Package ‘nsga3’

July 22, 2025

Type Package

Title An Implementation of Non-Dominated Sorting Genetic Algorithm III for Feature Selection

Version 0.0.3

Author Artem Shramko

Maintainer Artem Shramko <`art.shramko@gmail.com`>

Description An adaptation of Non-dominated Sorting Genetic Algorithm III for multi objective feature selection tasks. Non-dominated Sorting Genetic Algorithm III is a genetic algorithm that solves multiple optimization problems simultaneously by applying a non-dominated sorting technique. It uses a reference points based selection operator to explore solution space and preserve diversity. See the original paper by K. Deb and H. Jain (2014) <DOI:10.1109/TEVC.2013.2281534> for a detailed description.

Depends R (>= 2.10.0)

License GPL-3

Encoding UTF-8

LazyData true

Imports mlr, parallelMap, rPref, xgboost

Suggests testthat, knitr, rmarkdown

RoxygenNote 6.1.1

NeedsCompilation no

Repository CRAN

Date/Publication 2019-02-18 10:00:15 UTC

Contents

`german_credit` *The UCI "German Credit Data" Dataset*

Description

This dataset classifies people described by a set of attributes as good or bad credit risks.

Usage

```
german_credit
```

Format

A data frame with 1000 rows and 20 variables:

account_status Factor. Status of existing checking account

duration Numeric. Duration in month

purpose Factor. Purpose

credit_history Factor. Credit history

amount Numeric. Credit amount

savings Numeric. Savings account/bonds

employment Factor Present employment since

installment_rate Integer. Installment rate in percentage of disposable income

status_gender Factor. Personal status and gender

guarantors Factor. Other debtors / guarantors

resident_since Numeric. Present residence since

property Factor. Property

age Numeric. Age in years

other_plans Factor. Other installment plans

housing Factor. Housing

num_credits Numeric. num_credits

job Factor. Job

people_maintenance Numeric. Number of people being liable to provide maintenance for

phone Factor. Telephone

foreign Factor. foreign worker

BAD Factor. Target feature. 1 = BAD

...

Source

Professor Dr. Hofmann, Hans (1994). UCI Machine Learning Repository `https://archive.ics.uci.edu/ml/datasets/statlog+(german+credit+data)`. Hamburg, Germany: Universitaet Hamburg, Institut fuer Statistik und "Oekonometrie.

`nsga3fs` *NSGA III for Multi-Objective Feature Selection*

Description

An adaptation of Non-dominated Sorting Genetic Algorithm III for multi objective feature selection tasks. Non-dominated Sorting Genetic Algorithm III is a genetic algorithm that solves multiple optimization problems simultaneously by applying a non-dominated sorting technique. It uses a reference points based selection operator to explore solution space and preserve diversity. See the paper by K. Deb and H. Jain (2014) <DOI:10.1109/TEVC.2013.2281534> for a detailed description of the algorithm.

Usage

```
nsga3fs(df, target, obj_list, obj_names, pareto, pop_size, max_gen, model,
  resampling = FALSE, num_features = TRUE, mutation_rate = 0.1,
  threshold = 0.5, feature_cost = FALSE,
  r_measures = list(mlr::mmce), cpus = 1)
```

Arguments

`df`	An original dataset.
`target`	Name of a column (a string), which contains classification target variable.
`obj_list`	A List of objective functions to be optimizied. Must be a list of objects of type closure.
`obj_names`	A Vector of the names of objective functions. Must match the atguments passed to pareto.
`pareto`	A Pareto criteria for non-dominated sorting. Should be passed in a form: $low(objective_1)*high(objective_2)$ See description of `low` for more details.
`pop_size`	Size of the population.
`max_gen`	Number of generations.
`model`	A `makeLearner` object. A model to be used for classification task.
`resampling`	A `makeResampleDesc` object.
`num_features`	TRUE if algorithm should minimize number of features as one of objectives. You must pass a respective object to pareto as well as obj_names.
`mutation_rate`	Probability of switching the value of a certain gene to its opposite. Default value 0.1.
`threshold`	Threshold applied during majority vote when calculating final output. Default value 0.5.
`feature_cost`	A vector of feacure costs. Must be equal ncol(df)-1. You must pass a respective object to pareto as well as obj_names.
`r_measures`	A list of performance metrics for `makeResampleDesc` task. Default "mmce"
`cpus`	Number of sockets to be used for parallelisation. Default value is 1.

Value

A list with the final Pareto Front:

Raw A list containing two items:

1. A list with final Pareto Front individuals
2. A data.frame containing respective fitness values

Per individual Same content, structured per individual

Majority vote Pareto Front majority vote for dataset features

Stat Runtime, dataset details, model

Note

Be cautious with setting the size of population and maximum generations. Since NSGA III is a wrapper feature selection method, a model has to be retrained N*number of generation +1 times, which may involve high computational costs. A 100 x 100 setting should be enough.

This adaptation of NSGA III algorithm for Multi Objective Feature Selection is currently available only for classification tasks.

#'As any other Genetic Algorithm (GA), NSGA III includes following steps:

1. An initial population Pt of a size N is created
2. A model is trained on each individual (subset) and fitness values are assigned
3. An offsping population of a size N is created by crossover and mutation operators
4. The offspring population is combined with its parent population
5. A combined population of a size 2N is split into Pareto Fronts using non-dominated sorting technique
6. A next generation's population Pt+1 of size N is selected from the top Pareto Fronts with help of elitism based selection operator

The loop is repeated until the final generation is reached

Each generation is populated by individuals representing different subsets. Each individual is represented as a binary vector, where each gene represents a feature in the original dataset.

References

K. Deb, H. Jain (2014) <DOI:10.1109/TEVC.2013.2281534>

Examples

```
xgb_learner <- mlr::makeLearner("classif.xgboost", predict.type = "prob",
                             par.vals = list(
                             objective = "binary:logistic",
                             eval_metric = "error",nrounds = 2))

rsmp <- mlr::makeResampleDesc("CV", iters = 2)
measures <- list(mlr::mmce)
```

```
f_auc <- function(pred){auc <- mlr::performance(pred, auc)
                        return(as.numeric(auc))}
objective <- c(f_auc)
o_names <- c("AUC", "nf")
par <- rPref::high(AUC)*rPref::low(nf)

nsga3fs(df = german_credit, target = "BAD", obj_list = objective,
        obj_names = o_names, pareto = par, pop_size = 1, max_gen = 1,
        model = xgb_learner, resampling = rsmp,
        num_features = TRUE, r_measures = measures, cpus = 2)
```

Index