Package 'serp'

January 29, 2021

Type Package

Title Smooth Effects on Response Penalty for 'CLM'

Version 0.1.8

Author Ejike R. Ugba [aut, cre, cph]

Maintainer Ejike R. Ugba <ejike.ugba@outlook.com>

Description A regularization method for the cumulative link models. The 'smooth-effect-on-response penalty' ('SERP') provides flexible modelling of the ordinal model by enabling the smooth transition from the general cumulative link model to a coarser form of the same model. In other words, as the tuning parameter goes from zero to infinity, the subject-specific effects associated with each variable in the model tend to a unique global effect. The parameter estimates of the general cumulative model are mostly unidentifiable or at least only identifiable within a range of the entire parameter space. Thus, by maximizing a penalized rather than the usual non-penalized log-likelihood, this and other numerical problems common with the general model are to a large extent eliminated. Fitting is via a modified Newton's method. Several standard model performance and descriptive methods are also available. An outline of the penalty implemented here is found in Tutz, G and Gertheiss, J (2016) <doi:10.1177/1471082X16642560>.

License GPL-2

URL https://github.com/ejikeugba/serp

BugReports https://github.com/ejikeugba/serp/issues

Depends R (>= 3.2.0)

Imports ordinal (>= 2016-12-12), stats

Suggests covr, testthat, VGAM (>= 1.1-4)

Encoding UTF-8

LazyData True

RoxygenNote 7.1.1

NeedsCompilation no

Repository CRAN

Date/Publication 2021-01-29 08:50:02 UTC

R topics documented:

`AIC.serp` *AIC for an object of class 'serp'*

Description

Returns the akaike information criterion of a fitted object of class 'serp'

Usage

```
## S3 method for class 'serp'
AIC(object, ..., k = 2)
```

Arguments

`object`	An object of class 'serp'.
`...`	additional arguments.
`k`	fixed value equal to 2.

Value

A single numeric value of the model AIC.

See Also

serp

Examples

```
# See serp() documentation for examples.
```

---

anova.serp *ANOVA method for an object of class 'serp'*

Description

Provides an ANOVA table for comparing two or more 'serp' objects.

Usage

```
## S3 method for class 'serp'
anova(object, ..., test = c("Chisq", "none"))
```

Arguments

object	An object of class 'serp'.
...	additional arguments.
test	type of test to be conducted.

Value

An ANOVA table with the following components:

model the respective model aliases.

no.par the no of parameters in the model.

AIC the akaike information criterion.

logLik the realized log-likelihood.

Test the different pair(s) of test(s) conducted.

LR.stat the computed Likelihood ratio statistic.

df the degree of freedom.

Pr(chi) the p-value of test statitic.

See Also

serp

Examples

```
# See serp() documentation for examples.
```

BIC.serp — *BIC for an object of class 'serp'*

Description

Returns the bayesian information criterion of a fitted object of class 'serp'

Usage

```
## S3 method for class 'serp'
BIC(object, ...)
```

Arguments

object	An object of class 'serp'.
...	additional arguments.

Value

A single numeric value of the model.

See Also

serp

Examples

```
# See serp() documentation for examples.
```

coef.serp — *Coefficients for a serp object*

Description

Returns the coefficients of a fitted object of class 'serp'

Usage

```
## S3 method for class 'serp'
coef(object, ...)
```

Arguments

object	An object of class serp.
...	additional arguments.

Value

A vector of model coefficients.

See Also

serp

Examples

```
# See serp() documentation for examples.
```

---

confint.serp *Confidence interval for an object of class 'serp'*

Description

Provides the confidence interval of estimates for an object of class 'serp'.

Usage

```
## S3 method for class 'serp'
confint(object, ..., parm, level = 0.95)
```

Arguments

object	An object of class 'serp'.
...	additional arguments.
parm	unused argument.
level	significance level.

Value

A matrix of the the confidence intervals of fitted model.

See Also

serp

Examples

```
# See serp() documentation for examples.
```

`errorMetrics` *Performance metrics for categorical models*

Description

Calculates performance metrics of fitted categorical models, including binary and multi-categorical models.

Usage

```
errorMetrics(
             actual,
             predicted,
             model= c("multiclass", "binary"),
             type= c("brier", "logloss", "misclass"),
             eps=.Machine$double.eps)
```

Arguments

`actual`	vector of actual values observed
`predicted`	predicted probability matrix of a categorical model or a vector of fitted values for binary models.
`model`	specifies whether multi-categorical or binary model
`type`	specifies type of error metrics
`eps`	a near-zero value introduced only if the fitted probabilities go beyond a specified threshold. It helps to minimize the chances of running into numerical problems.

Value

A numeric value of computed performance metric determining how good a categorical model is compare to competing models.

brier the brier score of fitted model.

logloss the logloss of fitted model.

misclass the misclassification error of fitted model.

See Also

`serp`

Examples

```
f1 <- serp(rating ~ temp + contact, tuneMethod = "user",
slope = "penalize", lambda = 0.3, reverse = TRUE, link = "logit",
data = wine)
errorMetrics(f1, type = "brier")
errorMetrics(f1, type = "logloss")
errorMetrics(f1, type = "misclass")

## For non-serp object of class, 'actual'  and 'predicted' values
## must be provided
set.seed(1)
y <- as.factor(rbinom(50,1,0.5))
xx <- runif(50)
f2 <- glm(y ~ xx, family = binomial(link="logit"))
p2 <- f2$fitted.values

errorMetrics(actual=y, predicted=p2, model= "binary", type = "brier")
errorMetrics(actual=y, predicted=p2, model= "binary", type = "logloss")
errorMetrics(actual=y, predicted=p2, model= "binary", type = "misclass")
```

---

logLik.serp *Log-likelihood for a serp object.*

Description

Returns the Log-likelihood for a fitted object of class 'serp'

Usage

```
## S3 method for class 'serp'
logLik(object, ...)
```

Arguments

`object`	An object of class serp.
`...`	additional arguments.

Value

A single numeric value of model log-likelihood

See Also

`serp`

Examples

```
# See serp() documentation for examples.
```

predict.serp *Predict method for object of class 'serp'.*

Description

Returns the predicted probabilities, link and class for an object of class 'serp..

Usage

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

Arguments

object	An object of class serp.
type	could be any of these: response, link or terms.
newdata	fresh dataset with all relevant variables.
...	additional arguments.

Value

A vector of predicted classes with type equal to 'class' or a dataframe of predicted values for type equal to 'response' or 'link'.

See Also

serp

Examples

```
# See serp() documentation for examples.
```

print.serp *Print method for object of class 'serp'*

Description

Prints out a vector of coefficients of the fitted model with some additional goodness-of-fit measures.

Usage

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

Arguments

x	An object of class 'serp'.
...	additional arguments.

Value

No return value

See Also

serp

Examples

```
# See serp() documentation for examples.
```

---

print.summary.serp *Print method for object of class 'summary.serp'*

Description

Prints the data frame returned by the summary.serp method.

Usage

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

Arguments

x	An object of class 'summary.serp'.
...	additional arguments.

Value

No return value

See Also

serp

Examples

```
# See serp() documentation for examples.
```

serp *Smooth Effects on Response Penalty for CLM*

Description

Fits cumulative link models (CLMs) with the smooth-effect-on-response penalty (SERP) via a modified Newton-Raphson algorithm. SERP enables the regularization of the parameter space between the general and the restricted cumulative models, with a resultant shrinkage of all subject-specific effects to global effects. The minimum deviance and CV tuning among others, provide the means of arriving at an optimal model in a situation where a user-supplied tuning value is not available. The slope argument allows for the selection of a penalized, unparallel, parallel, or partial slope.

Usage

```
serp(
     formula,
     link = c("logit", "probit","loglog", "cloglog", "cauchit"),
     slope = c("penalize", "parallel", "unparallel", "partial"),
     tuneMethod = c("deviance", "cv", "finite", "user"),
     reverse = FALSE,
     lambdaGrid = NULL,
     cvMetric = c("brier", "logloss", "misclass"),
     gridType = c("discrete", "fine"),
     globalEff = NULL,
     data,
     subset,
     weights = NULL,
     weight.type = c("analytic", "frequency"),
     na.action = NULL,
     lambda = NULL,
     contrasts = NULL,
     control = list(),
     ...)
```

Arguments

formula	regression formula of the form: response ~ predictors. The response should be a factor (ordered).
link	sets the link function for the cumulative link model including: logit, probit, complementary log-log, cloglog, cauchit.
slope	selects the form of coefficients used in the model, with penalize denoting the penalized coefficients, unparallel, parallel and partial denoting the unpenalized non-parallel, parallel and semi-parallel coefficients respectively.
tuneMethod	sets the method of choosing an optimal shrinkage parameter, including: deviance, cv, finite and user. i.e., the lambda value along parameter shrinkage path at which the fit's residual deviance or the cross-validated test error is minimal. The

	finite tuning is used to obtain the model along parameter shrinkage for which the log-Likelihood exist (is finite). The 'user' tuning supports a user-supplied lambda value.
`reverse`	false by default, when true the sign of the linear predictor is reversed.
`lambdaGrid`	optional user-supplied lambda grid for the cv and deviance tuning methods, when the discrete `gridType` is chosen. Negative range of values are not allowed. A short lambda grid could increase computation time assuming large number of predictors and cases in the model.
`cvMetric`	sets the performance metric for the cv tuning, with the brier score used by default.
`gridType`	chooses if a discrete or a continuous lambda grid should be used to select the optimal tuning parameter. The former is used by default and could be adjusted as desired in `serp.control`. The latter is on the range (0, `maxPen`). A user-supplied grid is also possible, which automatically overrides the internal grid.
`globalEff`	specifies variable(s) to be assigned global effects during penalization or when `slope` is set to `partial`. Variables are specified as a formula with an empty left hand side, for instance, globalEff = ~predictors.
`data`	optional dataframe explaining the variables used in the formula.
`subset`	specifies which subset of the rows of the data should be used for fit. All observations are used by default.
`weights`	optional case weights in fitting. Negative weights are not allowed. Defaults to 1.
`weight.type`	chooses between analytic and frequency weights with the former used by default. The latter should be used when weights are mere case counts used to compress the data set.
`na.action`	a function to filter missing data.
`lambda`	a user-supplied single numeric value for the tuning parameter when using the `user` tuning method. Negative values are not allowed.
`contrasts`	a list of contrasts to be used for some or all of the factors appearing as variables in the model formula.
`control`	A list of fit control parameters to replace default values returned by `serp.control`. Values not set assume default values.
`...`	additional arguments.

Details

The `serp` function fits the cumulative link model (CLM) with smooth-effect-on-response penalty (SERP). The cumulative model developed by McCullagh (1980) is probably most frequently used ordinal model. When motivated by an underlying latent variable, a simple form of the model is expressed as follows:

$$P(Y \leq r|x) = F(\delta_{0r} + x^T\delta)$$

where x is a vector of covariates, δ a vector of regression parameters and F a continuous distribution function. This model assumes that the effect of x does not depend on the category. However, with this assumption relaxed, one obtains the following general cumulative model:

$$P(Y \leq r|x) = F(\delta_{0r} + x^T \delta_r),$$

where r=1,...,k-1. This model, however, has the stochastic ordering property, which implies that $P(Y \leq r-1|x) < P(Y \leq r|x)$ holds for all x and all categories r. Such assumption is often problematic, resulting in unstable likelihoods with ill-conditioned parameter space during the iterative procedure.

SERP offers a means of arriving at stable estimates of the general model. It provides a form of regularization that is based on minimizing the penalized log-likelihood:

$$l_p(\delta) = l(\delta) - J_\lambda(\delta)$$

where $l(\delta)$, is the log-likelihood of the general cumulative model and $J_\lambda(\delta) = \lambda J(\delta)$ the penalty function weighted by the turning parameter λ. Assuming an ordered categorical outcome $Y \in \{1, \ldots, k\}$, and considering that the corresponding parameters $\delta_{1j}, \ldots \delta_{k-1,j}$ vary smoothly over the categories, the following penalty (Tutz and Gertheiss, 2016),

$$J_\lambda(\delta) = \sum_{j=1}^{p} \sum_{r=1}^{k-2} (\delta_{r+1,j} - \delta_{rj})^2$$

enables the smoothing of response categories such that all category-specific effects associated with the response turn towards a common global effect. SERP could also be applied to a semi-parallel model with only the category-specific part of the model penalized.

Value

An object of class `serp` with the components listed below, depending on the type of slope modeled. Other summary methods include: `summary`, `coef`, `predict`, `vcov`, `anova`, `errorMetrics`, etc.

aic the akaike information criterion.

bic the bayesian information criterion.

call the matched call.

coef a vector of coefficients of the fitted model.

converged a character vector of fit convergence status.

contrasts (where relevant) the contrasts used in the model.

control list of control parameters from `serp.control`.

cvMetric the performance metric used for cv tuning.

deviance the residual deviance.

edf the (effective) number of degrees of freedom used by the model

fitted.values the fitted probabilities.

globalEff variable(s) in model treated as global effect(s)

gradient a column vector of gradients for the coefficients at the model convergence.

Hessian the hessian matrix for the coefficients at the model convergence.

iter number of interactions before convergence or non-convergence.

lambda a user-supplied single numeric value for the `user` tuning tuning method.

lambdaGrid a numeric vector of lambda values used to determine the optimum tuning parameter.

logLik the realized log-likelihood at the model convergence.

link character vector indicating the link function of the fit.

message character vector stating the type of convergence obtained

misc a list to hold miscellaneous fit information.

model model.frame having variables from formula.

na.action (where relevant) information on the treatment of NAs.

nobs the number of observations.

nrFold the number of k-fold cross validation for the cv tuning method. Default to k = 5.

reverse a logical vector indicating the the direction of the cumulative probabilities. Default to P(Y<=r).

slope a character vector indicating the type of slope parameters fitted. Default to `penalize`.

Terms the terms structure describing the model.

testError numeric value of the cross-validated test error at which the optimal tuning parameter emerged.

trainError numeric value of the cross-validated training error of the final model.

tuneMethod a character vector specifying the method for choosing an optimal shrinkage parameter.

value numeric value of the deviance or the minus log-likelihood of the optimal model for the 'deviance' and the 'finite' tuning methods respectively.

ylev the number of the response levels.

References

McCullagh, P (1980). Regression Models for Ordinal Data. *Journal of the Royal Statistical Society. Series B (Methodological)*, 42, pp. 109-142.

Tutz, G and Gertheiss, J (2016). Regularized Regression for Categorical Data (With Discussion and Rejoinder). *Statistical Modelling*, 16, pp. 161-260.

See Also

`anova.serp`, `summary.serp`, `predict.serp`, `confint.serp`, `vcov.serp`, `errorMetrics`

Examples

```
## The unpenalized non-proportional odds model. (with Unbounded estimates)
f1 <- serp(rating ~ temp + contact, slope = "unparallel",
           reverse = TRUE, link = "logit", data = wine)
coef(f1)
logLik(f1)
```

```
## The penalized non-proportional odds model (with Improved estimates)
f2 <- serp(rating ~ temp + contact, slope = "penalize",
           link = "logit", reverse = TRUE, tuneMethod = "deviance",
           lambdaGrid = 10^seq(1, -1, length.out=5), data = wine)
coef(f2)
predict(f2, type = "class")
anova(f1, f2)

## The unpenalized proportional odds model (with constrained estimates).
f3 <-  serp(rating ~ temp + contact, slope = "parallel",
            reverse = FALSE, link = "logit", data = wine)
summary(f3)
confint(f3)
errorMetrics(f3)
```

`serp.control` *Control parameters for serp fit*

Description

Default control parameters for 'serp' fit. User-supplied control parameters could be specified in the main function.

Usage

```
serp.control(
             maxits = 5e01,
             eps = 1e-07,
             maxpen = 1e05,
             trace = 0L,
             maxAdjIter = 5e0,
             max.half.iter = 1e01,
             relTol = 1e-03,
             nrFold = 5e0,
             cv.seed = 1e01,
             grid.length = 5e01,
             minP = .Machine$double.eps,
             ...)
```

Arguments

`maxits`	the maximum number of Newton's iterations. Default to 100.
`eps`	threshold value during optimization at which the iteration routine terminates. In other words, when the reported change in the log-likelihood goes below this threshold, convergence is achieved.
`maxpen`	the upper end point of the interval from zero to be searched for a tuning parameter.

trace	prints the Newton's fitting process at each iteration step.If 0 (default) no information is printed, if 1, 2 or 3 different shades of information are printed.
maxAdjIter	the maximum allowable number of Newton step adjustment to forestall an early optimization failure. Defaults to 5.
max.half.iter	the maximum number of iteration step-halfings. Defaults to 10.
relTol	relative convergence tolerance, defaults to 1e-03. checks relative changes in the parameter estimates between Newton iterations.
nrFold	the number of k-fold cross validation for the CV tuning method. Default to k = 5.
cv.seed	single numeric value to change the random seed in CV tuning.
grid.length	the length of the discrete lambda grid for the penalty method.
minP	A near zero minimum value the fitted probabilities are allowed to get during iteration to prevent numerical instability .
...	additional arguments.

Value

a list of control parameters.

See Also

serp

Examples

```
# See serp() documentation for examples.
```

---

summary.serp *Summary method for a serp object.*

Description

Summarizes the results of the fitted model in a dataframe.

Usage

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

Arguments

object	An object of class serp.
...	Not used. Additional summary arguments.

Value

an object of class "summary.serp", a list (depending on the type of `slope` used) with the components itemized below. Note that the 'components from object' are already defined in the main function.

call the component from object.

link the component from object.

edf the component from object.

ylev the component from object.

nobs the component from object.

gradient the component from object.

Hessian the component from object.

fitted.values the component from object.

slope the component from object.

Terms the component from object.

control the component from object.

reverse the component from object.

converged the component from object.

iter the component from object.

message the component from object.

misc the component from object.

model the component from object.

coefficients the matrix of coefficients, standard errors, z-values and p-values.

logLik the component from object.

deviance the component from object.

aic the component from object.

bic the component from object.

contrasts the component from object.

penalty list of penalization information when `slope` set to "penalize".

expcoefs the exponentiated coefficients.

cvMetric the component from object.

globalEff the component from object.

lambda the component from object.

lambdaGrid v

na.action the component from object.

nrFold the component from object.

testError the component from object.

trainError the component from object.

tuneMethod the component from object.

value the component from object.

See Also

serp

Examples

```
# See serp() documentation for examples.
```

---

vcov.serp *Variance covariance matrix for a serp object*

Description

Provides the Variance covariance matrix of an object of class 'serp'.

Usage

```
## S3 method for class 'serp'
vcov(object, ...)
```

Arguments

object	An object of class 'serp'.
...	additional arguments.

Value

A variance covariance matrix of a fitted model.

See Also

serp

Examples

```
# See serp() documentation for examples.
```

`wine` *Bitterness of wine*

Description

The `wine` dataset adopted from Randall(1989), represents the outcome of a factorial experiment on factors determining the bitterness of wine. Two treatment factors (temperature and contact) with two levels each are provided, with the rating of wine taken on a continuous scale in the interval from 0 (none) to 100 (intense). These were subsequently grouped into five ordered categories ranging from 1 = 'least bitter' to 5 = 'most bitter' Altogether, nine different judges assessed wine from two bottles and out of the four treatment conditions, making a total of 72 observations.

Usage

```
wine
```

Format

A data frame with 72 rows and 6 variables:

`response` scorings of wine bitterness on a 0—100 continuous scale.

`rating` ordered factor with 5 levels; a grouped version of `response`.

`contact` factor with two levels (`"no"` and `"yes"`).

`temp` temperature: factor with two levels.

`judge` factor with nine levels.

`bottle` factor with eight levels.

Source

Taken from Randall (1989).

References

Randall, J (1989). The analysis of sensory data by generalized linear model. *Biometrical journal 7*, pp. 781–793.

Examples

```
## Not run:
str(wine)
head(wine)

## End(Not run)
```

Index