Software Requirements Specification for Agolearn: subtitle describing software

Team 1, Agonaught(s)
Yiding Li

February 2, 2024

Contents

Revision History

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

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
$\mathbb{R}$	Real number
$\mathbb{R}^n$	Real vector
$\mathbb{R}^n \rightarrow \mathbb{R}$	Function from real vectors to real numbers

Hoffman and Strooper (1995) This thing has caused me great pain. [TODO: Save for later. List notations that are used in the document.]

[A reference document may not need to be read sequentially. For this document, the suggested order is: #NOTE]

- Goal Statement
- Instance Models
- Requirements
- Introduction
- Specific System Description

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 introduction includes the following subsections:

- Purpose of Document: The intent of this doocument
- Scope of requirements: [TODO: am I the scope of requirements, or scope of requirements of the project, or scope of this document, which lists the requirements?]
- Characteristics of Intended Reader: Assumptions about the reader, including their knowledge and goals. This document is intended for readers who satisfy these assumptions.

[I find it apt to explain these sections in detail. This will be where the reader makes first contact with the document - better make it count! tell the reader that "This document is intended for readers who satisfy these assumptions", so that they do not waste more time than necessary on if they should stay. #NOTE]

2.1 Purpose of Document

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

- Inputs and outputs of the software
- Assumptions and scope
- Broader problem and refined goals

- Mathematical model and theoretical foundation

This document repeats each item on several levels of abstraction. For example, [TODO: give example].

2.2 Scope of Requirements

[TODO: What am I?]

Genomes, in particular variators that operate upon genomes, are costly to implement. As such, Agolearn only conisders genomes of real vectors and genetic programs and only implements objective functions from these types.

Agolearn does not assume the shape of objective functions. As such, Agolearn does not guarantee optimality of the result, only that the result performs better than the initial guess.

Agolearn does not allow hard constraints on genome values. However, it should ensure that the result retains the same type, so that the result remains a valid solution.

2.3 Characteristics of Intended Reader

To understand the mathematical language in this document, the reader should be proficient in high school levels of mathematics, in addition to following items:

- Types: Real numbers, functions and functions of real numbers
- Optimization: Objective functions, hyperparameters, and iterative processes

The reader should understand the subject matter of evolutionary computing, in particular:

- Types of values and functions
- The balance between quality and novelty
- Evolutionary operators: (a) evaluator, (b) variator, (c) selectors, and their roles in the evolutionary process

2.4 Organization of Document

[I find the information on ”template” distracting. The reader should not care where the document comes from, only what it does. #NOTE]

This document begins with an overview of itself and the system, including (a) the purpose of this document, (b) the scope of requirements [TODO: of the project], and (c) characteristics of the intended reader. Then, this document describes the system in descending levels of abstraction. This document ends with precise, mathematical definitions of the problem, the system, as well as decisions that motivate earlier parts.

In particular, this document divides into the following major sections. These sections describe requirements in descending levels of abstraction.

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.

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.

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.

[TODO: After everything else]

3 General System Description

The program serves two objectives: (a) to optimize real-valued functions, and (b) to optimize higher-order functions.

[I still expereince a degree of .. "narrative whiplash", from the document switching between discussing itself and the project. I propose that I store information on "what the section does" in a small, compact space (such as "Organization of Document"), and leave the rest to describe the system #NOTE]

3.1 System Context

The system acts as an optimizer. The system takes (a) an objective function and (b) hyperparameters that instruct the optimization process. The system emits a solution that optimizes the objective function.

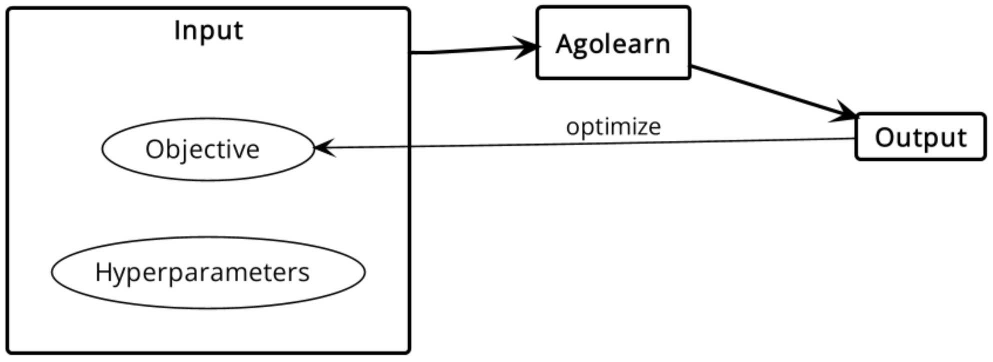


Figure 1: System Context

[Should I list these responsibilities if the user is... almost abstracted away as ”input”? Would it be redundant? #NOTE]

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

3.2 User Characteristics

[TODO: I think this is a good idea: to separate motivation and ability, or do we only consider capability, assuming that the ”motivation” is sufficient. I'm still working on my English.] The user should be motivated to, and able to, use Agolearn.

Motivation:

- The user should seek to optimize a function of real numbers, or a higher-order function.

Capability:

- Familiarity with making function calls to Agolearn
- Familiarity with evolutionary operators, and ability to choose operators from a pre-defined catalogue.

4 Specific System Description

[Add any project specific details that are relevant for the section overview. —TPLT]

[TODO: Hard to think of any - that is not appropriate for Problem Description]

4.1 Problem Description

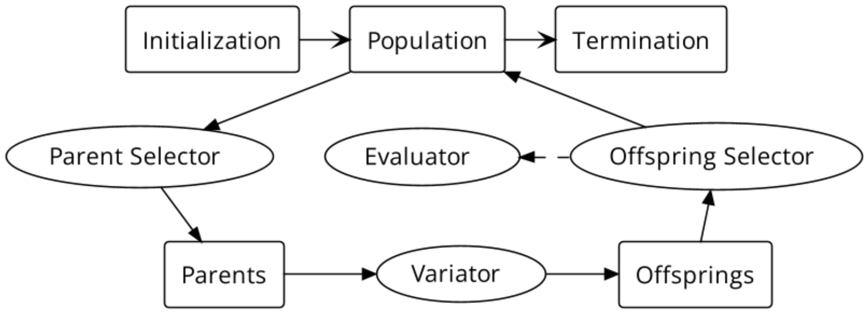
Optimization of black-box functions: functions that do not follow a particular form. [TODO: I find this duplicating Introduction]

4.1.1 Terminology and Definitions

[TODO: move the big boy here] [TODO: If I assume familiarity of the evolutionary process...] [TODO: Would it be appropriate to move it to either the beginning or the end, so that it is with other references?]

This subsection provides a list of terms that are used in the subsequent sections and their meaning, with the purpose of reducing ambiguity and making it easier to correctly understand the requirements:

- Evolutionary algorithm
- Genetic program
- Genetic programming algorithm
- Evolutionary operators
- Evolutionary operators
- Genome
- Evaluator
- Variator
- Selector
- Parents
- Offspring
- etc.



4.1.2 Goal Statements

Agolearn accepts objective function of two types. For each objective funnction, Agolearn seeks to produce a genome that optimizes the objective function.
In particular, Given the inputs, the goal statements are:

GS1: Optimize against real-valued functions

GS2: Optimize against higher-order functions

4.2 Solution Characteristics Specification

The system should produce a solution that 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.

[This section specifies the information in the solution domain of the system to be developed. This section is intended to express what is required in such a way that analysts and stakeholders get a clear picture, and the latter will accept it. The purpose of this section is to reduce the problem into one expressed in mathematical terms. Mathematical expertise is used to extract the essentials from the underlying physical description of the problem, and to collect and substantiate all physical data pertinent to the problem. —TPLT]

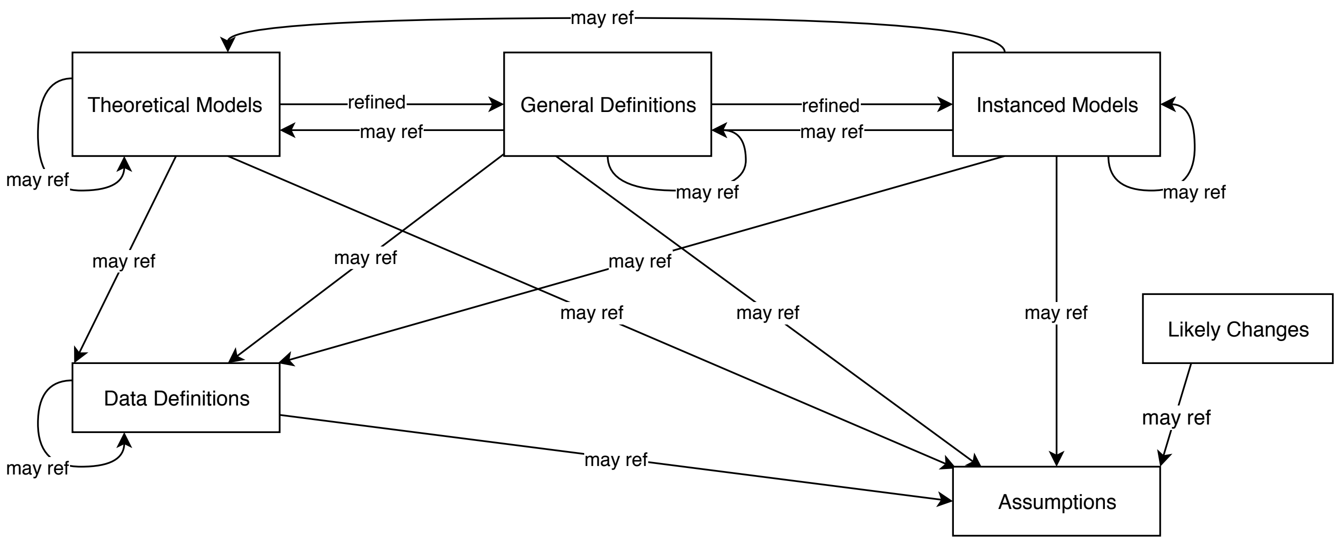
[This section presents the solution characteristics by successively refining models. It starts with the abstract/general Theoretical Models (TMs) and refines them to the concrete/specific Instance Models (IMs). If necessary there are intermediate refinements to General Definitions (GDs). All of these refinements can potentially use Assumptions (A) and Data Definitions (DD). TMs are refined to create new models, that are called GMs or IMs. DDs are not refined; they are just used. GDs and IMs are derived, or refined, from other models. DDs are not derived; they are just given. TMs are also just given, but they are refined, not used. If a potential DD includes a derivation, then that means it is refining other models, which would make it a GD or an IM. —TPLT]

[The above makes a distinction between "refined" and "used." A model is refined to another model if it is changed by the refinement. When we change a general 3D equation to a 2D equation, we are making a refinement, by applying the assumption that the third dimension does not matter. If we use a definition, like the definition of density, we aren't refining, or changing that definition, we are just using it. —TPLT]

[The same information can be a TM in one problem and a DD in another. It is about how the information is used. In one problem the definition of acceleration can be a TM, in another it would be a DD. —TPLT]

[There is repetition between the information given in the different chunks (TM, GDs etc) with other information in the document. For instance, the meaning of the symbols, the units etc are repeated. This is so that the chunks can stand on their own when being read by a reviewer/user. It also facilitates reuse of the models in a different context. —TPLT]

[The relationships between the parts of the document are show in the following figure. In this diagram "may ref" has the same role as "uses" above. The figure adds "Likely Changes," which are able to reference (use) Assumptions. —TPLT]



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

[TODO: Should I makde the scope decisions more concise, and move the assumptions here? That would "damage the flow", I find.] 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

[TODO: onlt because there is no rationale for assumptions. I feel that Scopes and Assumptions can reside on the same level, where one is "hard" and the other is "soft". Rationales for Scopes should not be on the same level as Assumptions, if that is the case.]

- [TODO: Should I number assumptions, or give more "expressive names"? I see more expressive names in the given examples.]

- This assumption might not lead to implementations: the program will - and I know this because I tried - just be less effective when the assuption does not hold. This assumption is here only so that the ”optimizaion” definition holds.

A1: To ensure that the result is optimized, the objective function should be consistant. That is, the objective function should involve minimal stochasticy and do not change with time.

[“Ref. By” is used repeatedly with the different types of information. This stands for Referenced By. It means that the models, definitions and assumptions listed reference the current model, definition or assumption. This information is given for traceability. Ref. By provides a pointer in the opposite direction to what we commonly do. You still need to have a reference in the other direction pointing to the current model, definition or assumption. As an example, if TM1 is referenced by GD2, that means that GD2 will explicitly include a reference to TM1. —TPLT]

4.3.2 Instance Models

[The motivation for this section is to reduce the problem defined in “Physical System Description” (Section **??**) to one expressed in mathematical terms. The IMs are built by refining the TMs and/or GDs. This section should remain abstract. The SRS should specify the requirements without considering the implementation. —TPLT]

[TODO: It is a refinement of the problem, or a refinement of the objective?]

This section transforms the problem defined in $Section\ Sec_pd$ into one which is expressed in mathematical terms. It uses concrete symbols defined in Section dratadef to replace the abstract symbols in the models identified in Sections **??** and **??**.

The goals [reference your goals —TPLT] are solved by [reference your instance models —TPLT]. [other details, with cross-references where appropriate. —TPLT] [Modify the examples below for your problem, and add additional models as appropriate. —TPLT]

Number	IM1
Label	**Optimize real-valued functions**
Input	$F : \mathbb{R}^n \rightarrow \mathbb{R}$
Output	$X : \mathbb{R}$
Description	[TODO: I don’t think the project needs this.]
Sources	Citation here
Ref. By	IM**??**

4.3.3 Properties of a Correct Solution

[TODO: The assumptions, in particular of ”the type of the output” and ”optimality not guaranteed”, as listed in Solution Characteristics Specification, do not need to be replicated.]

Table 1: Output Variables

Var	**Physical Constraints**
T_W	$T_{\text{init}} \leq T_W \leq T_C$ (by A??)

[This section is not for test cases or techniques for verification and validation. Those topics will be addressed in the Verification and Validation plan. —TPLT]

5 Requirements

5.1 Functional Requirements

R1: [Requirements for the inputs that are supplied by the user. This information has to be explicit. —TPLT]

R2: [It isn’t always required, but often echoing the inputs as part of the output is a good idea. —TPLT]

R3: [Calculation related requirements. —TPLT]

R4: [Verification related requirements. —TPLT]

R5: [Output related requirements. —TPLT]

[Every IM should map to at least one requirement, but not every requirement has to map to a corresponding IM. —TPLT]

5.2 Nonfunctional Requirements

[List your nonfunctional requirements. You may consider using a fit criterion to make them verifiable. —TPLT] [The goal is for the nonfunctional requirements to be unambiguous, abstract and verifiable. This isn’t easy to show succinctly, so a good strategy may be to give a “high level” view of the requirement, but allow for the details to be covered in the Verification and Validation document. —TPLT] [An absolute requirement on a quality of the system is rarely needed. For instance, an accuracy of 0.0101 % is likely fine, even if the requirement is for 0.01 % accuracy. Therefore, the emphasis will often be more on describing now well the quality is achieved, through experimentation, and possibly theory, rather than meeting some bar that was defined a priori. —TPLT] [You do not need an entry for correctness in your NFRs. The purpose of the SRS is to record the requirements that need to be satisfied for correctness. Any statement of correctness would just be redundant. Rather than discuss correctness, you can characterize how far away from the correct (true) solution you are allowed to be. This is discussed under accuracy. —TPLT]

NFR1: **Accuracy** [Characterize the accuracy by giving the context/use for the software. Maybe something like, "The accuracy of the computed solutions should meet the level needed for <engineering or scientific application>. The level of accuracy achieved by Agolearn shall be described following the procedure given in Section X of the Verification and Validation Plan." A link to the VnV plan would be a nice extra. —TPLT]

NFR2: **Usability** [Characterize the usability by giving the context/use for the software. You should likely reference the user characteristics section. The level of usability achieved by the software shall be described following the procedure given in Section X of the Verification and Validation Plan. A link to the VnV plan would be a nice extra. —TPLT]

NFR3: **Maintainability** [The effort required to make any of the likely changes listed for Agolearn should be less than FRACTION of the original development time. FRACTION is then a symbolic constant that can be defined at the end of the report. —TPLT]

NFR4: **Portability** [This NFR is easier to write than the others. The systems that Agolearn should run on should be listed here. When possible the specific versions of the potential operating environments should be given. To make the NFR verifiable a statement could be made that the tests from a given section of the VnV plan can be successfully run on all of the possible operating environments. —TPLT]

- Other NFRs that might be discussed include verifiability, understandability and reusability.

5.3 Rationale

[Provide a rationale for the decisions made in the documentation. Rationale should be provided for scope decisions, modelling decisions, assumptions and typical values. —TPLT]

6 Likely Changes

LC1: [Give the likely changes, with a reference to the related assumption (aref), as appropriate. —TPLT]

7 Unlikely Changes

LC2: [Give the unlikely changes. The design can assume that the changes listed will not occur. —TPLT]

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 with an "X" may have to be modified as well. Table 2 shows the dependencies of theoretical models, general definitions, data definitions, and instance models with each other. Table 3 shows the dependencies of instance models, requirements, and data constraints on each other. Table 4 shows the dependencies of theoretical models, general definitions, data definitions, instance models, and likely changes on the assumptions.

[You will have to modify these tables for your problem. —TPLT]

[The traceability matrix is not generally symmetric. If GD1 uses A1, that means that GD1's derivation or presentation requires invocation of A1. A1 does not use GD1. A1 is "used by" GD1. —TPLT]

[The traceability matrix is challenging to maintain manually. Please do your best. In the future tools (like Drasil) will make this much easier. —TPLT]

	TM??	TM??	TM??	GD??	GD??	DD??	DD??	DD??	DD??	IM1	IM??	IM?
TM??												
TM??			X									
TM??												
GD??												
GD??	X											
DD??				X								
DD??				X								
DD??												
DD??								X				
IM1					X	X	X				X	
IM??					X		X		X	X		
IM??		X										
IM??		X	X				X	X	X		X	

Table 2: Traceability Matrix Showing the Connections Between Items of Different Sections

	IM1	IM??	IM??	IM??	??	R??	R??
IM1		X				X	X
IM??	X			X		X	X
IM??						X	X
IM??		X				X	X
R??							
R??						X	
R??					X		
R2	X	X				X	X
R??	X						
R??		X					
R??			X				
R??				X			
R4			X	X			
R??		X					
R??		X					

Table 3: Traceability Matrix Showing the Connections Between Requirements and Instance Models

	A??	A??	A??	A??	A??	A??	A??	A??	A??	A??	A??	A??	A??	A??	A??	A??	A??	A??	A??
TM??	X																		
TM??																			
TM??																			
GD??		X																	
GD??			X	X	X	X													
DD??							X	X	X										
DD??			X	X						X									
DD??																			
DD??																			
IM1											X	X		X	X	X			X
IM??												X	X			X	X	X	
IM??														X					X
IM??													X					X	
LC??				X															
LC??								X											
LC??									X										
LC??											X								
LC??												X							
LC??															X				

Table 4: Traceability Matrix Showing the Connections Between Assumptions and Other Items

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. Figure **??** shows the dependencies of theoretical models, general definitions, data definitions, instance models, likely changes, and assumptions on each other. Figure **??** shows the dependencies of instance models, requirements, and data constraints on each other.

9 Development Plan

[This section is optional. It is used to explain the plan for developing the software. In particular, this section gives a list of the order in which the requirements will be implemented. In the context of a course this is where you can indicate which requirements will be implemented as part of the course, and which will be "faked" as future work. This section can be organized as a prioritized list of requirements, or it could should the requirements that will be implemented for "phase 1", "phase 2", etc. —TPLT]

10 Values of Auxiliary Constants

[Show the values of the symbolic parameters introduced in the report. —TPLT]

[The definition of the requirements will likely call for SYMBOLIC_CONSTANTS. Their values are defined in this section for easy maintenance. —TPLT]

[The value of FRACTION, for the Maintainability NFR would be given here. —TPLT]

References

Daniel M. Hoffman and Paul A. Strooper. *Software Design, Automated Testing, and Maintenance: A Practical Approach*. International Thomson Computer Press, New York, NY, USA, 1995. URL http://citeseer.ist.psu.edu/428727.html.

[The following is not part of the template, just some things to consider when filing in the template. —TPLT]
[Grammar, flow and LaTeXadvice:

- For Mac users `*.DS_Store` should be in `.gitignore`
- LaTeX and formatting rules
  - Variables are italic, everything else not, includes subscripts (link to document)
    - Conventions
    - Watch out for implied multiplication
  - Use BibTeX
  - Use cross-referencing
- Grammar and writing rules
  - Acronyms expanded on first usage (not just in table of acronyms)
  - "In order to" should be "to"

—TPLT]
[Advice on using the template:

- Difference between physical and software constraints
- Properties of a correct solution means *additional* properties, not a restating of the requirements (may be "not applicable" for your problem). If you have a table of output constraints, then these are properties of a correct solution.
- Assumptions have to be invoked somewhere
- "Referenced by" implies that there is an explicit reference
- Think of traceability matrix, list of assumption invocations and list of reference by fields as automatically generatable
- If you say the format of the output (plot, table etc), then your requirement could be more abstract

—TPLT]