Package ‘embed’

November 19, 2018

Version 0.0.2

Title Extra Recipes for Encoding Categorical Predictors

Description Factor predictors can be converted to one or more numeric representations using simple generalized linear models <arXiv:1611.09477> or nonlinear models <arXiv:1604.06737>. All encoding methods are supervised.

Depends R (>= 3.1), recipes (>= 0.1.4)

Imports rstanarm, keras, stats, dplyr, purrr, rlang, utils, generics, tidyr, tibble, lme4, tensorflow

Suggests testthat, knitr, rmarkdown, covr

License GPL-2

Encoding UTF-8

LazyData true

RoxygenNote 6.1.1

ByteCompile true

URL `https://tidymodels.github.io/embed`

BugReports `https://github.com/tidymodels/embed/issues`

NeedsCompilation no

Author Max Kuhn [aut, cre], RStudio [cph]

Maintainer Max Kuhn `<max@rstudio.com>`

Repository CRAN

Date/Publication 2018-11-19 19:00:15 UTC

R topics documented:

`step_embed` *Encoding Factors into Multiple Columns*

Description

`step_embed` creates a *specification* of a recipe step that will convert a nominal (i.e. factor) predictor into a set of scores derived from a tensorflow model via a word-embedding model. `embed_control` is a simple wrapper for setting default options.

Usage

```
step_embed(recipe, ..., role = "predictor", trained = FALSE,
  outcome = NULL, predictors = NULL, num_terms = 2,
  hidden_units = 0, options = embed_control(), mapping = NULL,
  history = NULL, skip = FALSE, id = rand_id("lencode_bayes"))

## S3 method for class 'step_embed'
tidy(x, ...)

embed_control(loss = "mse", metrics = NULL, optimizer = "sgd",
  epochs = 20, validation_split = 0, batch_size = 32, verbose = 0,
  callbacks = NULL)
```

Arguments

`recipe`	A recipe object. The step will be added to the sequence of operations for this recipe.
`...`	One or more selector functions to choose variables. For `step_embed`, this indicates the variables to be encoded into a numeric format. See `recipes::selections()` for more details. For the `tidy` method, these are not currently used.
`role`	For model terms created by this step, what analysis role should they be assigned?. By default, the function assumes that the embedding variables created will be used as predictors in a model.
`trained`	A logical to indicate if the quantities for preprocessing have been estimated.
`outcome`	A call to `vars` to specify which variable is used as the outcome in the neural network. Only numeric and two-level factors are currently supported.
`predictors`	An optional call to `vars` to specify any variables to be added as additional predictors in the neural network. These variables should be numeric and perhaps centered and scaled.
`num_terms`	An integer for the number of resulting variables.
`hidden_units`	An integer for the number of hidden units in a dense ReLu layer between the embedding and output later. Use a value of zero for no intermediate layer (see Details below).
`options`	A list of options for the model fitting process.

`mapping`	A list of tibble results that define the encoding. This is `NULL` until the step is trained by `recipes::prep.recipe()`.
`history`	A tibble with the convergence statistics for each term. This is `NULL` until the step is trained by `recipes::prep.recipe()`.
`skip`	A logical. Should the step be skipped when the recipe is baked by `recipes::bake.recipe()`? While all operations are baked when `recipes::prep.recipe()` is run, some operations may not be able to be conducted on new data (e.g. processing the outcome variable(s)). Care should be taken when using `skip = TRUE` as it may affect the computations for subsequent operations.
`id`	A character string that is unique to this step to identify it.
`x`	A `step_embed` object.
`optimizer, loss, metrics`	Arguments to pass to `keras::compile()`
`epochs, validation_split, batch_size, verbose, callbacks`	Arguments to pass to `keras::fit()`

Details

Factor levels are initially assigned at random to the new variables and these variables are used in a neural network to optimize both the allocation of levels to new columns as well as estimating a model to predict the outcome. See Section 6.1.2 of Francois and Allaire (2018) for more details.

The new variables are mapped to the specific levels seen at the time of model training and an extra instance of the variables are used for new levels of the factor.

One model is created for each call to `step_embed`. All terms given to the step are estimated and encoded in the same model which would also contain predictors give in `predictors` (if any).

When the outcome is numeric, a linear activation function is used in the last layer while softmax is used for factor outcomes (with any number of levels).

For example, the `keras` code for a numeric outcome, one categorical predictor, and no hidden units used here would be

```
keras_model_sequential() \%>\%
layer_embedding(
  input_dim = num_factor_levels_x + 1,
  output_dim = num_terms,
  input_length = 1
) \%>\%
layer_flatten() \%>\%
layer_dense(units = 1, activation = 'linear')
```

If a factor outcome is used and hidden units were requested, the code would be

```
keras_model_sequential() \%>\%
layer_embedding(
  input_dim = num_factor_levels_x + 1,
  output_dim = num_terms,
  input_length = 1
```

```
 ) \%>\%
layer_flatten() \%>\%
layer_dense(units = hidden_units, activation = "relu") \%>\%
layer_dense(units = num_factor_levels_y, activation = 'softmax')
```

Other variables specified by `predictors` are added as an additional dense layer after `layer_flatten` and before the hidden layer.

Also note that it may be difficult to obtain reproducible results using this step due to the nature of Tensorflow (see link in References).

tensorflow models cannot be run in parallel within the same session (via `foreach` or `futures`) or the `parallel` package. If using a recipes with this step with `caret`, avoid parallel processing.

Value

An updated version of `recipe` with the new step added to the sequence of existing steps (if any). For the `tidy` method, a tibble with columns `terms` (the selectors or variables for encoding), `level` (the factor levels), and several columns containing `embed` in the name.

References

Francois C and Allaire JJ (2018) *Deep Learning with R*, Manning

"How can I obtain reproducible results using Keras during development?" https://tinyurl.com/keras-repro

"Concatenate Embeddings for Categorical Variables with Keras" https://flovv.github.io/Embeddings_with_keras_part2/

Examples

```
data(okc)

rec <- recipe(Class ~ age + location, data = okc) %>%
  step_embed(location, outcome = vars(Class),
             options = embed_control(epochs = 10))

# See https://tidymodels.github.io/embed/ for examples
```

`step_lencode_bayes` *Supervised Factor Conversions into Linear Functions using Bayesian Likelihood Encodings*

Description

`step_lencode_bayes` creates a *specification* of a recipe step that will convert a nominal (i.e. factor) predictor into a single set of scores derived from a generalized linear model estimated using Bayesian analysis.

Usage

```
step_lencode_bayes(recipe, ..., role = NA, trained = FALSE,
  outcome = NULL, options = list(seed = sample.int(10^5, 1)),
  verbose = FALSE, mapping = NULL, skip = FALSE,
  id = rand_id("lencode_bayes"))

## S3 method for class 'step_lencode_bayes'
tidy(x, ...)
```

Arguments

`recipe`	A recipe object. The step will be added to the sequence of operations for this recipe.
`...`	One or more selector functions to choose variables. For `step_lencode_bayes`, this indicates the variables to be encoded into a numeric format. See `recipes::selections()` for more details. For the `tidy` method, these are not currently used.
`role`	Not used by this step since no new variables are created.
`trained`	A logical to indicate if the quantities for preprocessing have been estimated.
`outcome`	A call to `vars` to specify which variable is used as the outcome in the generalized linear model. Only numeric and two-level factors are currently supported.
`options`	A list of options to pass to `rstanarm::stan_glmer()`.
`verbose`	A logical to control the default printing by `rstanarm::stan_glmer()`.
`mapping`	A list of tibble results that define the encoding. This is `NULL` until the step is trained by `recipes::prep.recipe()`.
`skip`	A logical. Should the step be skipped when the recipe is baked by `recipes::bake.recipe()`? While all operations are baked when `recipes::prep.recipe()` is run, some operations may not be able to be conducted on new data (e.g. processing the outcome variable(s)). Care should be taken when using `skip = TRUE` as it may affect the computations for subsequent operations
`id`	A character string that is unique to this step to identify it.
`x`	A `step_lencode_bayes` object.

Details

For each factor predictor, a generalized linear model is fit to the outcome and the coefficients are returned as the encoding. These coefficients are on the linear predictor scale so, for factor outcomes, they are in log-odds units. The coefficients are created using a no intercept model and, when two factor outcomes are used, the log-odds reflect the event of interest being the *first* level of the factor.

For novel levels, a slightly timmed average of the coefficients is returned.

A hierarchical generalized linear model is fit using `rstanarm::stan_glmer()` and no intercept via

```
stan_glmer(outcome ~ (1 | predictor), data = data, ...)
```

where the `...` include the `family` argument (automatically set by the step) as well as any arguments given to the `options` argument to the step. Relevant options include `chains`, `iter`, `cores`, and arguments for the priors (see the links in the References below). `prior_intercept` is the argument that has the most effect on the amount of shrinkage.

Value

An updated version of `recipe` with the new step added to the sequence of existing steps (if any). For the `tidy` method, a tibble with columns `terms` (the selectors or variables for encoding), `level` (the factor levels), and `value` (the encodings).

References

Micci-Barreca D (2001) "A preprocessing scheme for high-cardinality categorical attributes in classification and prediction problems," ACM SIGKDD Explorations Newsletter, 3(1), 27-32.

Zumel N and Mount J (2017) "vtreat: a data.frame Processor for Predictive Modeling," arXiv:1611.09477

"Hierarchical Partial Pooling for Repeated Binary Trials" https://tinyurl.com/stan-pooling

"Prior Distributions for ‘rstanarm“ Models" https://tinyurl.com/stan-priors

"Estimating Generalized (Non-)Linear Models with Group-Specific Terms with `rstanarm`" https://tinyurl.com/stan-glm-grouped

Examples

```
library(recipes)
library(dplyr)

data(okc)

reencoded <- recipe(Class ~ age + location, data = okc) %>%
  step_lencode_bayes(location, outcome = vars(Class))

# See https://tidymodels.github.io/embed/ for examples
```

---

step_lencode_glm — *Supervised Factor Conversions into Linear Functions using Likelihood Encodings*

Description

`step_lencode_glm` creates a *specification* of a recipe step that will convert a nominal (i.e. factor) predictor into a single set of scores derived from a generalized linear model.

Usage

```
step_lencode_glm(recipe, ..., role = NA, trained = FALSE,
  outcome = NULL, mapping = NULL, skip = FALSE,
  id = rand_id("lencode_bayes"))

## S3 method for class 'step_lencode_glm'
tidy(x, ...)
```

Arguments

`recipe`	A recipe object. The step will be added to the sequence of operations for this recipe.
`...`	One or more selector functions to choose variables. For `step_lencode_glm`, this indicates the variables to be encoded into a numeric format. See `recipes::selections()` for more details. For the `tidy` method, these are not currently used.
`role`	Not used by this step since no new variables are created.
`trained`	A logical to indicate if the quantities for preprocessing have been estimated.
`outcome`	A call to `vars` to specify which variable is used as the outcome in the generalized linear model. Only numeric and two-level factors are currently supported.
`mapping`	A list of tibble results that define the encoding. This is `NULL` until the step is trained by `recipes::prep.recipe()`.
`skip`	A logical. Should the step be skipped when the recipe is baked by `recipes::bake.recipe()`? While all operations are baked when `recipes::prep.recipe()` is run, some operations may not be able to be conducted on new data (e.g. processing the outcome variable(s)). Care should be taken when using `skip = TRUE` as it may affect the computations for subsequent operations
`id`	A character string that is unique to this step to identify it.
`x`	A `step_lencode_glm` object.

Details

For each factor predictor, a generalized linear model is fit to the outcome and the coefficients are returned as the encoding. These coefficients are on the linear predictor scale so, for factor outcomes, they are in log-odds units. The coefficients are created using a no intercept model and, when two factor outcomes are used, the log-odds reflect the event of interest being the *first* level of the factor.

For novel levels, a slightly timmed average of the coefficients is returned.

Value

An updated version of `recipe` with the new step added to the sequence of existing steps (if any). For the `tidy` method, a tibble with columns `terms` (the selectors or variables for encoding), `level` (the factor levels), and `value` (the encodings).

References

Micci-Barreca D (2001) "A preprocessing scheme for high-cardinality categorical attributes in classification and prediction problems," ACM SIGKDD Explorations Newsletter, 3(1), 27-32.

Zumel N and Mount J (2017) "vtreat: a data.frame Processor for Predictive Modeling," arXiv:1611.09477

Examples

```
library(recipes)
library(dplyr)

data(okc)
```

```
glm_est <- recipe(Class ~ age + location, data = okc) %>%
  step_lencode_glm(location, outcome = vars(Class))

# See https://tidymodels.github.io/embed/ for examples
```

---

step_lencode_mixed — *Supervised Factor Conversions into Linear Functions using Bayesian Likelihood Encodings*

Description

step_lencode_mixed creates a *specification* of a recipe step that will convert a nominal (i.e. factor) predictor into a single set of scores derived from a generalized linear mixed model.

Usage

```
step_lencode_mixed(recipe, ..., role = NA, trained = FALSE,
  outcome = NULL, options = list(verbose = 0), mapping = NULL,
  skip = FALSE, id = rand_id("lencode_bayes"))

## S3 method for class 'step_lencode_mixed'
tidy(x, ...)
```

Arguments

recipe	A recipe object. The step will be added to the sequence of operations for this recipe.
...	One or more selector functions to choose variables. For step_lencode_mixed, this indicates the variables to be encoded into a numeric format. See recipes::selections() for more details. For the tidy method, these are not currently used.
role	Not used by this step since no new variables are created.
trained	A logical to indicate if the quantities for preprocessing have been estimated.
outcome	A call to vars to specify which variable is used as the outcome in the generalized linear model. Only numeric and two-level factors are currently supported.
options	A list of options to pass to lme4::lmer() or lme4::glmer().
mapping	A list of tibble results that define the encoding. This is NULL until the step is trained by recipes::prep.recipe().
skip	A logical. Should the step be skipped when the recipe is baked by recipes::bake.recipe()? While all operations are baked when recipes::prep.recipe() is run, some operations may not be able to be conducted on new data (e.g. processing the outcome variable(s)). Care should be taken when using skip = TRUE as it may affect the computations for subsequent operations
id	A character string that is unique to this step to identify it.
x	A step_lencode_mixed object.

Details

For each factor predictor, a generalized linear model is fit to the outcome and the coefficients are returned as the encoding. These coefficients are on the linear predictor scale so, for factor outcomes, they are in log-odds units. The coefficients are created using a no intercept model and, when two factor outcomes are used, the log-odds reflect the event of interest being the *first* level of the factor.

For novel levels, a slightly timmed average of the coefficients is returned.

A hierarchical generalized linear model is fit using `lme4::lmer()` or `lme4::glmer()`, depending on the nature of the outcome, and no intercept via

```
  lmer(outcome ~ 1 + (1 | predictor), data = data, ...)
```

where the `...` include the `family` argument (automatically set by the step) as well as any arguments given to the `options` argument to the step. Relevant options include `control` and others.

Value

An updated version of `recipe` with the new step added to the sequence of existing steps (if any). For the `tidy` method, a tibble with columns `terms` (the selectors or variables for encoding), `level` (the factor levels), and `value` (the encodings).

References

Micci-Barreca D (2001) "A preprocessing scheme for high-cardinality categorical attributes in classification and prediction problems," ACM SIGKDD Explorations Newsletter, 3(1), 27-32.

Zumel N and Mount J (2017) "vtreat: a data.frame Processor for Predictive Modeling," arXiv:1611.09477

Examples

```
library(recipes)
library(dplyr)

data(okc)

reencoded <- recipe(Class ~ age + location, data = okc) %>%
  step_lencode_mixed(location, outcome = vars(Class))

# See https://tidymodels.github.io/embed/ for examples
```

Index