Thomas Minkowski
MATH 3210-01

Research Report 5

Analysis Technique

Fully understanding Genetic Programming requires knowledge from different areas, as Genetic Programming is a result of other practices. The first thing that needs to be discussed is the darwinian theory of evolution. This theory states that all life is related and has descended from a common ancestor. The idea is that complex creatures evolve to what they become through a natural process. Random genetic mutations occur within an organisms genetic code, and beneficial mutations are preserved due to the organisms newfound ability to fit in better with their environment. Through reproduction, these traits are passed on to the next generation, creating a generation more fit for their environment. When that generation undergoes natural selection, the same cycle repeats, eventually leading to the organism drastically changing from what it originally was in order to fit in better with the environment.

Evolutionary computation is a broad field that uses the ideas behind Darwin's theory of evolution to create algorithms. These algorithms are called evolutionary algorithms, and one subset of this is Genetic Algorithms. Genetic algorithms use biological evolution as inspiration, by using reproduction, mutation, recombination, and survival of the fittest to come up with solutions. To understand how the algorithm works, relating it to the ideas of biological evolution are extremely helpful. In evolution, there is a population with certain individuals in that population. With genetic algorithms, the candidate solutions are the individuals. Each individual is a type of solution, but may not be the most fit for the environment that we would like. To account for this, there is a fitness function in the algorithm that places a value on how fit we would like the solution to be in the environment.

The manipulation of the individuals is what leads to more fit individuals. These manipulations are reproduction, mutation and recombination. Reproduction is using the same individual in the next generation of candidate solutions. Mutation involves changing specific parts of the individuals for the next generation. Recombination involves combining traits from two different individuals to create a new one for the next generation. These operations can be performed for a certain amount of time, or until the desired fitness level has been reached. How these manipulations occur can also be influenced by the user. The probability that recombination will take place could be made higher than reproduction, which may give different results. Along with this, the individuals with the highest fitness value in a generation can be given a certain probability that it will be used for the next generation over the other, less fit individuals.

Genetic Algorithms are what make genetic programming possible. Genetic programming uses genetic algorithms, but the individuals in the population are seen as programs. These programs can be represented through mathematic equations, involving variables, constants and operations. The way this is displayed is often through a tree, allowing for the user to understand how the programs are laid out. The idea that these programs can be manipulated through genetic programming and displayed in a way that a user can easily understand makes genetic programming an incredibly useful tool when trying to understand and manipulate things such as electrical circuits, antennas or other areas of design.

Genetic programming has many different applications. Symbolic Regression uses genetic programming to find relationships between independent and dependent variables within information. Through genetic programming, it is represented in a way that allows for the user to understand the information symbolically. Feature selection is for when not all independent variables influence the defendant variables. Through this, the subset of independent variables that actually influences the dependent variables can be found. One more application is through automatic programming. Automatic programming involves using the genetic algorithms to create programs that carry out a user defined task, without creating the program solution themselves.

The RGP package in R is a package that allows for genetic programming to be done through the R interface. There are many features that this package has which allows us to apply genetic programming to the different areas mentioned previously. RGP represents genetic programming individuals through R expressions. R expressions are laid out in a tree structure, which allows for the genetic programming done through the RGP package to be tree based as well. RGP allows for typed and untyped genetic programming, as well as function defining subtrees.

In the RGP package, there are three different types of operators. These operators are initiation, variation, and selection. These operators make up the basis for genetic programming. The operators have default values, but the user has a large degree of power to manipulate these values. The initialization operator involves how the initial trees are created. There are multiple different ways this can be done, and the user can decide this as well as combine different ways to create a tree in the same genetic programming problem. The variation operators involve how the individuals are manipulated for the next generation. There are multiple different types for each type of manipulation that can be used. The user can decide which ones to use, as well as how they are used. The probability of each manipulation technique can also be controlled. The final operator is the selection operator, which controls which individuals are selected for the next generation. The default selection tools add useful techniques, in which the selection can be made by weighting the strengths of the solution with the complexity. Along with being able to manipulate the default operations in the RGP package as explained above, entirely new ways to undergo these operations can be created and used.

RGP also allows different ways to visualize the different individuals and populations in the genetic program. The individuals can be viewed as trees, mathematic formulas, along with a few other visualization tools. Since RGP is a package in R, it also has access to the many visualization tools that R brings as well. Also, since R is extremely useful for statistical analysis, this can be done on the genetic program as well. The RGP package handles the genetic programming aspect, and the strength of using it in R is that everything else that R brings can also be applied to the findings.

To experiment with the RGP package in R, I will be doing two symbolic regression problems. With the first experiment, RGP will be use to create a polynomial expression that attempts to fit the sin function. The first thing to do is to define the search space. This means that we are telling the RGP package what the trees will be made up of. There will be three sets that make up this search space: variables, function symbols, and constants. The function symbols will be addition, subtraction, and multiplication since we are creating a polynomial. There will be one variable which will be labeled as ‘x’. To create the constants, RGP needs a function, so we will have it take a random constant from a normal distribution. Next the fitness function must be told to RGP. As stated, the goal is to create a polynomial that mimics the sign function. Therefore, the fitness function must be the sign function, since this is what we want our variables, symbols, and constants to organize into. This will direct the polynomial towards what we would like it to be. After these things are initialized, the genetic program function will run for five minutes until it terminates, and then it will select the fittest individual. Once our solution is given, we will plot it against the sin function and compare the result.

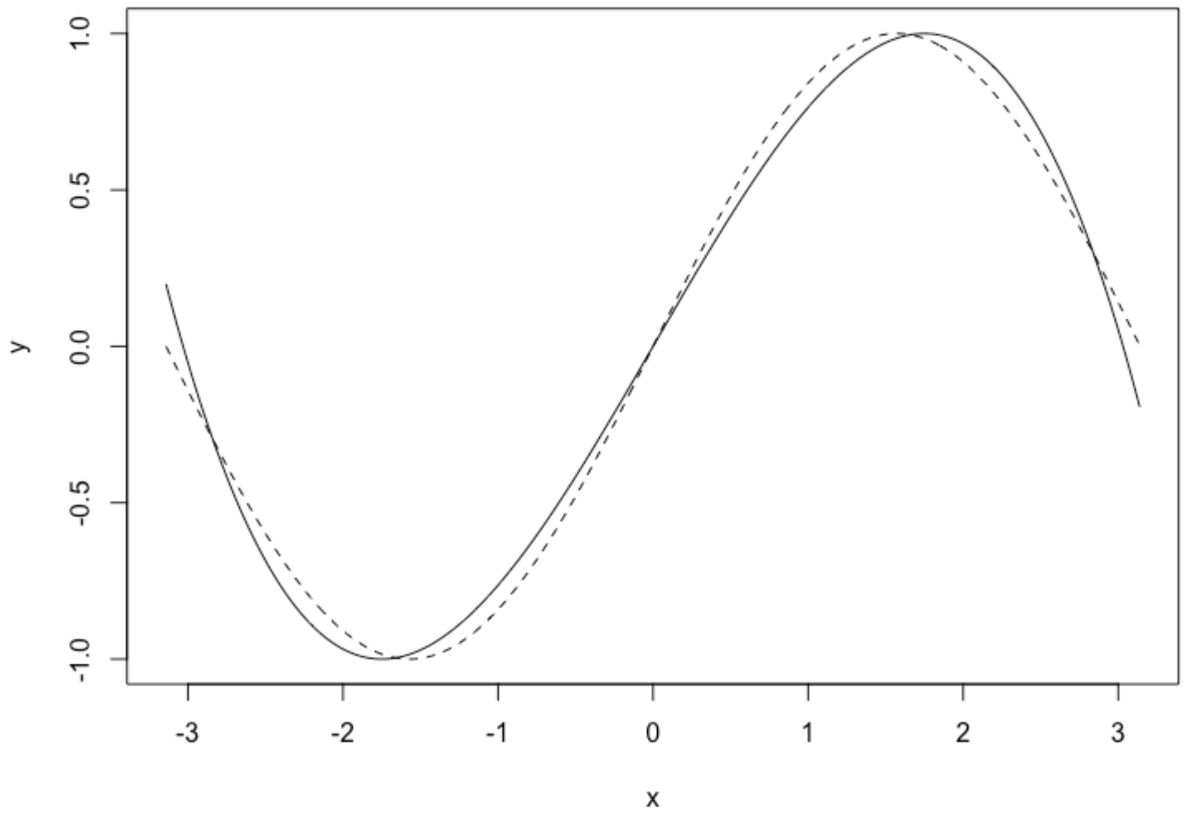
The second experiment is another symbolic regression problem, which will use the symbolic regression function within the RGP package. This experiment will involve using the equation for a damped pendulum to create a pendulum in R, which will be used to create data. Once this data is created, noise will be added by using a normal distribution, in order to simulate real data taken from the pendulum created. The data created consists of 512 samples over a specified time interval. With the data created, the symbolic regression function within RGP can be run. The search space in this example doesn't need to be specified, since the experiment isn't about creating a polynomial to fit the data. By not specifying the search space, all the different math functions will be used, which is the default setting for the search space. The time will be set to two minutes, and then the symbolic regression run will stop. The most fit individual will be chosen, and plotted against the data gathered from the pendulum.

Assumptions

- The time limit associated with each experiment is enough to show the potential of the RGP package.
- The probabilities associated with the selection of individuals involved with the genetic operations are efficient for the specific experiments.
- The probabilities associated with the genetic operators themselves are appropriate for the experiments underwent.

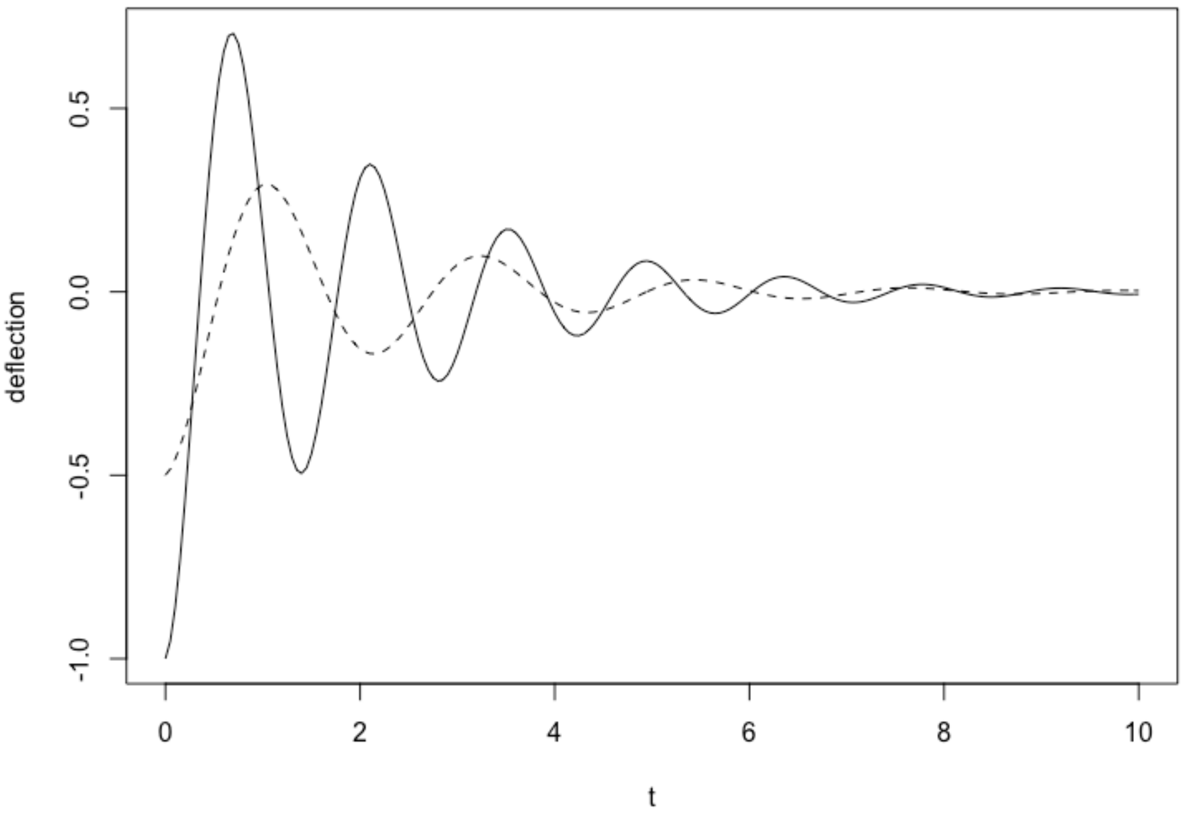
Results

After running the first experiment and plotting the results against the sin function, the graph below was created.

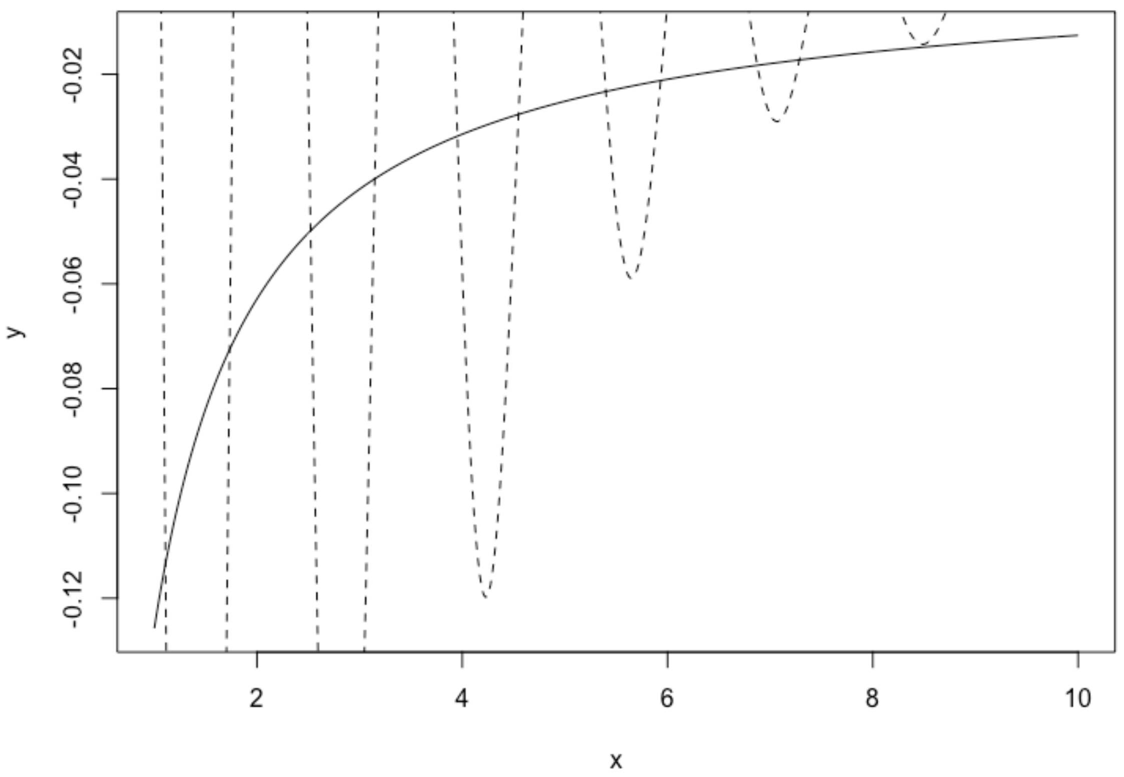


The solid line seen represents the sin function, and the dotted line represents the polynomial created through the genetic programming functionn in R. As is apparent, the RGP package did an excellent job at finding a polynomial that closely fits the sin function, in only a span of five minutes.

With the second experiment, the first task was to create a pendulum. Below is a graphical representation of two pendulums created using a formula for a damped pendulum.



The solid line is the pendulum that created the data used with the symbolic regression function in the RGP package. After the selecting the most fit individual from the symbolic regression run, the graph below was created, comparing the simulated data to the most fit individual.



The dotted line in the above graph represents the simulated data, and the solid line is the model created by the symbolic regression function. The symbolic regression did a good job at creating a model that loosely fits the data given.

Resources

Flasch, Oliver. "A Friendly Introduction to RGP." *Environmentally-Friendly Product Development* (n.d.): n. pag. *CRAN R Project*. RSymbolic Project, 7 Aug. 2014. Web. 1 Dec. 2016. <https://cran.r-project.org/web/packages/rgp/vignettes/rgp_introduction.pdf>.

Holland, John H. "Genetic Algorithms - John H. Holland." *Genetic Algorithms - John H. Holland*. N.p., n.d. Web. 10 Oct. 2016.

"Darwin's Theory Of Evolution." *Darwin's Theory Of Evolution*. All About Science, n.d. Web. 10 Oct. 2016.

Fogel, David B. *Evolutionary Computation: Toward a New Philosophy of Machine Intelligence*. New York: IEEE, 1995. Print.

Heitkotter, Jorg, and David Beasley. "The Hitch-Hiker's Guide to Evolutionary Computation." *Hitch-Hiker's Guide to Evolutionary Computation*. N.p., 12 Apr. 2001. Web. 10 Oct. 2016.

Koza, John R. "Genetic Programming." *Genetic-programming.org-Home-Page*. N.p., 8 July 2007. Web. 10 Oct. 2016.