Package 'owl'

February 11, 2020

Title Generalized Linear Models Regularized with the Sorted L1-Norm

Version 0.1.1

Description Efficient implementations for Sorted L-One Penalized Estimation (SLOPE): generalized linear models regularized with the sorted L1-norm (Bogdan et al. (2015) <doi:10/gfgwzt>) or, equivalently, ordered weighted L1-norm (OWL). Supported models include ordinary least-squares regression, binomial regression, multinomial regression, and Poisson regression. Both dense and sparse predictor matrices are supported. In addition, the package features predictor screening rules that enable fast and efficient solutions to high-dimensional problems.

License GPL-3

LazyData true

Depends R (>= 3.3.0)

Imports lattice, Matrix, methods, Rcpp

LinkingTo Rcpp, RcppArmadillo (>= 0.9.200.7.0)

Suggests caret, covr, glmnet, knitr, rmarkdown, SLOPE, spelling, testthat (>= 2.1.0)

RoxygenNote 7.0.2

Language en-US

Encoding UTF-8

URL https://github.com/jolars/owl, https://jolars.github.io/owl

BugReports https://github.com/jolars/owl/issues

VignetteBuilder knitr

NeedsCompilation yes

Author Johan Larsson [aut, cre] (<https://orcid.org/0000-0002-4029-5945>), Jonas Wallin [aut] (<https://orcid.org/0000-0003-0381-6593>), Malgorzata Bogdan [ctb] (code adapted from 'SLOPE'), Ewout van den Berg [ctb] (code adapted from 'SLOPE'), Chiara Sabatti [ctb] (code adapted from 'SLOPE'),

Emmanuel Candes [ctb] (code adapted from 'SLOPE'),
Evan Patterson [ctb] (code adapted from 'SLOPE'),
Weijie Su [ctb] (code adapted from 'SLOPE'),
Jerome Friedman [ctb] (code adapted from 'glmnet'),
Trevor Hastie [ctb] (code adapted from 'glmnet'),
Rob Tibshirani [ctb] (code adapted from 'glmnet'),
Balasubramanian Narasimhan [ctb] (code adapted from 'glmnet'),
Noah Simon [ctb] (code adapted from 'glmnet'),
Junyang Qian [ctb] (code adapted from 'glmnet')

Maintainer Johan Larsson <johan.larsson@stat.lu.se>

Repository CRAN

Date/Publication 2020-02-11 10:50:08 UTC

R topics documented:

`abalone` *Abalone*

Description

This data set contains observations of abalones, the common name for any of a group of sea snails. The goal is to predict the age of an individual abalone given physical measurements such as sex, weight, and height.

Usage

```
abalone
```

Format

A list with two items representing 211 observations from 9 variables

sex sex of abalone, 1 for female

infant indicates that the person is an infant

length longest shell measurement in mm

diameter perpendicular to length in mm

height height in mm including meat in shell

weight_whole weight of entire abalone

weight_shucked weight of meat

weight_viscera weight of viscera

weight_shell weight of shell

rings rings. +1.5 gives the age in years

Details

Only a stratified sample of 211 rows of the original data set are used here.

Source

Pace, R. Kelley and Ronald Barry, Sparse Spatial Autoregressions, Statistics and Probability Letters, 33 (1997) 291-297.

`bodyfat` *Bodyfat*

Description

The response (`y`) corresponds to estimates of percentage of body fat from application of Siri's 1956 equation to measurements of underwater weighing, as well as age, weight, height, and a variety of body circumference measurements.

Usage

```
bodyfat
```

Format

A list with two items representing 252 observations from 14 variables

age age (years)

weight weight (lbs)

height height (inches)

neck neck circumference (cm)

chest chest circumference (cm)

abdomen abdomen circumference (cm)

hip hip circumference (cm)

thigh thigh circumference (cm)

knee knee circumference (cm)

ankle ankle circumference (cm)

biceps biceps circumference (cm)

forearm forearm circumference (cm)

wrist wrist circumference (cm)

Source

http://lib.stat.cmu.edu/datasets/bodyfat

https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/regression.html

---

`caretSlopeOwl` *Model objects for model tuning with caret*

Description

This function can be used in a call to `caret::train()` to enable model tuning using caret. Note that this function does not properly work with sparse feature matrices and standardization due to the way resampling is implemented in caret. So for these cases, please check out `trainOwl()` instead.

Usage

```
caretSlopeOwl()
```

Value

A model description list to be used in the `method` argument in `caret::train()`.

See Also

`caret::train()`, `trainOwl()`, `owl()`

`coef.Owl` *Obtain coefficients*

Description

This function returns coefficients from a model fit by `owl()`.

Usage

```
## S3 method for class 'Owl'
coef(object, sigma = NULL, exact = FALSE, simplify = TRUE, ...)
```

Arguments

`object`	an object of class `'Owl'`.
`sigma`	penalty parameter for SLOPE models; if `NULL`, the values used in the original fit will be used
`exact`	if `TRUE` and the given parameter values differ from those in the original fit, the model will be refit by calling `stats::update()` on the object with the new parameters. If `FALSE`, the predicted values will be based on interpolated coefficients from the original penalty path.
`simplify`	if `TRUE`, `base::drop()` will be called before returning the coefficients to drop extraneous dimensions
`...`	arguments that are passed on to `stats::update()` (and therefore also to `owl()`) if `exact = TRUE` and the given penalty is not in `object`

Details

If `exact == FALSE` and `sigma` is not in `object`, then the returned coefficients will be approximated by linear interpolation. If coefficients from another type of penalty sequence (with a different `lambda`) are required, however, please use `owl()` to refit the model.

Value

Coefficients from the model.

Examples

```
fit <- owl(mtcars$mpg, mtcars$vs, n_sigma = 1)
coef(fit)
```

deviance.Owl *Model deviance*

Description

Model deviance

Usage

```
## S3 method for class 'Owl'
deviance(object, ...)
```

Arguments

object	an object of class 'Owl'.
...	ignored

Value

For Gaussian models this is twice the residual sums of squares. For all other models, two times the negative loglikelihood is returned.

Examples

```
fit <- owl(heart$x, heart$y, family = "binomial")
deviance(fit)
```

heart *Heart disease*

Description

Diagnostic attributes of patients classified as having heart disease or not.

Usage

```
heart
```

Format

270 observations from 17 variables represented as a list consisting of a binary factor response vector y, with levels 'absence' and 'presence' indicating the absence or presence of heart disease and x: a sparse feature matrix of class 'dgCMatrix' with the following variables:

age age

bp diastolic blood pressure

chol serum cholesterol in mg/dl

hr maximum heart rate achieved

old_peak ST depression induced by exercise relative to rest

vessels the number of major blood vessels (0 to 3) that were colored by fluoroscopy

sex sex of the participant: 0 for male, 1 for female

angina a dummy variable indicating whether the person suffered angina-pectoris during exercise

glucose_high indicates a fasting blood sugar over 120 mg/dl

cp_typical typical angina

cp_atypical atypical angina

cp_nonanginal non-anginal pain

ecg_abnormal indicates a ST-T wave abnormality (T wave inversions and/or ST elevation or depression of > 0.05 mV)

ecg_estes probable or definite left ventricular hypertrophy by Estes' criteria

slope_flat a flat ST curve during peak exercise

slope_downsloping a downwards-sloping ST curve during peak exercise

thal_reversible reversible defect

thal_fixed fixed defect

Preprocessing

The original dataset contained 13 variables. The nominal of these were dummycoded, removing the first category. No precise information regarding variables `chest_pain`, `thal` and `ecg` could be found, which explains their obscure definitions here.

Source

Dua, D. and Karra Taniskidou, E. (2017). UCI Machine Learning Repository http://archive.ics.uci.edu/ml. Irvine, CA: University of California, School of Information and Computer Science.

https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/binary.html#heart

owl *Generalized linear models regularized with the SLOPE (OWL) norm*

Description

Fit a generalized linear model regularized with the SLOPE (Sorted L-One Penalized Estimation) norm, which applies a decreasing λ (penalty sequence) to the coefficient vector (β) after having sorted it in decreasing order according to its absolute values.

Usage

```
owl(
  x,
  y,
  family = c("gaussian", "binomial", "multinomial", "poisson"),
  intercept = TRUE,
  standardize_features = TRUE,
  sigma = NULL,
  lambda = c("gaussian", "bh", "oscar"),
  lambda_min_ratio = if (n < p) 0.01 else 1e-04,
  n_sigma = 100,
  q = 0.1 * min(1, n/p),
  screening = TRUE,
  tol_dev_change = 1e-05,
  tol_dev_ratio = 0.995,
  max_variables = n * m,
  max_passes = 1e+06,
  tol_rel_gap = 1e-05,
  tol_infeas = 1e-04,
  diagnostics = FALSE,
  verbosity = 0
)
```

Arguments

x	the feature matrix, which can be either a dense matrix of the standard *matrix* class, or a sparse matrix inheriting from Matrix::sparseMatrix Data frames will be converted to matrices internally.
y	the response. For Gaussian models this must be numeric; for binomial models, it can be a factor.
family	response type. See **Families** for details.
intercept	whether to fit an intercept
standardize_features	whether to standardize features (predictors)
sigma	scale of lambda sequence

`lambda`	either a character vector indicating the method used to construct the lambda path or a vector with length equal to the number of coefficients in the model
`lambda_min_ratio`	smallest value for `lambda` as a fraction of `lambda_max`
`n_sigma`	length of regularization path
`q`	shape of lambda sequence
`screening`	whether the strong rule for SLOPE be used to screen variables for inclusion
`tol_dev_change`	the regularization path is stopped if the fractional change in deviance falls below this value. Note that this is automatically set to 0 if a sigma is manually entered
`tol_dev_ratio`	the regularization path is stopped if the deviance ratio $1-\text{deviance}/(\text{null}-\text{deviance})$ is above this threshold
`max_variables`	criterion for stopping the path in terms of the maximum number of unique, nonzero coefficients in absolute value in model
`max_passes`	maximum number of passes for optimizer
`tol_rel_gap`	stopping criterion for the duality gap
`tol_infeas`	stopping criterion for the level of infeasibility
`diagnostics`	should diagnostics be saved for the model fit (timings, primal and dual objectives, and infeasibility)
`verbosity`	level of verbosity for displaying output from the program. Setting this to 1 displays basic information on the path level, 2 a little bit more information on the path level, and 3 displays information from the solver.

Details

`owl()` tries to minimize the following composite objective, given in Lagrangian form.

$$f(\beta) + \sigma \sum_{i=j}^{p} \lambda_j |\beta|_{(j)},$$

where $f(\beta)$ is a smooth, convex function of β, whereas the second part is the SLOPE norm, which is convex but non-smooth. In ordinary least-squares regression, for instance, $f(\beta)$ is simply the squared norm of the least-squares residuals. See section **Families** for specifics regarding the various types of $f(\beta)$ (model families) that are allowed in `owl()`.

By default, `owl()` fits a path of `lambda` sequences, starting from the null (intercept-only) model to an almost completely unregularized model. The path will end prematurely, however, if the criteria related to *any* of the arguments `tol_dev_change`, `tol_dev_ratio`, or `max_variables` are reached before the path is complete. This means that unless these arguments are modified, the path is not guaranteed to be of length `n_sigma`.

Value

An object of class `"Owl"` with the following slots:

`coefficients`	a three-dimensional array of the coefficients from the model fit, including the intercept if it was fit. There is one row for each coefficient, one column for each target (dependent variable), and one slice for each penalty.

`nonzeros`	a three-dimensional boolean array indicating whether a coefficient was zero or not
`lambda`	the lambda vector that when multiplied by a value in `sigma` gives the penalty vector at that point along the regularization path
`sigma`	the vector of sigma, indicating the scale of the lambda vector
`class_names`	a character vector giving the names of the classes for binomial and multinomial families
`passes`	the number of passes the solver took at each path
`violations`	the number of violations of the screening rule
`active_sets`	a list where each element indicates the indices of the coefficients that were active at that point in the regularization path
`unique`	the number of unique predictors (in absolute value)
`deviance_ratio`	the deviance ratio (as a fraction of 1)
`null_deviance`	the deviance of the null (intercept-only) model
`family`	the name of the family used in the model fit
`diagnostics`	a `data.frame` of objective values for the primal and dual problems, as well as a measure of the infeasibility, time, and iteration. Only available if `diagnostics = TRUE` in the call to `owl()`.
`call`	the call used for fitting the model

Families

Gaussian

The Gaussian model (Ordinary Least Squares) minimizes the following objective.

$$||y - X\beta||_2^2$$

Binomial

The binomial model (logistic regression) has the following objective.

$$\sum_{i=1}^{n} \log\left(1 + \exp\left(-y_i\left(x_i^T\beta + \alpha\right)\right)\right)$$

with $y \in \{-1, 1\}$.

Poisson

In poisson regression, we use the following objective.

$$-\sum_{i=1}^{n}\left(y_i\left(x_i^T\beta + \alpha\right) - \exp\left(x_i^T\beta + \alpha\right)\right)$$

Multinomial

In multinomial regression, we minimize the full-rank objective

$$-\sum_{i=1}^{n}\left(\sum_{k=1}^{m-1} y_{ik}(x_i^T\beta_k + \alpha_k) - \log\sum_{k=1}^{m-1}\exp(x_i^T\beta_k + \alpha_k)\right)$$

with y_{ik} being the element in a n by $(m-1)$ matrix, where m is the number of classes in the response.

Regularization sequences

There are multiple ways of specifying the `lambda` sequence in `owl()`. It is, first of all, possible to select the sequence manually by using a non-increasing numeric vector as argument instead of a character. If all `lambda` are the same value, this will lead to the ordinary lasso penalty. The greater the differences are between consecutive values along the sequence, the more clustering behavior will the model exhibit. Note, also, that the scale of the λ vector makes no difference if `sigma = NULL`, since `sigma` will be selected automatically to ensure that the model is completely sparse at the beginning and almost unregularized at the end. If, however, both `sigma` and `lambda` are manually specified, both of the scales will matter.

Instead of choosing the sequence manually, one of the following automatically generated sequences may be chosen.

BH (Benjamini–Hochberg)

If `lambda = "bh"`, the sequence used is that referred to as $\lambda^{(\mathrm{BH})}$ by Bogdan et al, which sets λ according to

$$\lambda_i = \Phi^{-1}(1 - iq/(2p)),$$

where Φ^{-1} is the quantile function for the standard normal distribution and q is a parameter that can be set by the user in the call to `owl()`.

Gaussian

This penalty sequence is related to BH, such that

$$\lambda_i = \lambda_i^{(\mathrm{BH})} \sqrt{1 + w(i-1) \cdot \operatorname{cumsum}(\lambda^2)_i},$$

where $w(k) = 1/(n-k-1)$. We let $\lambda_1 = \lambda_1^{(\mathrm{BH})}$ and adjust the sequence to make sure that it's non-increasing. Note that if p is large relative to n, this option will result in a constant sequence, which is usually not what you would want.

OSCAR

This sequence comes from Bondell and Reich and is a linearly non-increasing sequence such that

$$\lambda_i = q(p - i) + 1.$$

References

Bogdan, M., van den Berg, E., Sabatti, C., Su, W., & Candès, E. J. (2015). SLOPE – adaptive variable selection via convex optimization. The Annals of Applied Statistics, 9(3), 1103–1140. https://doi.org/10/gfgwzt

Bondell, H. D., & Reich, B. J. (2008). Simultaneous Regression Shrinkage, Variable Selection, and Supervised Clustering of Predictors with OSCAR. Biometrics, 64(1), 115–123. JSTOR. https://doi.org/10.1111/j.1541-0420.2007.00843.x

See Also

`plot.Owl()`, `plotDiagnostics()`, `score()`, `predict.Owl()`, `trainOwl()`, `coef.Owl()`, `print.Owl()`

Examples

```
# Gaussian response, default lambda sequence

fit <- owl(bodyfat$x, bodyfat$y)

# Binomial response, BH-type lambda sequence

fit <- owl(heart$x, heart$y, family = "binomial", lambda = "bh")

# Poisson response, OSCAR-type lambda sequence

fit <- owl(abalone$x,
           abalone$y,
           family = "poisson",
           lambda = "oscar",
           q = 0.4)

# Multinomial response, custom sigma and lambda

m <- length(unique(wine$y)) - 1
p <- ncol(wine$x)

sigma <- 0.005
lambda <- exp(seq(log(2), log(1.8), length.out = p*m))

fit <- owl(wine$x,
           wine$y,
           family = "multinomial",
           lambda = lambda,
           sigma = sigma)
```

`plot.Owl` *Plot coefficients*

Description

Plot the model's coefficient along the regularization path.

Usage

```
## S3 method for class 'Owl'
plot(x, intercept = FALSE, ...)
```

Arguments

`x`	an object of class `"Owl"`
`intercept`	whether to plot the intercept
`...`	parameters that will be used to modify the call to `lattice::xyplot()`

Value

An object of class "trellis", which will be plotted on the current device unless stored in a variable.

See Also

lattice::xyplot(), owl(), plotDiagnostics()

Examples

```
fit <- owl(heart$x, heart$y)
plot(fit)
```

---

plot.TrainedOwl *Plot results from cross-validation*

Description

Plot results from cross-validation

Usage

```
## S3 method for class 'TrainedOwl'
plot(
  x,
  measure = c("auto", "mse", "mae", "deviance", "auc", "misclass"),
  plot_min = TRUE,
  ci_alpha = 0.2,
  ci_border = FALSE,
  ci_col = lattice::trellis.par.get("superpose.line")$col,
  ...
)
```

Arguments

x	an object of class 'TrainedOwl', typically from a call to trainOwl()
measure	any of the measures used in the call to trainOwl(). If measure = "auto" then deviance will be used for binomial and multinomial models, whilst mean-squared error will be used for Gaussian and Poisson models.
plot_min	whether to mark the location of the penalty corresponding to the best prediction score
ci_alpha	alpha (opacity) for fill in confidence limits
ci_border	color (or flag to turn off and on) the border of the confidence limits
ci_col	color for border of confidence limits
...	other arguments that are passed on to lattice::xyplot()

Value

An object of class 'trellis' is returned and, if used interactively, will most likely have its print function lattice::print.trellis()) invoked, which draws the plot on the current display device.

See Also

trainOwl(), lattice::xyplot(), lattice::panel.xyplot()

Examples

```
# Cross-validation for a SLOPE binomial model
set.seed(123)
tune <- trainOwl(subset(mtcars, select = c("mpg", "drat", "wt")),
                 mtcars$hp,
                 q = c(0.1, 0.2),
                 number = 10)
plot(tune, ci_col = "salmon", col = "black")
```

---

plotDiagnostics *Plot results from diagnostics collected during model fitting*

Description

This function plots various diagnostics collected during the model fitting resulting from a call to owl() *provided that* diagnostics = TRUE.

Usage

```
plotDiagnostics(
  object,
  ind = max(object$diagnostics$penalty),
  xvar = c("time", "iteration"),
  yvar = c("objectives", "infeasibility"),
  ...
)
```

Arguments

object	an object of class "Owl".
ind	either "last"
xvar	what to place on the x axis. iteration plots each iteration, time plots the wall-clock time.
yvar	what to place on the y axis. objectives returns the primal and dual objectives whereas infeasibility returns the infeasibility metric.
...	other arguments that will be used to modify the call to lattice::xyplot()

Value

An object of class "trellis", which, unless stored in a variable, will be plotted when its default print() method is called.

Examples

```
x <- owl(abalone$x, abalone$y, sigma = 2, diagnostics = TRUE)
plotDiagnostics(x)
```

---

predict.Owl *Generate predictions from owl models*

Description

Return predictions from models fit by owl().

Usage

```
## S3 method for class 'Owl'
predict(object, x, sigma = NULL, type = "link", simplify = TRUE, ...)

## S3 method for class 'OwlGaussian'
predict(
  object,
  x,
  sigma = NULL,
  type = c("link", "response"),
  simplify = TRUE,
  ...
)

## S3 method for class 'OwlBinomial'
predict(
  object,
  x,
  sigma = NULL,
  type = c("link", "response", "class"),
  simplify = TRUE,
  ...
)

## S3 method for class 'OwlPoisson'
predict(
  object,
  x,
  sigma = NULL,
  type = c("link", "response"),
```

```
  exact = FALSE,
  simplify = TRUE,
  ...
)

## S3 method for class 'OwlMultinomial'
predict(
  object,
  x,
  sigma = NULL,
  type = c("link", "response", "class"),
  exact = FALSE,
  simplify = TRUE,
  ...
)
```

Arguments

object	an object of class "owl", typically the result of a call to owl()
x	new data
sigma	penalty parameter for SLOPE models; if NULL, the values used in the original fit will be used
type	type of prediction; "link" returns the linear predictors, "response" returns the result of applying the link function, and "class" returns class predictions.
simplify	if TRUE, base::drop() will be called before returning the coefficients to drop extraneous dimensions
...	ignored and only here for method consistency
exact	if TRUE and the given parameter values differ from those in the original fit, the model will be refit by calling stats::update() on the object with the new parameters. If FALSE, the predicted values will be based on interpolated coefficients from the original penalty path.

Value

Predictions from the model with scale determined by type.

See Also

stats::predict(), stats::predict.glm()

Examples

```
fit <- with(mtcars, owl(cbind(mpg, hp), vs, family = "binomial"))
predict(fit, with(mtcars, cbind(mpg, hp)), type = "class")
```

print.Owl *Print results from owl fit*

Description

Print results from owl fit

Usage

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

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

Arguments

x	an object of class 'Owl' or 'TrainedOwl'
...	other arguments passed to print()

Value

Prints output on the screen

Examples

```
fit <- owl(wine$x, wine$y, family = "multinomial")
print(fit, digits = 1)
```

score *Compute one of several loss metrics on a new data set*

Description

This function is a unified interface to return various types of loss for a model fit with owl().

Usage

```
score(object, x, y, measure)

## S3 method for class 'OwlGaussian'
score(object, x, y, measure = c("mse", "mae"))

## S3 method for class 'OwlBinomial'
score(object, x, y, measure = c("mse", "mae", "deviance", "misclass", "auc"))
```

```
## S3 method for class 'OwlMultinomial'
score(object, x, y, measure = c("mse", "mae", "deviance", "misclass"))

## S3 method for class 'OwlPoisson'
score(object, x, y, measure = c("mse", "mae"))
```

Arguments

object	an object of class "Owl"
x	feature matrix
y	response
measure	type of target measure. "mse" returns mean squared error. "mae" returns mean absolute error, "misclass" returns misclassification rate, and "auc" returns area under the ROC curve.

Value

The measure along the regularization path depending on the value in measure.

Examples

```
x <- subset(infert, select = c("induced", "age", "pooled.stratum"))
y <- infert$case

fit <- owl(x, y, family = "binomial")
score(fit, x, y, measure = "auc")
```

---

student *Student performance*

Description

A data set of the attributes of 382 students in secondary education collected from two schools. The goal is to predict the grade in math and Portugese at the end of the third period. See the cited sources for additional information.

Usage

```
student
```

Format

382 observations from 13 variables represented as a list consisting of a binary factor response matrix y with two responses: portugese and math for the final scores in period three for the respective subjects. The list also contains x: a sparse feature matrix of class 'dgCMatrix' with the following variables:

school_ms student's primary school, 1 for Mousinho da Silveira and 0 for Gabriel Pereira

sex sex of student, 1 for male

age age of student

urban urban (1) or rural (0) home address

large_family whether the family size is larger than 3

cohabitation whether parents live together

Medu mother's level of education (ordered)

Fedu fathers's level of education (ordered)

Mjob_health whether the mother was employed in health care

Mjob_other whether the mother was employed as something other than the specified job roles

Mjob_services whether the mother was employed in the service sector

Mjob_teacher whether the mother was employed as a teacher

Fjob_health whether the father was employed in health care

Fjob_other whether the father was employed as something other than the specified job roles

Fjob_services whether the father was employed in the service sector

Fjob_teacher whether the father was employed as a teacher

reason_home school chosen for being close to home

reason_other school chosen for another reason

reason_rep school chosen for its reputation

nursery whether the student attended nursery school

internet Pwhether the student has internet access at home

Preprocessing

All of the grade-specific predictors were dropped from the data set. (Note that it is not clear from the source why some of these predictors are specific to each grade, such as which parent is the student's guardian.) The categorical variables were dummy-coded. Only the final grades (G3) were kept as dependent variables, whilst the first and second period grades were dropped.

Source

P. Cortez and A. Silva. Using Data Mining to Predict Secondary School Student Performance. In A. Brito and J. Teixeira Eds., Proceedings of 5th FUture BUsiness TEChnology Conference (FUBUTEC 2008) pp. 5-12, Porto, Portugal, April, 2008, EUROSIS, ISBN 978-9077381-39-7. http://www3.dsi.uminho.pt/pcortez/student.pdf

Dua, D. and Karra Taniskidou, E. (2017). UCI Machine Learning Repository http://archive.ics.uci.edu/ml. Irvine, CA: University of California, School of Information and Computer Science.

`trainOwl` *Train a owl model*

Description

This function trains a model fit by `owl()` by tuning its parameters through cross-validation.

Usage

```
trainOwl(
  x,
  y,
  q = 0.2,
  number = 10,
  repeats = 1,
  measure = c("mse", "mae", "deviance", "missclass", "auc"),
  cl = NULL,
  ...
)
```

Arguments

`x`	the feature matrix, which can be either a dense matrix of the standard *matrix* class, or a sparse matrix inheriting from Matrix::sparseMatrix Data frames will be converted to matrices internally.
`y`	the response. For Gaussian models this must be numeric; for binomial models, it can be a factor.
`q`	shape of lambda sequence
`number`	number of folds (cross-validation)
`repeats`	number of repeats for each fold (for repeated *k*-fold cross validation)
`measure`	measure to try to optimize; note that you may supply *multiple* values here and that, by default, all the possible measures for the given model will be used.
`cl`	cluster if parallel fitting is desired. Can be any cluster accepted by `parallel::parLapply()`.
`...`	other arguments to pass on to `owl()`

Details

Note that by default this method matches all of the available metrics for the given model family against those provided in the argument `measure`. Collecting these measures is not particularly demanding computationally so it is almost always best to leave this argument as it is and then choose which argument to focus on in the call to `plot.TrainedOwl()`.

Value

An object of class `"TrainedOwl"`, with the following slots:

`summary`	a summary of the results with means, standard errors, and 0.95 confidence levels
`data`	the raw data from the model training
`optima`	a `data.frame` of the best (mean) values for the different metrics and their corresponding parameter values
`measure`	a `data.frame` listing the used metrics and their labels
`model`	the model fit to the entire data set
`call`	the call

See Also

parallel::parallel, `plot.TrainedOwl()`

Examples

```
# 8-fold cross-validation repeated 5 times
tune <- trainOwl(subset(mtcars, select = c("mpg", "drat", "wt")),
                 mtcars$hp,
                 q = c(0.1, 0.2),
                 number = 8,
                 repeats = 5)
```

`wine` *Wine cultivars*

Description

A data set of results from chemical analysis of wines grown in Italy from three different cultivars.

Usage

```
wine
```

Format

178 observations from 13 variables represented as a list consisting of a categorical response vector `y` with three levels: *A*, *B*, and *C* representing different cultivars of wine as well as `x`: a sparse feature matrix of class 'dgCMatrix' with the following variables:

alcohol alcoholic content

malic malic acid

ash ash

alcalinity alcalinity of ash

magnesium magnemium

phenols total phenols

flavanoids flavanoids

nonflavanoids nonflavanoid phenols

proanthocyanins proanthocyanins

color color intensity

hue hue

dilution OD280/OD315 of diluted wines

proline proline

Source

Dua, D. and Karra Taniskidou, E. (2017). UCI Machine Learning Repository http://archive.ics.uci.edu/ml. Irvine, CA: University of California, School of Information and Computer Science.

https://raw.githubusercontent.com/hadley/rminds/master/1-data/wine.csv

https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/multiclass.html#wine

Index