will be warm_started with the solution from method = 'qp-heuristic' to potentially improve speed.

... Arguments passed to or from other methods.
formula a formula with specification mi(y, bags) ~ x which uses the mi function to create the bag-instance structure. This argument is an alternative to the x, y, bags arguments, but requires the data argument. See examples.
data If formula is provided, a data.frame or similar from which formula elements will be extracted.
.fns (argument for misvm.mild_df() method) list of functions to summarize instances over.
cor (argument for misvm.mild_df() method) logical, whether to include correlations between all features in the summarization.
Details

Several choices of fitting algorithm are available, including a version of the heuristic algorithm proposed by Andrews et al. (2003) and a novel algorithm that explicitly solves the mixed-integer programming (MIP) problem using the gurobi package optimization back-end.

Value

An object of class `misvm`. The object contains at least the following components:

- `*_fit`: A fit object depending on the method parameter. If method = 'heuristic', this will be an `svm` fit from the e1071 package. If method = 'mip', 'qp-heuristic' this will be `gurobi_fit` from a model optimization.
- `call_type`: A character indicating which method `misvm()` was called with.
- `features`: The names of features used in training.
- `levels`: The levels of y that are recorded for future prediction.
- `cost`: The cost parameter from function inputs.
- `weights`: The calculated weights on the cost parameter.
- `repr_inst`: The instances from positive bags that are selected to be most representative of the positive instances.
- `n_step`: If method %in% c('heuristic', 'qp-heuristic'), the total steps used in the heuristic algorithm.
- `x_scale`: If scale = TRUE, the scaling parameters for new predictions.

Methods (by class)

- `default`: Method for data.frame-like objects
- `formula`: Method for passing formula
- `mi_df`: Method for `mi_df` objects, automatically handling bag names, labels, and all covariates.
- `mild_df`: Method for `mild_df` objects. Summarize samples to the instance level based on specified functions, then perform `misvm()` on instance level data.

Author(s)

Sean Kent, Yifei Liu

References


See Also

- `predict.misvm()` for prediction on new data.
- `cv_misvm()` for cross-validation fitting.
Examples

```r
set.seed(8)
mil_data <- generate_mild_df(nbag = 20, 
  positive_prob = 0.15, 
  sd_of_mean = rep(0.1, 3))
df <- build_instance_feature(mil_data, seq(0.05, 0.95, length.out = 10))

# Heuristic method
mdl1 <- misvm(x = df[, 4:123], y = df$bag_label, 
  bags = df$bag_name, method = "heuristic")
mdl2 <- misvm(mi(bag_label, bag_name) ~ X1_mean + X2_mean + X3_mean, data = df)

# MIP method
if (require(gurobi)) {
  mdl3 <- misvm(x = df[, 4:123], y = df$bag_label, 
    bags = df$bag_name, method = "mip")
}
predict(mdl1, new_data = df, type = "raw", layer = "bag")

# summarize predictions at the bag layer
library(dplyr)
df %>%
  bind_cols(predict(mdl2, df, type = "class")) %>%
  bind_cols(predict(mdl2, df, type = "raw")) %>%
  distinct(bag_name, bag_label, .pred_class, .pred)
```

misvm_orova

Fit MI-SVM model to ordinal outcome data using One-vs-All

Description

This function uses the one-vs-all multiclass classification strategy to fit a series of MI-SVM models for predictions on ordinal outcome data. For an ordinal outcome with K levels, we fit K MI-SVM models to predict an individual level vs not.

Usage

```r
## Default S3 method:
misvm_orova(
  x, 
  y, 
  bags, 
  cost = 1, 
  method = c("heuristic", "mip", "qp-heuristic"), 
  weights = TRUE, 
  control = list(kernel = "linear", sigma = if (is.vector(x)) 1 else 1/ncol(x),
  ...) 
```

misvm_orova

Fit MI-SVM model to ordinal outcome data using One-vs-All

```r
## Default S3 method:
misvm_orova(
  x, 
  y, 
  bags, 
  cost = 1, 
  method = c("heuristic", "mip", "qp-heuristic"), 
  weights = TRUE, 
  control = list(kernel = "linear", sigma = if (is.vector(x)) 1 else 1/ncol(x),
  ...) 
```
misvm_orova

nystron_args = list(m = nrow(x), r = nrow(x), sampling = "random"), max_step = 500,
type = "C-classification", scale = TRUE, verbose = FALSE, time_limit = 60, start = FALSE),
...
)

## S3 method for class 'formula'
misvm_orova(formula, data, ...)

## S3 method for class 'mi_df'
misvm_orova(x, ...)

Arguments

x          A data.frame, matrix, or similar object of covariates, where each row represents
an instance. If a mi_df object is passed, y, bags are automatically extracted,
and all other columns will be used as predictors.

y          A numeric, character, or factor vector of bag labels for each instance. Must
satisfy length(y) == nrow(x). Suggest that one of the levels is 1, '1', or TRUE,
which becomes the positive class; otherwise, a positive class is chosen and a
message will be supplied.

bags       A vector specifying which instance belongs to each bag. Can be a string, nu-
meric, of factor.

cost       The cost parameter in SVM. If method = 'heuristic', this will be fed to kernlab::ksvm(),
otherwise it is similarly in internal functions.

method     The algorithm to use in fitting (default 'heuristic'). When method = 'heuristic',
which employs an algorithm similar to Andrews et al. (2003). When method = 
'mip', the novel MIP method will be used. When method = 'qp-heuristic,'
the heuristic algorithm is computed using the dual SVM. See details.

weights    named vector, or TRUE, to control the weight of the cost parameter for each
possible y value. Weights multiply against the cost vector. If TRUE, weights are
calculated based on inverse counts of instances with given label, where we only
count one positive instance per bag. Otherwise, names must match the levels of
y.

control    list of additional parameters passed to the method that control computation with
the following components:

  • kernel either a character the describes the kernel ('linear' or 'radial') or a
    kernel matrix at the instance level.
  • sigma argument needed for radial basis kernel.
  • nystrom_args a list of parameters to pass to kfm_nystrom(). This is used
    when method = 'mip' and kernel = 'radial' to generate a Nystrom ap-
    proximation of the kernel features.
  • max_step argument used when method = 'heuristic'. Maximum steps
    of iteration for the heuristic algorithm.
  • type: argument used when method = 'heuristic'. The type argument is
    passed to e1071::svm().
**misvm_orova**

- scale argument used for all methods. A logical for whether to rescale the input before fitting.
- verbose argument used when method = 'mip'. Whether to message output to the console.
- time_limit argument used when method = 'mip'. FALSE, or a time limit (in seconds) passed to gurobi() parameters. If FALSE, no time limit is given.
- start argument used when method = 'mip'. If TRUE, the mip program will be warm_started with the solution from method = 'qp-heuristic' to potentially improve speed.

... Arguments passed to or from other methods.

**formula**

A formula with specification mi(y, bags) ~ x which uses the mi function to create the bag-instance structure. This argument is an alternative to the x, y, bags arguments, but requires the data argument. See examples.

**data**

If formula is provided, a data.frame or similar from which formula elements will be extracted

**Value**

An object of class misvm_orova The object contains at least the following components:

- fits: a list of misvm objects with length equal to the number of classes in y. See misvm() for details on the misvm object.
- call_type: A character indicating which method misvm_orova() was called with.
- features: The names of features used in training.
- levels: The levels of y that are recorded for future prediction.

**Methods (by class)**

- default: Method for data.frame-like objects
- formula: Method for passing formula
- mi_df: Method for mi_df objects, automatically handling bag names, labels, and all covariates.

**Author(s)**

Sean Kent

**References**


**See Also**

predict.misvm_orova() for prediction on new data.
Examples

data(“ordmvnorm”)  
x <- ordmvnorm[, 3:7]  
y <- ordmvnorm$bag_label  
bags <- ordmvnorm$bag_name

mdl1 <- misvm_orova(x, y, bags)  
predict(mdl1, x, new_bags = bags)

---

mi_df

Build a multiple instance (MI) data frame

Description

mi_df() constructs a data frame that corresponds to Multiple Instance (MI) data. A mi_df object must have two special columns:

- bag_label, determines the label of each bag, typically from \(\{0, 1\}\)
- bag_name, character or factor that specifies the bag that each sample belongs to.

Usage

mi_df(
  bag_label = character(),
  bag_name = character(),
  ...
)

Arguments

- bag_label: A character, factor, or numeric vector.
- bag_name: A character or factor vector.
- ...: A set of name-value pairs. These construct the covariates for a mi_df.
- instance_label: A character, factor, or numeric vector, or NULL.

Details

We refer to the rows of a mi_df as instances. Each instance is contained in a bag, with a corresponding label. Bags will typically have several instances within them. Instance labels can be provided, but they will be pulled in as an attribute.

Value

A `mi_df` object. This data.frame-like has columns bag_label, bag_name, and those specified in .... It also inherits from the `tbl_df` and `tbl` classes.
omisvm

Author(s)
Sean Kent

See Also
• as.mi_df() to convert data.frames to midfs.

Examples

```r
mi_df('bag_label' = factor(c(1, 1, 0)),
     'bag_name' = c(rep('bag_1', 2), 'bag_2'),
     'X1' = c(-0.4, 0.5, 2),
     'instance_label' = c(0, 1, 0))
```

---

omisvm **Fit MI-SVM-OR model to ordinal outcome data**

Description

This function fits a modification of MI-SVM to ordinal outcome data based on the research method proposed by Kent and Yu.

Usage

```r
## Default S3 method:
omisvm(
x, y, bags,
cost = 1,
h = 1,
s = Inf,
method = c("qp-heuristic"),
weights = TRUE,
control = list(kernel = "linear", sigma = if (is.vector(x)) 1 else 1/ncol(x),
max_step = 500, type = "C-classification", scale = TRUE, verbose = FALSE, time_limit = 60),
...
)

## S3 method for class 'formula'
omisvm(formula, data, ...)

## S3 method for class 'mi_df'
omisvm(x, ...)
```
Arguments

x  A data.frame, matrix, or similar object of covariates, where each row represents an instance. If a mi_df object is passed, y, bags are automatically extracted, and all other columns will be used as predictors.

y  A numeric, character, or factor vector of bag labels for each instance. Must satisfy length(y) == nrow(x). Suggest that one of the levels is 1, '1', or TRUE, which becomes the positive class; otherwise, a positive class is chosen and a message will be supplied.

bags  A vector specifying which instance belongs to each bag. Can be a string, numeric, or factor.

cost  The cost parameter in SVM. If method = 'heuristic', this will be fed to kernlab::ksvm(), otherwise it is similarly in internal functions.

h  A scalar that controls the trade-off between maximizing the margin and minimizing distance between hyperplanes.

s  An integer for how many replication points to add to the dataset. If k represents the number of labels in y, must have 1 <= s <= k-1. The default, Inf, uses the maximum number of replication points, k-1.

method  The algorithm to use in fitting (default 'heuristic'). When method = 'heuristic', which employs an algorithm similar to Andrews et al. (2003). When method = 'mip', the novel MIP method will be used. When method = 'qp-heuristic', the heuristic algorithm is computed using the dual SVM. See details.

weights  named vector, or TRUE, to control the weight of the cost parameter for each possible y value. Weights multiply against the cost vector. If TRUE, weights are calculated based on inverse counts of instances with given label, where we only count one positive instance per bag. Otherwise, names must match the levels of y.

control  list of additional parameters passed to the method that control computation with the following components:

  • kernel either a character the describes the kernel ('linear' or 'radial') or a kernel matrix at the instance level.
  • sigma argument needed for radial basis kernel.
  • nystrom_args a list of parameters to pass to kfm_nystrom(). This is used when method = 'mip' and kernel = 'radial' to generate a Nystrom approximation of the kernel features.
  • max_step argument used when method = 'heuristic'. Maximum steps of iteration for the heuristic algorithm.
  • type: argument used when method = 'heuristic'. The type argument is passed to e1071::svm().
  • scale argument used for all methods. A logical for whether to rescale the input before fitting.
  • verbose argument used when method = 'mip'. Whether to message output to the console.
  • time_limit argument used when method = 'mip'. FALSE, or a time limit (in seconds) passed to gurobi() parameters. If FALSE, no time limit is given.
omisvm

- start argument used when method = 'mip'. If TRUE, the mip program will be warm_started with the solution from method = 'qp-heuristic' to potentially improve speed.

... Arguments passed to or from other methods.

formula a formula with specification mi(y, bags) ~ x which uses the mi function to create the bag-instance structure. This argument is an alternative to the x, y, bags arguments, but requires the data argument. See examples.

data If formula is provided, a data.frame or similar from which formula elements will be extracted

Details

Currently, the only method available is a heuristic algorithm in linear SVM space. Additional methods should be available shortly.

Value

An object of class omisvm. The object contains at least the following components:

- _x_fit: A fit object depending on the method parameter. If method = 'qp-heuristic' this will be gurobifit from a model optimization.
- call_type: A character indicating which method omisvm() was called with.
- features: The names of features used in training.
- levels: The levels of y that are recorded for future prediction.
- cost: The cost parameter from function inputs.
- weights: The calculated weights on the cost parameter.
- repr_inst: The instances from positive bags that are selected to be most representative of the positive instances.
- n_step: If method == 'qp-heuristic', the total steps used in the heuristic algorithm.
- x_scale: If scale = TRUE, the scaling parameters for new predictions.

Methods (by class)

- default: Method for data.frame-like objects
- formula: Method for passing formula
- mi_df: Method for mi_df objects, automatically handling bag names, labels, and all covariates.

Author(s)

Sean Kent

See Also

predict.omisvm() for prediction on new data.
Examples

```r
if (require(gurobi)) {
  data("ordmvnorm")
  x <- ordmvnorm[, 3:7]
  y <- ordmvnorm$bag_label
  bags <- ordmvnorm$bag_name

  mdl1 <- omisvm(x, y, bags, weights = NULL)
  predict(mdl1, x, new_bags = bags)
}
```

ordmvnorm

Sample ordinal MIL data using mvnorm

Description

A data set that demonstrates the ordinal multiple-instance learning structure with feature columns randomly sampled from a multivariate normal distribution.

Usage

ordmvnorm

Format

An MI data frame with 1000 rows 8 variables, and 5 bags. Instance labels can be accessed via attr(ordmvnorm, "instance_label").

- **bag_label**: outcome label at the bag level. This is the maximum of the inst_label for each bag
- **bag_name**: indicator of each bag
- **V1**: Variable with mean equal to 2 * inst_label
- **V2**: Variable with mean equal to -1 * inst_label
- **V3**: Variable with mean equal to 1 * inst_label
- **V4**: Variable with mean 0, essentially noise
- **V5**: Variable with mean 0, essentially noise
predict.cv_misvm

Description

Predict method for cv_misvm object

Usage

## S3 method for class 'cv_misvm'
predict(
  object,  
  new_data,  
  type = c("class", "raw"),  
  layer = c("bag", "instance"),  
  new_bags = "bag_name",  
  ...  
)

Arguments

- **object**: An object of class cv_misvm.
- **new_data**: A data frame to predict from. This needs to have all of the features that the data was originally fitted with.
- **type**: If 'class', return predicted values with threshold of 0 as -1 or +1. If 'raw', return the raw predicted scores.
- **layer**: If 'bag', return predictions at the bag level. If 'instance', return predictions at the instance level.
- **new_bags**: A character or character vector. Can specify a singular character that provides the column name for the bag names in new_data (default 'bag_name'). Can also specify a vector of length nrow(new_data) that has bag name for each row.
- **...**: Arguments passed to or from other methods.

Value

A tibble with nrow(new_data) rows. If type = 'class', the tibble will have a column '.pred_class'. If type = 'raw', the tibble will have a column '.pred'.

Author(s)

Sean Kent
Examples

mil_data <- generate_mild_df(
  nbag = 10,
  nsample = 20,
  positive_degree = 3
)
df1 <- build_instance_feature(mil_data, seq(0.05, 0.95, length.out = 10))
mdl1 <- cv_misvm(x = df1[, 4:123], y = df1$bag_label,
  bags = df1$bag_name, cost_seq = 2^(-2:2),
  n_fold = 3, method = "heuristic")
predict(mdl1, new_data = df1, type = "raw", layer = "bag")

# summarize predictions at the bag layer
suppressWarnings(library(dplyr))
df1 %>%
  bind_cols(predict(mdl1, df1, type = "class")) %>%
  bind_cols(predict(mdl1, df1, type = "raw")) %>
  distinct(bag_name, bag_label, .pred_class, .pred)

---

`predict.mior`  
*Predict method for mior object*

Description

Predict method for mior object

Usage

```r
## S3 method for class 'mior'
predict(
  object,
  new_data,
  type = c("class", "raw"),
  layer = c("bag", "instance"),
  new_bags = "bag_name",
  ...
)
```

Arguments

- `object`  
  An object of class mior

- `new_data`  
  A data frame to predict from. This needs to have all of the features that the data was originally fitted with.

- `type`  
  If 'class', return predicted values with threshold of 0 as -1 or +1. If 'raw', return the raw predicted scores.
predict.mior

layer

If 'bag', return predictions at the bag level. If 'instance', return predictions at the instance level.

new_bags

A character or character vector. Can specify a singular character that provides the column name for the bag names in new_data (default 'bag_name'). Can also specify a vector of length nrow(new_data) that has bag name for each row.

... Arguments passed to or from other methods.

Details

When the object was fitted using the formula method, then the parameters new_bags and new_instances are not necessary, as long as the names match the original function call.

Value

A tibble with nrow(new_data) rows. If type = 'class', the tibble will have a column .pred_class. If type = 'raw', the tibble will have a column .pred.

Author(s)

Sean Kent

See Also

mior() for fitting the mior object.

Examples

if (require(gurobi)) {
  set.seed(8)
  # make some data
  n <- 15
  X <- rbind(
    mvtnorm::rmvnorm(n/3, mean = c(4, -2, 0)),
    mvtnorm::rmvnorm(n/3, mean = c(0, 0, 0)),
    mvtnorm::rmvnorm(n/3, mean = c(-2, 1, 0))
  )
  score <- X %*% c(2, -1, 0)
  y <- as.numeric(cut(score, c(-Inf, quantile(score, probs = 1:2 / 3), Inf)))
  bags <- 1:length(y)
  # add in points outside boundaries
  X <- rbind(
    X,
    mvtnorm::rmvnorm(n, mean = c(6, -3, 0)),
    mvtnorm::rmvnorm(n, mean = c(-6, 3, 0))
  )
  y <- c(y, rep(-1, 2*n))
  bags <- rep(bags, 3)
  repr <- c(repr(1, n), rep(0, 2*n))
  y_bag <- classify_bags(y, bags, condense = FALSE)
mdl1 <- mior(X, y_bag, bags)
# summarize predictions at the bag layer
library(dplyr)
df1 <- bind_cols(y = y_bag, bags = bags, as.data.frame(X))
df1 %>%
  bind_cols(predict(mdl1, df1, new_bags = bags, type = "class")) %>%
  bind_cols(predict(mdl1, df1, new_bags = bags, type = "raw")) %>%
  distinct(y, bags, .pred_class, .pred)

---

### predict.mismm

**Predict method for mismm object**

**Description**

Predict method for mismm object

**Usage**

```r
## S3 method for class 'mismm'
predict(
  object,
  new_data,
  type = c("class", "raw"),
  layer = c("bag", "instance"),
  new_bags = "bag_name",
  new_instances = "instance_name",
  kernel = NULL,
  ...
)
```

**Arguments**

- **object**
  An object of class mismm.

- **new_data**
  A data frame to predict from. This needs to have all of the features that the data was originally fitted with.

- **type**
  If 'class', return predicted values with threshold of 0 as -1 or +1. If 'raw', return the raw predicted scores.

- **layer**
  If 'bag', return predictions at the bag level. If 'instance', return predictions at the instance level.

- **new_bags**
  A character or character vector. Can specify a singular character that provides the column name for the bag names in new_data (default 'bag_name'). Can also specify a vector of length nrow(new_data) that has bag name for each row.
predict.mismm

new_instances A character or character vector. Can specify a singular character that provides the column name for the instance names in new_data (default 'instance_name'). Can also specify a vector of length nrow(new_data) that has instance name for each row.

kernel An optional pre-computed kernel matrix at the instance level or NULL (default NULL). The rows should correspond to instances in the new data to predict, and columns should correspond to instances in the original training data, such as a call to kme().

... Arguments passed to or from other methods.

Details

When the object was fitted using the formula method, then the parameters new_bags and new_instances are not necessary, as long as the names match the original function call.

Value

A tibble with nrow(new_data) rows. If type = 'class', the tibble will have a column .pred_class. If type = 'raw', the tibble will have a column .pred.

Author(s)

Sean Kent

See Also

mismm() for fitting the mismm object.

Examples

mil_data <- generate_mild_df(nbag = 15, nsample = 20, positive_prob = 0.15,
                             sd_of_mean = rep(0.1, 3))

mdl1 <- mismm(mil_data, control = list(sigma = 1/5))

# bag level predictions
library(dplyr)
mil_data %>%
  bind_cols(predict(mdl1, mil_data, type = "class")) %>%
  bind_cols(predict(mdl1, mil_data, type = "raw")) %>%
  distinct(bag_name, bag_label, .pred_class, .pred)

# instance level prediction
mil_data %>%
  bind_cols(predict(mdl1, mil_data, type = "class", layer = "instance")) %>%
  bind_cols(predict(mdl1, mil_data, type = "raw", layer = "instance")) %>%
  distinct(bag_name, instance_name, bag_label, .pred_class, .pred)
predict.misvm

# S3 method for class 'misvm'
predict(
  object, 
  new_data, 
  type = c("class", "raw"), 
  layer = c("bag", "instance"), 
  new_bags = "bag_name", 
  ... 
)

Arguments

object An object of class misvm.
new_data A data frame to predict from. This needs to have all of the features that the data was originally fitted with.
type If 'class', return predicted values with threshold of 0 as -1 or +1. If 'raw', return the raw predicted scores.
layer If 'bag', return predictions at the bag level. If 'instance', return predictions at the instance level.
new_bags A character or character vector. Can specify a singular character that provides the column name for the bag names in new_data (default 'bag_name'). Can also specify a vector of length nrow(new_data) that has bag name for each row.
...
Arguments passed to or from other methods.

Details

When the object was fitted using the formula method, then the parameters new_bags and new_instances are not necessary, as long as the names match the original function call.

Value

A tibble with nrow(new_data) rows. If type = 'class', the tibble will have a column .pred_class. If type = 'raw', the tibble will have a column .pred.

Author(s)

Sean Kent
predict.misvm_orova

See Also

- `misvm()` for fitting the misvm object.
- `cv_misvm()` for fitting the misvm object with cross-validation.

Examples

```r
mil_data <- generate_mild_df(nbag = 20,
    positive_prob = 0.15,
    sd_of_mean = rep(0.1, 3))
df1 <- build_instance_feature(mil_data, seq(0.05, 0.95, length.out = 10))
mdl1 <- misvm(x = df1[, 4:63], y = df1$bag_label,
    bags = df1$bag_name, method = "heuristic")
predict(mdl1, new_data = df1, type = "raw", layer = "bag")

# summarize predictions at the bag layer
library(dplyr)
df1 %>%
    bind_cols(predict(mdl1, df1, type = "class")) %>%
    bind_cols(predict(mdl1, df1, type = "raw")) %>%
    distinct(bag_name, bag_label, .pred_class, .pred)
```

predict.misvm_orova  Predict method for misvm_orova object

Description

Predict method for misvm_orova object. Predictions use the K fitted MI-SVM models. For class predictions, we return the class whose MI-SVM model has the highest raw predicted score. For raw predictions, a full matrix of predictions is returned, with one column for each model.

Usage

```r
## S3 method for class 'misvm_orova'
predict(
    object,
    new_data,
    type = c("class", "raw"),
    layer = c("bag", "instance"),
    new_bags = "bag_name",
    ...
)
```
predict.misvm_orova

Arguments

object
A data frame to predict from. This needs to have all of the features that the data was originally fitted with.

new_data
If 'class', return predicted values based on the highest output of an individual model. If 'raw', return the raw predicted scores for each model.

layer
If 'bag', return predictions at the bag level. If 'instance', return predictions at the instance level.

new_bags
A character or character vector. Can specify a singular character that provides the column name for the bag names in new_data (default 'bag_name'). Can also specify a vector of length nrow(new_data) that has bag name for each row.

Details

When the object was fitted using the formula method, then the parameters new_bags and new_instances are not necessary, as long as the names match the original function call.

Value

A tibble with nrow(new_data) rows. If type = 'class', the tibble will have a column .pred_class. If type = 'raw', the tibble will have K columns .pred_{class_name} corresponding to the raw predictions of the K models.

Author(s)

Sean Kent

See Also

misvm_orova() for fitting the misvm_orova object.

Examples

data("ordmvnorm")
x <- ordmvnorm[, 3:7]
y <- ordmvnorm$bag_label
bags <- ordmvnorm$bag_name
mdl1 <- misvm_orova(x, y, bags)

# summarize predictions at the bag layer
library(dplyr)
df1 <- bind_cols(y = y, bags = bags, as.data.frame(x))
df1 %>%
  bind_cols(predict(mdl1, df1, new_bags = bags, type = "class")) %>%
  bind_cols(predict(mdl1, df1, new_bags = bags, type = "raw")) %>%
  select(-starts_with("V")) %>%
  distinct()
Description

Predict method for omisvm object

Usage

```r
## S3 method for class 'omisvm'
predict(
  object,
  new_data,
  type = c("class", "raw"),
  layer = c("bag", "instance"),
  new_bags = "bag_name",
  ...
)
```

Arguments

- **object**: An object of class omisvm
- **new_data**: A data frame to predict from. This needs to have all of the features that the data was originally fitted with.
- **type**: If 'class', return predicted values with threshold of 0 as -1 or +1. If 'raw', return the raw predicted scores.
- **layer**: If 'bag', return predictions at the bag level. If 'instance', return predictions at the instance level.
- **new_bags**: A character or character vector. Can specify a singular character that provides the column name for the bag names in new_data (default 'bag_name'). Can also specify a vector of length nrow(new_data) that has bag name for each row.
- **...**: Arguments passed to or from other methods.

Details

When the object was fitted using the formula method, then the parameters new_bags and new_instances are not necessary, as long as the names match the original function call.

Value

A tibble with nrow(new_data) rows. If type = 'class', the tibble will have a column .pred_class. If type = 'raw', the tibble will have a column .pred.

Author(s)

Sean Kent
See Also

`omisvm()` for fitting the `omisvm` object.

Examples

```r
if (require(gurobi)) {
  data("ordmvnorm")
  x <- ordmvnorm[, 3:7]
  y <- ordmvnorm$bag_label
  bags <- ordmvnorm$bag_name

  mdl1 <- omisvm(x, y, bags, weights = NULL)

  # summarize predictions at the bag layer
  library(dplyr)
  df1 <- bind_cols(y = y, bags = bags, as.data.frame(x))
  df1 %>%
    bind_cols(predict(mdl1, df1, new_bags = bags, type = "class")) %>%
    bind_cols(predict(mdl1, df1, new_bags = bags, type = "raw")) %>%
    distinct(y, bags, .pred_class, .pred)
}
```

--

### predict.smm

**Predict method for smm object**

**Description**

Predict method for `smm` object

**Usage**

```r
## S3 method for class 'smm'
predict(
  object,
  new_data,
  type = c("class", "raw"),
  layer = "instance",
  new_instances = "instance_name",
  new_bags = "bag_name",
  kernel = NULL,
  ...
)
```

**Arguments**

- `object` an object of class `smm`
predict.smm

new_data  A data frame to predict from. This needs to have all of the features that the data was originally fitted with.

type  If 'class', return predicted values with threshold of 0 as -1 or +1. If 'raw', return the raw predicted scores.

layer  If 'instance', return predictions at the instance level. Option 'bag' returns predictions at the bag level, but only if the model was fit with smm.mild_df().

new_instances  A character or character vector. Can specify a singular character that provides the column name for the instance names in new_data (default 'instance_name'). Can also specify a vector of length nrow(new_data) that has instance name for each row.

new_bags  A character or character vector. Only relevant when fit with smm.mild_df(), which contains bag level information. Can specify a singular character that provides the column name for the bag names in new_data, default = "bag_name". Can also specify a vector of length nrow(new_data) that has bag name for each instance.

kernel  An optional pre-computed kernel matrix at the instance level or NULL (default NULL). The rows should correspond to instances in the new data to predict, and columns should correspond to instances in the original training data, such as a call to kme().

...  Arguments passed to or from other methods.

Details

When the object was fitted using the formula method, then the parameters new_bags and new_instances are not necessary, as long as the names match the original function call.

Value

tibble with nrow(new_data) rows. If type = 'class', the tibble will have a column named .pred_class. If type = 'raw', the tibble will have a column name .pred.

Author(s)

Sean Kent

See Also

smm() for fitting the smm object.

Examples

set.seed(8)
n_instances <- 10
n_samples <- 20
y <- rep(c(1, -1), each = n_samples * n_instances / 2)
instances <- as.character(rep(1:n_instances, each = n_samples))
x <- data.frame(x1 = rnorm(length(y), mean = 1*(y==1)), x2 = rnorm(length(y), mean = 2*(y==1)),

x3 = rnorm(length(y), mean = 3*(y==1)))

mdl <- smm(x, y, instances, control = list(sigma = 1/3))

# instance level predictions (training data)
suppressWarnings(library(dplyr))
data.frame(instance_name = instances, y = y, x) %>%
  bind_cols(predict(mdl, type = "raw", new_data = x, new_instances = instances)) %>%
  bind_cols(predict(mdl, type = "class", new_data = x, new_instances = instances)) %>%
  distinct(instance_name, y, .pred, .pred_class)

# test data
new_inst <- rep(c("11", "12"), each = 30)
new_y <- rep(c(1, -1), each = 30)
new_x <- data.frame(x1 = rnorm(length(new_inst), mean = 1*(new_inst=="11")),
  x2 = rnorm(length(new_inst), mean = 2*(new_inst=="11")),
  x3 = rnorm(length(new_inst), mean = 3*(new_inst=="11")))

# instance level predictions (test data)
data.frame(instance_name = new_inst, y = new_y, new_x) %>%
  bind_cols(predict(mdl, type = "raw", new_data = new_x, new_instances = new_inst)) %>%
  bind_cols(predict(mdl, type = "class", new_data = new_x, new_instances = new_inst)) %>%
  distinct(instance_name, y, .pred, .pred_class)

---

**predict.svor_exc**

*Predict method for svor_exc object*

**Description**

Predict method for svor_exc object

**Usage**

```r
## S3 method for class 'svor_exc'
predict(
  object,
  new_data,
  type = c("class", "raw"),
  layer = c("instance", "bag"),
  new_bags = "bag_name",
  ...
)
```

**Arguments**

- `object` An object of class svor_exc.
- `new_data` A data frame to predict from. This needs to have all of the features that the data was originally fitted with.
type If 'class', return predicted values with threshold of 0 as -1 or +1. If 'raw', return the raw predicted scores.

layer If 'bag', return predictions at the bag level. If 'instance', return predictions at the instance level.

new_bags A character or character vector. Can specify a singular character that provides the column name for the bag names in new_data (default 'bag_name'). Can also specify a vector of length nrow(new_data) that has bag name for each row.

Arguments passed to or from other methods.

Details

When the object was fitted using the formula method, then the parameter new_bags is not necessary, as long as the names match the original function call.

Value

A tibble with nrow(new_data) rows. If type = 'class', the tibble will have a column .pred_class. If type = 'raw', the tibble will have a column .pred.

Author(s)

Sean Kent

See Also

svor_exc() for fitting the svor_exc object.

Examples

data("ordmvnorm")
x <- ordmvnorm[, 3:7]
y <- attr(ordmvnorm, "instance_label")
mdl1 <- svor_exc(x, y)
predict(mdl1, x)
predict(mdl1, x, type = "raw")

Description

Function to carry out support measure machines algorithm which is appropriate for multiple instance learning. The algorithm calculates the kernel matrix of different empirical measures using kernel mean embedding. The data set should be passed in with rows corresponding to samples from a set of instances. SMM will compute a kernel on the instances and pass that to kernlab::ksvm() to train the appropriate SVM model.
smm

Usage

## Default S3 method:
smm(
  x,
  y,
  instances,
  cost = 1,
  weights = TRUE,
  control = list(kernel = "radial", sigma = if (is.vector(x)) 1 else 1/ncol(x), scale = TRUE),
  ...
)

## S3 method for class 'formula'
smm(formula, data, instances = "instance_name", ...)

## S3 method for class 'mild_df'
smm(x, ...)

Arguments

x A data.frame, matrix, or similar object of covariates, where each row represents a sample. If a mild_df object is passed, y, instances are automatically extracted, bags is ignored, and all other columns will be used as predictors.

y A numeric, character, or factor vector of bag labels for each instance. Must satisfy length(y) == nrow(x). Suggest that one of the levels is 1, '1', or TRUE, which becomes the positive class; otherwise, a positive class is chosen and a message will be supplied.

instances A vector specifying which samples belong to each instance. Can be a string, numeric, of factor.

cost The cost parameter in SVM, fed to the C argument in kernlab::ksvm().

weights named vector, or TRUE, to control the weight of the cost parameter for each possible y value. Weights multiply against the cost vector. If TRUE, weights are calculated based on inverse counts of instances with given label, where we only count one positive instance per bag. Otherwise, names must match the levels of y.

control A list of additional parameters passed to the method that control computation with the following components:
  • kernel either a character the describes the kernel ('linear' or 'radial') or a kernel matrix at the instance level.
  • sigma argument needed for radial basis kernel.
  • scale argument used for all methods. A logical for whether to rescale the input before fitting.

... Arguments passed to or from other methods.

formula A formula with specification y ~ x. This argument is an alternative to the x, y arguments, but requires the data and instances argument. See examples.
data If formula is provided, a data.frame or similar from which formula elements will be extracted.

Value

An object of class smm The object contains at least the following components:

- ksvm_fit: A fit of class ksvm from the kernlab package.
- call_type: A character indicating which method smm() was called with.
- x: The training data needed for computing the kernel matrix in prediction.
- features: The names of features used in training.
- levels: The levels of y that are recorded for future prediction.
- cost: The cost parameter from function inputs.
- sigma: The radial basis function kernel parameter.
- weights: The calculated weights on the cost parameter, if applicable.
- x_scale: If scale = TRUE, the scaling parameters for new predictions.

Methods (by class)

- default: Method for data.frame-like objects
- formula: Method for passing formula
- mild_df: Method for mild_df objects. Use the bag_label as y at the instance level, then perform smm() ignoring the MIL structure.

Author(s)

Sean Kent, Yifei Liu

References


See Also

predict.smm() for prediction on new data.

Examples

```r
set.seed(8)
n_instances <- 10
n_samples <- 20
y <- rep(c(1, -1), each = n_samples * n_instances / 2)
instances <- as.character(rep(1:n_instances, each = n_samples))
x <- data.frame(x1 = rnorm(length(y), mean = 1*(y==1)),
                x2 = rnorm(length(y), mean = 2*(y==1)),
                x3 = rnorm(length(y), mean = 3*(y==1)))
```
df <- data.frame(instance_name = instances, y = y, x)
mdl <- smm(x, y, instances)
mdl2 <- smm(y ~ ., data = df)

# instance level predictions
suppressWarnings(library(dplyr))
df %>%
dplyr::bind_cols(predict(mdl, type = "raw", new_data = x, new_instances = instances)) %>%
dplyr::bind_cols(predict(mdl, type = "class", new_data = x, new_instances = instances)) %>%
dplyr::distinct(instance_name, y, .pred, .pred_class)

summarize_samples

Description
Summarize a numeric data frame based on specified grouping columns and a list of functions. This is useful in summarizing a mild_df object from the sample level to the instance level.

Usage
## Default S3 method:
summarize_samples(data, group_cols, .fns = list(mean = mean), cor = FALSE, ...)

## S3 method for class 'mild_df'
summarize_samples(data, ...)

Arguments
data A data.frame, 'mild_df' object, or similar of data to summarize.
group_cols A character vector of column(s) that describe groups to summarize across.
.fns A list of functions (default list(mean = mean)).
cor A logical (default FALSE) for whether to include correlations between all features in the summarization.
... Arguments passed to or from other methods.

Value
A tibble with summarized data. There will be one row for each set of distinct groups specified by group_cols. There will be one column for each of the group_cols, plus length(.fns) columns for each of the features in data, plus correlation columns if specified.

Methods (by class)
- default: Method for data.frame-like objects.
- mild_df: Method for mild_df objects.
svor_exc

Author(s)

Sean Kent

Examples

```r
fns <- list(mean = mean, sd = sd)
summarize_samples(mtcars, group_cols = c("cyl", "gear"), .fns = fns)
summarize_samples(mtcars, group_cols = c("cyl", "gear"), .fns = fns, cor = TRUE)
```

svor_exc

Fit SVOR-EXC model to ordinal outcome data

Description

This function fits the Support Vector Ordinal Regression with Explicit Constraints based on the research of Chu and Keerthi (2007).

Usage

```r
## Default S3 method:
svor_exc(
  x, y, cost = 1, method = c("smo"), weights = NULL, control = list(kernel = "linear", sigma = if (is.vector(x)) 1 else 1/ncol(x), max_step = 500, scale = TRUE, verbose = FALSE), ...
)

## S3 method for class 'formula'
svor_exc(formula, data, ...)

## S3 method for class 'mi_df'
svor_exc(x, ...)
```

Arguments

- `x` A data.frame, matrix, or similar object of covariates, where each row represents an instance. If a `mi_df` object is passed, `y` is automatically extracted, bags is ignored, and all other columns will be used as predictors.
- `y` A numeric, character, or factor vector of bag labels for each instance. Must satisfy `length(y) == nrow(x)`. Suggest that one of the levels is 1, '1', or TRUE, which becomes the positive class; otherwise, a positive class is chosen and a message will be supplied.
cost  The cost parameter in SVM.
method The algorithm to use in fitting (default 'smo'). When method = 'smo', the modified SMO algorithm from Chu and Keerthi (2007) is used.
weights NULL, since weights are not implemented for this function.
control list of additional parameters passed to the method that control computation with the following components:
  • kernel either a character the describes the kernel ('linear' or 'radial') or a kernel matrix at the instance level.
  • sigma argument needed for radial basis kernel.
  • max_step argument used when method = 'heuristic'. Maximum steps of iteration for the heuristic algorithm.
  • scale argument used for all methods. A logical for whether to rescale the input before fitting.
  • verbose argument used when method = 'mip'. Whether to message output to the console.
... Arguments passed to or from other methods.
formula A formula with specification y ~ x. This argument is an alternative to the x, y arguments, but requires the data argument. See examples.
data If formula is provided, a data.frame or similar from which formula elements will be extracted.

Value
An object of class svor_exc The object contains at least the following components:
  • smo_fit: A fit object from running the modified ordinal smo algorithm.
  • call_type: A character indicating which method svor_exc() was called with.
  • features: The names of features used in training.
  • levels: The levels of y that are recorded for future prediction.
  • cost: The cost parameter from function inputs.
  • n_step: The total steps used in the heuristic algorithm.
  • x_scale: If scale = TRUE, the scaling parameters for new predictions.

Methods (by class)
  • default: Method for data.frame-like objects
  • formula: Method for passing formula
  • mi_df: Method for mi_df objects, automatically handling bag names, labels, and all covariates. Use the bag_label as y at the instance level, then perform svor_exc() ignoring the MIL structure and bags.

Author(s)
Sean Kent
References


See Also

`predict.svor_exc()` for prediction on new data.

Examples

```r
data("ordmvnorm")
x <- ordmvnorm[, 3:7]
y <- attr(ordmvnorm, "instance_label")

mdl1 <- svor_exc(x, y)
predict(mdl1, x)
```
Index

* datasets
  ordmvnorm, 40

* kernel feature map functions
  kfm_exact, 16
  kfm_nystrom, 17

* multiple instance formula helper functions
  mi, 20
  mild, 21

as_mi_df, 4
as_mi_df(), 37
as_mild_df, 3
as_mild_df(), 23

bag_instance_sampling, 5
build_fm, 6
build_instance_feature, 7

classify_bags, 8
cv_misvm, 9
cv_misvm(), 32, 47

formatting, 12

generate_mild_df, 13
generate_mild_df(), 23

kfm_exact, 16, 19
kfm_exact(), 7
kfm_nystrom, 17, 17
kfm_nystrom(), 6, 7, 10, 27, 31, 34, 38
kme, 19
kme(), 45, 51

mi, 20, 21
mi_df, 36
mi_df(), 4, 5
mild, 21, 21
mild_df, 22
mild_df(), 3
miord(), 43
mismm, 26
mismm(), 45
misvm, 30
misvm(), 9, 11, 35, 47
misvm_orova, 33
misvm_orova(), 48

omisvm, 37
omisvm(), 50
option, 13
ordmvnorm, 40

predict.cv_misvm, 41
predict.mior, 42
predict.mior(), 25
predict.mismm, 44
predict.mismm(), 29
predict.misvm, 46
predict.misvm(), 25, 32
predict.misvm_orova, 47
predict.misvm_orova(), 35
predict.omisvm, 49
predict.omisvm(), 39
predict.smm, 50
predict.smm(), 55
predict.svor_exc, 52
predict.svor_exc(), 59
print.mi_df (formatting), 12
print.mild_df (formatting), 12
print.tbl(), 12

smm, 53
smm(), 27, 51
summarize_samples, 56
summarize_samples(), 8, 23
svor_exc, 57
svor_exc(), 53