Package 'roclab'

November 4, 2025

Title ROC-Optimizing Binary Classifiers

Version 0.1.4

Description Implements ROC (Receiver Operating Characteristic)–Optimizing Binary Classifiers, supporting both linear and kernel models. Both model types provide a variety of surrogate loss functions. In addition, linear models offer multiple regularization penalties, whereas kernel models support a range of kernel functions. Scalability for large datasets is achieved through approximation-based options, which accelerate training and make fitting feasible on large data. Utilities are provided for model training, prediction, and cross-validation. The implementation builds on the ROC-Optimizing Support Vector Machines. For more information, see Hernàndez-Orallo, José, et al. (2004) <doi:10.1145/1046456.1046489>, presented in the ROC Analysis in AI Workshop (ROCAI-2004).

License MIT + file LICENSE

Encoding UTF-8

Imports stats, graphics, utils, ggplot2, fastDummies, kernlab, pracma, rsample, dplyr, caret, pROC

RoxygenNote 7.3.2

Suggests mlbench, knitr, rmarkdown, testthat (>= 3.0.0)

Config/testthat/edition 3

VignetteBuilder knitr

URL https://github.com/gimunBae/roclab

BugReports https://github.com/gimunBae/roclab/issues

NeedsCompilation no

Author Gimun Bae [aut, cre],
Seung Jun Shin [aut]

Maintainer Gimun Bae <gimunbae0201@gmail.com>

Repository CRAN

Date/Publication 2025-11-04 07:10:02 UTC

Contents

---

auc *Generic function for AUC*

Description

Compute AUC (Area Under the ROC Curve) for a fitted model. Dispatches to class-specific methods such as `auc.roclearn`.

Usage

```
auc(object, ...)
```

Arguments

`object`	A fitted model object.
`...`	Additional arguments passed to methods.

Value

Numeric scalar: estimated AUC.

auc.kroclearn *Compute AUC for a fitted kernel model*

Description

Estimate the AUC (Area Under the ROC Curve) for a fitted kernel model on new data.

Usage

```
## S3 method for class 'kroclearn'
auc(object, newdata, y, ...)
```

Arguments

object	A fitted model object of class "kroclearn" (kernel model).
newdata	A matrix or data.frame of test predictors. Must have the same structure as the training data (categorical variables are dummy-aligned automatically).
y	Response vector of test labels ({-1, 1} or convertible).
...	Not used.

Value

A numeric scalar giving the estimated AUC.

Examples

```
set.seed(123)

n_train <- 100
r_train <- sqrt(runif(n_train, 0.05, 1))
theta_train <- runif(n_train, 0, 2*pi)
X_train <- cbind(r_train * cos(theta_train), r_train * sin(theta_train))
y_train <- ifelse(r_train < 0.5, 1, -1)

n_test <- 10
r_test <- sqrt(runif(n_test, 0.05, 1))
theta_test <- runif(n_test, 0, 2*pi)
X_test <- cbind(r_test * cos(theta_test), r_test * sin(theta_test))
y_test <- ifelse(r_test < 0.5, 1, -1)

fit <- kroclearn(X_train, y_train, lambda = 0.1,
  kernel = "radial", approx=TRUE)

auc(fit, X_test, y_test)
```

`auc.roclearn` *Compute AUC for a fitted linear model*

Description

Estimate the AUC (Area Under the ROC Curve) for a fitted linear model on new data.

Usage

```
## S3 method for class 'roclearn'
auc(object, newdata, y, ...)
```

Arguments

`object`	A fitted model object of class `"roclearn"` (linear model).
`newdata`	A matrix or data.frame of test predictors. Must have the same structure as the training data (categorical variables are dummy-aligned automatically).
`y`	Response vector of test labels ({-1, 1} or convertible).
`...`	Not used.

Value

A numeric scalar giving the estimated AUC.

Examples

```
set.seed(123)

n_train <- 100
n_pos <- round(0.2 * n_train)
n_neg <- n_train - n_pos

X_train <- rbind(
  matrix(rnorm(2 * n_neg, mean = -1), ncol = 2),
  matrix(rnorm(2 * n_pos, mean =  1), ncol = 2)
)
y_train <- c(rep(-1, n_neg), rep(1, n_pos))

n_test <- 10
n_pos_test <- round(0.2 * n_test)
n_neg_test <- n_test - n_pos_test

X_test <- rbind(
  matrix(rnorm(2 * n_neg_test, mean = -1), ncol = 2),
  matrix(rnorm(2 * n_pos_test, mean =  1), ncol = 2)
)
y_test <- c(rep(-1, n_neg_test), rep(1, n_pos_test))
```

```
fit <- roclearn(X_train, y_train, lambda = 0.1, approx=TRUE)

auc(fit, X_test, y_test)
```

`cv.kroclearn` *Cross-validation for kernel models*

Description

Perform k-fold cross-validation over a sequence of λ values and select the optimal model based on AUC.

Usage

```
cv.kroclearn(
  X,
  y,
  lambda.vec = NULL,
  lambda.length = 30,
  kernel = "radial",
  param.kernel = NULL,
  loss = "hinge",
  approx = NULL,
  intercept = TRUE,
  nfolds = 10,
  target.perf = list(),
  param.convergence = list()
)
```

Arguments

`X`	Predictor matrix or data.frame (categorical variables are automatically one-hot encoded).
`y`	Response vector with class labels in {-1, 1}. Labels given as {0, 1} or as a two-level factor/character are automatically converted to this format.
`lambda.vec`	Optional numeric vector of regularization parameters (lambda values). If `NULL` (default), a decreasing sequence is generated automatically.
`lambda.length`	Number of λ values to generate if `lambda.vec` is `NULL`. Default is 30.
`kernel`	Kernel type: `"radial"` (default), `"polynomial"`, `"linear"`, or `"laplace"`.
`param.kernel`	Kernel-specific parameter: • σ for `"radial"` and `"laplace"` kernels (default $1/p$, where p is the number of predictors after preprocessing, i.e., after categorical variables are one-hot encoded). • Degree for `"polynomial"` kernel (default 2). • Ignored for `"linear"` kernel.

loss	Surrogate loss function type. One of: "hinge" (default), "hinge2" (squared hinge), "logistic", or "exponential".
approx	Logical; enables a scalable approximation to accelerate training. The default is TRUE when nrow(X) >= 1000, and FALSE otherwise. For details about how approximation is applied, see the details section of the kroclearn function.
intercept	Logical; include an intercept in the model (default TRUE).
nfolds	Number of cross-validation folds (default 10).
target.perf	List with target sensitivity and specificity used when estimating the intercept (defaults to 0.9 each).
param.convergence	List of convergence controls (e.g., maxiter, eps). Default is list(maxiter = 5e4, eps = 1e-4).

Value

An object of class "cv.kroclearn" with:

- optimal.lambda — selected λ.
- optimal.fit — model trained at optimal.lambda.
- lambda.vec — grid of penalty values considered.
- auc.mean, auc.sd — mean and sd of cross-validated AUC.
- auc.result — fold-by-lambda AUC matrix.
- time.mean, time.sd — mean and sd of training time.
- time.result — fold-by-lambda training time matrix.
- nfolds, loss, kernel — settings.

See Also

kroclearn

Examples

```
set.seed(123)
n <- 100
r <- sqrt(runif(n, 0.05, 1))
theta <- runif(n, 0, 2*pi)
X <- cbind(r * cos(theta), r * sin(theta))
y <- ifelse(r < 0.5, 1, -1)

cvfit <- cv.kroclearn(
  X, y,
  lambda.vec = exp(seq(log(0.01), log(5), length.out = 3)),
  kernel = "radial",
  approx=TRUE, nfolds = 2
)
cvfit$optimal.lambda
```

`cv.roclearn` *Cross-validation for linear models*

Description

Perform k-fold cross-validation over a sequence of λ values and select the optimal model based on AUC.

Usage

```
cv.roclearn(
  X,
  y,
  lambda.vec = NULL,
  lambda.length = 30,
  penalty = "ridge",
  param.penalty = NULL,
  loss = "hinge",
  approx = NULL,
  intercept = TRUE,
  nfolds = 10,
  target.perf = list(),
  param.convergence = list()
)
```

Arguments

`X`	Predictor matrix or data.frame (categorical variables are automatically one-hot encoded).
`y`	Response vector with class labels in {-1, 1}. Labels given as {0, 1} or as a two-level factor/character are automatically converted to this format.
`lambda.vec`	Optional numeric vector of regularization parameters (lambda values). If `NULL` (default), a decreasing sequence is generated automatically.
`lambda.length`	Number of λ values to generate if `lambda.vec` is `NULL`. Default is 30.
`penalty`	Regularization penalty type: `"ridge"` (default), `"lasso"`, `"elastic"`, `"alasso"`, `"scad"`, or `"mcp"`.
`param.penalty`	Penalty-specific parameter: • Ignored for `"ridge"` and `"lasso"`. • Mixing parameter $\alpha \in (0, 1)$ for `"elastic"`. Default is 0.5. • Adaptive weight exponent $\gamma > 0$ for `"alasso"`. Default is 1. • Tuning parameter (default 3.7) for `"scad"` and `"mcp"`.
`loss`	Surrogate loss function type. One of: `"hinge"` (default), `"hinge2"` (squared hinge), `"logistic"`, or `"exponential"`.

approx	Logical; enables a scalable approximation to accelerate training. The default is TRUE when nrow(X) >= 1000, and FALSE otherwise. For details about how approximation is applied, see the details section of the roclearn function.
intercept	Logical; include an intercept in the model (default TRUE).
nfolds	Number of cross-validation folds (default 10).
target.perf	List with target sensitivity and specificity used when estimating the intercept (defaults to 0.9 each).
param.convergence	List of convergence controls (e.g., maxiter, eps). Default is list(maxiter = 5e4, eps = 1e-4).

Value

An object of class "cv.roclearn" with:

- optimal.lambda — selected λ.
- optimal.fit — model refit on the full data at optimal.lambda.
- lambda.vec — grid of penalty values considered.
- auc.mean, auc.sd — mean and sd of cross-validated AUC.
- auc.result — fold-by-lambda AUC matrix.
- time.mean, time.sd — mean and sd of training time.
- time.result — fold-by-lambda training time matrix.
- nfolds, loss, penalty — settings.

See Also

roclearn

Examples

```
set.seed(123)
n <- 100
n_pos <- round(0.2 * n)
n_neg <- n - n_pos

X <- rbind(
  matrix(rnorm(2 * n_neg, mean = -1), ncol = 2),
  matrix(rnorm(2 * n_pos, mean =  1), ncol = 2)
)
y <- c(rep(-1, n_neg), rep(1, n_pos))

cvfit <- cv.roclearn(
  X, y,
  lambda.vec = exp(seq(log(0.01), log(5), length.out = 3)),
  approx=TRUE, nfolds = 2
)
cvfit$optimal.lambda
```

kroclearn *Fit a kernel model*

Description

Fit a kernel model

Usage

```
kroclearn(
  X,
  y,
  lambda,
  kernel = "radial",
  param.kernel = NULL,
  loss = "hinge",
  approx = NULL,
  intercept = TRUE,
  target.perf = list(),
  param.convergence = list()
)
```

Arguments

X	Predictor matrix or data.frame (categorical variables are automatically one-hot encoded).
y	Response vector with class labels in {-1, 1}. Labels given as {0, 1} or as a two-level factor/character are automatically converted to this format.
lambda	Positive scalar regularization parameter.
kernel	Kernel type: "radial" (default), "polynomial", "linear", or "laplace".
param.kernel	Kernel-specific parameter: • σ for "radial" and "laplace" kernels (default $1/p$, where p is the number of predictors after preprocessing, i.e., after categorical variables are one-hot encoded). • Degree for "polynomial" kernel (default 2). • Ignored for "linear" kernel.
loss	Surrogate loss function type. One of: "hinge" (default), "hinge2" (squared hinge), "logistic", or "exponential".
approx	Logical; enables a scalable approximation to accelerate training. The default is TRUE when nrow(X) >= 1000, and FALSE otherwise. For details about how approximation is applied, see the details section.
intercept	Logical; include an intercept in the model (default TRUE).
target.perf	List with target sensitivity and specificity used when estimating the intercept (defaults to 0.9 each).

`param.convergence`
: List of convergence controls (e.g., `maxiter`, `eps`). Default is `list(maxiter = 5e4, eps = 1e-4)`.

Details

For large-scale data, the model is computationally prohibitive because its loss is a U-statistic involving a double summation. To reduce this burden, the package adopts an efficient algorithm based on an incomplete U-statistic, which approximates the loss with a single summation. In kernel models, a Nyström low-rank approximation is further applied to efficiently compute the kernel matrix. These approximations substantially reduce computational cost and accelerate training, while maintaining accuracy, making the model feasible for large-scale datasets. This option is available when `@param approx = TRUE`.

Value

An object of class `"kroclearn"`, a list containing:

- `theta.hat` — estimated dual coefficient vector.
- `intercept` — fitted intercept (if applicable).
- `lambda`, `kernel`, `param.kernel`, `loss`.
- `approx`, `B` (number of sampled pairs if approximation used).
- `time` — training time (seconds).
- `nobs`, `p` — number of observations and predictors.
- `converged`, `n.iter` — convergence information.
- `kfunc` — kernel function object.
- `nystrom` — low rank kernel approximation details (if used).
- `X` — training data (post-preprocessing).
- `preprocessing` — details on categorical variables, removed columns, and column names.
- `call` — the function call.

Examples

```
set.seed(123)
n <- 100
r <- sqrt(runif(n, 0.05, 1))
theta <- runif(n, 0, 2*pi)
X <- cbind(r * cos(theta), r * sin(theta))
y <- ifelse(r < 0.5, 1, -1)

fit <- kroclearn(X, y, lambda = 0.1, kernel = "radial", approx=TRUE)
```

`plot.cv.kroclearn` *Visualize Cross-Validation results for kernel models*

Description

Produce a visualization of cross-validation results from a fitted `cv.kroclearn` object. The plot shows the mean AUC across regularization parameters λ, with error bars reflecting the cross-validation standard deviation. Optionally, the selected optimal λ is highlighted with a dashed line and marker.

Usage

```
## S3 method for class 'cv.kroclearn'
plot(x, highlight = TRUE, ...)
```

Arguments

`x`	A cross-validation object of class `"cv.kroclearn"`.
`highlight`	Logical; if `TRUE`, mark the selected optimal λ with a vertical dashed line with a red point (default `TRUE`).
`...`	Additional arguments passed to underlying **ggplot2** functions.

Details

This function is a method for the generic `plot()` function, designed specifically for cross-validation objects from `cv.kroclearn`. The x-axis is displayed on a log scale for λ, and the y-axis represents AUC values. Error bars show variability across folds. This is the kernel counterpart of `plot.cv.roclearn`.

Value

A `ggplot2` object is returned and drawn to the current device.

Examples

```
set.seed(123)
n <- 100
r <- sqrt(runif(n, 0.05, 1))
theta <- runif(n, 0, 2*pi)
X <- cbind(r * cos(theta), r * sin(theta))
y <- ifelse(r < 0.5, 1, -1)

cvfit <- cv.kroclearn(
  X, y,
  lambda.vec = exp(seq(log(0.01), log(5), length.out = 3)),
  kernel = "radial",
  approx=TRUE, nfolds = 2
)
plot(cvfit)
```

`plot.cv.roclearn` *Visualize Cross-Validation results for linear models*

Description

Produce a visualization of cross-validation results from a fitted `cv.roclearn` object. The plot shows the mean AUC across regularization parameters λ, with error bars reflecting the cross-validation standard deviation. Optionally, the selected optimal λ is highlighted with a dashed line and marker.

Usage

```
## S3 method for class 'cv.roclearn'
plot(x, highlight = TRUE, ...)
```

Arguments

`x`	A cross-validation object of class `"cv.roclearn"`.
`highlight`	Logical; if `TRUE`, mark the selected optimal λ with a vertical dashed line with a red point (default `TRUE`).
`...`	Additional arguments passed to underlying **ggplot2** functions.

Details

This function is a method for the generic `plot()` function, designed specifically for cross-validation objects from `cv.roclearn`. The x-axis is displayed on a log scale for λ, and the y-axis represents AUC values. Error bars show variability across folds.

Value

A `ggplot2` object is returned and drawn to the current device.

Examples

```
set.seed(123)

n <- 100
n_pos <- round(0.2 * n)
n_neg <- n - n_pos

X <- rbind(
  matrix(rnorm(2 * n_neg, mean = -1), ncol = 2),
  matrix(rnorm(2 * n_pos, mean =  1), ncol = 2)
)
y <- c(rep(-1, n_neg), rep(1, n_pos))

cvfit <- cv.roclearn(
  X, y,lambda.vec = exp(seq(log(0.01), log(5), length.out = 3)),
  approx=TRUE, nfolds = 2)
```

```
plot(cvfit)
```

---

plot_roc *Plot Receiver Operating Characteristic (ROC) curve using ggroc*

Description

Draws an ROC curve based on decision values. There is an option to display the AUC in the plot title and to print the ROC summary object.

Usage

```
plot_roc(
  y_true,
  y_score,
  col = "blue",
  size = 1.2,
  title = TRUE,
  summary = FALSE,
  ...
)
```

Arguments

y_true	Response vector with class labels in {-1, 1}. Labels given as {0, 1} or as a two-level factor/character are automatically converted to this format.
y_score	Numeric vector of predicted scores or decision values.
col	Line color.
size	Line width.
title	Logical; if TRUE, displays AUC in the plot title.
summary	Logical; if TRUE, prints the ROC object summary.
...	Additional arguments passed to pROC::ggroc().

Value

A ggplot object of the ROC curve.

Examples

```
set.seed(123)
n <- 100
n_pos <- round(0.2 * n)
n_neg <- n - n_pos
X <- rbind(
  matrix(rnorm(2 * n_neg, mean = -1), ncol = 2),
```

```
  matrix(rnorm(2 * n_pos, mean =  1), ncol = 2)
)
y <- c(rep(-1, n_neg), rep(1, n_pos))

fit <- roclearn(X, y, lambda = 0.1, approx=TRUE)

y_score <- predict(fit, X, type = "response")

plot_roc(y, y_score)
```

`predict.kroclearn` *Predictions from a fitted kernel model*

Description

Generate predictions from a fitted kernel model.

Usage

```
## S3 method for class 'kroclearn'
predict(object, newdata, type = c("class", "response"), ...)
```

Arguments

`object`	A fitted model object of class `"kroclearn"` (kernel).
`newdata`	A data frame or matrix of predictors for which predictions are desired. Categorical variables are automatically dummy-encoded and aligned to the training structure.
`type`	Prediction type: `"class"` for {-1, 1} labels, or `"response"` for raw decision scores.
`...`	Not used.

Value

A numeric vector of predictions ({-1, 1}) if `type = "class"`, or raw decision scores if `type = "response"`.

See Also

`kroclearn`, `cv.kroclearn`

Examples

```
set.seed(123)
n <- 100
r <- sqrt(runif(n, 0.05, 1))
theta <- runif(n, 0, 2*pi)
X <- cbind(r * cos(theta), r * sin(theta))
y <- ifelse(r < 0.5, 1, -1)

fit <- kroclearn(X, y, lambda = 0.1, kernel = "radial", approx=TRUE)

# Predict classes {-1, 1}
predict(fit, X, type = "class")

# Predict decision scores
predict(fit, X, type = "response")
```

---

predict.roclearn *Predictions from a fitted linear model*

Description

Generate predictions from a fitted linear model.

Usage

```
## S3 method for class 'roclearn'
predict(object, newdata, type = c("class", "response"), ...)
```

Arguments

object	A fitted model object of class "roclearn" (linear).
newdata	A data frame or matrix of predictors for which predictions are desired. Categorical variables are automatically dummy-encoded and aligned to the training structure.
type	Prediction type: "class" for {-1, 1} labels, or "response" for raw decision scores.
...	Not used.

Value

A numeric vector of predictions ({-1, 1}) if type = "class", or raw decision scores if type = "response".

See Also

roclearn, cv.roclearn

Examples

```
set.seed(123)
n <- 100
n_pos <- round(0.2 * n)
n_neg <- n - n_pos
X <- rbind(
  matrix(rnorm(2 * n_neg, mean = -1), ncol = 2),
  matrix(rnorm(2 * n_pos, mean =  1), ncol = 2)
)
y <- c(rep(-1, n_neg), rep(1, n_pos))

fit <- roclearn(X, y, lambda = 0.1, approx=TRUE)

# Predict classes {-1, 1}
predict(fit, X, type = "class")

# Predict decision scores
predict(fit, X, type = "response")
```

`roclearn` *Fit a linear model*

Description

Fit a linear model

Usage

```
roclearn(
  X,
  y,
  lambda,
  penalty = "ridge",
  param.penalty = NULL,
  loss = "hinge",
  approx = NULL,
  intercept = TRUE,
  target.perf = list(),
  param.convergence = list()
)
```

Arguments

`X`	Predictor matrix or data.frame (categorical variables are automatically one-hot encoded).
`y`	Response vector with class labels in {-1, 1}. Labels given as {0, 1} or as a two-level factor/character are automatically converted to this format.
`lambda`	Positive scalar regularization parameter.

`penalty` Regularization penalty type: `"ridge"` (default), `"lasso"`, `"elastic"`, `"alasso"`, `"scad"`, or `"mcp"`.

`param.penalty` Penalty-specific parameter:

- Ignored for `"ridge"` and `"lasso"`.
- Mixing parameter $\alpha \in (0, 1)$ for `"elastic"`. Default is 0.5.
- Adaptive weight exponent $\gamma > 0$ for `"alasso"`. Default is 1.
- Tuning parameter (default 3.7) for `"scad"` and `"mcp"`.

`loss` Surrogate loss function type. One of: `"hinge"` (default), `"hinge2"` (squared hinge), `"logistic"`, or `"exponential"`.

`approx` Logical; enables a scalable approximation to accelerate training. The default is `TRUE` when `nrow(X) >= 1000`, and `FALSE` otherwise. For details about how approximation is applied, see the `details` section.

`intercept` Logical; include an intercept in the model (default `TRUE`).

`target.perf` List with target sensitivity and specificity used when estimating the intercept (defaults to 0.9 each).

`param.convergence` List of convergence controls (e.g., `maxiter`, `eps`). Default is `list(maxiter = 5e4, eps = 1e-4)`.

Details

For large-scale data, the model is computationally prohibitive because its loss is a U-statistic involving a double summation. To reduce this burden, the package adopts an efficient algorithm based on an incomplete U-statistic, which approximates the loss with a single summation. These approximations substantially reduce computational cost and accelerate training, while maintaining accuracy, making the model feasible for large-scale datasets. This option is available when `approx = TRUE`.

Value

An object of class `"roclearn"`, a list containing:

- `beta.hat` — estimated coefficient vector.
- `intercept` — fitted intercept (if applicable).
- `lambda`, `penalty`, `param.penalty`, `loss`.
- `approx`, `B` (number of sampled pairs if approximation used).
- `time` — training time (seconds).
- `nobs`, `p` — number of observations and predictors.
- `converged`, `n.iter` — convergence information.
- `preprocessing` — details on categorical variables, removed columns, and column names.
- `call` — the function call.

Examples

```
set.seed(123)
n <- 100
n_pos <- round(0.2 * n)
n_neg <- n - n_pos
X <- rbind(
  matrix(rnorm(2 * n_neg, mean = -1), ncol = 2),
  matrix(rnorm(2 * n_pos, mean =  1), ncol = 2)
)
y <- c(rep(-1, n_neg), rep(1, n_pos))

fit <- roclearn(X, y, lambda = 0.1, penalty = "ridge", approx=TRUE)
```

---

summary.cv.kroclearn *Summarize Cross-Validation results for kernel models*

Description

Print a concise summary of cross-validation results for a kernel model.

Usage

```
## S3 method for class 'cv.kroclearn'
summary(object, ...)
```

Arguments

`object`	A fitted cross-validation object of class `"cv.kroclearn"` (kernel).
`...`	Not used.

Details

This is a method for the generic `summary()` function, applied to objects of class `"cv.kroclearn"`. It prints training settings (loss, kernel type, number of folds, the set of candidate λ), the selected optimal λ, the corresponding mean and standard deviation of cross-validated AUC, and a truncated table of AUC results across candidate λ values.

Value

Invisibly returns the input `object`, after printing a summary to the console.

Examples

```
set.seed(123)

n <- 100
r <- sqrt(runif(n, 0.05, 1))
theta <- runif(n, 0, 2*pi)
```

```
X <- cbind(r * cos(theta), r * sin(theta))
y <- ifelse(r < 0.5, 1, -1)

cvfit <- cv.kroclearn(
  X, y,
  lambda.vec = exp(seq(log(0.01), log(5), length.out = 3)),
  kernel = "radial",
  approx=TRUE, nfolds = 2
)

summary(cvfit)
```

`summary.cv.roclearn` *Summarize Cross-Validation results for linear models*

Description

Print a concise summary of cross-validation results for a linear model.

Usage

```
## S3 method for class 'cv.roclearn'
summary(object, ...)
```

Arguments

`object`	A fitted cross-validation object of class `"cv.roclearn"` (linear).
`...`	Not used.

Details

This is a method for the generic `summary()` function, applied to objects of class `"cv.roclearn"`. It prints training settings (loss, penalty, number of folds, the set of candidate λ), the selected optimal λ, the corresponding mean and standard deviation of cross-validated AUC, and a truncated table of AUC results across candidate λ values.

Value

Invisibly returns the input `object`, after printing a summary to the console.

Examples

```
set.seed(123)

n <- 100
n_pos <- round(0.2 * n)
n_neg <- n - n_pos
X <- rbind(
  matrix(rnorm(2 * n_neg, mean = -1), ncol = 2),
```

```
  matrix(rnorm(2 * n_pos, mean =  1), ncol = 2)
)
y <- c(rep(-1, n_neg), rep(1, n_pos))

cvfit <- cv.roclearn(
  X, y,
  lambda.vec = exp(seq(log(0.01), log(5), length.out = 3)),
  approx=TRUE, nfolds = 2
)

summary(cvfit)
```

`summary.kroclearn` *Summarize a fitted kernel model*

Description

Display key information from a fitted `"kroclearn"` object, including: data dimensions, kernel specification, convergence status, training time, and leading coefficient estimates.

Usage

```
## S3 method for class 'kroclearn'
summary(object, ...)
```

Arguments

`object`	A fitted model of class `"kroclearn"`.
`...`	Unused.

Value

Invisibly returns `object` after printing a formatted summary.

See Also

`kroclearn`, `summary.roclearn`, `cv.kroclearn`, `cv.roclearn`

Examples

```
set.seed(123)
n <- 100
r <- sqrt(runif(n, 0.05, 1))
theta <- runif(n, 0, 2*pi)
X <- cbind(r * cos(theta), r * sin(theta))
y <- ifelse(r < 0.5, 1, -1)

fit <- kroclearn(X, y, lambda = 0.1, kernel = "radial", approx=TRUE)
summary(fit)
```

summary.roclearn *Summarize a fitted linear model*

Description

Display key information from a fitted "roclearn" object, including: data dimensions, model specification, convergence status, training time, and leading coefficient estimates.

Usage

```
## S3 method for class 'roclearn'
summary(object, ...)
```

Arguments

object	A fitted model of class "roclearn".
...	Unused.

Value

Invisibly returns object after printing a formatted summary.

See Also

roclearn, summary.kroclearn, cv.roclearn, cv.kroclearn

Examples

```
set.seed(123)
n <- 100
n_pos <- round(0.2 * n)
n_neg <- n - n_pos
X <- rbind(
  matrix(rnorm(2 * n_neg, mean = -1), ncol = 2),
  matrix(rnorm(2 * n_pos, mean =  1), ncol = 2)
)
y <- c(rep(-1, n_neg), rep(1, n_pos))

fit <- roclearn(X, y, lambda = 0.1, approx=TRUE)
summary(fit)
```

Index