Package ‘monmlp’

July 23, 2015

Type Package
Title Monotone Multi-Layer Perceptron Neural Network
Version 1.1.3
Author Alex J. Cannon
Maintainer Alex J. Cannon <acannon@eos.ubc.ca>
Description Train and make predictions from a multi-layer perceptron neural
    network with partial monotonicity constraints.
License GPL-2
LazyLoad yes
Repository CRAN
NeedsCompilation no
Date/Publication 2015-07-23 07:05:17

R topics documented:

monmlp-package .................................................. 2
gam.style ......................................................... 3
linear ............................................................. 4
linear.prime ..................................................... 5
logistic ........................................................... 5
logistic.prime ................................................... 6
monmlp.cost ..................................................... 6
monmlp.fit ....................................................... 7
monmlp.initialize .............................................. 9
monmlp.nlm ...................................................... 9
monmlp.predict ............................................... 11
monmlp.reshape ............................................... 12
tansig .......................................................... 12
tansig.prime ................................................ 13

Index 14
Description

The monmlp package implements the monotone multi-layer perceptron neural network (MON-MLP) model following Zhang and Zhang (1999). The main feature is the monotone constraint, which guarantees monotonically increasing behaviour of model outputs with respect to specified covariates. The package also features model architectures with one or two hidden layers, analytical calculation of the gradient via backpropagation, optimization using the nlm routine, and optional use of early stopping in conjunction with bootstrap aggregation to control overfitting. The model reduces to a standard multi-layer perceptron neural network if the monotone constraint is not invoked.

MONMLP models are fit using the monmlp.fit function. Predictions from a fitted model are made using the monmlp.predict function. The gam.style function can be used to investigate fitted predictor/predictand relationships. Most other functions are used internally and should not need to be called directly by the user.

Details

<table>
<thead>
<tr>
<th>Package:</th>
<th>monmlp</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type:</td>
<td>Package</td>
</tr>
<tr>
<td>License:</td>
<td>GPL-2</td>
</tr>
<tr>
<td>LazyLoad:</td>
<td>yes</td>
</tr>
</tbody>
</table>

Author(s)

Alex J. Cannon

Maintainer: Alex J. Cannon <acannon@eos.ubc.ca>

References


GAM-style effects plots for interpreting MONMLP models

Description

GAM-style effects plots provide a graphical means of interpreting fitted MONMLP predictor/predictor relationships. From Plate et al. (2000): The effect of the i-th input variable at a particular input point $\Delta_i x$ 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 $(x_i, \Delta_i x)$, 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 predictor 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 predictor 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]`.
- **seg.cols**: colors of effects line segments.
- **plot**: if `TRUE` (the default) then an effects plots for each predictand variable is produced.
- **return.results**: if `TRUE` then values of effects and partial derivatives for each predictand variable are returned.
- **...**: further arguments to be passed to `plot`.

Value

A list with elements:

- **effects**: a matrix of predictor effects.
- **partials**: a matrix of predictor partial derivatives.
References


See Also

monmlp.fit, monmlp.predict

Examples

```r
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)
for (i in 1:ncol(x)) gam.style(x, weights = w, column = i)
```

```
linear

Identity function

Description

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

Usage

linear(x)

Arguments

x numeric vector.

See Also

linear.prime
linear.prime

Description
Derivative of the linear function.

Usage
linear.prime(x)

Arguments
x numeric vector.

See Also
linear

logistic

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

Usage
logistic(x)

Arguments
x numeric vector.

See Also
logistic.prime
logistic.prime \hspace{1em} \textit{Derivative of the logistic sigmoid function}

**Description**

Derivative of the logistic sigmoid function.

**Usage**

\texttt{logistic.prime(x)}

**Arguments**

- \texttt{x} \hspace{1em} numeric vector.

**See Also**

\texttt{logistic}

---

\texttt{monmlp.cost} \hspace{1em} \textit{Least squares cost function for MONMLP fitting}

**Description**

MONMLP mean squared error cost function with analytical calculation of its gradient via back-propagation.

**Usage**

\texttt{monmlp.cost(\texttt{weights}, \texttt{x}, \texttt{y}, \texttt{hidden1}, \texttt{hidden2}, \texttt{Th}, \texttt{To}, \texttt{Th.prime},
\texttt{To.prime}, \texttt{monotone = NULL})}

**Arguments**

- \texttt{weights} \hspace{1em} vector of weights.
- \texttt{x} \hspace{1em} covariate matrix with number of rows equal to the number of samples and number of columns equal to the number of covariates.
- \texttt{y} \hspace{1em} predictand matrix with number of rows equal to the number of samples and number of columns equal to the number of predictands.
- \texttt{hidden1} \hspace{1em} number of hidden nodes in the first hidden layer.
- \texttt{hidden2} \hspace{1em} number of hidden nodes in the second hidden layer.
- \texttt{Th} \hspace{1em} hidden layer transfer function.
- \texttt{To} \hspace{1em} output layer transfer function.
- \texttt{Th.prime} \hspace{1em} derivative of the hidden layer transfer function.
- \texttt{To.prime} \hspace{1em} derivative of the output layer transfer function.
- \texttt{monotone} \hspace{1em} column indices of covariates for which the monotonicity constraint should hold.
monmlp.fit

Value
	numeric value giving the mean squared error with associated gradient attached as an attribute.

See Also

monmlp.cost, monmlp.nlm, monmlp.fit

monmlp.fit

Fit a MONMLP model or an ensemble of MONMLP models

Description

Fit a MONMLP model or an ensemble of MONMLP models using the nlm optimization routine. 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.

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 = c(-0.5, 0.5),
            max.exceptions = 10, silent = FALSE, ...)

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
predictand matrix with number of rows equal to the number of samples and number of columns equal to the number of predictands.

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 nlm 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 predictand 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.

T.o output layer transfer function.

Th.prime derivative of the hidden layer transfer function.

T.o.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 or an initial weight vector.

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

silent logical determining if diagnostic messages should be suppressed.

... additional parameters passed to the nlm optimization routine.

Value

list containing fitted weight matrices with attributes including called values of x, y, Th, T.o, Th.prime, T.o.prime, monotone, bag, iter.max, and iter.stopped, along with values of covariate/predictand 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, monmlp.nlm, monmlp.cost, gam.style

Examples

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

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)
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.stopped = 50)
lines(x, attr(w.stp, "y.pred"), col = "orange", lwd = 3)
```
monmlp.initialize

Initialize a MONMLP weight vector

Description

Uniform random initialization of the weight vector used during fitting of a MONMLP model.

Usage

monmlp.initialize(x, y, hidden1, hidden2, init.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.
- **y**: predictand column matrix with number of rows equal to the number of samples.
- **hidden1**: number of hidden nodes in the first hidden layer.
- **hidden2**: number of hidden nodes in the second hidden layer.
- **init.weights**: vector giving the minimum and maximum allowable values of the random weights.

See Also

monmlp.reshape

monmlp.nlm

Fit MONMLP model via nlm optimization function

Description

Helper function used to fit a MONMLP model via the nlm routine.

Usage

monmlp.nlm(x, y, hidden1, hidden2 = 0, iter.max = 5000, n.trials = 1, Th = tansig, To = linear, Th.prime = tansig.prime, To.prime = linear.prime, monotone = NULL, init.weights = c(-0.5, 0.5), max.exceptions = 10, silent = FALSE, ...)

```r
## MONMLP w/ 2 hidden nodes
w.mon <- monmlp.fit(x = x, y = y, hidden1 = 2, monotone = 1)
lines(x, attr(w.mon, "y.pred"), col = "blue", lwd = 3)
```
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  predictand matrix with number of rows equal to the number of samples and number of columns equal to the number of predictands.

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 nlm optimization algorithm.

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

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 or an initial weight vector.

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

silent  logical determining if diagnostic messages should be suppressed.

...  additional parameters passed to the nlm optimization routine.

Value

a list containing elements

weights  final weight vector

cost  final value of the cost function

code  termination code from nlm

See Also

monmlp.fit
**Description**

Make predictions from a fitted MONMLP model or ensemble of 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 MONMLP 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 predictand 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.

**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)

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)
p.mon <- monmlp.predict(x = x, weights = w.mon)

## Plot predictions from ensemble members
for(i in 1:15)
  lines(x, attr(p.mon, "ensemble")[[i]], col = "cyan")

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

**Reshape a MONMLP weight vector**

**Description**

Reshapes a weight vector used during fitting of a MONMLP model into the appropriate weight matrices.

**Usage**

```r
monmlp.reshape(x, y, weights, hidden1, hidden2)
```

**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`: predictand matrix with number of rows equal to the number of samples and number of columns equal to the number of predictands.
- `weights`: weight vector of length returned by `monmlp.initialize`.
- `hidden1`: number of hidden nodes in the first hidden layer.
- `hidden2`: number of hidden nodes in the second hidden layer.

**See Also**

- `monmlp.initialize`

---

### tansig

**Hyperbolic tangent sigmoid function**

**Description**

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

**Usage**

```r
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, 4, 5
logistic, 5, 6
logistic.prime, 5, 6

monmlp (monmlp-package), 2
monmlp-package, 2
monmlp.cost, 6, 7, 8
monmlp.fit, 2–4, 7, 7, 10, 11
monmlp.initialize, 9, 12
monmlp.nlm, 7, 8, 9
monmlp.predict, 2, 4, 7, 8, 11
monmlp.reshape, 9, 12

nlm, 2, 7–10

tansig, 12, 13
tansig.prime, 12, 13