Package 'deeptrafo'

November 22, 2022

Title Fitting Deep Conditional Transformation Models

Version 0.1-1

Description Allows for the specification of deep conditional transformation
models (DCTMs) and ordinal neural network transformation models, as
described in Baumann et al (2021) <doi:10.1007/978-3-030-86523-8_1> and
Kook et al (2022) <doi:10.1016/j.patcog.2021.108263>. Extensions such as
autoregressive DCTMs (Ruegamer et al, 2022, <doi:10.48550/arXiv.2110.08248>)
and transformation ensembles (Kook et al, 2022, <doi:10.48550/arXiv.2205.12729>)
are implemented.

Depends R (>= 4.0.0), Formula, tensorflow (>= 2.2.0), keras (>=
2.2.0), tfprobability (>= 0.15), deepregression

Suggests testthat, knitr, ordinal, tram, cotram, covr

Imports mlt, variables, stats, purrr, survival, R6, reticulate

License GPL-3

Encoding UTF-8

RoxygenNote 7.1.2

NeedsCompilation no

Author Lucas Kook [aut, cre],
Philipp Baumann [aut],
David Ruegamer [aut]

Maintainer Lucas Kook <lucasheinrich.kook@gmail.com>

Repository CRAN

Date/Publication 2022-11-22 09:50:02 UTC

R topics documented:

---

atm_init   *Initializes the Processed Additive Predictor for ATMs*

Description

Initializes the Processed Additive Predictor for ATMs

Usage

```
atm_init(atmnr, h1nr)
```

Arguments

atmnr, h1nr   positions of the atm and h1 formula

Value

returns a subnetwork_init function with pre-defined arguments

---

BoxCoxNN   *BoxCox-type neural network transformation models*

Description

BoxCox-type neural network transformation models

Usage

```
BoxCoxNN(
  formula,
  data,
  response_type = get_response_type(data[[all.vars(formula)[1]]]),
  order = get_order(response_type, data[[all.vars(formula)[1]]]),
  addconst_interaction = 0,
  latent_distr = "normal",
  monitor_metrics = NULL,
 trafo_options = trafo_control(order_bsp = order, response_type = response_type),
  ...
)
```

Arguments

formula	Formula specifying the response, interaction, shift terms as `response	interacting ~ shifting`. auto-regressive transformation models (ATMs).
data	Named `list` or `data.frame` which may contain both structured and unstructured data.
response_type	Character; type of response. One of `"continuous"`, `"survival"`, `"count"`, or `"ordered"`. If not supplied manually it is determined by the first entry in `data[[response]]`.
order	Integer; order of the response basis. Default 10 for Bernstein basis or number of levels minus one for ordinal responses.
addconst_interaction	
	Positive constant; a constant added to the additive predictor of the interaction term. If NULL, terms are left unchanged. If 0 and predictors have negative values in their design matrix, the minimum value of all predictors is added to ensure positivity. If > 0, the minimum value plus the `addconst_interaction` is added to each predictor in the interaction term. This ensures a monotone non-decreasing transformation function in the response when using (tensor product) spline bases in the interacting term.
latent_distr	A `tfd_distribution` or character; the base distribution for transformation models. If character, can be `"normal"`, `"logistic"`, `"gumbel"` or `"gompertz"`.
monitor_metrics	
	See [deepregression](#)
trafo_options	Options for transformation models such as the basis function used, see [trafo_control](#) for more details.
...	Additional arguments passed to deepregression

Value

See return statement of [deeptrafo](#)

Examples

```
df <- data.frame(y = rnorm(50), x = rnorm(50))
if (reticulate::py_module_available("tensorflow") &
    reticulate::py_module_available("keras") &
    reticulate::py_module_available("tensorflow_probability")) {
    m <- BoxCoxNN(y ~ x, data = df)
    coef(m)
}
```

ColrNN   *Deep continuous outcome logistic regression*

Description

Deep continuous outcome logistic regression

Usage

```
ColrNN(
  formula,
  data,
  response_type = get_response_type(data[[all.vars(formula)[1]]]),
  order = get_order(response_type, data[[all.vars(formula)[1]]]),
  addconst_interaction = 0,
  latent_distr = "logistic",
  monitor_metrics = NULL,
 trafo_options = trafo_control(order_bsp = order, response_type = response_type),
  ...
)
```

Arguments

formula	Formula specifying the response, interaction, shift terms as `response	interacting ~ shifting`. auto-regressive transformation models (ATMs).
data	Named `list` or `data.frame` which may contain both structured and unstructured data.
response_type	Character; type of response. One of `"continuous"`, `"survival"`, `"count"`, or `"ordered"`. If not supplied manually it is determined by the first entry in `data[[response]]`.
order	Integer; order of the response basis. Default 10 for Bernstein basis or number of levels minus one for ordinal responses.
addconst_interaction	
	Positive constant; a constant added to the additive predictor of the interaction term. If `NULL`, terms are left unchanged. If 0 and predictors have negative values in their design matrix, the minimum value of all predictors is added to ensure positivity. If > 0, the minimum value plus the `addconst_interaction` is

added to each predictor in the interaction term. This ensures a monotone non-decreasing transformation function in the response when using (tensor product) spline bases in the interacting term.

latent_distr   A tfd_distribution or character; the base distribution for transformation models. If character, can be "normal", "logistic", "gumbel" or "gompertz".

monitor_metrics

See [deepregression](#)

trafo_options   Options for transformation models such as the basis function used, see [trafo_control](#) for more details.

...   Additional arguments passed to deepregression

Value

See return statement of [deeptrafo](#)

Examples

```
df <- data.frame(y = rnorm(50), x = rnorm(50))
if (reticulate::py_module_available("tensorflow") &
    reticulate::py_module_available("keras") &
    reticulate::py_module_available("tensorflow_probability")) {
    m <- ColrNN(y ~ x, data = df)
    coef(m)
}
```

---

cotramNN   *Deep distribution-free count regression*

Description

Deep distribution-free count regression

Usage

```
cotramNN(
  formula,
  data,
  response_type = get_response_type(data[[all.vars(formula)[1]]]),
  order = get_order(response_type, data[[all.vars(formula)[1]]]),
  addconst_interaction = 0,
  latent_distr = "logistic",
  monitor_metrics = NULL,
  ...
)
```

Arguments

formula	Formula specifying the response, interaction, shift terms as response | interacting ~ shifting. auto-regressive transformation models (ATMs).
data	Named list or data.frame which may contain both structured and unstructured data.
response_type	Character; type of response. One of "continuous", "survival", "count", or "ordered". If not supplied manually it is determined by the first entry in data[[response]].
order	Integer; order of the response basis. Default 10 for Bernstein basis or number of levels minus one for ordinal responses.
addconst_interaction	
	Positive constant; a constant added to the additive predictor of the interaction term. If NULL, terms are left unchanged. If 0 and predictors have negative values in their design matrix, the minimum value of all predictors is added to ensure positivity. If > 0, the minimum value plus the addconst_interaction is added to each predictor in the interaction term. This ensures a monotone non-decreasing transformation function in the response when using (tensor product) spline bases in the interacting term.
latent_distr	A tfd_distribution or character; the base distribution for transformation models. If character, can be "normal", "logistic", "gumbel" or "gompertz".
monitor_metrics	
	See [deepregression](#)
...	Additional arguments passed to deepregression

Value

See return statement of [deeptrafo](#)

Examples

```
set.seed(1)
df <- data.frame(y = as.integer(abs(1 + rnorm(50, sd = 10))), x = rnorm(50))
if (reticulate::py_module_available("tensorflow") &
    reticulate::py_module_available("keras") &
    reticulate::py_module_available("tensorflow_probability")) {
    m <- cotramNN(y ~ 0 + x, data = df, order = 6)

    optimizer <- optimizer_adam(learning_rate = 0.1, decay = 4e-4)
    m <- cotramNN(y ~ 0 + x, data = df, optimizer = optimizer, order = 6)
    library(cotram)
    fit(m, epochs = 800L, validation_split = 0)
    logLik(mm <- cotram(y ~ x, data = df, method = "logit")); logLik(m)
    coef(mm, with_baseline = TRUE); unlist(c(coef(m, which = "interacting"),
                                             coef(m, which = "shifting")))

}
```

CoxphNN *Cox proportional hazards type neural network transformation models*

Description

Cox proportional hazards type neural network transformation models

Usage

```
CoxphNN(
  formula,
  data,
  response_type = get_response_type(data[[all.vars(formula)[1]]]),
  order = get_order(response_type, data[[all.vars(formula)[1]]]),
  addconst_interaction = 0,
  latent_distr = "gompertz",
  monitor_metrics = NULL,
  trafo_options = trafo_control(order_bsp = order, response_type = response_type),
  ...
)
```

Arguments

formula	Formula specifying the response, interaction, shift terms as `response	interacting ~ shifting`. auto-regressive transformation models (ATMs).
data	Named `list` or `data.frame` which may contain both structured and unstructured data.
response_type	Character; type of response. One of `"continuous"`, `"survival"`, `"count"`, or `"ordered"`. If not supplied manually it is determined by the first entry in `data[[response]]`.
order	Integer; order of the response basis. Default 10 for Bernstein basis or number of levels minus one for ordinal responses.
addconst_interaction	
	Positive constant; a constant added to the additive predictor of the interaction term. If NULL, terms are left unchanged. If 0 and predictors have negative values in their design matrix, the minimum value of all predictors is added to ensure positivity. If > 0, the minimum value plus the `addconst_interaction` is added to each predictor in the interaction term. This ensures a monotone non-decreasing transformation function in the response when using (tensor product) spline bases in the interacting term.
latent_distr	A `tfd_distribution` or character; the base distribution for transformation models. If character, can be `"normal"`, `"logistic"`, `"gumbel"` or `"gompertz"`.
monitor_metrics	
	See deepregression

trafo_options   Options for transformation models such as the basis function used, see trafo_control
                     for more details.

...   Additional arguments passed to deepregression

Value

See return statement of deeptrafo

Examples

```
df <- data.frame(y = rnorm(50), x = rnorm(50))
if (reticulate::py_module_available("tensorflow") &
    reticulate::py_module_available("keras") &
    reticulate::py_module_available("tensorflow_probability")) {
    m <- CoxphNN(y ~ x, data = df)
    coef(m)
}
```

---

dctm   *Deep conditional transformation models with alternative formula interface*

Description

Deep conditional transformation models with alternative formula interface

Usage

```
dctm(
  response,
  intercept = NULL,
  shift = NULL,
  shared = NULL,
  data,
  response_type = get_response_type(data[[all.vars(response)[1]]]),
  order = get_order(response_type, data[[all.vars(response)[1]]]),
  addconst_interaction = 0,
  latent_distr = "logistic",
  monitor_metrics = NULL,
 trafo_options = trafo_control(order_bsp = order, response_type = response_type),
  ...
)
```

Arguments

`response`	Formula for the response; e.g. ~ y
`intercept`	Formula for the intercept function; e.g., ~ x, for which interacting bases with the response will be set up
`shift`	Formula for the shift part of the model; e.g., ~ s(x)
`shared`	Formula for sharing weights between predictors in the intercept and shift part of the model
`data`	Named `list` or `data.frame` which may contain both structured and unstructured data.
`response_type`	Character; type of response. One of `"continuous"`, `"survival"`, `"count"`, or `"ordered"`. If not supplied manually it is determined by the first entry in `data[[response]]`.
`order`	Integer; order of the response basis. Default 10 for Bernstein basis or number of levels minus one for ordinal responses.
`addconst_interaction`	
	Positive constant; a constant added to the additive predictor of the interaction term. If `NULL`, terms are left unchanged. If 0 and predictors have negative values in their design matrix, the minimum value of all predictors is added to ensure positivity. If > 0, the minimum value plus the `addconst_interaction` is added to each predictor in the interaction term. This ensures a monotone non-decreasing transformation function in the response when using (tensor product) spline bases in the interacting term.
`latent_distr`	A `tfd_distribution` or character; the base distribution for transformation models. If character, can be `"normal"`, `"logistic"`, `"gumbel"` or `"gompertz"`.
`monitor_metrics`	
	See [deepregression](#)
`trafo_options`	Options for transformation models such as the basis function used, see [trafo_control](#) for more details.
`...`	Additional arguments passed to deepregression

Value

See return statement of [deeptrafo](#)

Examples

```
df <- data.frame(y = rnorm(50), x = rnorm(50))
if (reticulate::py_module_available("tensorflow") &
    reticulate::py_module_available("keras") &
    reticulate::py_module_available("tensorflow_probability")) {
  m <- dctm(response = ~ y, shift = ~ 0 + x, data = df)
  coef(m)
}
```

---

deeptrafo   *Deep Conditional Transformation Models*

Description

Deep Conditional Transformation Models

Usage

```
deeptrafo(
  formula,
  data,
  response_type = get_response_type(data[[all.vars(fml)[1]]]),
  order = get_order(response_type, data[[all.vars(fml)[1]]]),
  addconst_interaction = 0,
  latent_distr = "logistic",
  monitor_metrics = NULL,
 trafo_options = trafo_control(order_bsp = order, response_type = response_type),
  return_data = FALSE,
  ...
)
```

Arguments

formula	Formula specifying the response, interaction, shift terms as `response	interacting ~ shifting`. auto-regressive transformation models (ATMs).
data	Named `list` or `data.frame` which may contain both structured and unstructured data.
response_type	Character; type of response. One of `"continuous"`, `"survival"`, `"count"`, or `"ordered"`. If not supplied manually it is determined by the first entry in `data[[response]]`.
order	Integer; order of the response basis. Default 10 for Bernstein basis or number of levels minus one for ordinal responses.
addconst_interaction	
	Positive constant; a constant added to the additive predictor of the interaction term. If `NULL`, terms are left unchanged. If 0 and predictors have negative values in their design matrix, the minimum value of all predictors is added to ensure positivity. If > 0, the minimum value plus the `addconst_interaction` is added to each predictor in the interaction term. This ensures a monotone non-decreasing transformation function in the response when using (tensor product) spline bases in the interacting term.
latent_distr	A `tfd_distribution` or character; the base distribution for transformation models. If character, can be `"normal"`, `"logistic"`, `"gumbel"` or `"gompertz"`.
monitor_metrics	
	See [deepregression](#)

trafo_options	Options for transformation models such as the basis function used, see [trafo_control](#) for more details.
return_data	Include full data in the returned object. Defaults to FALSE. Set to TRUE if inteded to use [simulate](#) afterwards.
...	Additional arguments passed to deepregression

Details

deeptrafo is the main function for setting up neural network transformation models and is called by all aliases for the more special cases (see e.g. [ColrNN](#)). The naming convention of the aliases follow the 'tram' package (see e.g. [Colr](#)) and add the suffix "NN" to the function name.

Value

An object of class c("deeptrafo", "deepregression")

Examples

```
data("wine", package = "ordinal")
wine$z <- rnorm(nrow(wine))
wine$x <- rnorm(nrow(wine))

nn <- \(x) x |>
    layer_dense(input_shape = 1L, units = 2L, activation = "relu") |>
    layer_dense(1L)

fml <- rating ~ 0 + temp + contact + s(z, df = 3) + nn(x)
if (reticulate::py_module_available("tensorflow") &
    reticulate::py_module_available("keras") &
    reticulate::py_module_available("tensorflow_probability")) {
m <- deeptrafo(fml, wine, latent_distr = "logistic", monitor_metric = NULL,
    return_data = TRUE, list_of_deep_models = list(nn = nn))

print(m)

    m %>% fit(epochs = 10, batch_size = nrow(wine))
    coef(m, which_param = "interacting")
    coef(m, which_param = "shifting")
    fitted(m)
    predict(m, type = "pdf")
    predict(m, type = "pdf", newdata = wine[, -2])
    logLik(m)
    logLik(m, newdata = wine[1:10, ])
    plot(m)
    mcv <- cv(m, cv_folds = 3)
    ens <- ensemble(m, n_ensemble = 3)
    coef(ens)
}
```

---

ensemble.deeptrafo   *Deep ensembling for neural network transformation models*

Description

Deep ensembling for neural network transformation models

Usage

```
## S3 method for class 'deeptrafo'
ensemble(
  x,
  n_ensemble = 5,
  reinitialize = TRUE,
  mylapply = lapply,
  verbose = FALSE,
  patience = 20,
  plot = TRUE,
  print_members = TRUE,
  stop_if_nan = TRUE,
  save_weights = TRUE,
  callbacks = list(),
  save_fun = NULL,
  ...
)
```

Arguments

x	Object of class "deeptrafo".
n_ensemble	Numeric; number of ensemble members to fit.
reinitialize	Logical; if TRUE (default), model weights are initialized randomly prior to fitting each member. Fixed weights are not affected.
mylapply	Function; lapply function to be used; defaults to lapply
verbose	Logical; whether to print training in each fold.
patience	Integer; number of patience for early stopping.
plot	Logical; whether to plot the resulting losses in each fold.
print_members	Logical; print results for each member.
stop_if_nan	Logical; whether to stop ensembling if NaN values occur
save_weights	Logical; whether to save the ensemble weights.
callbacks	List; callbacks used for fitting.
save_fun	Function; function to be applied to each member to be stored in the final result.
...	Further arguments passed to object$fit_fun.

Value

Ensemble of "deeptrafo" models with list of training histories and fitted weights included in ensemble_results. For details see the return statment in [ensemble](#).

---

from_preds_to_trafo   *Define Predictor of Transformation Model*

Description

Define Predictor of Transformation Model

Usage

```
from_preds_to_trafo(
  atm_toplayer = function(x) layer_dense(x, units = 1L, name = "atm_toplayer"),
  const_ia = NULL
)
```

Arguments

atm_toplayer	Function to be applied on top of the transformed lags.
const_ia	See addconst_interaction in [deeptrafo](#) or [deepregression](#).

Details

Not intended to be used directly by the end user.

Value

A function of list_pred_param returning a list of output tensors that is passed to model_fun of deepregression

---

h1_init   *Initializes the Processed Additive Predictor for TM's Interaction*

Description

Initializes the Processed Additive Predictor for TM's Interaction

Usage

```
h1_init(yterms, h1pred, add_const_positiv = 0)
```

Arguments

yterms   Terms for the response

h1pred   Interacting predictor

add_const_positiv
                    Shift basis for the predictors to be strictly positive

Value

returns a subnetwork_init function with pre-defined arguments

---

LehmanNN   *Lehmann-type neural network transformation models*

Description

Lehmann-type neural network transformation models

Usage

```
LehmanNN(
  formula,
  data,
  response_type = get_response_type(data[[all.vars(formula)[1]]]),
  order = get_order(response_type, data[[all.vars(formula)[1]]]),
  addconst_interaction = 0,
  latent_distr = "gumbel",
  monitor_metrics = NULL,
 trafo_options = trafo_control(order_bsp = order, response_type = response_type),
  ...
)
```

Arguments

formula   Formula specifying the response, interaction, shift terms as response | interacting
                    ~ shifting. auto-regressive transformation models (ATMs).

data   Named list or data.frame which may contain both structured and unstruc-
                    tured data.

response_type   Character; type of response. One of "continuous", "survival", "count",
                    or "ordered". If not supplied manually it is determined by the first entry in
                    data[[response]].

order   Integer; order of the response basis. Default 10 for Bernstein basis or number of
                    levels minus one for ordinal responses.

addconst_interaction

> Positive constant; a constant added to the additive predictor of the interaction term. If NULL, terms are left unchanged. If 0 and predictors have negative values in their design matrix, the minimum value of all predictors is added to ensure positivity. If > 0, the minimum value plus the addconst_interaction is added to each predictor in the interaction term. This ensures a monotone non-decreasing transformation function in the response when using (tensor product) spline bases in the interacting term.

latent_distr   A tfd_distribution or character; the base distribution for transformation models. If character, can be "normal", "logistic", "gumbel" or "gompertz".

monitor_metrics

> See deepregression

trafo_options   Options for transformation models such as the basis function used, see trafo_control for more details.

...   Additional arguments passed to deepregression

Value

See return statement of deeptrafo

Examples

```
df <- data.frame(y = rnorm(50), x = rnorm(50))
if (reticulate::py_module_available("tensorflow") &
    reticulate::py_module_available("keras") &
    reticulate::py_module_available("tensorflow_probability")) {
    m <- LehmanNN(y ~ 0 + x, data = df)
    coef(m)
}
```

LmNN   *Deep normal linear regression*

Description

Deep normal linear regression

Usage

```
LmNN(
  formula,
  data,
  response_type = get_response_type(data[[all.vars(formula)[1]]]),
  order = get_order(response_type, data[[all.vars(formula)[1]]]),
  addconst_interaction = 0,
  latent_distr = "normal",
```

```
    monitor_metrics = NULL,
    trafo_options = trafo_control(order_bsp = 1L, response_type = response_type,
     y_basis_fun = eval_lin, y_basis_fun_lower = .empty_fun(eval_lin), y_basis_fun_prime =
       eval_lin_prime, basis = "shiftscale"),
    ...
  )
```

Arguments

formula	Formula specifying the response, interaction, shift terms as response | interacting ~ shifting. auto-regressive transformation models (ATMs).
data	Named list or data.frame which may contain both structured and unstructured data.
response_type	Character; type of response. One of "continuous", "survival", "count", or "ordered". If not supplied manually it is determined by the first entry in data[[response]].
order	Integer; order of the response basis. Default 10 for Bernstein basis or number of levels minus one for ordinal responses.
addconst_interaction	
	Positive constant; a constant added to the additive predictor of the interaction term. If NULL, terms are left unchanged. If 0 and predictors have negative values in their design matrix, the minimum value of all predictors is added to ensure positivity. If > 0, the minimum value plus the addconst_interaction is added to each predictor in the interaction term. This ensures a monotone non-decreasing transformation function in the response when using (tensor product) spline bases in the interacting term.
latent_distr	A tfd_distribution or character; the base distribution for transformation models. If character, can be "normal", "logistic", "gumbel" or "gompertz".
monitor_metrics	
	See deepregression
trafo_options	Options for transformation models such as the basis function used, see trafo_control for more details.
...	Additional arguments passed to deepregression

Value

See return statement of deeptrafo

Examples

```
set.seed(1)
df <- data.frame(y = 10 + rnorm(50), x = rnorm(50))
if (reticulate::py_module_available("tensorflow") &
    reticulate::py_module_available("keras") &
    reticulate::py_module_available("tensorflow_probability")) {
    m <- LmNN(y ~ 0 + x, data = df)

    optimizer <- optimizer_adam(learning_rate = 0.01, decay = 4e-4)
```

```
m <- LmNN(y ~ 0 + x, data = df, optimizer = optimizer)
library(tram)
fit(m, epochs = 900L, validation_split = 0)
logLik(mm <- Lm(y ~ x, data = df)); logLik(m)
coef(mm, with_baseline = TRUE); unlist(c(coef(m, which = "interacting"),
                                         coef(m, which = "shifting")))

}
```

---

nll   *Generic negative log-likelihood for transformation models*

Description

Generic negative log-likelihood for transformation models

Usage

```
nll(base_distribution)
```

Arguments

base_distribution

> Target distribution, character or `tfd_distribution`. If character, can be either
> "logistic", "normal", "gumbel", "gompertz".

Value

A function for computing the negative log-likelihood of a neural network transformation model
with generic response.

---

ontram   *Ordinal neural network transformation models*

Description

Ordinal neural network transformation models

Usage

```
ontram(
  response,
  intercept = NULL,
  shift = NULL,
  shared = NULL,
  data,
  response_type = "ordered",
  order = get_order(response_type, data[[all.vars(response)[1]]]),
  addconst_interaction = 0,
  latent_distr = "logistic",
  monitor_metrics = NULL,
 trafo_options = trafo_control(order_bsp = order, response_type = response_type),
  ...
)
```

Arguments

response	Formula for the response; e.g., ~ y
intercept	Formula for the intercept function; e.g., ~ x, for which interacting bases with the response will be set up
shift	Formula for the shift part of the model; e.g., ~ s(x)
shared	Formula for sharing weights between predictors in the intercept and shift part of the model
data	Named list or data.frame which may contain both structured and unstructured data.
response_type	Character; type of response. One of "continuous", "survival", "count", or "ordered". If not supplied manually it is determined by the first entry in data[[response]].
order	Integer; order of the response basis. Default 10 for Bernstein basis or number of levels minus one for ordinal responses.
addconst_interaction	
	Positive constant; a constant added to the additive predictor of the interaction term. If NULL, terms are left unchanged. If 0 and predictors have negative values in their design matrix, the minimum value of all predictors is added to ensure positivity. If > 0, the minimum value plus the addconst_interaction is added to each predictor in the interaction term. This ensures a monotone non-decreasing transformation function in the response when using (tensor product) spline bases in the interacting term.
latent_distr	A tfd_distribution or character; the base distribution for transformation models. If character, can be "normal", "logistic", "gumbel" or "gompertz".
monitor_metrics	
	See [deepregression](#)
trafo_options	Options for transformation models such as the basis function used, see [trafo_control](#) for more details.
...	Additional arguments passed to deepregression

Value

See return statement of deeptrafo

References

Kook, L. & Herzog, L., Hothorn, T., Dürr, O., & Sick, B. (2022). Deep and interpretable regression models for ordinal outcomes. Pattern Recognition, 122, 108263. DOI 10.1016/j.patcog.2021.108263

Examples

```
df <- data.frame(y = ordered(sample.int(6, 50, TRUE)), x = rnorm(50))
if (reticulate::py_module_available("tensorflow") &
    reticulate::py_module_available("keras") &
    reticulate::py_module_available("tensorflow_probability")) {
    m <- ontram(response = ~ y, shift = ~ x, data = df)
  coef(m)
}
```

---

plot.deeptrafo   *Generic methods for neural network transformation models*

Description

Generic methods for neural network transformation models

Usage

```
## S3 method for class 'deeptrafo'
plot(
  x,
  which = NULL,
  type = c("smooth", "trafo", "pdf", "cdf"),
  newdata = NULL,
  which_param = c("shifting", "interacting"),
  only_data = FALSE,
  K = 40,
  q = NULL,
  ...
)

## S3 method for class 'deeptrafo'
coef(
  object,
  which_param = c("shifting", "interacting", "autoregressive"),
  type = NULL,
  ...
```

```
)

## S3 method for class 'deeptrafo'
predict(
  object,
  newdata = NULL,
  type = c("trafo", "pdf", "cdf", "interaction", "shift", "terms"),
  batch_size = NULL,
  K = 100,
  q = NULL,
  ...
)

## S3 method for class 'deeptrafo'
fitted(object, newdata = NULL, batch_size = NULL, convert_fun = as.matrix, ...)

## S3 method for class 'deeptrafo'
logLik(
  object,
  newdata = NULL,
  convert_fun = function(x, ...) -sum(x, ...),
  ...
)

## S3 method for class 'deeptrafo'
simulate(object, nsim = 1, seed = NULL, newdata = NULL, ...)

## S3 method for class 'deeptrafo'
print(x, print_model = FALSE, print_coefs = TRUE, with_baseline = FALSE, ...)

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

Arguments

x	Object of class "deeptrafo".
which	Which effect to plot, default selects all smooth effects in the shift term.
type	Either NULL (all types of coefficients are returned), "linear" for linear coefficients or "smooth" for coefficients of; Note that type is currently not used for "interacting".
newdata	Named list or data.frame; optional new data.
which_param	Character; either "shifting", "interacting", or "autoregressive" (only for autoregressive transformation models).
only_data	Logical, if TRUE, only the data for plotting is returned.
K	Integer; grid length for the response to evaluate predictions at, if newdata does not contain the response.

q	Numeric or factor; user-supplied grid of response values to evaluate the predictions. Defaults to NULL. If overwritten, K is ignored.
...	Further arguments supplied to print.deeptrafo
object	Object of class "deeptrafo".
batch_size	Integer; optional, useful if data is too large.
convert_fun	Function; applied to the log-likelihood values of all observations.
nsim	Integer; number of simulations; defaults to 1.
seed	Seed for generating samples; defaults to NULL.
print_model	Logical; print keras model.
print_coefs	Logical; print coefficients.
with_baseline	Logical; print baseline coefs.

Details

If no new data is supplied, predictions are computed on the training data (i.e. in-sample). If new data is supplied without a response, predictions are evaluated on a grid of length K.

Value

Returns vector or matrix of predictions, depending on the supplied type.

Returns matrix of fitted values.

PolrNN	*Deep (proportional odds) logistic regression*

Description

Deep (proportional odds) logistic regression

Usage

```
PolrNN(
  formula,
  data,
  response_type = get_response_type(data[[all.vars(formula)[1]]]),
  order = get_order(response_type, data[[all.vars(formula)[1]]]),
  addconst_interaction = 0,
  latent_distr = "logistic",
  monitor_metrics = NULL,
 trafo_options = trafo_control(order_bsp = order, response_type = response_type),
  ...
)
```

Arguments

formula	Formula specifying the response, interaction, shift terms as response | interacting ~ shifting. auto-regressive transformation models (ATMs).
data	Named list or data.frame which may contain both structured and unstructured data.
response_type	Character; type of response. One of "continuous", "survival", "count", or "ordered". If not supplied manually it is determined by the first entry in data[[response]].
order	Integer; order of the response basis. Default 10 for Bernstein basis or number of levels minus one for ordinal responses.
addconst_interaction	
	Positive constant; a constant added to the additive predictor of the interaction term. If NULL, terms are left unchanged. If 0 and predictors have negative values in their design matrix, the minimum value of all predictors is added to ensure positivity. If > 0, the minimum value plus the addconst_interaction is added to each predictor in the interaction term. This ensures a monotone non-decreasing transformation function in the response when using (tensor product) spline bases in the interacting term.
latent_distr	A tfd_distribution or character; the base distribution for transformation models. If character, can be "normal", "logistic", "gumbel" or "gompertz".
monitor_metrics	
	See [deepregression](#)
trafo_options	Options for transformation models such as the basis function used, see [trafo_control](#) for more details.
...	Additional arguments passed to deepregression

Value

See return statement of [deeptrafo](#)

Examples

```
df <- data.frame(y = ordered(sample.int(5, 50, replace = TRUE)),
    x = rnorm(50))
if (reticulate::py_module_available("tensorflow") &
    reticulate::py_module_available("keras") &
    reticulate::py_module_available("tensorflow_probability")) {
    m <- PolrNN(y ~ x, data = df)
    coef(m)
}
```

| SurvregNN | *Deep parametric survival regression* |

Description

Deep parametric survival regression

Usage

```
SurvregNN(
  formula,
  data,
  response_type = get_response_type(data[[all.vars(formula)[1]]]),
  order = get_order(response_type, data[[all.vars(formula)[1]]]),
  addconst_interaction = 0,
  latent_distr = "gompertz",
  monitor_metrics = NULL,
  trafo_options = NULL,
  ...
)
```

Arguments

formula   Formula specifying the response, interaction, shift terms as `response | interacting ~ shifting`. auto-regressive transformation models (ATMs).

data   Named `list` or `data.frame` which may contain both structured and unstructured data.

response_type   Character; type of response. One of `"continuous"`, `"survival"`, `"count"`, or `"ordered"`. If not supplied manually it is determined by the first entry in `data[[response]]`.

order   Integer; order of the response basis. Default 10 for Bernstein basis or number of levels minus one for ordinal responses.

addconst_interaction

                Positive constant; a constant added to the additive predictor of the interaction term. If NULL, terms are left unchanged. If 0 and predictors have negative values in their design matrix, the minimum value of all predictors is added to ensure positivity. If > 0, the minimum value plus the `addconst_interaction` is added to each predictor in the interaction term. This ensures a monotone non-decreasing transformation function in the response when using (tensor product) spline bases in the interacting term.

latent_distr   A `tfd_distribution` or character; the base distribution for transformation models. If character, can be `"normal"`, `"logistic"`, `"gumbel"` or `"gompertz"`.

monitor_metrics

                See deepregression

trafo_options   Options for transformation models such as the basis function used, see trafo_control
                for more details.

...   Additional arguments passed to deepregression

Value

See return statement of deeptrafo

Examples

```
set.seed(1)
df <- data.frame(y = abs(1 + rnorm(50)), x = rnorm(50))
if (reticulate::py_module_available("tensorflow") &
    reticulate::py_module_available("keras") &
    reticulate::py_module_available("tensorflow_probability")) {
    m <- SurvregNN(y ~ 0 + x, data = df)

    optimizer <- optimizer_adam(learning_rate = 0.01, decay = 4e-4)
    m <- SurvregNN(y ~ 0 + x, data = df, optimizer = optimizer)
    library(tram)
    fit(m, epochs = 500L, validation_split = 0)
    logLik(mm <- Survreg(y ~ x, data = df, dist = "loglogistic")); logLik(m)
    coef(mm, with_baseline = TRUE); unlist(c(coef(m, which = "interacting"),
                                             coef(m, which = "shifting")))

}
```

---

trafo_control   *Options for transformation models*

Description

Options for transformation models

Usage

```
trafo_control(
  order_bsp = 10L,
  support = function(y) range(y),
  y_basis_fun = NULL,
  y_basis_fun_lower = NULL,
  y_basis_fun_prime = NULL,
  penalize_bsp = 0,
  order_bsp_penalty = 2,
  tf_bsps = FALSE,
  response_type = c("continuous", "ordered", "survival", "count"),
 atm_toplayer = function(x) layer_dense(x, units = 1L, name = "atm_toplayer", use_bias
```

```
      = FALSE),
  basis = c("bernstein", "ordered", "shiftscale")
)
```

Arguments

order_bsp
: The order of Bernstein polynomials in case `y_basis_fun` is a Bernstein polynomial defined by `eval_bsp` or (one less than) the number of classes of an ordinal outcome.

support
: A function returning a vector with two elements, namely the support for the basis of y.

y_basis_fun
: Function; basis function for Y

y_basis_fun_lower
: Function; basis function for lower bound of interval censored response

y_basis_fun_prime
: Function; basis function derivative

penalize_bsp
: Scalar value > 0; controls amount of penalization of Bernstein polynomials.

order_bsp_penalty
: Integer; order of Bernstein polynomial penalty. 0 results in a penalty based on integrated squared second order derivatives, values >= 1 in difference penalties.

tf_bsps
: Logical; whether to use a TensorFlow implementation of the Bernstein polynomial functions.

response_type
: Character; type of response can be continuous, ordered, survival, or count.

atm_toplayer
: Function; a function specifying the layer on top of ATM lags.

basis
: Character or function; implemented options are `"bernstein"` (a Bernstein polynomial basis), `"ordered"` (for ordinal responses), or `"shiftscale"` for (log-) linear bases

Value

Returns a named `list` with all options, basis functions, support, and penalties.

Index