Package 'abclass'

May 28, 2022

Title Angle-Based Large-Margin Classifiers

Version 0.3.0

Description Multi-category angle-based large-margin classifiers.
See Zhang and Liu (2014) <doi:10.1093/biomet/asu017> for details.

Depends R (>= 3.5.0)

Imports Rcpp, stats

LinkingTo Rcpp, RcppArmadillo

Suggests tinytest

Copyright Eli Lilly and Company

License GPL (>= 3)

URL https://wwenjie.org/abclass,
https://github.com/wenjie2wang/abclass

BugReports https://github.com/wenjie2wang/abclass/issues

Encoding UTF-8

RoxygenNote 7.1.2

NeedsCompilation yes

Author Wenjie Wang [aut, cre] (<https://orcid.org/0000-0003-0363-3180>),
Eli Lilly and Company [cph]

Maintainer Wenjie Wang <wang@wwenjie.org>

Repository CRAN

Date/Publication 2022-05-28 18:10:02 UTC

R topics documented:

abclass-package *Multi-Category Angle-Based Large-Margin Classifiers*

Description

This package provides implementations of the multi-category angle-based classifiers (Zhang & Liu, 2014) with the large-margin unified machines (Liu, et al., 2011) for high-dimensional data.

References

Zhang, C., & Liu, Y. (2014). Multicategory Angle-Based Large-Margin Classification. *Biometrika*, 101(3), 625–640.

Liu, Y., Zhang, H. H., & Wu, Y. (2011). Hard or soft classification? large-margin unified machines. *Journal of the American Statistical Association*, 106(493), 166–177.

abclass *Angle-Based Classification*

Description

Multi-category angle-based large-margin classifiers with regularization by the elastic-net penalty.

Usage

```
abclass(
  x,
  y,
  intercept = TRUE,
  weight = NULL,
  loss = c("logistic", "boost", "hinge-boost", "lum"),
  control = list(),
  ...
)

abclass.control(
  lambda = NULL,
  alpha = 0.5,
  nlambda = 50,
  lambda_min_ratio = NULL,
  grouped = TRUE,
  group_weight = NULL,
  group_penalty = c("lasso", "scad", "mcp"),
  dgamma = 1,
  nfolds = 0,
  stratified_cv = TRUE,
```

```
  alignment = c("fraction", "lambda"),
  lum_a = 1,
  lum_c = 1,
  boost_umin = -5,
  maxit = 1e+05,
  epsilon = 0.001,
  standardize = TRUE,
  varying_active_set = TRUE,
  verbose = 0,
  ...
)
```

Arguments

x	A numeric matrix representing the design matrix. No missing valus are allowed. The coefficient estimates for constant columns will be zero. Thus, one should set the argument `intercept` to `TRUE` to include an intercept term instead of adding an all-one column to `x`.
y	An integer vector, a character vector, or a factor vector representing the response label.
intercept	A logical value indicating if an intercept should be considered in the model. The default value is `TRUE` and the intercept is excluded from regularization.
weight	A numeric vector for nonnegative observation weights. Equal observation weights are used by default.
loss	A character value specifying the loss function. The available options are `"logistic"` for the logistic deviance loss, `"boost"` for the exponential loss approximating Boosting machines, `"hinge-boost"` for hybrid of SVM and AdaBoost machine, and `"lum"` for largin-margin unified machines (LUM). See Liu, et al. (2011) for details.
control	A list of control parameters. See `abclass.control()` for details.
...	Other control parameters passed to `abclass.control`.
lambda	A numeric vector specifying the tuning parameter *lambda* of elastic-net penalty. A data-driven *lambda* sequence will be generated and used according to specified `alpha`, `nlambda` and `lambda_min_ratio` if this argument is `NULL` by default. The specified `lambda` will be sorted in decreasing order internally and only the unique values will be kept.
alpha	A numeric value in [0, 1] representing the mixing parameter *alpha* in elastic-net penalty.
nlambda	A positive integer specifying the length of the internally generated *lambda* sequence. This argument will be ignored if a valid `lambda` is specified. The default value is `50`.
lambda_min_ratio	A positive number specifying the ratio of the smallest lambda parameter to the largest lambda parameter. The default value is set to `1e-4` if the sample size is larger than the number of predictors, and `1e-2` otherwise.
grouped	A logicial value. Experimental flag to apply group Lasso.

`group_weight` A numerical vector with nonnegative values representing the adaptive penalty factors for grouped lasso.

`group_penalty` A character vector specifying the name of the group penalty.

`dgamma` A positive number specifying the increment to the minimal gamma parameter for group SCAD or group MCP.

`nfolds` A nonnegative integer specifying the number of folds for cross-validation. The default value is `0` and no cross-validation will be performed if `nfolds < 2`.

`stratified_cv` A logical value indicating if the cross-validation procedure should be stratified by the response label. The default value is `TRUE`.

`alignment` A character vector specifying how to align the lambda sequence used in the main fit with the cross-validation fits. The available options are `"fraction"` for allowing cross-validation fits to have their own lambda sequences and `"lambda"` for using the same lambda sequence of the main fit. The option `"lambda"` will be applied if a meaningful `lambda` is specified. The default value is `"fraction"`.

`lum_a` A positive number greater than one representing the parameter a in LUM, which will be used only if `loss = "lum"`. The default value is `1.0`.

`lum_c` A nonnegative number specifying the parameter c in LUM, which will be used only if `loss = "hinge-boost"` or `loss = "lum"`. The default value is `1.0`.

`boost_umin` A negative number for adjusting the boosting loss for the internal majorization procedure.

`maxit` A positive integer specifying the maximum number of iteration. The default value is `10^5`.

`epsilon` A positive number specifying the relative tolerance that determines convergence. The default value is `1e-3`.

`standardize` A logical value indicating if each column of the design matrix should be standardized internally to have mean zero and standard deviation equal to the sample size. The default value is `TRUE`. Notice that the coefficient estimates are always returned on the original scale.

`varying_active_set`
A logical value indicating if the active set should be updated after each cycle of coordinate-majorization-descent algorithm. The default value is `TRUE` for usually more efficient estimation procedure.

`verbose` A nonnegative integer specifying if the estimation procedure should print out intermediate steps/results. The default value is `0` for silent estimation procedure.

Value

The function `abclass()` returns an object of class `abclass` representing a trained classifier; The function `abclass.control()` returns an object of class abclass.control representing a list of control parameters.

References

Zhang, C., & Liu, Y. (2014). Multicategory Angle-Based Large-Margin Classification. *Biometrika*, 101(3), 625–640.

Liu, Y., Zhang, H. H., & Wu, Y. (2011). Hard or soft classification? large-margin unified machines. *Journal of the American Statistical Association*, 106(493), 166–177.

Examples

```
library(abclass)
set.seed(123)

## toy examples for demonstration purpose
## reference: example 1 in Zhang and Liu (2014)
ntrain <- 100 # size of training set
ntest <- 100  # size of testing set
p0 <- 5       # number of actual predictors
p1 <- 5       # number of random predictors
k <- 5        # number of categories

n <- ntrain + ntest; p <- p0 + p1
train_idx <- seq_len(ntrain)
y <- sample(k, size = n, replace = TRUE)         # response
mu <- matrix(rnorm(p0 * k), nrow = k, ncol = p0) # mean vector
## normalize the mean vector so that they are distributed on the unit circle
mu <- mu / apply(mu, 1, function(a) sqrt(sum(a ^ 2)))
x0 <- t(sapply(y, function(i) rnorm(p0, mean = mu[i, ], sd = 0.25)))
x1 <- matrix(rnorm(p1 * n, sd = 0.3), nrow = n, ncol = p1)
x <- cbind(x0, x1)
train_x <- x[train_idx, ]
test_x <- x[- train_idx, ]
y <- factor(paste0("label_", y))
train_y <- y[train_idx]
test_y <- y[- train_idx]

## Regularization through ridge penalty
model1 <- abclass(train_x, train_y, nlambda = 5, nfolds = 3,
                  loss = "logistic", alpha = 0, lambda_min_ratio = 1e-2)
pred1 <- predict(model1, test_x)
table(test_y, pred1)
mean(test_y == pred1) # accuracy

## groupwise regularization via group lasso
model2 <- abclass(train_x, train_y, nlambda = 5, nfolds = 3,
                  grouped = TRUE, loss = "boost")
pred2 <- predict(model2, test_x)
table(test_y, pred2)
mean(test_y == pred2) # accuracy
```

---

coef.abclass *Coefficient Estimates of A Trained Angle-Based Classifier*

Description

Extract coefficient estimates from an abclass object.

Usage

```
## S3 method for class 'abclass'
coef(object, selection = c("cv_min", "cv_1se", "all"), ...)
```

Arguments

`object`	An object of class `abclass`.
`selection`	A character value specifying how to select a particular set of coefficient estimates from the entire solution path. If the spcified `abclass` object contains the cross-validation results, one may set `selection` to `"cv_min"` (or `"cv_1se"`) for the estimates giving the smallest cross-validation error (or the set of estimates resulted from the largest *lambda* within one standard error of the smallest cross-validation error). The entire solution path will be returned in an array if `selection = "all"` or no cross-validation results are available in the input `abclass` object.
`...`	Other arguments not used now.

Value

A vector representing the predictions or an array representing the entire solution path.

`predict.abclass` *Prediction by A Trained Angle-Based Classifier*

Description

Predict class labels or estimate conditional probabilities for the specified new data.

Usage

```
## S3 method for class 'abclass'
predict(
  object,
  newx,
  type = c("class", "probability"),
  selection = c("cv_min", "cv_1se", "all"),
  ...
)
```

Arguments

`object`	An object of class `abclass`.
`newx`	A numeric matrix representing the design matrix for predictions.
`type`	A character value specifying the desired type of predictions. The available options are `"class"` for predicted labels and `"probability"` for class conditional probability estimates.

`selection` A character value specifying how to select a particular set of coefficient estimates from the entire solution path for the predictions. If the spcified `abclass` object contains the cross-validation results, one may set `selection` to `"cv_min"` (or `"cv_1se"`) for predictions from the set of estimates having the smallest cross-validation error (or the set of estimates resulted from the largest *lambda* within one standard error of the smallest cross-validation error). The predictions for the entire solution path will be returned if `selection = "all"` or no cross-validation results are available in the input `abclass` object.

`...` Other arguments not used now.

Value

A vector representing the predictions or a list containing the predictions for each set of estimates along the solution path.

Examples

```
## see examples of `abclass()`.
```

Index