Package ‘nnet’

January 24, 2021

Priority recommended
Version 7.3-15
Date 2021-01-21
Depends R (>= 3.0.0), stats, utils
Suggests MASS
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
ByteCompile yes
License GPL-2 | GPL-3
URL http://www.stats.ox.ac.uk/pub/MASS4/
NeedsCompilation yes
Author Brian Ripley [aut, cre, cph],
William Venables [cph]
Maintainer Brian Ripley <ripley@stats.ox.ac.uk>
Repository CRAN
Date/Publication 2021-01-24 09:27:44 UTC

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

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

`formula`  
a formula expression as for regression models, of the form `response ~ predictors`.  
The response should be a factor or a matrix with K columns, which will be interpreted as counts for each of K 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 K columns if the response is either a matrix with K columns or a factor with K >= 2 classes, or a numeric vector for a response factor with 2 levels. See the documentation of `formula()` for other details.

`data`  
an optional data frame in which to interpret the variables occurring in `formula`.

`weights`  
optional case weights in fitting.

`subset`  
expression saying which subset of the rows of the data should be used in the fit.  
All observations are included by default.

`na.action`  
a function to filter missing data.

`contrasts`  
a list of contrasts to be used for some or all of the factors appearing as variables in the model formula.

`Hess`  
logical for whether the Hessian (the observed/expected information matrix) should be returned.

`summ`  
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.

`censored`  
If Y is a matrix with K columns, interpret the entries as one for possible classes, zero for impossible classes, rather than as counts.

`model`  
logical. If true, the model frame is saved as component `model` of the returned object.

`...`  
additional arguments for `nnet`

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:

`deviance`  
the residual deviance, compared to the full saturated model (that explains individual observations exactly). Also, minus twice log-likelihood.

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

`AIC`  
the AIC for this fit.

`Hessian`  
(if `Hess` is true).

`model`  
(if `model` is true).

References

nnet

See Also

nnet

Examples

oc <- options(contrasts = c("contr.treatment", "contr.poly"))
library(MASS)
example(birthwt)
(bwt.mu <- multinom(low ~ ., bwt))
options(oc)

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.
y matrix or data frame of target values for examples.
weights (case) weights for each example – if missing defaults to 1.
size number of units in the hidden layer. Can be zero if there are skip-layer units.
data Data frame from which variables specified in formula are preferentially to be taken.
subset An index vector specifying the cases to be used in the training sample. (NOTE: If given, this argument must be named.)
**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`. 

```r
na.action
contrasts
Wts
mask
linout
entropy
softmax
censored
skip
rang
decay
maxit
Hess
trace
MaxNWts
abstol
reltol
...`
Value

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

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

References


See Also

- `predict.nnet`
- `nnetHess`

Examples

```r
# 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))
ir1 <- 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(ir1, ir[-samp,]))
```

```r
# 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
- net: object of class nnet as returned by nnet.
- x: training data.
- y: classes for training data.
- weights: the (case) weights used in the nnet fit.

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

References

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))
ir1 <- nnet(ir[samp,], targets[samp,], size=2, rang=0.1, decay=5e-4, maxit=200)
eigen(nnetHess(ir1, ir[samp,], targets[samp,], TRUE)$values
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

- **object**: an object of class nnet as returned by nnet.
- **newdata**: matrix or data frame of test examples. A vector is considered to be a row vector comprising a single case.
- **type**: Type of output
- **...**: 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


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))
ir1 <- nnet(ir[,], targets[,], 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(ir1, 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

See Also

max.col, which.max which takes the first of ties.

Examples

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

## End(Not run)
Index

* models
  multinom, 2
* multiple logistic
  multinom, 2
* neural
  class.ind, 2
  multinom, 2
  nnet, 4
  nnetHess, 7
  predict.nnet, 8
* utilities
  class.ind, 2
  which.is.max, 8, 9
  which.max, 10

add.net (nnet), 4
add1.multinom (multinom), 2
anova.multinom (multinom), 2
class.ind, 2
coeff.multinom (multinom), 2
coeff.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, 5
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