Package ‘monmlp’

December 5, 2017

Type    Package
Title   Multi-Layer Perceptron Neural Network with Optional Monotonicity Constraints
Version 1.1.5
Author  Alex J. Cannon
Maintainer Alex J. Cannon <alex.cannon@canada.ca>
Description Train and make predictions from a multi-layer perceptron neural network with optional partial monotonicity constraints.
License GPL-2
LazyLoad yes
Depends optimx
Repository CRAN
NeedsCompilation no
Date/Publication 2017-12-05 00:05:01 UTC

R topics documented:

monmlp-package .......................................................... 2
gam.style ................................................................. 3
linear ................................................................. 4
linear.prime ......................................................... 5
logistic ......................................................... 5
logistic.prime ..................................................... 6
monmlp.fit .......................................................... 6
monmlp.predict ..................................................... 8
tansig .............................................................. 9
tansig.prime ..................................................... 10

Index 11
Description

The monmlp package implements one and two hidden-layer multi-layer perceptron neural network (MLP) models. An optional monotone constraint, which guarantees monotonically increasing behaviour of model outputs with respect to specified covariates, can be added to the MLP. The resulting monotone MLP (MONMLP) regression model is based on Zhang and Zhang (1999). Early stopping can be combined with bootstrap aggregation to control overfitting. The model reduces to a standard MLP neural network if the monotone constraint is not invoked.

MLP and MONMLP models are fit using the \texttt{monmlp.fit} function. Predictions from a fitted model are made using the \texttt{monmlp.predict} function. The \texttt{gam.style} function can be used to investigate fitted covariate/response relationships.

Details

Package: monmlp
Type: Package
License: GPL-2
LazyLoad: yes

References


Examples

```r
set.seed(123)
x <- as.matrix(seq(-10, 10, length = 100))
y <- logistic(x) + rnorm(100, sd = 0.2)

dev.new()
plot(x, y)
lines(x, logistic(x), lwd = 10, col = "gray")
```
## mlp w/ 2 hidden nodes
w.mlp <- monmlp.fit(x = x, y = y, hidden1 = 2, iter.max = 500)
lines(x, attr(w.mlp, "y.pred"), col = "red", lwd = 3)

## mlp w/ 2 hidden-layers (2 nodes each) and early stopping
w.stp <- monmlp.fit(x = x, y = y, hidden1 = 2, hidden2 = 2,
                      bag = TRUE, iter.max = 500, iter.stopped = 10)
lines(x, attr(w.stp, "y.pred"), col = "orange", lwd = 3)

## MONMLP w/ 2 hidden nodes
w.mon <- monmlp.fit(x = x, y = y, hidden1 = 2, monotone = 1,
                     iter.max = 500)
lines(x, attr(w.mon, "y.pred"), col = "blue", lwd = 3)

---

### Description

GAM-style effects plots provide a graphical means of interpreting fitted covariate/response relationships. From Plate et al. (2000): The effect of the $i$th input variable at a particular input point $\Delta_i \times$ is the change in $f$ resulting from changing $x_1$ to $x_1$ from $b_1$ (the baseline value [...]) while keeping the other inputs constant. The effects are plotted as short line segments, centered at $(x_i, \Delta_i \times)$, where the slope of the segment is given by the partial derivative. Variables that strongly influence the function value have a large total vertical range of effects. Functions without interactions appear as possibly broken straight lines (linear functions) or curves (nonlinear functions). Interactions show up as vertical spread at a particular horizontal location, that is, a vertical scattering of segments. Interactions are present when the effect of a variable depends on the values of other variables.

### Usage

```r
gam.style(x, weights, column, baseline = mean(x[,column]),
          epsilon = 1e-5, seg.len = 0.02, seg.cols = "black",
          plot = TRUE, return.results = FALSE, ...)
```

### Arguments

- `x`: matrix with number of rows equal to the number of samples and number of columns equal to the number of covariate variables.
- `weights`: list returned by `monmlp.fit`.
- `column`: column of `x` for which effects plots should be returned.
- `baseline`: value of `x[,column]` to be used as the baseline for calculation of covariate effects; defaults to `mean(x[,column])`.
- `epsilon`: step-size used in the finite difference calculation of the partial derivatives.
- `seg.len`: length of effects line segments expressed as a fraction of the range of `x[,column]`.
linear

Description
Computes a trivial identity function. Used as the hidden layer transfer function for linear MLP or MONMLP models.

Usage
linear(x)

Examples
set.seed(1)
x <- matrix(runif(350*6), ncol=6)
y <- as.matrix(5*sin(10*x[,1]*x[,2]) + 20*(x[,3]-0.5)^2 - 10*x[,4] + 20*x[,5]*x[,6])
w <- monmlp.fit(x = x, y = y, hidden1 = 4, n.trials = 1, iter.max = 500)
for (i in seq(ncol(x))) gam.style(x, weights = w, column = i)
**linear.prime**

**Arguments**

- *x*  numeric vector.

**See Also**

- linear.prime

---

**linear.prime**  Derivative of the linear function

**Description**

Derivative of the linear function.

**Usage**

linear.prime(x)

**Arguments**

- *x*  numeric vector.

**See Also**

- linear

---

**logistic**  Logistic sigmoid function

**Description**

Computes the logistic sigmoid function. Used as a hidden layer transfer function for nonlinear MLP or MONMLP models.

**Usage**

logistic(x)

**Arguments**

- *x*  numeric vector.

**See Also**

- logistic.prime
logistic.prime  

Derivative of the logistic sigmoid function

Description

Derivative of the logistic sigmoid function.

Usage

logistic.prime(x)

Arguments

x  numeric vector.

See Also

logistic

monmlp.fit  

Fit one or more MLP or MONMLP models

Description

Fit an individual model or ensemble of MLP or MONMLP regression models using optimx optimization routines to minimize a least squares cost function. Optional stopped training and bootstrap aggregation (bagging) can be used to help avoid overfitting.

If invoked, the monotone argument enforces increasing behaviour between specified columns of x and model outputs. In this case, the exp function is applied to the relevant weights following initialization and during optimization; manual adjustment of init.weights may be needed.

Note: x and y are automatically standardized prior to fitting and predictions are automatically rescaled by monmlp.predict. This behaviour can be suppressed for y by the scale.y argument.

Usage

monmlp.fit(x, y, hidden1, hidden2 = 0, iter.max = 5000, n.trials = 1, n.ensemble = 1, bag = FALSE, cases.specified = NULL, iter.stopped = NULL, scale.y = TRUE, Th = tansig, To = linear, Th.prime = tansig.prime, To.prime = linear.prime, monotone = NULL, init.weights = NULL, max.exceptions = 10, silent = FALSE, method = "BFGS", control = list(trace = 0))
monmlp.fit

Arguments

x  
covariate matrix with number of rows equal to the number of samples and number of columns equal to the number of covariates.

y  
response matrix with number of rows equal to the number of samples and number of columns equal to the number of response variables.

hidden1  
number of hidden nodes in the first hidden layer.

hidden2  
number of hidden nodes in the second hidden layer.

iter.max  
maximum number of iterations of the optimization algorithm.

n.trials  
number of repeated trials used to avoid local minima.

n.ensemble  
number of ensemble members to fit.

bag  
logical variable indicating whether or not bootstrap aggregation (bagging) should be used.

cases.specified  
if bag = TRUE, a list that specifies the bootstrapped cases to be used in each ensemble member.

iter.stopped  
if bag = TRUE, specifies the number of stopped training iterations between calculation of the cost function on the out-of-bootstrap cases.

scale.y  
logical determining if columns of the response matrix should be scaled to zero mean and unit variance prior to fitting. Set this to FALSE if using an output layer transfer function that limits the range of predictions.

Th  
hidden layer transfer function.

To  
output layer transfer function.

Th.prime  
derivative of the hidden layer transfer function.

To.prime  
derivative of the output layer transfer function.

monotone  
column indices of covariates for which the monotonicity constraint should hold.

init.weights  
either a vector giving the minimum and maximum allowable values of the random weights, an initial weight vector, or NULL to calculate based on fan-in.

max.exceptions  
maximum number of exceptions of the optimization routine before fitting is terminated with an error.

silent  
logical determining if diagnostic messages should be suppressed.

method  
optimx optimization method.

control  
list of optimx control parameters.

Value

list containing fitted weight matrices with attributes including called values of x, y, Th, To, Th.prime, To.prime, monotone, bag, iter.max, and iter.stopped, along with values of covariate/response column means and standard deviations (x.center, x.scale, y.center, y.scale), out-of-bootstrap cases oob, predicted values y_pred, and, if stopped training is switched on, the iteration iter.best and value of the cost function cost.best that minimized the out-of-bootstrap validation error.
See Also

`monmlp.predict`, `gam.style`

Examples

```r
set.seed(123)
x <- as.matrix(seq(-10, 10, length = 100))
y <- logistic(x) + rnorm(100, sd = 0.2)

dev.new()
plot(x, y)
lines(x, logistic(x), lwd = 10, col = "gray")

## MLP w/ 2 hidden nodes
w.mlp <- monmlp.fit(x = x, y = y, hidden1 = 2, iter.max = 500)
lines(x, attr(w.mlp, "y.pred"), col = "red", lwd = 3)

## MLP w/ 2 hidden nodes and stopped training
w.stp <- monmlp.fit(x = x, y = y, hidden1 = 2, bag = TRUE,
                     iter.max = 500, iter.stopped = 10)
lines(x, attr(w.stp, "y.pred"), col = "orange", lwd = 3)

## MONMLP w/ 2 hidden nodes
w.mon <- monmlp.fit(x = x, y = y, hidden1 = 2, monotone = 1,
                     iter.max = 500)
lines(x, attr(w.mon, "y.pred"), col = "blue", lwd = 3)
```

### monmlp.predict

Make predictions from a fitted MLP or MONMLP model

Description

Make predictions from a fitted model or ensemble of MLP or MONMLP models.

Usage

```r
monmlp.predict(x, weights)
```

Arguments

- `x`: covariate matrix with number of rows equal to the number of samples and number of columns equal to the number of covariates.
- `weights`: list containing weight matrices and other parameters from `monmlp.fit`.

Value

a matrix with number of rows equal to the number of samples and number of columns equal to the number of response variables. If `weights` is from an ensemble of models, the matrix is the ensemble mean and the attribute `ensemble` contains a list with predictions for each ensemble member.
tansig

See Also

monmlp.fit

Examples

```r
set.seed(123)
x <- as.matrix(seq(-10, 10, length = 100))
y <- logistic(x) + rnorm(100, sd = 0.2)

dev.new()
plot(x, y)
lines(x, logistic(x), lwd = 10, col = "gray")

## ensemble of monmlp models w/ 3 hidden nodes
w.mon <- monmlp.fit(x = x, y = y, hidden1 = 3, monotone = 1,
                     n.ensemble = 15, bag = TRUE, iter.max = 500,
                     control = list(trace = 0))
p.mon <- monmlp.predict(x = x, weights = w.mon)

## Plot predictions from ensemble members
matlines(x = x, y = do.call(cbind, attr(p.mon, "ensemble")),
          col = "cyan", lty = 2)

## Plot ensemble mean
lines(x, p.mon, col = "blue", lwd = 3)
```

tansig

Hyperbolic tangent sigmoid function

Description

Computes the hyperbolic tangent sigmoid function. Used as a hidden layer transfer function for nonlinear MLP or MONMLP models.

Usage

tansig(x)

Arguments

x numeric vector.

See Also
tansig.prime
tansig.prime

Derivative of the hyperbolic tangent function

Description

Derivative of the hyperbolic tangent function.

Usage

tansig.prime(x)

Arguments

x numeric vector.

See Also

tansig
Index

*Topic package
  monmlp-package, 2

  gam.style, 2, 3, 8

  linear, 4, 5
  linear.prime, 5, 5
  logistic, 5, 6
  logistic.prime, 5, 6

  monmlp (monmlp-package), 2
  monmlp-package, 2
  monmlp.fit, 2–4, 6, 8, 9
  monmlp.predict, 2, 4, 6, 8, 8

  optimx, 6, 7

  tansig, 9, 10
  tansig.prime, 9, 10