Software Requirements Specification for Agolearn: An Evolutionary Black-Box Optimizer

Team 1, Agonaught(s)
Yiding Li

February 16, 2024

Contents

Revision History

Date	Version	Notes
2024-01–22	0.1	Initial draft
2024-02–02	0.2	First complete draft
2024-02–02	0.2	First coherent draft

This document is developed from a template based on Smith and Lai (2005); Smith et al. (2007); Smith and Koothoor (2016). At the time of writing, this template is located at https://github.com/smiths/capTemplate

1 Reference Material

This section records information for easy reference.

1.1 Abbreviations and Acronyms

symbol	description
A	Assumption
DD	Data Definition
GD	General Definition
GS	Goal Statement
IM	Instance Model
LC	Likely Change
PS	Physical System Description
R	Requirement
SRS	Software Requirements Specification
Agolearn	Agnostic Learner
TM	Theoretical Model

1.2 Mathematical Notation

symbol	description
$\mathbb{R}$	Type of real numbers
$\mathbb{R}^n$	Type of real vectors
$A \to B$	Type of functions from type A to type B
$\mathbb{R}^n \to \mathbb{R}$	Type of functions from real vectors to real numbers

2 Introduction

Black-box optimization problems deal with objective functions that are unknown, unexploitable, or non-existent*. As evolutionary algorithms lend well to optimizing against black-box objective functions, Agolearn seeks to implement an black-box optimizer using evolutionary methods.

This section includes the following subsections:

- **2.1 Purpose of Document**: A rough overview of this document: its intent, its contents, and its position in the development process.
- **2.2 Scope of Project**: Scope of problems that Agolearn seeks to address, an abstraction of assumptions.
- **2.3 Characteristics of Intended Reader**: Assumptions about the reader, including their knowledge and goals. This document is intended for readers who satisfy these assumptions.
- **2.4 Organization of Document**: Major sections of this document that follow the introduction.

2.1 Purpose of Document

This document records requirement specifications of Agolearn, in particular of the software that should satisfy these requirements. This document explains requirement specifications in the following parts:

- Scope and refined assumptions
- Problem and refined goals
- Theoretical foundation and refined models
- Inputs and outputs
- Functional and nonfunctional requirements

2.2 Scope of Project

Genomes, in particular variators that operate upon genomes, are costly to implement. As such, Agolearn only implements genomes of real vectors and genetic programs and only implements objective functions from these types. This scope decision refines to A1 and A2 Agolearn use evolutionary algorithms, which are iterative, do not guarantee optimality, and do not incorporate hard constraints. This scope decision refines to A3, A4, A5, and A6.

Agolearn is a library. This assumption removes the need to consider complex user interfaces. This scope decision refines to A7.

2.3 Characteristics of Intended Reader

Readers of this document should have proficiency in at least high school level math [link]. In addition, readers should be familiar with the following concepts relating to optimization and evolutionary learning:

- Optimization: Objective functions, hyper-parameters, and iterative processes
- Evolutionary operators: (a) evaluator, (b) variator, (c) selectors, and their roles in the evolutionary process
- Evolutionary pressure induced by different evolutionary operators, and their effect on exploration and exploitation

Familiarity with terminologies defined in **4.1.1 Terminology and Definitions** are helpful, but not required to understand this document.

2.4 Organization of Document

This document begins with an overview of this document and Agolearn, including: (a) the purpose of Agolearn, (b) the scope of Agolearn, and (c) characteristics of the intended reader.

Subsequent sections describes the system in increasing specificity, ending with a mathematical definitions of the system. These sections are:

3 General System Description: General information about the system. It identifies the interfaces between the system and its environment, describes the user characteristics and lists the system constraints.

4 Specific System Description: This section first presents the problem description, which gives a high-level view of the problem to be solved. This is followed by the solution characteristics specification, which presents the assumptions, theories, definitions and finally the instance models.

5 Requirements: This section provides the functional requirements, the business tasks that the software is expected to complete, and the nonfunctional requirements, the qualities that the software is expected to exhibit.

6 Likely Changes and **7 Unlikely Changes**: These sections document anticipated changes to requirements. This information instructs design and implementation.

8 Traceability Matrices and Graphs: This section connects requirement items in this document. Requirement items include (a) assumptions, (b) goal statements, (c) instance models, (d) functional requirements, (e) nonfunctional requirements, (f) likely changes, and (g) unlikely changes.

3 General System Description

This section includes the following subsections:

- **3.1 System Context**: Abstract interface and responsibilities of the software
- **3.2 User Characteristics**: Characteristics of expected users of the software

3.1 System Context

Agolearnimplements an optimizer. As an optimizer, Agolearntakes (a) an objective function and (b) hyper-parameters that instruct the optimization process. As an optimizer, Agolearnoutputs a solution that optimizes the given objective function. Figure 1 describes the interaction between Agolearnand the user.

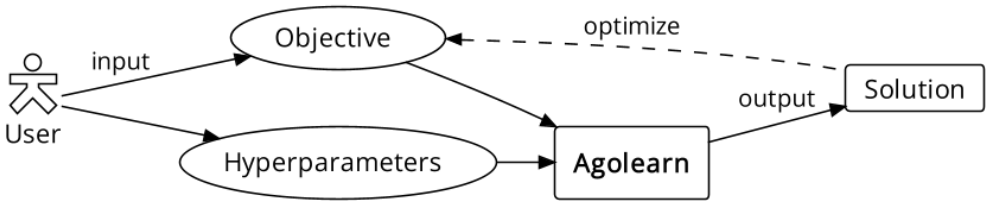

Figure 1: System Context

Responsibilities of Agolearnand the user are as follows. Appropriate performance of Agolearnto its specification requires that both responsibilities are satisfied.

- User Responsibility:
  - Provide correctly typed inputs
- Agolearn Responsibilities:
  - Detect and report mistyped inputs
  - Detect and report constraint violations of the input
  - Calculate required outputs

3.2 User Characteristics

The intended user of Agolearn has the following characteristics:

- Motivation: The user seeks to optimize an objective function of real numbers or functions.

- Capability: The user can make function calls to libraries
- Capability: The user is familiarity with evolutionary operators and can choose from a pre-defined catalogue of operators.

4 Specific System Description

This section includes the following subsections:

- **4.1 Problem Description**: Description and refinements of the problem
- **4.2 Solution Characteristics Specification**: Characteristics of a good solution, which Agolearn should produce
- **4.3 Scope Decisions**: Scope decisions that affect design and implementation

4.1 Problem Description

Optimization of black-box functions: functions that do not follow a particular form.

4.1.1 Terminology and Definitions

The subsequent sections use the following terms. Definitions are given according to Eiben and Smith (2003).

- Genetic program: A evolutionary algorithm that generates genetic programs
- Genetic programming algorithm: A program that is evolutionarily produced
- Evolutionary operators: Operators that constitute a evolutionary algorithm
- Genome: A mathematical representation of a solution to the objective function
- Evaluator: An evolutionary operator that evaluates the fitness of a genome
- Variator: An evolutionary operator that modifies genomes
- Selector: An evolutionary operator that selects from a pool of genomes
- Parents: A pool of genomes about to undergo variation
- Offspring: The output of a variator
- Fitness: The score of a genome according to the evaluator
- Higher-Order Function: A function from a function to a value

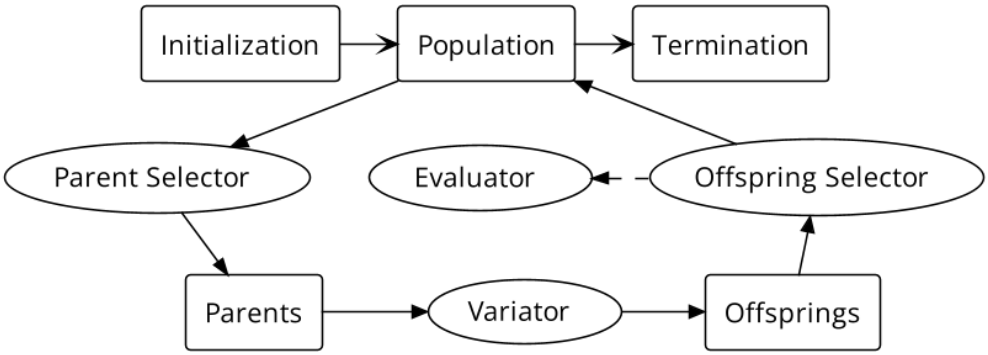

Figure 2: Evolutionary Operators and their Positions in an Evolutionary Algorithm

4.1.2 Goal Statements

Agolearn accepts objective function as evaluators. For each objective function, Agolearn returns a genome with optimal fitness.
In particular, given the assumed inputs, the goal statements are:

GS1: Optimize against real-valued functions

GS2: Optimize against higher-order functions

4.2 Solution Characteristics Specification

The output of Agolearn optimizes the given objective function. The solution should have the same type of the initial population.

The highest score of the output should be greater or equal to that of the initial population.

The instance models that govern Agolearn are presented in Subsection 4.3.2. The information to understand the meaning of the instance models and their derivation is also presented, so that the instance models can be verified.

4.3 Scope Decisions

This system only optimizes functions of real numbers, as well as higher-order functions.

An evolutionary algorithm must define a variator for each individual selector. Such variators are costly to implement and it would be difficult to require them as user inputs. The decision is made to restrict representations to two types: vectors of real values and higher-order functions.

Consequently, the restriction on the types of solutions restricts the range of inputs

4.3.1 Assumptions

A1: The input objective function can be real-valued (GS1)

A2: The input objective function can be a higher-order function (GS2)

A3: Agolearn implements evolutionary algorithms

A4: Agolearn may not output the global optimum of the input objective function.

A5: Agolearn is an iterative optimizer.

A6: Agolearn does not accept explicit constraints.

A7: Agolearn is a library.

4.3.2 Instance Models

The following instance models describe systems that corresponds to goals.

Number	IM1
Label	**Optimize real-valued functions**
Input	$F : \mathbb{R}^n \to \mathbb{R}$
Output	$X : \mathbb{R}^n$
Description	The system receives a real-valued objective function, then emits a real vector that optimizes the objective function.
Goal	GS1

Number	IM2
Label	**Optimize higher-order functions**
Input	$F : (A^n \to B) \to \mathbb{R}$
Output	$X : (A^n \to B)$
Description	The system receives a higher-order objective function, then emits a function that optimizes the given objective function.
Goal	GS2

4.3.3 Properties of a Correct Solution

The output of Agolearn seeks, but not necessarily, optimize the input objective function (A4).

Because Agolearn is an iterative optimizer (A5), the best genome in each generation has a better fitness than the best genome in the last generation.

5 Requirements

This section includes the following subsections:

- **5.1 Functional Requirements**: Requirements that make explicit tasks and behaviours the software should complete
- **5.2 Nonfunctional Requirements**: Requirements that facilitate the satisfaction of functional requirements
- **5.3 Rationale**: Rationale for scope decisions, modeling decision, and assumptions

5.1 Functional Requirements

R1: The system checks if input values are correctly typed and within input constraints.

R2: If input values are correctly typed and within input constraints, then the system should operate normally (satisfy all requirements).

R3: Evolutionary operators of the same type can be used interchangeably

R4: Each iteration of the algorithm (A5) should produce a genome (A4).

R5: Each genome produced by the algorithm (A4) have equal or higher fitness than that of the previous generation (A4).

5.2 Nonfunctional Requirements

NFR1: **Accuracy** Each iteration of the algorithm should produce a solution whose value, when given to the given objective function, is better than that of the best solution of the previous iteration.

NFR2: **Usability-Implementation** The software should be implemented as a library and expose an interface in the form of function calls

NFR3: **Usability-Docs** Every public variables, functions, and class should be described.

NFR4: **Maintainability** Implementation of the software should be typed. Where types cannot be enforced, types must be hinted.

NFR5: **Portability** The system should be able to run on Windows 11, up to 024-01 Cumulative Update for Windows 11 Version 22H2 for x64-based Systems (KB5034123)

5.3 Rationale

[None at the moment #NOTE]

6 Likely Changes

LC1: The selection of evolution operators have not been defined and will change.

LC2: Hyper-parameters are not concretely defined. The types of hyperparameter inputs will change.

LC3: Implementation of other types of objective functions.

7 Unlikely Changes

LC4: Implement Agolearnwith another technology than evolutionary models

LC5: Add assurance that the result is a global optimum

8 Traceability Matrices and Graphs

The purpose of the traceability matrices is to provide easy references on what has to be additionally modified if a certain component is changed. Every time a component is changed, the items in the column of that component that are marked Table 2 shows the dependencies of instance models, requirements, and data constraints on each other. Table 1 shows the dependencies of theoretical models, general definitions, data definitions, instance models, and likely changes on the assumptions.

The purpose of the traceability graphs is also to provide easy references on what has to be additionally modified if a certain component is changed. The arrows in the graphs represent dependencies. The component at the tail of an arrow is depended on by the component at the head of that arrow. Therefore, if a component is changed, the components that it points to should also be changed.

9 Development Plan

Develop an abstract framework that represents the evolutionary process, then craft instances of these operators.

10 Values of Auxiliary Constants

[None at the moment #NOTE]

	GS1	GS2	R4	R5	A4	A5
A1	$\leftarrow$					
A2		$\leftarrow$				
A3					$\rightarrow$	$\rightarrow$
A4				$\rightarrow$		
A5			$\rightarrow$			
A6						
A7						

Table 1: Traceability Matrix Showing the Connections Between Assumptions and Other Items

	IM1	IM2
A3	X	X
A2	X	X
A1	X	X
A5	X	X

Table 2: Traceability Matrix Showing the Connections Between Requirements and Instance Models

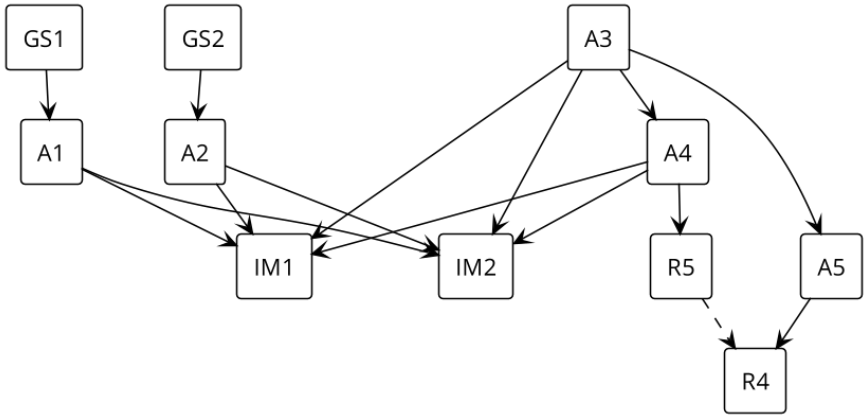

Figure 3: Traceability Matrix Showing the Connections Between Assumptions, Goal Statements, and Instance Models

References

A. Eiben and Jim Smith. *Introduction To Evolutionary Computing*, volume 45. 01 2003. ISBN 978-3-642-07285-7. doi: 10.1007/978-3-662-05094-1.

W. Spencer Smith and Nirmitha Koothoor. A document-driven method for certifying scientific computing software for use in nuclear safety analysis. *Nuclear Engineering and Technology*, 48(2):404–418, April 2016. ISSN 1738-5733. doi: http://dx.doi.org/10.1016/j.net.2015.11.008. URL http://www.sciencedirect.com/science/article/pii/S1738573315002582.

W. Spencer Smith and Lei Lai. A new requirements template for scientific computing. In J. Ralyté, P. Ågerfalk, and N. Kraiem, editors, *Proceedings of the First International Workshop on Situational Requirements Engineering Processes – Methods, Techniques and Tools to Support Situation-Specific Requirements Engineering Processes, SREP'05*, pages 107–121, Paris, France, 2005. In conjunction with 13th IEEE International Requirements Engineering Conference.

W. Spencer Smith, Lei Lai, and Ridha Khedri. Requirements analysis for engineering computation: A systematic approach for improving software reliability. *Reliable Computing, Special Issue on Reliable Engineering Computation*, 13(1):83–107, February 2007.