Package 'arf'

February 6, 2023

Title Adversarial Random Forests

Version 0.1.3

Date 2023-02-06

Maintainer Marvin N. Wright <cran@wrig.de>

Description Adversarial random forests (ARFs) recursively partition data into fully factorized leaves, where features are jointly independent. The procedure is iterative, with alternating rounds of generation and discrimination. Data becomes increasingly realistic at each round, until original and synthetic samples can no longer be reliably distinguished. This is useful for several unsupervised learning tasks, such as density estimation and data synthesis. Methods for both are implemented in this package. ARFs naturally handle unstructured data with mixed continuous and categorical covariates. They inherit many of the benefits of random forests, including speed, flexibility, and solid performance with default parameters. For details, see Watson et al. (2022) <arXiv:2205.09435>.

License GPL (>= 3)

URL https://github.com/bips-hb/arf, https://bips-hb.github.io/arf/

BugReports https://github.com/bips-hb/arf/issues

Imports data.table, ranger, foreach, truncnorm, matrixStats

Encoding UTF-8

RoxygenNote 7.2.3

Suggests ggplot2, doParallel, mlbench, knitr, rmarkdown, testthat (>= 3.0.0)

Config/testthat/edition 3

VignetteBuilder knitr

NeedsCompilation no

Author Marvin N. Wright [aut, cre] (<https://orcid.org/0000-0002-8542-6291>), David S. Watson [aut] (<https://orcid.org/0000-0001-9632-2159>)

Repository CRAN

Date/Publication 2023-02-06 20:22:32 UTC

R topics documented:

`adversarial_rf` *Adversarial Random Forests*

Description

Implements an adversarial random forest to learn independence-inducing splits.

Usage

```
adversarial_rf(
  x,
  num_trees = 10L,
  min_node_size = 2L,
  delta = 0,
  max_iters = 10L,
  early_stop = TRUE,
  verbose = TRUE,
  parallel = TRUE,
  ...
)
```

Arguments

`x`	Input data. Integer variables are recoded as ordered factors with a warning. See Details.
`num_trees`	Number of trees to grow in each forest. The default works well for most generative modeling tasks, but should be increased for likelihood estimation. See Details.
`min_node_size`	Minimal number of real data samples in leaf nodes.
`delta`	Tolerance parameter. Algorithm converges when OOB accuracy is < 0.5 + `delta`.
`max_iters`	Maximum iterations for the adversarial loop.
`early_stop`	Terminate loop if performance fails to improve from one round to the next?
`verbose`	Print discriminator accuracy after each round?
`parallel`	Compute in parallel? Must register backend beforehand, e.g. via `doParallel`.
`...`	Extra parameters to be passed to `ranger`.

Details

The adversarial random forest (ARF) algorithm partitions data into fully factorized leaves where features are jointly independent. ARFs are trained iteratively, with alternating rounds of generation and discrimination. In the first instance, synthetic data is generated via independent bootstraps of each feature, and a RF classifier is trained to distinguish between real and synthetic samples. In subsequent rounds, synthetic data is generated separately in each leaf, using splits from the previous forest. This creates increasingly realistic data that satisfies local independence by construction. The algorithm converges when a RF cannot reliably distinguish between the two classes, i.e. when OOB accuracy falls below 0.5 + `delta`.

ARFs are useful for several unsupervised learning tasks, such as density estimation (see `forde`) and data synthesis (see `forge`). For the former, we recommend increasing the number of trees for improved performance (typically on the order of 100-1000 depending on sample size).

Integer variables are treated as ordered factors by default. If the ARF is passed to `forde`, the estimated distribution for these variables will only have support on observed factor levels (i.e., the output will be a pmf, not a pdf). To override this behavior and assign nonzero density to intermediate values, explicitly recode the features as numeric.

Note: convergence is not guaranteed in finite samples. The `max_iter` argument sets an upper bound on the number of training rounds. Similar results may be attained by increasing `delta`. Even a single round can often give good performance, but data with strong or complex dependencies may require more iterations. With the default `early_stop = TRUE`, the adversarial loop terminates if performance does not improve from one round to the next, in which case further training may be pointless.

Value

A random forest object of class `ranger`.

References

Watson, D., Blesch, K., Kapar, J., & Wright, M. (2022). Adversarial random forests for density estimation and generative modeling. *arXiv* preprint, 2205.09435.

See Also

`forde`, `forge`

Examples

```
arf <- adversarial_rf(iris)
```

col_rename *Adaptive column renaming*

Description

This function renames columns in case the input data.frame includes any colnames required by internal functions (e.g., "y").

Usage

```
col_rename(df, old_name)
```

Arguments

df	Input data.frame.
old_name	Name of column to be renamed.

forde *Forests for Density Estimation*

Description

Uses a pre-trained ARF model to estimate leaf and distribution parameters.

Usage

```
forde(
  arf,
  x,
  oob = FALSE,
  family = "truncnorm",
  alpha = 0,
  epsilon = 0,
  parallel = TRUE
)
```

Arguments

arf	Pre-trained adversarial_rf. Alternatively, any object of class ranger.
x	Training data for estimating parameters.
oob	Only use out-of-bag samples for parameter estimation? If TRUE, x must be the same dataset used to train arf.
family	Distribution to use for density estimation of continuous features. Current options include truncated normal (the default family = "truncnorm") and uniform (family = "unif"). See Details.

alpha	Optional pseudocount for Laplace smoothing of categorical features. This avoids zero-mass points when test data fall outside the support of training data. Effectively parametrizes a flat Dirichlet prior on multinomial likelihoods.
epsilon	Optional slack parameter on empirical bounds when family = "unif". This avoids zero-density points when test data fall outside the support of training data. The gap between lower and upper bounds is expanded by a factor of 1 + epsilon.
parallel	Compute in parallel? Must register backend beforehand, e.g. via doParallel.

Details

forde extracts leaf parameters from a pretrained forest and learns distribution parameters for data within each leaf. The former includes coverage (proportion of data falling into the leaf) and split criteria. The latter includes proportions for categorical features and mean/variance for continuous features. These values are stored in a data.table, which can be used as input to various other functions.

Currently, forde only provides support for a limited number of distributional families: truncated normal or uniform for continuous data, and multinomial for discrete data. Future releases will accommodate a larger set of options.

Though forde was designed to take an adversarial random forest as input, the function's first argument can in principle be any object of class ranger. This allows users to test performance with alternative pipelines (e.g., with supervised forest input). There is also no requirement that x be the data used to fit arf, unless oob = TRUE. In fact, using another dataset here may protect against overfitting. This connects with Wager & Athey's (2018) notion of "honest trees".

Value

A list with 5 elements: (1) parameters for continuous data; (2) parameters for discrete data; (3) leaf indices and coverage; (4) metadata on variables; and (5) the data input class. This list is used for estimating likelihoods with lik and generating data with forge.

References

Watson, D., Blesch, K., Kapar, J., & Wright, M. (2022). Adversarial random forests for density estimation and generative modeling. *arXiv* preprint, 2205.09435.

Wager, S. & Athey, S. (2018). Estimation and inference of heterogeneous treatment effects using random forests. *J. Am. Stat. Assoc.*, *113*(523): 1228-1242.

See Also

adversarial_rf, forge, lik

Examples

```
arf <- adversarial_rf(iris)
psi <- forde(arf, iris)
head(psi)
```

forge *Forests for Generative Modeling*

Description

Uses pre-trained FORDE model to simulate synthetic data.

Usage

```
forge(params, n_synth, parallel = TRUE)
```

Arguments

params	Parameters learned via forde.
n_synth	Number of synthetic samples to generate.
parallel	Compute in parallel? Must register backend beforehand, e.g. via doParallel.

Details

forge simulates a synthetic dataset of n_synth samples. First, leaves are sampled in proportion to their coverage. Then, each feature is sampled independently within each leaf according to the probability mass or density function learned by forde. This will create realistic data so long as the adversarial RF used in the previous step satisfies the local independence criterion. See Watson et al. (2022).

Value

A dataset of n_synth synthetic samples.

References

Watson, D., Blesch, K., Kapar, J., & Wright, M. (2022). Adversarial random forests for density estimation and generative modeling. *arXiv* preprint, 2205.09435.

See Also

adversarial_rf, forde

Examples

```
arf <- adversarial_rf(iris)
psi <- forde(arf, iris)
x_synth <- forge(psi, n_synth = 100)
```

lik *Likelihood Estimation*

Description

Compute the density of input data.

Usage

```
lik(arf, params, x, oob = FALSE, log = TRUE, batch = NULL, parallel = TRUE)
```

Arguments

arf	Pre-trained adversarial_rf. Alternatively, any object of class ranger.
params	Parameters learned via forde.
x	Input data. Densities will be computed for each sample.
oob	Only use out-of-bag leaves for likelihood estimation? If TRUE, x must be the same dataset used to train arf.
log	Return likelihoods on log scale? Recommended to prevent underflow.
batch	Batch size. The default is to compute densities for all of x in one round, which is always the fastest option if memory allows. However, with large samples or many trees, it can be more memory efficient to split the data into batches. This has no impact on results.
parallel	Compute in parallel? Must register backend beforehand, e.g. via doParallel.

Details

This function computes the density of input data according to a FORDE model using a pre-trained ARF. Each sample's likelihood is a weighted average of its likelihood in all leaves whose split criteria it satisfies. Intra-leaf densities are fully factorized, since ARFs satisfy the local independence criterion by construction. See Watson et al. (2022).

Value

A vector of likelihoods, optionally on the log scale.

References

Watson, D., Blesch, K., Kapar, J., & Wright, M. (2022). Adversarial random forests for density estimation and generative modeling. *arXiv* preprint, 2205.09435.

See Also

adversarial_rf, forge

Examples

```
# Estimate average log-likelihood
arf <- adversarial_rf(iris)
psi <- forde(arf, iris)
ll <- lik(arf, psi, iris, log = TRUE)
mean(ll)
```

Index