

# Package ‘neuralGAM’

September 20, 2023

**Type** Package

**Title** Interpretable Neural Network Based on Generalized Additive Models

**Version** 1.1.0

**Maintainer** Ines Ortega-Fernandez <iortega@gradient.org>

**Description** Neural network framework based on Generalized Additive Models from Hastie & Tibshirani (1990, ISBN:9780412343902), which trains a different neural network to estimate the contribution of each feature to the response variable. The networks are trained independently leveraging the local scoring and backfitting algorithms to ensure that the Generalized Additive Model converges and it is additive. The resultant Neural Network is a highly accurate and interpretable deep learning model, which can be used for high-risk AI practices where decision-making should be based on accountable and interpretable algorithms.

**License** MPL-2.0

**BugReports** <https://github.com/inesortega/neuralGAM/issues>

**Encoding** UTF-8

**Imports** tensorflow, keras, ggplot2, magrittr, reticulate, formula.tools, gridExtra

**SystemRequirements** python (>= 3.8), keras, tensorflow

**RoxygenNote** 7.2.3

**Suggests** covr, testthat (>= 3.0.0), fs, withr

**Config/testthat/edition** 3

**URL** <https://inesortega.github.io/neuralGAM/>,  
<https://github.com/inesortega/neuralGAM>

**NeedsCompilation** no

**Author** Ines Ortega-Fernandez [aut, cre, cph]  
(<<https://orcid.org/0000-0002-8041-6860>>),  
Marta Sestelo [aut, cph] (<<https://orcid.org/0000-0003-4284-6509>>)

**Repository** CRAN

**Date/Publication** 2023-09-20 18:20:02 UTC

**R topics documented:**

autoplot.neuralGAM . . . . .	2
install_neuralGAM . . . . .	3
neuralGAM . . . . .	4
plot.neuralGAM . . . . .	6
predict.neuralGAM . . . . .	8
print.neuralGAM . . . . .	10
summary.neuralGAM . . . . .	11

<b>Index</b>	<b>13</b>
--------------	-----------

---

autoplot.neuralGAM	<i>Advanced neuralGAM visualization with ggplot2 library</i>
--------------------	--

---

**Description**

Advanced neuralGAM visualization with ggplot2 library

**Usage**

```
## S3 method for class 'neuralGAM'
autoplot(object, select, xlab = NULL, ylab = NULL, ...)
```

**Arguments**

object	a fitted neuralGAM object as produced by neuralGAM().
select	selects the term to be plotted.
xlab	A title for the x axis.
ylab	A title for the y axis.
...	other graphics parameters to pass on to plotting commands. See details for ggplot2::geom_line options

**Value**

A ggplot object, so you can use common features from the ggplot2 package to manipulate the plot.

**Author(s)**

Ines Ortega-Fernandez, Marta Sestelo.

## Examples

```
## Not run:
n <- 24500

seed <- 42
set.seed(seed)

x1 <- runif(n, -2.5, 2.5)
x2 <- runif(n, -2.5, 2.5)
x3 <- runif(n, -2.5, 2.5)

f1 <- x1 ** 2
f2 <- 2 * x2
f3 <- sin(x3)
f1 <- f1 - mean(f1)
f2 <- f2 - mean(f2)
f3 <- f3 - mean(f3)

eta0 <- 2 + f1 + f2 + f3
epsilon <- rnorm(n, 0.25)
y <- eta0 + epsilon
train <- data.frame(x1, x2, x3, y)

library(neuralGAM)
ngam <- neuralGAM(y ~ s(x1) + x2 + s(x3), data = train,
  num_units = 1024, family = "gaussian",
  activation = "relu",
  learning_rate = 0.001, bf_threshold = 0.001,
  max_iter_backfitting = 10, max_iter_ls = 10,
  seed = seed
)
autoplot(ngam, select="x1")

# add custom title
autoplot(ngam, select="x1") + ggplot2::ggtitle("Main Title")
# add labels
autoplot(ngam, select="x1") + ggplot2::xlab("test") + ggplot2::ylab("my y lab")
# plot multiple terms:
plots <- lapply(c("x1", "x2", "x3"), function(x) autoplot(ngam, select = x))
gridExtra::grid.arrange(grobs = plots, ncol = 3, nrow = 1)

## End(Not run)
```

---

install\_neuralGAM

*Install neuralGAM python requirements*

---

## Description

Creates a conda environment (installing miniconda if required) and set ups the Python requirements to run neuralGAM (Tensorflow and Keras).

Miniconda and related environments are generated in the user's cache directory given by `tools::R_user_dir('neuralGAM', 'cache')` if no previous installation of miniconda is found on the system.

### Usage

```
install_neuralGAM()
```

### Value

No return value.

---

neuralGAM	<i>Fit a neuralGAM model</i>
-----------	------------------------------

---

### Description

Fits a neuralGAM model by building a neural network to attend to each covariate.

### Usage

```
neuralGAM(
  formula,
  data,
  num_units,
  family = "gaussian",
  learning_rate = 0.001,
  activation = "relu",
  kernel_initializer = "glorot_normal",
  kernel_regularizer = NULL,
  bias_regularizer = NULL,
  bias_initializer = "zeros",
  activity_regularizer = NULL,
  loss = "mse",
  w_train = NULL,
  bf_threshold = 0.001,
  ls_threshold = 0.1,
  max_iter_backfitting = 10,
  max_iter_ls = 10,
  seed = NULL,
  verbose = 1,
  ...
)
```

### Arguments

<code>formula</code>	An object of class "formula": a description of the model to be fitted. You can add smooth terms using <code>s()</code> .
----------------------	--

<code>data</code>	A data frame containing the model response variable and covariates required by the formula. Additional terms not present in the formula will be ignored.
<code>num_units</code>	Defines the architecture of each neural network. If a scalar value is provided, a single hidden layer neural network with that number of units is used. If a vector of values is provided, a multi-layer neural network with each element of the vector defining the number of hidden units on each hidden layer is used.
<code>family</code>	This is a family object specifying the distribution and link to use for fitting. By default, it is "gaussian" but also works to "binomial" for logistic regression.
<code>learning_rate</code>	Learning rate for the neural network optimizer.
<code>activation</code>	Activation function of the neural network. Defaults to <code>relu</code>
<code>kernel_initializer</code>	Kernel initializer for the Dense layers. Defaults to Xavier Initializer ( <code>glorot_normal</code> ).
<code>kernel_regularizer</code>	Optional regularizer function applied to the kernel weights matrix.
<code>bias_regularizer</code>	Optional regularizer function applied to the bias vector.
<code>bias_initializer</code>	Optional initializer for the bias vector.
<code>activity_regularizer</code>	Optional regularizer function applied to the output of the layer
<code>loss</code>	Loss function to use during neural network training. Defaults to the mean squared error.
<code>w_train</code>	Optional sample weights
<code>bf_threshold</code>	Convergence criterion of the backfitting algorithm. Defaults to <code>0.001</code>
<code>ls_threshold</code>	Convergence criterion of the local scoring algorithm. Defaults to <code>0.1</code>
<code>max_iter_backfitting</code>	An integer with the maximum number of iterations of the backfitting algorithm. Defaults to <code>10</code> .
<code>max_iter_ls</code>	An integer with the maximum number of iterations of the local scoring Algorithm. Defaults to <code>10</code> .
<code>seed</code>	A positive integer which specifies the random number generator seed for algorithms dependent on randomization.
<code>verbose</code>	Verbosity mode ( <code>0 = silent</code> , <code>1 = print messages</code> ). Defaults to <code>1</code> .
<code>...</code>	Additional parameters for the Adam optimizer (see <code>?keras::optimizer_adam</code> )

### Details

The function builds one neural network to attend to each feature in `x`, using the backfitting and local scoring algorithms to fit a weighted additive model using neural networks as function approximators. The adjustment of the dependent variable and the weights is determined by the distribution of the response `y`, adjusted by the `family` parameter.

### Value

A trained `neuralGAM` object. Use `summary(ngam)` to see details.

**Author(s)**

Ines Ortega-Fernandez, Marta Sestelo.

**References**

Hastie, T., & Tibshirani, R. (1990). Generalized Additive Models. London: Chapman and Hall, 1931(11), 683–741.

**Examples**

```
## Not run:
n <- 24500

seed <- 42
set.seed(seed)

x1 <- runif(n, -2.5, 2.5)
x2 <- runif(n, -2.5, 2.5)
x3 <- runif(n, -2.5, 2.5)

f1 <- x1 ** 2
f2 <- 2 * x2
f3 <- sin(x3)
f1 <- f1 - mean(f1)
f2 <- f2 - mean(f2)
f3 <- f3 - mean(f3)

eta0 <- 2 + f1 + f2 + f3
epsilon <- rnorm(n, 0.25)
y <- eta0 + epsilon
train <- data.frame(x1, x2, x3, y)

library(neuralGAM)
ngam <- neuralGAM(y ~ s(x1) + x2 + s(x3), data = train,
  num_units = 1024, family = "gaussian",
  activation = "relu",
  learning_rate = 0.001, bf_threshold = 0.001,
  max_iter_backfitting = 10, max_iter_ls = 10,
  seed = seed
)

ngam

## End(Not run)
```

**Description**

Visualization of neuralGAM object. Plots the learned partial effects by the neuralGAM object.

**Usage**

```
## S3 method for class 'neuralGAM'  
plot(x, select = NULL, xlab = NULL, ylab = NULL, ...)
```

**Arguments**

<code>x</code>	a fitted neuralGAM object as produced by neuralGAM().
<code>select</code>	allows to plot a set of selected terms. e.g. if you just want to plot the first term, <code>select="X0"</code>
<code>xlab</code>	if supplied, this value will be used as the x label for all plots
<code>ylab</code>	if supplied, this value will be used as the y label for all plots
<code>...</code>	other graphics parameters to pass on to plotting commands.

**Value**

Returns the partial effects plot.

**Author(s)**

Ines Ortega-Fernandez, Marta Sestelo.

**Examples**

```
## Not run:  
  
n <- 24500  
  
seed <- 42  
set.seed(seed)  
  
x1 <- runif(n, -2.5, 2.5)  
x2 <- runif(n, -2.5, 2.5)  
x3 <- runif(n, -2.5, 2.5)  
  
f1 <- x1**2  
f2 <- 2*x2  
f3 <- sin(x3)  
f1 <- f1 - mean(f1)  
f2 <- f2 - mean(f2)  
f3 <- f3 - mean(f3)  
  
eta0 <- 2 + f1 + f2 + f3  
epsilon <- rnorm(n, 0.25)  
y <- eta0 + epsilon  
train <- data.frame(x1, x2, x3, y)
```

```

library(neuralGAM)
ngam <- neuralGAM(y ~ s(x1) + x2 + s(x3), data = train,
  num_units = 1024, family = "gaussian",
  activation = "relu",
  learning_rate = 0.001, bf_threshold = 0.001,
  max_iter_backfitting = 10, max_iter_ls = 10,
  seed = seed
)

plot(ngam)

## End(Not run)

```

---

predict.neuralGAM      *Produces predictions from a fitted neuralGAM object*

---

### Description

Takes a fitted neuralGAM object produced by neuralGAM() and produces predictions given a new set of values for the model covariates.

### Usage

```

## S3 method for class 'neuralGAM'
predict(object, newdata = NULL, type = "link", terms = NULL, verbose = 1, ...)

```

### Arguments

object	a fitted 'neuralGAM' object
newdata	A data frame or list containing the values of covariates at which predictions are required. If not provided, the function returns the predictions for the original training data.
type	when type="link" (default), the linear predictor is returned. When type="terms" each component of the linear predictor is returned separately on each column of a data.frame. When type="response" predictions on the scale of the response are returned.
terms	If type="terms", then only results for the terms named in this list will be returned. If NULL then no terms are excluded (default).
verbose	Verbosity mode (0 = silent, 1 = print messages). Defaults to 1.
...	Other options.

### Value

Predicted values according to type parameter.



**Examples**

```
## Not run:

n <- 24500

seed <- 42
set.seed(seed)

x1 <- runif(n, -2.5, 2.5)
x2 <- runif(n, -2.5, 2.5)
x3 <- runif(n, -2.5, 2.5)

f1 <- x1**2
f2 <- 2*x2
f3 <- sin(x3)
f1 <- f1 - mean(f1)
f2 <- f2 - mean(f2)
f3 <- f3 - mean(f3)

eta0 <- 2 + f1 + f2 + f3
epsilon <- rnorm(n, 0.25)
y <- eta0 + epsilon
train <- data.frame(x1, x2, x3, y)

library(neuralGAM)
ngam <- neuralGAM(y ~ s(x1) + x2 + s(x3), data = train,
  num_units = 1024, family = "gaussian",
  activation = "relu",
  learning_rate = 0.001, bf_threshold = 0.001,
  max_iter_backfitting = 10, max_iter_ls = 10,
  seed = seed
)

n <- 5000
x1 <- runif(n, -2.5, 2.5)
x2 <- runif(n, -2.5, 2.5)
x3 <- runif(n, -2.5, 2.5)
test <- data.frame(x1, x2, x3)

# Obtain linear predictor
eta <- predict(ngam, test, type = "link")

# Obtain predicted response
yhat <- predict(ngam, test, type = "response")

# Obtain each component of the linear predictor
terms <- predict(ngam, test, type = "terms")

# Obtain only certain terms:
terms <- predict(ngam, test, type = "terms", terms = c("x1", "x2"))

## End(Not run)
```

---

print.neuralGAM      *Short neuralGAM summary*

---

**Description**

Default print statement for a neuralGAM object.

**Usage**

```
## S3 method for class 'neuralGAM'  
print(x, ...)
```

**Arguments**

x	neuralGAM object.
...	Other arguments.

**Value**

The printed output of the object:

- Distribution family
- Formula
- Intercept value
- Mean Squared Error (MSE)
- Training sample size

**Author(s)**

Ines Ortega-Fernandez, Marta Sestelo.

**Examples**

```
## Not run:  
  
n <- 24500  
  
seed <- 42  
set.seed(seed)  
  
x1 <- runif(n, -2.5, 2.5)  
x2 <- runif(n, -2.5, 2.5)  
x3 <- runif(n, -2.5, 2.5)  
  
f1 <- x1**2  
f2 <- 2*x2  
f3 <- sin(x3)  
f1 <- f1 - mean(f1)
```

```

f2 <- f2 - mean(f2)
f3 <- f3 - mean(f3)

eta0 <- 2 + f1 + f2 + f3
epsilon <- rnorm(n, 0.25)
y <- eta0 + epsilon
train <- data.frame(x1, x2, x3, y)

library(neuralGAM)
ngam <- neuralGAM(y ~ s(x1) + x2 + s(x3), data = train,
                 num_units = 1024, family = "gaussian",
                 activation = "relu",
                 learning_rate = 0.001, bf_threshold = 0.001,
                 max_iter_backfitting = 10, max_iter_ls = 10,
                 seed = seed
                 )
print(ngam)

## End(Not run)

```

---

```
summary.neuralGAM      neuralGAM summary
```

---

## Description

Summary of a fitted neuralGAM object. Prints the distribution family, model formula, intercept value, sample size, as well as neural network architecture and training history.

## Usage

```
## S3 method for class 'neuralGAM'
summary(object, ...)
```

## Arguments

```
object      neuralGAM object.
...         Other options.
```

## Value

The summary of the object:

- Distribution family
- Formula
- Intercept value
- Mean Squared Error (MSE)
- Training sample size
- Training History
- Model Architecture

**Author(s)**

Ines Ortega-Fernandez, Marta Sestelo.

**Examples**

```
## Not run:

n <- 24500

seed <- 42
set.seed(seed)

x1 <- runif(n, -2.5, 2.5)
x2 <- runif(n, -2.5, 2.5)
x3 <- runif(n, -2.5, 2.5)

f1 <- x1**2
f2 <- 2*x2
f3 <- sin(x3)
f1 <- f1 - mean(f1)
f2 <- f2 - mean(f2)
f3 <- f3 - mean(f3)

eta0 <- 2 + f1 + f2 + f3
epsilon <- rnorm(n, 0.25)
y <- eta0 + epsilon
train <- data.frame(x1, x2, x3, y)

library(neuralGAM)
ngam <- neuralGAM(y ~ s(x1) + x2 + s(x3), data = train,
  num_units = 1024, family = "gaussian",
  activation = "relu",
  learning_rate = 0.001, bf_threshold = 0.001,
  max_iter_backfitting = 10, max_iter_ls = 10,
  seed = seed
)

summary(ngam)

## End(Not run)
```

# Index

`autoplot.neuralGAM`, [2](#)

`install_neuralGAM`, [3](#)

`neuralGAM`, [4](#)

`plot.neuralGAM`, [6](#)

`predict.neuralGAM`, [8](#)

`print.neuralGAM`, [10](#)

`summary.neuralGAM`, [11](#)