Description

Corrected AIC calculation.

Usage

\texttt{AICc(object, k=2)}

Arguments

- \texttt{object} Some model object that you can call \texttt{logLik} on (such as a \texttt{gamlr} or \texttt{glm} fit).
- \texttt{k} The usual AIC complexity penalty. \texttt{k} defaults to 2.

Details

This works just like usual AIC, but instead calculates the small sample (or high dimensional) corrected version from Hurvich and Tsai

\[
AIC_c = -2 \log LHD + k \times df \times \frac{n}{n - df - 1}.
\]

Value

A numeric value for every model evaluated.

Author(s)

Matt Taddy <mataddy@gmail.com>

References


See Also

gamlr, hockey
Cross Validation for gamlr

Description

Cross validation for gamma lasso penalty selection.

Usage

```r
cv.gamlr(x, y, nfold=5, foldid=NULL, verb=FALSE, cl=NULL, ...)
## S3 method for class 'cv.gamlr'
plot(x, select=TRUE, df=TRUE, ...)
## S3 method for class 'cv.gamlr'
coef(object, select=c("1se","min"), ...)
## S3 method for class 'cv.gamlr'
predict(object, newdata, select=c("1se","min"), ...)
```

Arguments

- `x`: Covariates; see `gamlr`.
- `y`: Response; see `gamlr`.
- `nfold`: The number of cross validation folds.
- `foldid`: An optional length-n vector of fold memberships for each observation. If specified, this dictates `nfold`.
- `verb`: Whether to print progress through folds.
- `cl`: Possible parallel library cluster. If this is not-NULL, the CV folds are executed in parallel. This copies the data `nfold` times, so make sure you have the memory space.
- `...`: Arguments to `gamlr`.
- `object`: A `gamlr` object.
- `newdata`: New `x` data for prediction.
- `select`: In prediction and coefficient extraction, select which "best" model to return: `select="min"` is that with minimum average OOS deviance, and `select="1se"` is that whose average OOS deviance is no more than 1 standard error away from the minimum. In `plot`, whether to draw these selections.
- `df`: Whether to add to the plot degrees of freedom along the top axis.

Details

Fits a `gamlr` regression to the full dataset, and then performs `nfold` cross validation to evaluate out-of-sample (OOS) performance for different penalty weights.

`plot.cv.gamlr` can be used to plot the results: it shows mean OOS deviance with 1se error bars.
Value

- `gamlr` The full-data fitted `gamlr` object.
- `nfold` The number of CV folds.
- `foldid` The length-n vector of fold memberships.
- `cvm` Mean OOS deviance by `gamlr$lambda`
- `cvs` The standard errors on `cvm`.
- `seg.min` The index of minimum `cvm`.
- `seg.1se` The index of 1se `cvm` (see details).
- `lambda.min` Penalty at minimum `cvm`.
- `lambda.1se` Penalty at 1se `cvm`.

Author(s)

Matt Taddy <mataddy@gmail.com>

References


See Also

gamlr, hockey

Examples

```r
n <- 100
p <- 100
xvar <- matrix(ncol=p,nrow=p)
for(i in 1:p) for(j in i:p) xvar[i,j] <- 0.5^abs(i-j)
x <- matrix(rnorm(p*n), nrow=n)%*%chol(xvar)
beta <- matrix( (-1)^(1:p)*exp(-(1:p)/10) )
mu = x%*%beta
y <- mu + rnorm(n,sd=sd(as.vector(mu))/2)

## fit with gamma=1 concavity
cvfit <- cv.gamlr(x, y, gamma=1, verb=TRUE)

coef(cvfit)[1:3,] # 1se default
coef(cvfit, select="min")[1:3,] # min OOS deviance
predict(cvfit, x[1:2,:], select="min")
predict(cvfit$gamlr, x[1:2,:], select=cvfit$seg.min)

par(mfrow=c(1,2))
plot(cvfit)
plot(cvfit$gamlr)
```
**doubleML**

**Description**

double (i.e., double) Machine Learning for treatment effect estimation

**Usage**

doubleML(x, d, y, nfold=2, foldid=NULL, family="gaussian", cl=NULL, ...)

**Arguments**

- `x` Covariates; see `gamlr`.
- `d` The matrix of treatment variables. Each column is used as a response by `gamlr` during the residualization procedure.
- `y` Response; see `gamlr`.
- `nfold` The number of cross validation folds.
- `foldid` An optional length-n vector of fold memberships for each observation. If specified, this dictates `nfold`.
- `family` Response model type for the treatment prediction; either "gaussian", "poisson", or "binomial". This can be either be a single family shared by all columns of `d` or a vector of families of length `ncol(d)`
- `cl` possible parallel library cluster. If this is not-NULL, the CV folds are executed in parallel. This copies the data `nfold` times, so make sure you have the memory space.
- `...` Arguments to all the `gamlr` regressions.

**Details**

Performs the double ML procedure of Chernozhukov et al. (2017) to produce an unbiased estimate of the average linear treatment effects of `d` on `y`. This procedure uses `gamlr` to regress `y` and each column of `d` onto `x`. In the cross-fitting routine described in Taddy (2019), these regressions are trained on a portion of the data and the out-of-sample residuals are calculated on the left-out fold. Model selection for these residualization steps is based on the AICc selection rule. The response residuals are then regressed onto the treatment residuals using `lm` and the resulting estimates and standard errors are unbiased for the treatment effects under the assumptions of Chernozhukov et al.

**Value**

A fitted `lm` object estimating the treatment effect of `d` on `y`. The `lm` function has been called with `x=TRUE, y=TRUE` such that this object contains the residualized `d` as `x` and residualized `y` as `y`.

**Author(s)**

Matt Taddy <mataddy@gmail.com>
References

See Also
gamlr, hockey, AICc

Examples

```r
data(hockey)
who <- which(colnames(player) == "SIDNEY_CROSBY")
s <- sample.int(nrow(player), 10000) # subsample for a fast example
doubleML(x = player[s, -who], d = player[s, who], y = goal$homegoal[s], standardize = FALSE)
```

---

### gamlr

**Gamma-Lasso regression**

**Description**

Adaptive L1 penalized regression estimation.

**Usage**

```r
gamlr(x, y, 
   family = c("gaussian", "binomial", "poisson"),
   gamma = 0, nlambda = 100, lambda.start = Inf,
   lambda.min.ratio = 0.01, free = NULL, standardize = TRUE,
   obsweight = NULL, varweight = NULL,
   prexx = (p < 500),
   tol = 1e-7, maxit = 1e5, verb = FALSE, ...)
```

```r
## S3 method for class 'gamlr'
plot(x, against = c("pen", "dev"),
     col = NULL, select = TRUE, df = TRUE, ...)
## S3 method for class 'gamlr'
coef(object, select = NULL, k = 2, corrected = TRUE, ...)
## S3 method for class 'gamlr'
predict(object, newdata, 
     type = c("link", "response"), ...)
## S3 method for class 'gamlr'
logLik(object, ...)
```
Arguments

- **x**: A dense matrix or sparse Matrix of covariates, with \( \text{ncol}(x) \) variables and \( \text{nrow}(x) = \text{length}(y) \) observations. This should not include the intercept.

- **y**: A vector of response values. There is almost no argument checking, so be careful to match \( y \) with the appropriate family.

- **family**: Response model type; either "gaussian", "poisson", or "binomial". Note that for "binomial", \( y \) is in \([0, 1]\).

- **gamma**: Penalty concavity tuning parameter; see details. Zero (default) yields the lasso, and higher values correspond to a more concave penalty.

- **nlambda**: Number of regularization path segments.

- **lambda.start**: Initial penalty value. Default of \( \text{Inf} \) implies the infimum lambda that returns all zero coefficients. This is the largest absolute coefficient gradient at the null model.

- **lambda.min.ratio**: The smallest penalty weight (expected L1 cost) as a ratio of the path start value. Our default is always 0.01; note that this differs from glmnet whose default depends upon the dimension of \( x \).

- **free**: Free variables: indices of the columns of \( x \) which will be unpenalized.

- **standardize**: Whether to standardize the coefficients to have standard deviation of one. This is equivalent to multiplying the L1 penalty by each coefficient standard deviation.

- **obsweight**: For family="gaussian" only, weights on each observation in the weighted least squares objective. For other response families, obsweights are overwritten by IRLS weights. Defaults to \( \text{rep}(1,n) \).

- **varweight**: Multipliers on the penalty associated with each covariate coefficient. Must be non-negative. These are further multiplied by \( \text{sd}(x_j) \) if standardize=TRUE. Defaults to \( \text{rep}(1,p) \).

- **prexx**: Only possible for family="gaussian": whether to use pre-calculated weighted variable covariances in gradient calculations. This leads to massive speed-ups for big-n datasets, but can be slow for \( p > n \) datasets. See note.

- **tol**: Optimization convergence tolerance relative to the null model deviance for each inner coordinate-descent loop. This is measured against the maximum coordinate change times deviance curvature after full parameter-set update.

- **maxit**: Max iterations for a single segment coordinate descent routine.

- **verb**: Whether to print some output for each path segment.

- **object**: A gamlr object.

- **against**: Whether to plot paths against log penalty or deviance.

- **select**: In coef (and predict, which calls coef), the index of path segments for which you want coefficients or prediction (e.g., do select=which.min(BIC(object)) for BIC selection). If null, the segments are selected via our AICc function with \( k \) as specified (see also corrected). If select=0 all segments are returned. In plot, select is just a flag for whether to add lines marking AICc and BIC selected models.
If `select=NULL` in `coef` or `predict`, the AICc complexity penalty. `k` defaults to the usual 2.

`corrected` A flag that swaps corrected (for high dimensional bias) AICc in for the standard AIC. You almost always want `corrected=TRUE`, unless you want to apply the BIC in which case use `k=log(n),corrected=FALSE`.

`newdata` New x data for prediction.

`type` Either "link" for the linear equation, or "response" for predictions transformed to the same domain as `y`.

`col` A single plot color, or vector of length `ncol(x)` colors for each coefficient regularization path. `NULL` uses the `matplot` default 1:6.

`df` Whether to add to the plot degrees of freedom along the top axis.

`...` Extra arguments to each method. Most importantly, from `predict.gamlr` these are arguments to `coef.gamlr`.

### Details

Finds posterior modes along a regularization path of adapted L1 penalties via coordinate descent.

Each path segment $t$ minimizes the objective $-(\phi/n)\log LHD(\beta_1, ..., \beta_p) + \sum \omega_j |\beta_j|$, where $\phi$ is the exponential family dispersion parameter ($\sigma^2$ for `family="gaussian"`, one otherwise). Weights $\omega_j$ are set as $1/(1 + \gamma |\beta_j|)$ where $\beta_j$ is our estimate of $\beta_j$ for the previous path segment (or zero if $t = 0$). This adaptation is what makes the penalization 'concave'; see Taddy (2013) for details.

`plot.gamlr` can be used to graph the results: it shows the regularization paths for penalized $\beta$, with degrees of freedom along the top axis and minimum AICc selection marked.

`logLik.gamlr` returns log likelihood along the regularization path. It is based on the deviance, and is correct only up to static constants; e.g., for a Poisson it is off by $\sum_i y_i (\log y_i - 1)$ (the saturated log likelihood) and for a Gaussian it is off by likelihood constants $(n/2)(1 + \log 2\pi)$.

### Value

- `lambda` The path of fitted prior expected L1 penalties.
- `nobs` The number of observations.
- `alpha` Intercepts.
- `beta` Regression coefficients.
- `df` Approximate degrees of freedom.
- `deviance` Fitted deviance: $(-2/\phi)(\log LHD.fitted - \log LHD.saturated)$.
- `iter` Number of optimization iterations by segment, broken into coordinate descent cycles and IRLS re-weightings for `family="gaussian"`.
- `family` The exponential family model.

### Note

Under `prexx=TRUE` (requires `family="gaussian"`), weighted covariances $(V X)' X$ and $(V X)' y$, weighted column sums of $V X$, and column means $\bar{x}$ will be pre-calculated. Here $V$ is the diagonal matrix of least squares weights (obsweights, so $V$ defaults to I). It is not necessary (they will be
built by gamlr otherwise), but you have the option to pre-calculate these sufficient statistics yourself as arguments vxx (matrix or dspMatrix), vxy, vxsum, and xbar (all vectors) respectively. Search PREXX in gamlr.R to see the steps involved, and notice that there is very little argument checking – do at your own risk. Note that xbar is an unweighted calculation, even if $V \neq I$. For really Big Data you can then run with $x=\text{NULL}$ (e.g., if these statistics were calculated on distributed machines and full design is unavailable). Beware: in this $x=\text{NULL}$ case our deviance (and df, if $\gamma>0$) calculations are incorrect and selection rules will always return the smallest-lambda model.

Author(s)
Matt Taddy <mataddy@gmail.com>

References

See Also
cv.gamlr, AICc, hockey

Examples

### a low-D test (highly multi-collinear)

```r
n <- 1000
p <- 3
xvar <- matrix(0.9, nrow=p, ncol=p)
diag(xvar) <- 1
x <- matrix(rnorm(p*n), nrow=n)%*%chol(xvar)
y <- 4 + 3*x[,1] + -1*x[,2] + rnorm(n)

## run models to extra small lambda 1e-3*lambda.start
fitlasso <- gamlr(x, y, gamma=0, lambda.min.ratio=1e-3) # lasso
fitgl <- gamlr(x, y, gamma=2, lambda.min.ratio=1e-3) # small gamma
fitglbv <- gamlr(x, y, gamma=10, lambda.min.ratio=1e-3) # big gamma

par(mfrow=c(1,3))
ylim = range(c(fitglbv$beta@x))
plot(fitlasso, ylim=ylim, col="navy")
plot(fitgl, ylim=ylim, col="maroon")
plot(fitglbv, ylim=ylim, col="darkorange")
```
hockey

NHL hockey data

Description

Every NHL goal from fall 2002 through the 2014 cup finals.

Details

The data comprise of information about play configuration and the players on ice (including goalies) for every goal from 2002-03 to 2012-14 NHL seasons. Collected using A. C. Thomas’s `n1hscrapr` package. See the Chicago hockey analytics project at github.com/mataddy/hockey.

Value

- **goal**: Info about each goal scored, including homegoal – an indicator for the home team scoring.
- **player**: Sparse Matrix with entries for who was on the ice for each goal: +1 for a home team player, -1 for an away team player, zero otherwise.
- **team**: Sparse Matrix with indicators for each team*season interaction: +1 for home team, -1 for away team.
- **config**: Special teams info. For example, S5v4 is a 5 on 4 powerplay, +1 if it is for the home-team and -1 for the away team.

Author(s)

Matt Taddy, <mataddy@gmail.com>

References


See Also

gamlr

Examples

```r
## design
data(hockey)
x <- cbind(config, team, player)
y <- goal$homegoal

## fit the plus-minus regression model
```
## (non-player effects are unpenalized)

```r
t <- gamlr(x, y,
  lambda.min.ratio=0.05, nlambda=40, ## just so it runs in under 5 sec
  free=1:(ncol(config)+ncol(team)),
  standardize=FALSE, family="binomial")
plot(t)
```

## look at estimated player [career] effects

```r
B <- coef(t)[colnames(player),]
sum(B!=0) # number of measurable effects (AICc selection)
B[order(-B)[1:10]] # 10 biggest
```

## convert to 2013-2014 season partial plus-minus

```r
now <- goal$season=="20132014"
pm <- colSums(player[now,names(B)]*-1,1)[y[now]+1] # traditional plus minus
ng <- colSums(abs(player[now,names(B)])) # total number of goals
# The individual effect on probability that a
given goal is for vs against that player’s team
p <- 1/(1+exp(-B))
# multiply ng*p - ng*(1-p) to get expected plus-minus
ppm <- ng*(2*p-1)
```

# organize the data together and print top 20

```r
effect <- data.frame(b=round(B,3),ppm=round(ppm,3),pm)
effect <- effect[order(-effect$ppm),]
print(effect[1:20,])
```

---

### naref

<table>
<thead>
<tr>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Set NA as the reference level for factor variables and do imputation on missing values for numeric variables. This is useful to build model matrices for regularized regression, and for dealing with missing values, as in Taddy 2019.</td>
</tr>
</tbody>
</table>

### Usage

```r
naref(x, impute=FALSE, pzero=0.5)
```

### Arguments

- **x**: A data frame.
- **impute**: Logical, whether to impute missing values in numeric columns.
- **pzero**: If impute==TRUE, then if more than pzero of the values in a column are zero do zero imputation, else do mean imputation.
Details

For every factor or character column in x, naref sets NA as the reference level for a factor variable. Columns coded as character class are first converted to factors via \texttt{Rfactor(x)}. If \texttt{impute=TRUE} then the numeric columns are converted to two columns, one appended \texttt{.x} that contains imputed values and another appended \texttt{.miss} which is a binary variable indicating whether the original value was missing. Numeric columns are returned without change if \texttt{impute=FALSE} or if they do not contain any missing values.

Value

A data frame where the factor and character columns have been converted to factors with reference level NA, and if \texttt{impute=TRUE} the missing values in numeric columns have been imputed and a flag for missingness has been added. See details.

Author(s)

Matt Taddy <mataddy@gmail.com>

References


Examples

\begin{verbatim}
(x <- data.frame(a=factor(c(1,2,3)),b=c(1,NA,3)))
naref(x, impute=TRUE)
\end{verbatim}
Index

AICc, 2

coeff.cv.gamlr (cv.gamlr), 3
coeff.gamlr (gamlr), 6
config (hockey), 10
cv.gamlr, 3
doubleML, 5
gamlr, 6
goal (hockey), 10
hockey, 10
logLik.gamlr (gamlr), 6
naref, 11
player (hockey), 10
plot.cv.gamlr (cv.gamlr), 3
plot.gamlr (gamlr), 6
predict.cv.gamlr (cv.gamlr), 3
predict.gamlr (gamlr), 6
team (hockey), 10