

Package ‘nnet’

October 28, 2009

Priority recommended

Version 7.3-1

Date 2009-05-09

Depends R (>= 2.5.0), stats, utils

Suggests MASS

Author Brian Ripley <ripley@stats.ox.ac.uk>.

Maintainer Brian Ripley <ripley@stats.ox.ac.uk>

Description Software for feed-forward neural networks with a single hidden layer, and for multinomial log-linear models.

Title Feed-forward Neural Networks and Multinomial Log-Linear Models

License GPL-2 | GPL-3

URL <http://www.stats.ox.ac.uk/pub/MASS4/>

LazyLoad yes

Repository CRAN

Date/Publication 2009-10-28 07:17:10

R topics documented:

class.ind	2
multinom	2
nnet	4
nnetHess	7
predict.nnet	8
which.is.max	9

Index	11
--------------	-----------

class.ind *Generates Class Indicator Matrix from a Factor*

Description

Generates a class indicator function from a given factor.

Usage

```
class.ind(cl)
```

Arguments

cl factor or vector of classes for cases.

Value

a matrix which is zero except for the column corresponding to the class.

References

Venables, W. N. and Ripley, B. D. (2002) *Modern Applied Statistics with S*. Fourth edition. Springer.

Examples

```
# The function is currently defined as
class.ind <- function(cl)
{
  n <- length(cl)
  cl <- as.factor(cl)
  x <- matrix(0, n, length(levels(cl)) )
  x[(1:n) + n*(unclass(cl)-1)] <- 1
  dimnames(x) <- list(names(cl), levels(cl))
  x
}
```

multinom *Fit Multinomial Log-linear Models*

Description

Fits multinomial log-linear models via neural networks.

Usage

```
multinom(formula, data, weights, subset, na.action,
          contrasts = NULL, Hess = FALSE, summ = 0, censored = FALSE,
          model = FALSE, ...)
```

Arguments

<code>formula</code>	a formula expression as for regression models, of the form <code>response ~ predictors</code> . The response should be a factor or a matrix with <code>K</code> columns, which will be interpreted as counts for each of <code>K</code> classes. A log-linear model is fitted, with coefficients zero for the first class. An offset can be included: it should be a numeric matrix with <code>K</code> columns if the response is either a matrix with <code>K</code> columns or a factor with <code>K > 2</code> classes, or a numeric vector for a response factor with 2 levels. See the documentation of <code>formula()</code> for other details.
<code>data</code>	an optional data frame in which to interpret the variables occurring in <code>formula</code> .
<code>weights</code>	optional case weights in fitting.
<code>subset</code>	expression saying which subset of the rows of the data should be used in the fit. All observations are included by default.
<code>na.action</code>	a function to filter missing data.
<code>contrasts</code>	a list of contrasts to be used for some or all of the factors appearing as variables in the model formula.
<code>Hess</code>	logical for whether the Hessian (the observed/expected information matrix) should be returned.
<code>summ</code>	integer; if non-zero summarize by deleting duplicate rows and adjust weights. Methods 1 and 2 differ in speed (2 uses C); method 3 also combines rows with the same X and different Y, which changes the baseline for the deviance.
<code>censored</code>	If Y is a matrix with <code>K > 2</code> columns, interpret the entries as one for possible classes, zero for impossible classes, rather than as counts.
<code>model</code>	logical. If true, the model frame is saved as component <code>model</code> of the returned object.
<code>...</code>	additional arguments for <code>nnet</code>

Details

`multinom` calls `nnet`. The variables on the rhs of the formula should be roughly scaled to [0,1] or the fit will be slow or may not converge at all.

Value

A `nnet` object with additional components:

<code>deviance</code>	the residual deviance, compared to the full saturated model (that explains individual observations exactly). Also, minus twice log-likelihood.
<code>edf</code>	the (effective) number of degrees of freedom used by the model
<code>AIC</code>	the AIC for this fit.
<code>Hessian</code>	(if <code>Hess</code> is true).
<code>model</code>	(if <code>model</code> is true).

References

Venables, W. N. and Ripley, B. D. (2002) *Modern Applied Statistics with S*. Fourth edition. Springer.

See Also[nnet](#)**Examples**

```

options(contrasts = c("contr.treatment", "contr.poly"))
library(MASS)
example(birthwt)
(bwt.mu <- multinom(low ~ ., bwt))
## Not run: Call:
multinom(formula = low ~ ., data = bwt)

Coefficients:
(Intercept)          age          lwt raceblack raceother
  0.823477 -0.03724311 -0.01565475  1.192371  0.7406606
  smoke          ptd          ht          ui          ftv1          ftv2+
  0.7555234  1.343648  1.913213  0.6802007 -0.4363238  0.1789888

Residual Deviance: 195.4755
AIC: 217.4755
## End(Not run)

```

nnet

*Fit Neural Networks***Description**

Fit single-hidden-layer neural network, possibly with skip-layer connections.

Usage

```

nnet(x, ...)

## S3 method for class 'formula':
nnet(formula, data, weights, ...,
      subset, na.action, contrasts = NULL)

## Default S3 method:
nnet(x, y, weights, size, Wts, mask,
      linout = FALSE, entropy = FALSE, softmax = FALSE,
      censored = FALSE, skip = FALSE, rang = 0.7, decay = 0,
      maxit = 100, Hess = FALSE, trace = TRUE, MaxNWts = 1000,
      abstol = 1.0e-4, reltol = 1.0e-8, ...)

```

Arguments

`formula` A formula of the form `class ~ x1 + x2 + ...`

`x` matrix or data frame of `x` values for examples.

<code>y</code>	matrix or data frame of target values for examples.
<code>weights</code>	(case) weights for each example – if missing defaults to 1.
<code>size</code>	number of units in the hidden layer. Can be zero if there are skip-layer units.
<code>data</code>	Data frame from which variables specified in <code>formula</code> are preferentially to be taken.
<code>subset</code>	An index vector specifying the cases to be used in the training sample. (NOTE: If given, this argument must be named.)
<code>na.action</code>	A function to specify the action to be taken if NAs are found. The default action is for the procedure to fail. An alternative is <code>na.omit</code> , which leads to rejection of cases with missing values on any required variable. (NOTE: If given, this argument must be named.)
<code>contrasts</code>	a list of contrasts to be used for some or all of the factors appearing as variables in the model formula.
<code>Wts</code>	initial parameter vector. If missing chosen at random.
<code>mask</code>	logical vector indicating which parameters should be optimized (default all).
<code>linout</code>	switch for linear output units. Default logistic output units.
<code>entropy</code>	switch for entropy (= maximum conditional likelihood) fitting. Default by least-squares.
<code>softmax</code>	switch for softmax (log-linear model) and maximum conditional likelihood fitting. <code>linout</code> , <code>entropy</code> , <code>softmax</code> and <code>censored</code> are mutually exclusive.
<code>censored</code>	A variant on <code>softmax</code> , in which non-zero targets mean possible classes. Thus for <code>softmax</code> a row of $(0, 1, 1)$ means one example each of classes 2 and 3, but for <code>censored</code> it means one example whose class is only known to be 2 or 3.
<code>skip</code>	switch to add skip-layer connections from input to output.
<code>rang</code>	Initial random weights on $[-rang, rang]$. Value about 0.5 unless the inputs are large, in which case it should be chosen so that $rang * \max(x)$ is about 1.
<code>decay</code>	parameter for weight decay. Default 0.
<code>maxit</code>	maximum number of iterations. Default 100.
<code>Hess</code>	If true, the Hessian of the measure of fit at the best set of weights found is returned as component <code>Hessian</code> .
<code>trace</code>	switch for tracing optimization. Default TRUE.
<code>MaxNWts</code>	The maximum allowable number of weights. There is no intrinsic limit in the code, but increasing <code>MaxNWts</code> will probably allow fits that are very slow and time-consuming.
<code>abstol</code>	Stop if the fit criterion falls below <code>abstol</code> , indicating an essentially perfect fit.
<code>reltol</code>	Stop if the optimizer is unable to reduce the fit criterion by a factor of at least $1 - reltol$.
<code>...</code>	arguments passed to or from other methods.

Details

If the response in `formula` is a factor, an appropriate classification network is constructed; this has one output and entropy fit if the number of levels is two, and a number of outputs equal to the number of classes and a softmax output stage for more levels. If the response is not a factor, it is passed on unchanged to `nnet.default`.

Optimization is done via the BFGS method of `optim`.

Value

object of class "nnet" or "nnet.formula". Mostly internal structure, but has components

<code>wts</code>	the best set of weights found
<code>value</code>	value of fitting criterion plus weight decay term.
<code>fitted.values</code>	the fitted values for the training data.
<code>residuals</code>	the residuals for the training data.
<code>convergence</code>	1 if the maximum number of iterations was reached, otherwise 0.

References

Ripley, B. D. (1996) *Pattern Recognition and Neural Networks*. Cambridge.

Venables, W. N. and Ripley, B. D. (2002) *Modern Applied Statistics with S*. Fourth edition. Springer.

See Also

[predict.nnet](#), [nnetHess](#)

Examples

```
# use half the iris data
ir <- rbind(iris3[,,1],iris3[,,2],iris3[,,3])
targets <- class.ind( c(rep("s", 50), rep("c", 50), rep("v", 50)) )
samp <- c(sample(1:50,25), sample(51:100,25), sample(101:150,25))
irl <- nnet(ir[samp,], targets[samp,], size = 2, rang = 0.1,
           decay = 5e-4, maxit = 200)
test.cl <- function(true, pred) {
  true <- max.col(true)
  cres <- max.col(pred)
  table(true, cres)
}
test.cl(targets[-samp,], predict(irl, ir[-samp,]))

# or
ird <- data.frame(rbind(iris3[,,1], iris3[,,2], iris3[,,3]),
                 species = factor(c(rep("s",50), rep("c", 50), rep("v", 50))))
ir.nn2 <- nnet(species ~ ., data = ird, subset = samp, size = 2, rang = 0.1,
              decay = 5e-4, maxit = 200)
table(ird$species[-samp], predict(ir.nn2, ird[-samp,], type = "class"))
```

`nnetHess`*Evaluates Hessian for a Neural Network*

Description

Evaluates the Hessian (matrix of second derivatives) of the specified neural network. Normally called via argument `Hess=TRUE` to `nnet` or via `vcov.multinom`.

Usage

```
nnetHess(net, x, y, weights)
```

Arguments

<code>net</code>	object of class <code>nnet</code> as returned by <code>nnet</code> .
<code>x</code>	training data.
<code>y</code>	classes for training data.
<code>weights</code>	the (case) weights used in the <code>nnet</code> fit.

Value

square symmetric matrix of the Hessian evaluated at the weights stored in the net.

References

Ripley, B. D. (1996) *Pattern Recognition and Neural Networks*. Cambridge.

Venables, W. N. and Ripley, B. D. (2002) *Modern Applied Statistics with S*. Fourth edition. Springer.

See Also

[nnet](#), [predict.nnet](#)

Examples

```
# use half the iris data
ir <- rbind(iris3[,1], iris3[,2], iris3[,3])
targets <- matrix(c(rep(c(1,0,0),50), rep(c(0,1,0),50), rep(c(0,0,1),50)),
  150, 3, byrow=TRUE)
samp <- c(sample(1:50,25), sample(51:100,25), sample(101:150,25))
irl <- nnet(ir[samp,], targets[samp,], size=2, rang=0.1, decay=5e-4, maxit=200)
eigen(nnetHess(irl, ir[samp,], targets[samp,]), TRUE)$values
```

`predict.nnet`*Predict New Examples by a Trained Neural Net*

Description

Predict new examples by a trained neural net.

Usage

```
## S3 method for class 'nnet':  
predict(object, newdata, type = c("raw", "class"), ...)
```

Arguments

<code>object</code>	an object of class <code>nnet</code> as returned by <code>nnet</code> .
<code>newdata</code>	matrix or data frame of test examples. A vector is considered to be a row vector comprising a single case.
<code>type</code>	Type of output
<code>...</code>	arguments passed to or from other methods.

Details

This function is a method for the generic function `predict()` for class `"nnet"`. It can be invoked by calling `predict(x)` for an object `x` of the appropriate class, or directly by calling `predict.nnet(x)` regardless of the class of the object.

Value

If `type = "raw"`, the matrix of values returned by the trained network; if `type = "class"`, the corresponding class (which is probably only useful if the net was generated by `nnet.formula`).

References

Ripley, B. D. (1996) *Pattern Recognition and Neural Networks*. Cambridge.

Venables, W. N. and Ripley, B. D. (2002) *Modern Applied Statistics with S*. Fourth edition. Springer.

See Also

[nnet](#), [which.is.max](#)

Examples

```

# use half the iris data
ir <- rbind(iris3[,,1], iris3[,,2], iris3[,,3])
targets <- class.ind( c(rep("s", 50), rep("c", 50), rep("v", 50)) )
samp <- c(sample(1:50,25), sample(51:100,25), sample(101:150,25))
irl <- nnet(ir[samp,], targets[samp,],size = 2, rang = 0.1,
           decay = 5e-4, maxit = 200)
test.cl <- function(true, pred){
  true <- max.col(true)
  cres <- max.col(pred)
  table(true, cres)
}
test.cl(targets[-samp,], predict(irl, ir[-samp,]))

# or
ird <- data.frame(rbind(iris3[,,1], iris3[,,2], iris3[,,3]),
                 species=factor(c(rep("s",50), rep("c", 50), rep("v", 50))))
ir.nn2 <- nnet(species ~ ., data = ird, subset = samp, size = 2, rang = 0.1,
              decay = 5e-4, maxit = 200)
table(ird$species[-samp], predict(ir.nn2, ird[-samp,], type = "class"))

```

which.is.max

Find Maximum Position in Vector

Description

Find the maximum position in a vector, breaking ties at random.

Usage

```
which.is.max(x)
```

Arguments

x a vector

Details

Ties are broken at random.

Value

index of a maximal value.

References

Venables, W. N. and Ripley, B. D. (2002) *Modern Applied Statistics with S*. Fourth edition. Springer.

See Also

[max.col](#), [which.max](#) which takes the first of ties.

Examples

```
## Not run:  
pred <- predict(nnet, test)  
table(true, apply(pred, 1, which.is.max))  
## End(Not run)
```

Index

*Topic **models**

multinom, 2

*Topic **neural**

class.ind, 2

multinom, 2

nnet, 4

nnetHess, 7

predict.nnet, 8

*Topic **utilities**

class.ind, 2

which.is.max, 9

add.net (*nnet*), 4

add1.multinom (*multinom*), 2

anova.multinom (*multinom*), 2

class.ind, 2

coef.multinom (*multinom*), 2

coef.nnet (*nnet*), 4

drop1.multinom (*multinom*), 2

eval.nn (*nnet*), 4

extractAIC.multinom (*multinom*), 2

formula, 3

logLik.multinom (*multinom*), 2

max.col, 10

model.frame.multinom (*multinom*), 2

multinom, 2

nnet, 3, 4, 4, 7, 8

nnetHess, 6, 7

norm.net (*nnet*), 4

optim, 6

predict.multinom (*multinom*), 2

predict.nnet, 6, 7, 8

print.multinom (*multinom*), 2

print.nnet (*nnet*), 4

print.summary.multinom

(*multinom*), 2

print.summary.nnet (*nnet*), 4

summary.multinom (*multinom*), 2

summary.nnet (*nnet*), 4

vcov.multinom (*multinom*), 2

which.is.max, 8, 9

which.max, 10