Package: glmfsa (via r-universe)

November 3, 2026

Type Package

Title Feature Selection with Annealing for Sparse Generalized Linear Models

Version 1.0.1

Date 2026-10-21

Description Fits sparse generalized linear models under an exact cardinality constraint using Feature Selection with Annealing (FSA), the method of Barbu, She, Ding and Gramajo (2017) <doi:10.1109/TPAMI.2016.2544315>. Rather than inducing sparsity through an L1 (lasso-type) penalty, FSA alternates a gradient step on the loss with a step that removes the lowest-magnitude coefficients according to an annealing schedule, so the user specifies the number of variables directly and the surviving coefficients carry no shrinkage bias. The working design matrix physically shrinks as iterations proceed, which makes the cost of a fit fall over time and the method well suited to problems with very many features. Gaussian, binomial, Poisson, multinomial, multi-response Gaussian and Cox proportional hazards families are supported, with robust Lorenz and smoothed support vector machine (SVM) losses available for classification with noisy labels. Dense and sparse design matrices are both handled, and the interface mirrors that of the 'glmnet' package.

License GPL-3

Encoding UTF-8

LazyData true

LazyDataCompression xz

Depends R (>= 3.5.0)

Imports Rcpp (>= 1.0.0), Matrix, methods, graphics, grDevices, stats, tools, utils

LinkingTo Rcpp, RcppArmadillo

Suggests testthat (>= 3.0.0), knitr, rmarkdown, glmnet, survival, foreach, doParallel, MASS

VignetteBuilder knitr

URL https://github.com/barbua/FSA

BugReports https://github.com/barbua/FSA/issues

Config/testthat/edition 3

NeedsCompilation yes

Config/roxygen2/version 8.1.0

Author Adrian Barbu [aut, cre, cph], Yiyuan She [aut], Liangjing Ding [ctb], Gary Gramajo [ctb]

Maintainer Adrian Barbu <abarbu@fsu.edu>

Repository https://cran.r-universe.dev

Date/Publication 2026-11-03 21:12:58 UTC

RemoteUrl https://github.com/cran/glmfsa

RemoteRef HEAD

RemoteSha 81c50becfa91ce2ce6efd0987e92bf751c36fc34

Contents

coef.fsa . . . 2
cv.fsa . . . 3
cv.fsa-methods . . . 5
deviance.fsa . . . 6
dexter . . . 6
fetch_gisette . . . 7
fsa . . . 8
fsa_losses . . . 12
plot.fsa . . . 13
plot_schedule . . . 13
plot_trace . . . 14
predict.fsa . . . 15
print.fsa . . . 16

Index **18**

`coef.fsa` *Extract coefficients from a fitted FSA model*

Description

Extract coefficients from a fitted FSA model

Usage

```
## S3 method for class 'fsa'
coef(object, k = NULL, ...)
```

Arguments

object	A fitted ‘"fsa"‘ object.
k	Which model size(s) to extract. Defaults to all fitted values. Values must match (or will be matched to the nearest) fitted ‘k‘.
...	Ignored.

Value

A sparse matrix with ‘nvars + 1‘ rows (the intercept first), or a list of such matrices for ‘"multinomial"‘ and ‘"mgaussian"‘ fits.

Examples

```
set.seed(1)
x <- matrix(rnorm(100 * 40), 100, 40)
y <- x[, 3] - 2 * x[, 17] + rnorm(100)
coef(fsa(x, y, k = 2))
```

---

cv.fsa *Cross-validation for FSA*

Description

Chooses the model size ‘k‘ by K-fold cross-validation. The interface and the returned object deliberately mirror [glmnet::cv.glmnet()], so existing analysis code needs only the names changed.

Usage

```
cv.fsa(
  x,
  y,
  family = "gaussian",
  k = NULL,
  s = 0,
  type.measure = c("default", "deviance", "mse", "mae", "class", "auc", "C"),
  nfolds = 10L,
  foldid = NULL,
  weights = NULL,
  parallel = FALSE,
  keep = FALSE,
  ...
)
```

Arguments

`x`, `y`, `family`, `...`	Passed to [fsa()]. '...' may contain any 'fsa()' argument except 'k' and 's', which are controlled here.
`k`	Vector of model sizes to evaluate. If 'NULL', the default grid from [fsa()] is used.
`s`	Ridge coefficient. A single value is passed straight to [fsa()]; a vector is cross-validated jointly with 'k', and the chosen value is returned as 's.min'. Tuning 's' matters when 'p' is much larger than 'n': on the bundled [dexter] data (n = 300, p = 20000) moving from 's = 0' to a cross-validated 's' is the difference between losing to the lasso and beating it.
`type.measure`	Loss used for cross-validation. '"default"' selects the family's deviance. Also available: '"mse"', '"mae"', '"class"' (misclassification rate), '"auc"' (binomial only) and '"C"' (Harrell's concordance, Cox only).
`nfolds`	Number of folds. Ignored if 'foldid' is supplied.
`foldid`	Optional vector of fold memberships, one per observation.
`weights`	Optional observation weights.
`parallel`	Evaluate folds in parallel with **foreach**? Requires a registered parallel backend.
`keep`	Keep the matrix of pre-validated out-of-fold linear predictors.

Value

An object of class '"cv.fsa"' with components 'k', 'cvm', 'cvsd', 'cvup', 'cvlo', 'nzero', 'k.min', 'k.1se', 's.min', 'index', 'name', 'fsa.fit' and, if 'keep = TRUE', 'fit.preval' and 'foldid'. When 's' was a vector, 'cvm.s' holds the full 'length(k) x length(s)' surface.

See Also

[fsa()], [plot.cv.fsa()]

Examples

```
set.seed(1)
n <- 200; p <- 300
x <- matrix(rnorm(n * p), n, p)
b <- numeric(p); b[c(7, 23, 61, 140)] <- 1.5
y <- as.numeric(x %*% b + rnorm(n))

cvfit <- cv.fsa(x, y, k = c(20, 10, 6, 4, 2, 1), nfolds = 5)
cvfit$k.min
coef(cvfit, k = "k.min")[1:5, ]
```

`cv.fsa-methods` *Methods for cross-validated FSA fits*

Description

Methods for cross-validated FSA fits

Usage

```
## S3 method for class 'cv.fsa'
predict(object, newx, k = "k.min", ...)

## S3 method for class 'cv.fsa'
coef(object, k = "k.min", ...)

## S3 method for class 'cv.fsa'
print(x, digits = max(3, getOption("digits") - 3), ...)

## S3 method for class 'cv.fsa'
plot(x, sign.lambda = 1, ...)
```

Arguments

`object`, `x`	A '"cv.fsa"' object.
`newx`	Matrix of new values of 'x'.
`k`	Either '"k.min"', '"k.1se"', or a numeric model size.
`...`	Passed on to the corresponding 'fsa' method.
`digits`	Number of significant digits.
`sign.lambda`	Reverse the horizontal axis (kept for familiarity with **glmnet**; '-1' plots decreasing 'k' left to right).

Value

'predict' returns predictions at the chosen 'k'; 'coef' returns the coefficient vector; 'plot' draws the cross-validation curve and returns 'NULL' invisibly; 'print' returns 'x' invisibly.

Examples

```
set.seed(1)
x <- matrix(rnorm(150 * 60), 150, 60)
y <- as.numeric(x[, c(3, 9)] %*% c(2, -2) + rnorm(150))
cvfit <- cv.fsa(x, y, k = c(10, 5, 2, 1), nfolds = 5)
plot(cvfit)
head(predict(cvfit, x, k = "k.1se"))
```

`deviance.fsa` *Deviance of a fitted FSA model*

Description

Deviance of a fitted FSA model

Usage

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

Arguments

`object`	A fitted ‘"fsa"‘ object.
`...`	Ignored.

Details

Returns ‘2 * sum(weights) * loss‘, where ‘loss‘ is the weighted mean training loss. For ‘family = "gaussian"‘ this is the residual sum of squares; for ‘"binomial"‘ with the logistic loss, ‘"poisson"‘ and ‘"multinomial"‘ it is the usual deviance up to the saturated-model constant; for ‘"cox"‘ it is minus twice the Breslow partial log-likelihood.

Value

A numeric vector with one deviance per fitted value of ‘k‘.

Examples

```
set.seed(1)
x <- matrix(rnorm(100 * 30), 100, 30)
deviance(fsa(x, x[, 1] + rnorm(100), k = c(5, 2)))
```

`dexter` *Dexter: text categorization with 20,000 features and 300 observations*

Description

One of the five datasets of the NIPS 2003 Feature Selection Challenge. The task is to decide whether a Reuters newswire article is about corporate acquisitions. Documents are represented as a bag of words, and the feature set was deliberately padded: of the 20,000 columns, 10,053 are **probes** – randomly permuted features with no relationship to the label, included so that a method’s false-discovery behavior can be measured. The design is very sparse (0.47 which makes it a natural test for cardinality-constrained fitting.

Usage

```
dexter
```

Format

A list with four elements:

x 300 x 20000 sparse ‘dgCMatrix‘ of word counts (training).

y Factor of length 300 with levels ‘"neg"‘ and ‘"pos"‘, balanced 150/150.

xvalid 300 x 20000 sparse ‘dgCMatrix‘ (validation).

yvalid Factor of length 300, balanced 150/150.

Details

Test-set labels were never released, so the published validation split is supplied here as the held-out set.

Source

UCI Machine Learning Repository, https://archive.ics.uci.edu/dataset/168/dexter. Licensed CC BY 4.0.

References

Guyon, I., Gunn, S., Ben-Hur, A. and Dror, G. (2004). Result analysis of the NIPS 2003 feature selection challenge. *Advances in Neural Information Processing Systems* **17**.

Examples

```
data(dexter)
dim(dexter$x)
fit <- fsa(dexter$x, dexter$y, family = "binomial", k = 20)
mean(predict(fit, dexter$xvalid, k = 20, type = "class") !=
       as.character(dexter$yvalid))
```

---

fetch_gisette *Download the Gisette dataset*

Description

Gisette is another dataset from the NIPS 2003 Feature Selection Challenge: separating the handwritten digits 4 and 9, with 5,000 features of which 2,500 are probes (random features carrying no signal). At 6,000 training observations it is too large to ship inside an R package, so it is downloaded from the UCI Machine Learning Repository on first use and cached.

Usage

```
fetch_gisette(cache_dir = tools::R_user_dir("glmfsa", "cache"), quiet = FALSE)
```

Arguments

cache_dir	Directory for the cached copy. Defaults to ‘tools::R_user_dir("glmfsa", "cache")‘.
quiet	Suppress download progress?

Value

A list shaped like [dexter]: ‘x‘ (6000 x 5000 numeric matrix), ‘y‘ (factor, 3000/3000), ‘xvalid‘ (1000 x 5000), ‘yvalid‘ (factor, 500/500).

Source

UCI Machine Learning Repository, https://archive.ics.uci.edu/dataset/170/gisette. Licensed CC BY 4.0.

References

Guyon, I., Gunn, S., Ben-Hur, A. and Dror, G. (2004). Result analysis of the NIPS 2003 feature selection challenge. *Advances in Neural Information Processing Systems* **17**.

Examples

```
## Not run during package checks: the first call downloads ~21 MB from UCI.
## Not run:
g <- fetch_gisette()
fit <- fsa(g$x, g$y, family = "binomial", k = 50)
mean(predict(fit, g$xvalid, k = 50, type = "class") !=
       as.character(g$yvalid))

## End(Not run)
```

---

fsa *Fit a sparse generalized linear model by Feature Selection with Annealing*

Description

‘fsa()‘ fits a linear model under an *exact* constraint on the number of non-zero coefficients, using the Feature Selection with Annealing algorithm of Barbu et al. (2017). It alternates a gradient step on the loss with a step that removes the lowest-magnitude coefficients according to an annealing schedule, so that the working design matrix physically shrinks as the fit proceeds.

Usage

```
fsa(
  x,
  y,
 family = c("gaussian", "binomial", "poisson", "multinomial", "mgaussian", "cox"),
  k = NULL,
```

```
  nk = 10L,
  loss = NULL,
  s = 0,
  niter = 500L,
  mu = 300,
  eta = NULL,
  mom = 0.9,
  nesterov = FALSE,
  minibatch = 0L,
  schedule = c("inverse", "exponential", "linear"),
  perc_exp = 0.8,
  standardize = TRUE,
  intercept = TRUE,
  weights = NULL,
  offset = NULL,
  include = NULL,
  exclude = NULL,
  nwarm = 10L,
  adapt = TRUE,
  cap = TRUE,
  balance = TRUE,
  keep_trace = FALSE,
  verbose = 0L
)
```

Arguments

x	Input matrix of dimension ‘nobs x nvars‘. Either a dense ‘matrix‘ or any sparse matrix that can be coerced to ‘Matrix::dgCMatrix‘.
y	Response. A numeric vector for ‘"gaussian"‘ and ‘"poisson"‘; a two-level factor (or 0/1, or -1/1 vector) for ‘"binomial"‘; a factor for ‘"multinomial"‘; a numeric matrix for ‘"mgaussian"‘; a two-column matrix of ‘(time, status)‘ or a ‘survival::Surv‘ object for ‘"cox"‘.
family	Model family. One of ‘"gaussian"‘, ‘"binomial"‘, ‘"poisson"‘, ‘"multinomial"‘, ‘"mgaussian"‘, ‘"cox"‘.
k	Target number of selected variables: a single value, or a decreasing vector to fit a whole path. If ‘NULL‘, a log-spaced grid of ‘nk‘ values is used.
nk	Length of the default ‘k‘ grid when ‘k‘ is ‘NULL‘.
loss	Per-observation loss for ‘family = "binomial"‘: ‘"logistic"‘ (default), ‘"lorenz"‘, ‘"svm"‘ or ‘"hinge"‘. The Lorenz and smoothed support vector machine (SVM) losses are substantially more robust to mislabeled training data; see [fsa_losses()]. Ignored for other families.
s	Coefficient of the ridge prior ‘s * ||beta||^2‘ (paper eq. 4). A small positive value is worth using when predictors are strongly correlated.
niter	Number of annealing iterations per path segment.
mu	Annealing rate of the schedule; larger drops variables faster.

eta	Gradient step size. Defaults depend on ‘family‘ and ‘loss‘.
mom	Momentum coefficient for the gradient moving average, in ‘[0, 1)‘.
nesterov	Use Nesterov lookahead when computing the gradient.
minibatch	If positive, use mini-batch stochastic gradient descent with this batch size instead of full-batch updates. Not available for ‘family = "cox"‘, whose partial likelihood is not separable over observations.
schedule	Annealing schedule: ‘"inverse"‘ (paper eq. 3, the default), ‘"exponential"‘ or ‘"linear"‘.
perc_exp	Decay base for the exponential schedule.
standardize	Standardize the columns of ‘x‘ to unit variance before fitting? Coefficients are always returned on the original scale.
intercept	Fit an intercept? Forced to ‘FALSE‘ for ‘family = "cox"‘.
weights	Optional observation weights.
offset	Optional offset; a vector, or an ‘nobs x K‘ matrix for multi-response families.
include	Indices of variables that must never be eliminated (the analogue of ‘penalty.factor = 0‘ in **glmnet**).
exclude	Indices of variables to drop before fitting.
nwarm	Number of initial iterations run with no elimination, so the coefficients can develop before the first keep-or-kill decision. Follows the reference implementation’s ‘nEpNoSel‘. Raise it on strongly correlated designs, where one gradient step from zero is little more than marginal screening.
adapt	Halve the step size automatically when the objective stops decreasing. This is what makes the default ‘eta‘ safe on strongly correlated designs, where the Hessian’s largest eigenvalue is big enough that a fixed step oscillates rather than converging. Full-batch fits only.
cap	Cap ‘eta‘ at the largest value for which the gradient step is guaranteed to descend, estimated from the largest eigenvalue of the scaled Gram matrix (Theorem 2.1 of the paper). Costs about a dozen matrix-vector products and removes the main way an unlucky ‘eta‘ can silently produce a bad fit. The bound is re-estimated as the design shrinks, since eliminated columns take their curvature with them: ‘eta_cap‘ reports the bound for the full design and ‘eta_final‘ the step in use at the end of the fit.
balance	For ‘family = "binomial"‘, reweight the two classes to equal total weight. Matches the reference implementation’s behavior for the logistic loss.
keep_trace	Record the coefficient trajectory of the first path segment so it can be drawn with [plot_trace()]. Costs memory proportional to the area under the annealing schedule.
verbose	‘0‘ silent, ‘1‘ progress every 50 iterations, ‘2‘ every iteration.

Details

Where [glmnet::glmnet()] asks for a penalty ‘lambda‘ and gives you whatever model size that implies, ‘fsa()‘ asks for the model size ‘k‘ directly. The surviving coefficients are not shrunk toward zero by a sparsity penalty, so the "fit the lasso to select, then refit on the selected variables to de-bias" two-step is unnecessary.

Value

An object of class ‘"fsa"‘, a list with components:

`beta`	Coefficients: a sparse ‘nvars x length(k)‘ matrix, or for ‘"multinomial"‘/‘"mgaussian"‘ a list of such matrices, one per response.
`a0`	Intercepts, one per value of ‘k‘ (a ‘K x length(k)‘ matrix for multi-response families).
`k`, `df`	Requested and achieved numbers of selected variables.
`selected`	List of selected variable indices, one element per ‘k‘.
`loss`, `nulldev`, `dev.ratio`	Training loss, null-model loss, and the fraction of the null loss explained.
`trace`	Coefficient trajectory, if ‘keep_trace = TRUE‘.
`call`, `family`, `loss`, `dim`, `nobs`, `npasses`, `time`	Fit metadata.

Choosing the tuning parameters

The defaults follow the paper and are deliberately not aggressive. ‘niter‘ controls how long the anneal runs, ‘mu‘ how fast variables are dropped (large ‘mu‘ drops most variables almost immediately and is much cheaper; ‘mu = 0‘ drops them linearly), and ‘eta‘ is the gradient step size. Figure 6 of the paper shows prediction performance is flat over roughly an order of magnitude in each of the three, so tuning them is rarely the best use of effort – tune ‘k‘ instead, with [cv.fsa()].

The ‘k‘ path

Passing a decreasing vector to ‘k‘ returns one model per entry, computed in a single annealed pass: the run anneals down to ‘k[1]‘, records the fit, then continues annealing to ‘k[2]‘, and so on. The supports are therefore nested and the whole path costs little more than the single fit at ‘min(k)‘.

References

Barbu, A., She, Y., Ding, L. and Gramajo, G. (2017). Feature Selection with Annealing for Computer Vision and Big Data Learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence* **39**(2), 272–286. doi:10.1109/TPAMI.2016.2544315

See Also

[cv.fsa()] to choose ‘k‘, [predict.fsa()], [plot.fsa()], [plot_trace()]

Examples

```
set.seed(1)
n <- 200; p <- 500
x <- matrix(rnorm(n * p), n, p)
beta <- numeric(p); beta[c(10, 50, 100, 200, 300)] <- 1.5
y <- as.numeric(x %*% beta + rnorm(n))

fit <- fsa(x, y, family = "gaussian", k = 5)
```

```
which(coef(fit)[-1] != 0)        # the five true variables

## a whole path of model sizes in one pass
fit2 <- fsa(x, y, k = c(20, 10, 5, 2))
fit2
```

`fsa_losses` *Per-observation classification losses used by FSA*

Description

Evaluates the margin-based losses available for ‘family = "binomial"‘, as defined in equations (6)-(8) of Barbu et al. (2017). Useful for seeing why the Lorenz loss is the robust choice: it is the only one of the four whose value stays bounded in growth as the margin becomes very negative, so a handful of mislabeled points cannot dominate the fit.

Usage

```
fsa_losses(u, loss = c("logistic", "lorenz", "svm", "hinge"), h = 0.1)
```

Arguments

`u`	Numeric vector of margins ‘y * x’beta‘.
`loss`	One of ‘"logistic"‘, ‘"lorenz"‘, ‘"svm"‘, ‘"hinge"‘.
`h`	Huberization half-width for the smoothed support vector machine (SVM) loss.

Value

A numeric vector of loss values.

Examples

```
u <- seq(-6, 3, length.out = 200)
matplot(u, sapply(c("logistic", "lorenz", "svm", "hinge"),
                  function(l) fsa_losses(u, l)),
        type = "l", lty = 1, ylab = "loss", xlab = "margin")
```

plot.fsa *Plot the coefficient path of a fitted FSA model*

Description

Draws each selected coefficient against the model size ‘k‘. Because FSA supports along a path are nested, a coefficient enters the plot at the largest ‘k‘ where it survives and continues to the sparsest model that retains it.

Usage

```
## S3 method for class 'fsa'
plot(x, xvar = c("k", "df"), label = FALSE, nlabel = 10L, response = 1L, ...)
```

Arguments

x	A fitted ‘"fsa"‘ object with more than one value of ‘k‘.
xvar	Horizontal axis: ‘"k"‘ (requested model size) or ‘"df"‘ (achieved number of non-zero coefficients).
label	Annotate the most prominent curves with their variable names.
nlabel	How many curves to annotate when ‘label = TRUE‘.
response	For ‘"multinomial"‘ and ‘"mgaussian"‘ fits, which response to plot.
...	Passed to [graphics::matplot()].

Value

‘NULL‘, invisibly. Called for its side effect of drawing the plot.

Examples

```
set.seed(1)
x <- matrix(rnorm(200 * 100), 200, 100)
y <- as.numeric(x[, c(5, 20, 60)] %*% c(2, -1.5, 1) + rnorm(200))
plot(fsa(x, y, k = c(30, 20, 15, 10, 6, 4, 3, 2, 1)))
```

plot_schedule *Plot an annealing schedule*

Description

Draws the number of surviving variables against the iteration index for the schedules implemented by the package – the figure that makes the cost of a fit obvious, since total work is proportional to the area under the curve (paper Fig. 3 and Table 1).

Usage

```
plot_schedule(
  M = 1000,
  k = 10,
  niter = 500,
  mu = c(0, 10, 100, 300),
  schedule = c("inverse", "exponential", "linear"),
  perc_exp = 0.8,
  ...
)
```

Arguments

M	Number of variables at the start.
k	Target number of variables.
niter	Number of iterations.
mu	Annealing rate, or a vector of rates to overlay.
schedule	‘"inverse"‘, ‘"exponential"‘ or ‘"linear"‘.
perc_exp	Decay base for the exponential schedule.
...	Passed to [graphics::matplot()].

Value

Invisibly, a matrix of retained-variable counts with one column per value of ‘mu‘.

Examples

```
plot_schedule(M = 1000, k = 10, niter = 500, mu = c(0, 1, 10, 100, 300))
```

---

plot_trace *Plot the annealing trajectory*

Description

Reproduces Figure 2 of Barbu et al. (2017): every coefficient's value against the iteration number, with curves terminating as the annealing schedule eliminates their variables. It is the most direct picture of what the algorithm does, and a quick diagnostic – if coefficients are still moving sharply at the final iteration, ‘niter‘ is too small or ‘eta‘ too large.

Usage

```
plot_trace(object, highlight = NULL, show_size = TRUE, ...)
```

Arguments

object	A fit produced with ‘keep_trace = TRUE‘.
highlight	Optional vector of variable indices to draw in color.
show_size	Overlay the number of surviving variables on a second axis.
...	Passed to [graphics::matplot()].

Value

‘NULL‘, invisibly. Called for its side effect of drawing the plot.

Examples

```
set.seed(1)
x <- matrix(rnorm(200 * 200), 200, 200)
y <- as.numeric(x[, c(10, 50, 120)] %*% c(2, -2, 1.5) + rnorm(200))
fit <- fsa(x, y, k = 3, niter = 120, keep_trace = TRUE)
plot_trace(fit, highlight = c(10, 50, 120))
```

---

predict.fsa *Predict from a fitted FSA model*

Description

Predict from a fitted FSA model

Usage

```
## S3 method for class 'fsa'
predict(
  object,
  newx,
  k = NULL,
  type = c("link", "response", "class", "coefficients", "nonzero"),
  newoffset = NULL,
  ...
)

## S3 method for class 'fsa'
fitted(object, ...)
```

Arguments

object	A fitted ‘"fsa"‘ object.
newx	Matrix of new values for ‘x‘. Required for all types except ‘"coefficients"‘ and ‘"nonzero"‘.
k	Which model size(s) to predict from. Defaults to all fitted values.

type	One of ‘"link"‘ (the linear predictor), ‘"response"‘, ‘"class"‘ (classification families only), ‘"coefficients"‘ or ‘"nonzero"‘ (indices of the selected variables).
newoffset	Offset to add to the linear predictor, if the fit used one.
...	Ignored.

Details

For ‘family = "binomial"‘ with a margin-based ‘loss‘ (‘"lorenz"‘, ‘"svm"‘, ‘"hinge"‘), ‘type = "response"‘ returns ‘plogis()‘ of the margin. That is monotone in the decision value – so it ranks observations correctly and is fine for ROC or AUC – but it is not a calibrated probability, because those losses do not estimate one. Use ‘loss = "logistic"‘ if you need probabilities.

Value

A matrix of predictions with one column per value of ‘k‘, or an array ‘nobs x K x length(k)‘ for multi-response families.

Examples

```
set.seed(1)
x <- matrix(rnorm(200 * 50), 200, 50)
y <- factor(x[, 5] + x[, 20] > 0)
fit <- fsa(x, y, family = "binomial", k = c(5, 2))
head(predict(fit, x, k = 2, type = "response"))
predict(fit, type = "nonzero")
```

---

print.fsa *Print a fitted FSA model*

Description

Print a fitted FSA model

Usage

```
## S3 method for class 'fsa'
print(x, digits = max(3, getOption("digits") - 3), ...)

## S3 method for class 'fsa'
summary(object, digits = max(3, getOption("digits") - 3), ...)
```

Arguments

digits	Number of significant digits.
...	Ignored.
object, x	A fitted ‘"fsa"‘ object.

Value

‘x‘, invisibly. Called for the printed summary of the ‘k‘ path.

Examples

```
set.seed(1)
x <- matrix(rnorm(100 * 30), 100, 30)
print(fsa(x, x[, 1] + rnorm(100), k = c(5, 2)))
```

Index

∗ **datasets**
dexter, 6

coef.cv.fsa *(*cv.fsa-methods*)*, 5
coef.fsa, 2
cv.fsa, 3
cv.fsa-methods, 5

deviance.fsa, 6
dexter, 6

fetch_gisette, 7
fitted.fsa *(*predict.fsa*)*, 15
fsa, 8
fsa_losses, 12

plot.cv.fsa *(*cv.fsa-methods*)*, 5
plot.fsa, 13
plot_schedule, 13
plot_trace, 14
predict.cv.fsa *(*cv.fsa-methods*)*, 5
predict.fsa, 15
print.cv.fsa *(*cv.fsa-methods*)*, 5
print.fsa, 16

summary.fsa *(*print.fsa*)*, 16