CallEshotgun-Interface in R

Lennard Reese
lennard.reese@smail.th-koeln.de

July 21, 2022

Abstract

The CallEshotgun-package specializes in providing an interface to the novel e-shotgun infill criterion for Bayesian Optimization. The algorithm was initially developed in Python and not directly accessible to other programming languages such as R. Therefore, the package enables the usage of the new algorithm within the R-ecosystem. This paper discusses and motivates the development of the interface, provides an insight into the included functions and exemplary use of the e-shotgun algorithm.

1 Introduction

The R-package aims to enable the use of the novel e-shotgun algorithm for Bayesian optimization inside the R-ecosystem provided through an open, flexible and easy to use interface. The e-shotgun algorithm was initially developed and published in Python by George de Ath Et al. [DEFR20] as a GitHub-Repository. Further the project contains multiple different acquisition functions, data sets and test problems. In the current implementation, the usage of the e-shotgun is limited to an array of preselected test problems. To apply the e-shotgun algorithm to a different test problem or real-world application it has to be added to the test problem folder of the project in the form of a new file or as an addition to an existing file including a modification in the import management accordingly to the conducted changes, in the folder structure.

The approach used for the interface development is twofold, first to extract the Python implementation of the e-shotgun from the original project disregarding other acquisition functions, test problems and data. Therefore, decouple the test problems from the e-shotgun implementation, thereby enabling the user to change between optimization problems without a modification of the underlying Python source code. Secondly, developing the R side of the interface that allows for the execution of the algorithm and further handling the data exchange between R and Python.

After the introduction started with a short discussion of the current algorithm implementation, including the tools used for development. The method section introduces the details of the functions included in the interface. Followed by the explanation and conduction of the test strategy that should ensure the prevention of errors at run time and distorted results, caused by incorrect parameters. The example section showcases how valid and invalid function calls could look like as well as possible error messages that can occur. The paper concludes with a discussion of the results.

2 Related

With an increased need for optimization across various disciplines of engineering and related fields the complexity of the simulation programs increases as well, the use of such programs gets ever more expensive and resource-intensive. To reduce the needed search time it is possible to use surrogate models that approximate the expensive real-world function through observed input-output data. Then optimizing the cheap-to-query model via an infill criterion that chooses the point to evaluate next with the expensive function. [JIA20]

The value of batch optimization algorithms lies in the premise that they help to reduce the wall-clock time for optimization problems that can be evaluated in parallel. The e-shotgun is an infill criterion used to maximize the surrogate model and choose a batch of points to evaluate next.

[DEFR20]

The R-package was developed with the help of multiple existing R-packages including reticulate [UAT21], testthat [Wic11] and roxygen2 [WDCE20].

Reticulate is an Interface that enables the interoperability between R and Python. Functionalities contained in the package are the call of Python functions in R, the import of and use of Python Libraries into R as well as the translation between classes and data types from Python to R and vice versa. Reticulate served a critical role in the project by allowing for the reuse of the existing Python code and the embedding of Python functions into R.

Testthat is an R-Package that offers a set of functionalities for unit testing in R, enabling to check for equality, a given return type or expected warnings or errors.

The Roxygen2 R-package is used for In-line documentation and the automatic generation of documentation for functions and objects.

3 Methods

The package includes the source code of the interface, the core implementation of the e-shotgun in Python, test problems used for the unit tests as well as a simple optimizer for functionality testing during the development phase of the e-shotgun interface.

Reticulate enabled the use of the Python implementation of the e-shotgun in R without the need to port or rebuild the original code, further reticulate handles the translation of data types and parameters between Python and R.

The implementation of the interface mainly consists of three functions. The first function is named “checkLibraries”, the function has to be called one time initially to check that all imports of libraries that will be necessary for the interface and Python code to run, exist and fulfil the version requirements. If a requirement is not fulfiled or the version is outdated the function downloads or updates the package to prevent errors at run time or compilation. ”checkLibraries” has no parameters and no return type.

The second function is called “callEshotgun”, the function first validates the parameters (as seen in table 1) through several tests. If a test is failed the function call is terminated and an error message will be displayed (error messages are further discussed in the example section). The third function is called ”callEShotgunV2” and has an equal workflow as ”callEshotgun” on the R-side of the function. But with the distinction that a different implementation of the function is called on the Python side. In the original paper two versions of the e-shotgun was published one with Pareto front selection represented by ”callEshotgun” and one with random selection represented by ”callEShotgunV2” (for technical and mathematical refinements see [DEFR20]). When all tests are passed the parameters are transmitted between R and Python via the reticulate interface followed by the execution of the e-shotgun algorithm in Python. The result is then transmitted back to R and returned by the function. The test strategy gives more insight into how the parameters are validated.

Table 1: Parameters of callEshotgun and callEshotgunV2.

Parameter	Description	Data Type	Default Value
Xtr	A matrix containing a set of points	Matrix	none
Ytr	A matrix containing the result of Xtr applied to a given function	Matrix	none
f_lb	Lower bound of the search space	Vector	none
f_ub	Upper bound of the search space	Vector	none
q	The amount of values the e-shotgun should evaluate	Integer	10
epsilon	Probability of choosing to explore	Double	0.1
pf	Use Pareto Front	Boolean	false

Table 2: Functions of the interface.

Name	Parameters	Return Value
checkLibraries	none	none
callEshotgun	Xtr, Ytr, f_lb, f_ub, q, epsilon, pf	Matrix
callEshotgunV2	Xtr, Ytr, f_lb, f_ub, q, epsilon, pf	Matrix

4 Test Strategy

After changing the workflow of the original project by removing parts and functions responsible for the validation and testing of parameters. It is necessary to substitute testing and validation on the caller side. The following test strategy section discusses which parameters are tested and how the tests are conducted to prevent run time errors or allowing the passing of values that can affect the result negatively. Both callEshotgun-functions take the same list of parameters and have identical requirements that must be fulfilled therefore the test strategy is generalized and applies mutually.

The unit tests strategy can be divided into 3 main groups, focusing on different parameters and aspects.

The first parameter that has to validated is epsilon, the value passed as epsilon has to be in the interval of [0,1] to prevent the usage of a probability to explore greater than 1 or less than 0.

Another important group of validations is concerned with the lower and upper bound of the search space. The vectors of both bounds have to be equal in length and also the same length as the columns of the Xtr matrix. Furthermore, every value in the lower bound at the index i has to be strictly smaller than the value at the corresponding i in the upper bound.

The last group of unit tests verifies that the returned data type of the function call in Python is transmitted as expected. Additionally, that the returned matrix has the same amount of rows as specified with the parameter q.

5 Examples

This section aims to showcase how invalid and valid function calls can look like including the error an invalid call should provoke. For the following examples, the callEshotgun is interchangeable with callEshtogunV2. Function calls succeed and fail equally for the used parameters.

Xtr is a matrix with randomly generated values that can represent an initial design or a batch of points generated by a prior call of the callEshotgun. Ytr is the result of Xtr applied to the sphere function a commonly used test function for optimization algorithms. The lower and upper bound of the search space can be selected as preferred:

$$f(x) = \sum_{i=1}^{d} X_i^2$$

```
Xtr <- matrix(runif(20),ncol=2)
Ytr <- sphere(Xtr)
```

- A valid function call:

  ```
  callEshotgun(Xtr, Ytr, f_lb=c(-5.12, -5.12),
                  f_ub=c(5.12, 5.12), q=10L, epsilon=0.2)
  ```

- An invalid call that violates the equality of the length of the lower and upper bound:

  ```
  callEshotgun(Xtr, Ytr, f_lb=c(-5.12, -5.12, -5.12),
                  f_ub=c(5.12, 5.12), q=10L, epsilon=0.3)
  ```

  caused Error:
  Dimension of bounds don't match!

Dimension of Lower bound: 3
Dimension of Upper bound: 2

- Invalid call that violates the rule that every value of the lower bound has to be strictly lower than the values of the upper bound:

```
callEshotgun(Xtr, Ytr, f_lb=c(-5.12, -5.12),
                f_ub=c(5.12, -6.12), q=25L, epsilon=0.6)
```

caused Error:
All Values of the lower bound have to be strictly smaller.

- Function call that violates the valid epsilon interval:

```
callEshotgun(Xtr, Ytr, f_lb=c(-5.12, -5.12),
                f_ub=c(5.12, 5.12), q=10L, epsilon=2.0)
```

caused Error:
Epsilon has to be between 0.0 and 1.0
Passed Epsilon is 2

Assume that for the example below Ytr is an "eight by one"-matrix instead of the "ten by one"-matrix:

- ```
  callEshotgun(Xtr, Ytr, f_lb=c(-5.12, -5.12),
                  f_ub=c(5.12, 5.12), q=10L, epsilon=0.7)
  ```

caused Error:
Xtr and Ytr have unequal rows
Xtr is 10 and Ytr is 8

To reiterate, all the criteria needed for a successful function call: an epsilon value between [0,1], a lower bound vector that has the same dimension as the upper bound additionally strictly smaller values at corresponding indices. Furthermore, both bounds of the search space have to have the same dimension as the columns of the Xtr matrix, and that Xtr and Ytr have to be equal in the number of rows and the parameter pf has to be either false or true.

6 Discussion

The two main goals of the paper and the developed interface were, to make the e-shotgun algorithm available within the R-ecosystem in a reusable and openly accessible way, this criteria was fulfilled by the development of an R-package. Secondly, design the interface in a way that allows for changing between different optimization problems with minimal effort. This goal was reached by separating the implementation of the e-shotgun from the test problems. The e-shotgun is now capable of optimizing a variety of tasks whilst the stated parameter requirements are satisfied.

Further, the interface enables the use of a novel algorithm for Bayesian optimization in general and therefore broadens the existing landscape. Thereby helping the community to choose from a wider array of algorithms to find the best-suited one for the individual needs and requirements of a given problem.

References

[DEFR20] George De Ath, Richard M. Everson, Jonathan E. Fieldsend, and Alma A. M. Rahat. -shotgun : -greedy batch bayesian optimisation. In *Proceedings of the Genetic and Evolutionary Computation Conference*, New York, NY, USA, 2020. Association for Computing Machinery.

[JIA20] Qi ZHOU und Xinyu SHAO JIANG, Ping. *Surrogate Model-Based Engineering Design and Optimization*, 2020.

[UAT21] Kevin Ushey, JJ Allaire, and Yuan Tang. *reticulate: Interface to 'Python'*, 2021. R package version 1.20.
  ```

[WDCE20] Hadley Wickham, Peter Danenberg, Gábor Csárdi, and Manuel Eugster. *roxygen2: In-Line Documentation for R*, 2020. R package version 7.1.1.

[Wic11] Hadley Wickham. testthat: Get started with testing. *The R Journal*, 3:5–10, 2011.