STAT 243 Final Project

Lulude Sun, Wei-Hsiang Sun, Chloe Zhang

December 2021

GitHub username:
xinyuz/GA

1 Introduction

Genetic Algorithms, referred to as GAs in this paper, simulates the process of natural selection and reproduction. Species that are able to adapt to environmental changes and survive with high probability are selected to reproduce the next generation with superior characteristics, according to the idea of "survival of the fittest". Each individual in the original population is being represented as a string of integers where each integer represent the quality of a certain characteristic. These strings are analogous to genetic information such as chromosomes. Individuals within a population compete for resources and the fittest, meaning having the most resources and the highest probability of surviving, are able to produce more offsprings than the others. Some individuals, the ones deemed as least fittest, are being filtered out meaning that they failed to obtain resources and mates. Offsprings that are reproduced oftentimes are even better than their parents, thus more suitable and fit in the new generation. Each successive population is more superior and fitted in the environment.

Our project aims to implement these genetic algorithms by creating an R package for variable selection in regression and optimization problems, including both linear regression and GLMs. Our algorithms include initialization, evaluation, selection, crossover, and mutation. We will discuss the detailed implementation of each genetic algorithm in section 3. In section 4, we will have a thorough discusssion of the process of testing our functions. In the last section, we will have a final discussion of the testing results.

2 Contributions

- Chloe: Wrote *selection*, formal testings, final report writing, documentations, helper functions, test ran the entire package, result visualization

- Lulude: Wrote *crossover* function, wrote and built package, formal testings, final report writing, documnetation, helper functions, finalized report
- Wei-Hsiang: Wrote *initialization* function, *mutation*, *fitness*, *select* function, formal testing, final report writing, documentations, helper functions

3 Algorithms

3.1 Fitness

The *fitness* function takes the population as an input and produces an output that is an indicator of how fit is the input with respect to the desired variable. In GAs, the solutions are represented as strings of binary numbers to represent each individuals' chromosomes. Because we want to select the most fitted individuals to further produce offsprings in the selection and crossover process, we would need to know how fit each individual is. We have to test the solutions and come up with the best solutions for regression problems including both linear regression and GLMs. Therefore, for each desired variable/characteristics, individuals need an indicator that shows how close they are or how fit they are to the corresponding variable/characteristics. The scores are then generated through the fitness function.

3.2 Selection

The *selection* function takes in the fitness or the rank of the current parent generation as input and returns a matrix in size of $p \times 2$, where p denotes the population size. Each row of the matrix is a pair of parents, recording the index of both parents. Totally, p pairs of parents will be generated by the selection function, as parents with high fitness to perform the *crossover* function. There are mainly four methods for selection phase – fitness proportionate selection, tournament selection, rank selection, and random selection. As random selection has no selection pressure towards fitter individuals and this strategy is usually avoided, we only coded for the other three methods, which have various selection pressure and thus different convergence speed so that users could customize selection method according to their data set and requirements.

As for fitness proportionate selection, we coded three methods all taking fitness as inputs and returning a matrix in size of $p \times 2$ – *onepropselection twopropselection* and *sus*. The *onepropselection* function selects the first parent with probability proportional to fitness and selects the other randomly. The *twopropselection* function selects both parents proportionally according to fitness. And the *sus* function denotes stochastic universal sampling which places two fixed points on a Roulette Wheel. After rolling the wheel and let it stop randomly, two parents will be selected proportionally to fitness.

As for rank based selection, we coded *rankselection* function by which one parent will be selected with probability proportional to rank. For example, for parent a, b, c, they are ranked as 2, 1, 3. Then a higher rank of parent will have a larger probability of being selected. Thus their rank proportion will be $\frac{2}{6}, \frac{3}{6}, \frac{1}{6}$. The other parent will be selected randomly.

As for a K-way tournament selection, we coded *tournament* function which takes an int K and fitness as inputs. This method selects K individuals from the population at random and select the best out of these to become a parent. The other parent is selected randomly.

3.3 Crossover

The *crossover* function takes the genetic information of two parents whom are selected from the *selection* function to create new offsprings by crossing over their chromosomes. The crossover process is used to vary the structures of chromosomes from the previous generation to create a new generation as a genetic operator. This genetic operator acts during the process of sexual reproduction where we selected parents from the original population pool based on tournament selection to produce superior offsprings in order to create a new superior population pool. The method we used is single point crossover. A single random point on the parents' chromosomes is selected during the crossover process of each pair of parents. All chromosomes beyond the selected point is being swapped between each pair of parents' chromosome information. An offspring is therefore produced with the newly crossed over chromosomes.

3.4 Mutation

The *mutation* function takes the offsprings from crossover as input, and randomly flip bits with prior designated probability μ. The mutation process is used to maintain diversity of each population, like biological mutations that happens as one generation passed down its genetic information to the next. Because we have selected superior parents to produce more offpsrings than the others, there are possibilities that the produced offsprings are very similar to each other. By doing mutation to mimic the natural biological mutation process, we try to minimize the probability of the chromosomes of each individuals being too similar. The method we used for mutation is single point mutation. A single random point is selected and the genetic information at the selected point is being mutated to produce a new chromosome.

3.5 Main Function and Converging Criterion

The *select* function combines the above functions into one function. A new individual is selected from the parent population randomly if one of the indi-

viduals have a binary gene containing all zeros, to prevent error in the algorithm.

The converging criteria is set according to the

1. the homogeneity between the offspring population after mutation and its parent population
2. the homogeneity within the offspring population

The homogeneity between generations is tested by checking whether the absolute difference in AIC between the current population and the parent population is less than a fraction of the current best AIC.

The homogeneity within the current generation is tested by checking whether the absolute difference between the best AIC and the average AIC is less than the fraction of the median AIC in the population.

The best AIC values in the converging population and the binary vector corresponding to whether a variable is included (1) or not (0) is then reported in a list.

4 Testing

In order to test the validity and funcionality of our functions, we did testing for each functions and the results are shown in the *tests* and *testthat* folders. We conducted a set of test for the basic functions that we need for the main select function. For each of the functions, we used the testthat function to test the function and compared the output to some known truths. For example, for crossover's formal testing, we randomly generated a population and chromosomes information. By using the *crossover* function, we were able to generate a new population. We compared the dimension of the output to the known truth which is the size of the population since we should be creating a new crossovered population that is equal in size to the original parent population. After we performed testing for each of the individual tests, we performed formal testing on the entire genetic algorithms by using examples. In the first test example, we used a real dataset while in the second testing example, we randomly generated a dataframe to test. We were able to successfully passed through all the tests to ensure the validity of all functions. The *select* function tests the functionality of the entire genetic algorithm process and the testing process is contained in the *testthat* folder under *test_select.R* file.

5 Results

We ran our function on a real dataset as well as a randomly generated dataframe and we were able to see improvements that are shown through the decreased AICs. We will illustrate the results of genetic algorithms in the following graphs.

Figure 1: Best AICs Trends of GLM Models

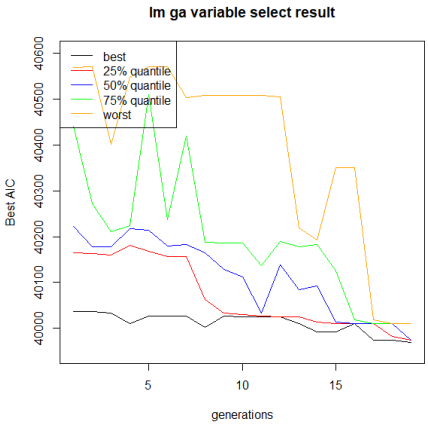

Figure 2: Best AICs Trends of LM Models

We used AIC (Akaike information criterion) to measure the performance of our model. AIC is an estimator of prediction error therefore a low AIC value essentially indicates a well fitted model. As we can see in the following graph, we see an decreasing trend in the values of AIC, which shows that the model is becoming more well-fitted.

References

[1] Mallawaarachchi, Vijini. "How to Define a Fitness Function in a Genetic Algorithm?" Medium, Towards Data Science, 10 Nov. 2017, https://towardsdatascience.com/how-to-define-a-fitness-function-in-a-genetic-algorithm-be572b9ea3b4.

[2] "Genetic Algorithm." Wikipedia, Wikimedia Foundation, 22 Oct. 2021, https://en.wikipedia.org/wiki/Genetic_algorithm.