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.)
`na.action`	A function to specify the action to be taken if `NA`s are found. The default action is for the procedure to fail. An alternative is na.omit, which leads to rejection of cases with missing values on any required variable. (NOTE: If given, this argument must be named.)
`contrasts`	a list of contrasts to be used for some or all of the factors appearing as variables in the model formula.
`Wts`	initial parameter vector. If missing chosen at random.
`mask`	logical vector indicating which parameters should be optimized (default all).
`linout`	switch for linear output units. Default logistic output units.
`entropy`	switch for entropy (= maximum conditional likelihood) fitting. Default by least-squares.
`softmax`	switch for softmax (log-linear model) and maximum conditional likelihood fitting. `linout`, `entropy`, `softmax` and `censored` are mutually exclusive.
`censored`	A variant on `softmax`, in which non-zero targets mean possible classes. Thus for `softmax` a row of `(0, 1, 1)` means one example each of classes 2 and 3, but for `censored` it means one example whose class is only known to be 2 or 3.

skip	switch to add skip-layer connections from input to output.
rang	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.
decay	parameter for weight decay. Default 0.
maxit	maximum number of iterations. Default 100.
Hess	If true, the Hessian of the measure of fit at the best set of weights found is returned as component Hessian.
trace	switch for tracing optimization. Default TRUE.
MaxNWts	The maximum allowable number of weights. There is no intrinsic limit in the code, but increasing MaxNWts will probably allow fits that are very slow and time-consuming.
abstol	Stop if the fit criterion falls below abstol, indicating an essentially perfect fit.
reltol	Stop if the optimizer is unable to reduce the fit criterion by a factor of at least1 - reltol.
...	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

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.

…

```
###################"
data(iris)
Mydata=iris

# Create training and test set
samplesize =floor(0.60 * nrow(Mydata))
set.seed(80)
indice=sample(1:nrow(Mydata),samplesize)
```

```
datatrain = Mydata[ indice, ]
datatest = Mydata[ -indice, ]

### Scale datatrain for fitting a neural network
# A normalization between 0 and 1 is used for neural network

max = apply(datatrain[,1:4] , 2 , max) # apply : Returns a vector of values obtained by applying a function to margins (1=rows, 2=columns) of matrix
min = apply(datatrain[,1:4], 2 , min)
Scdatatrain = scale(datatrain[,1:4], center = min, scale = max-min)

apply(Scdatatrain[,1:4],2,range)

Scdatatrain=as.data.frame(Scdatatrain)
# Add the target
Scdatatrain=cbind(Scdatatrain,datatrain$Species)
names(Scdatatrain)=names(datatrain)

### Fitting a Neural Network
library(nnet)
library(NeuralNetTools)
#NN=neuralnet(Species~.,data=datatrain,hidden=2)
NN=nnet(Species~.,data=Scdatatrain,size=2,entropy=TRUE)
plotnet(NN)

attributes(NN)

# Optimization of hyper_parameters
library(e1071)
opt=tune.nnet(Species~.,data=Scdatatrain,size=1:5,decay=c(0.01,0.1,1,2,3),maxit=100)
plot(opt)
opt$best.parameters

### Prediction
predict(NN,Scdatatrain[,1:4],type="raw")
predict(NN,Scdatatrain[,1:4],type="class")
```