Genetic Algorithm-R Package Final Report

David Chen, Qi Chen, Emily Suter and Xinyi(Cindy) Zhang

December 14, 2017

Abstract

Genetic Algorithm (GA) is a search-based optimization technique based on the principles of Genetics and Natural Selection. It is frequently used to find optimal or near-optimal solutions to difficult problems which otherwise would take a lifetime to solve. In this report, we first introduce how we set up a Genetic Algorithm and tested its functionality in Section 1. We then describe the testing procedure carried out in Section 2. In section 3, we apply our GA algorithm to two example datasets, one set of simulated data and one real world dataset. Contributions of each team member are collected in section 4.

Github Username: dchen49

1 Introduction to Our Genetic Algorithm

Our package is comprised of 4 major functions: *select()*, *regress()*, *mate()*, and *evolve()*.

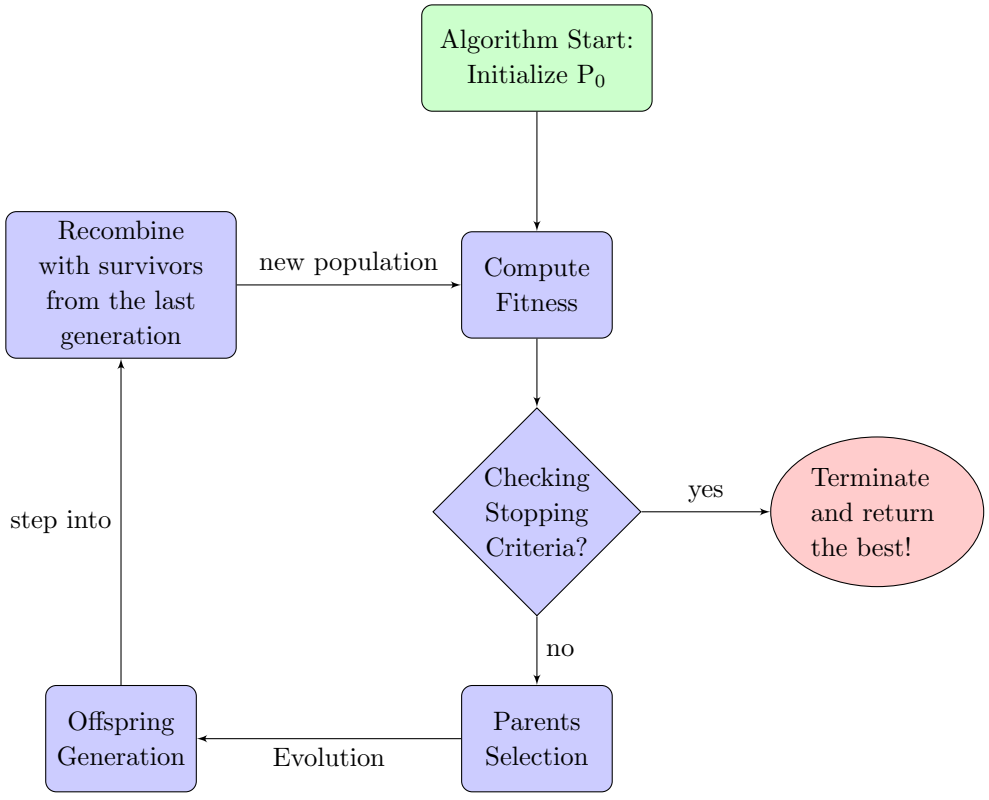
select() is the main, exported function which takes in all user arguments and wraps all other functions. *select()* also loops over generations, checks stopping criteria, and creates the output list object with 4 components: *optimum*, *fitPlot*, *fitStats*, and *GA*. *optimum* contains the names of the selected variables, the fitness value achieved, and the regression object. *fitPlot* is a ggplot of the mean, median, and maximum fitness per generation and is generated from the table in *fitStats*. *GA* contains the elite subset of genotypes and all fitness values for each generation.

regress() calculates the fitness of the regresssion model using a particular group of covariates and returns fitness metric as a single number to be maximized. In our GA implementation, *regress()* is called in an apply loop to operate over the entire genotype poulation and return a vector of fitness values.

mate() selects parents using one of 4 selection methods: tournament selection, linear ranking selection, exponential ranking selection, or roulette wheel selection. Each method is defined as a sub-function which is then called by *mate()*. The output is a genotype population of parents to be passed into *evolve()*.

evolve() performs crossing-over of genotypes and mutates single alleles by calling subfunctions *singlecrossover()*, *multiplecrossover()*, and *mutate()*. It returns a population of altered genotypes to be combined with elite survivors to produce the next generation.

To better introduce our genetic algorithm, the following Figure 1 diagrams the workflow



2 Testing

We have tested the input and output format for each individual functions in our GA algorithm, as well as the stopping criteria. To test the function select() as a whole, we tested its ability to find max fitness using lm/glm and AIC/BIC on a manufactured dataset.

To investigate whether our GA algorithm can attain the same optimum as global search for all possible genotypes, we create another dataset to test this. We generated the data as follows:

Consider number of variables $p = 5$, and we generate covariates $X = (X_1, \cdots, X_5)^{\mathrm{T}}$ from different distributions, where $X_1 \sim N(0, 25)$, $X_2 \sim \mathrm{Unif}(0, 1)$, $X_3 \sim \mathrm{Poisson}(1)$, $X_4 \sim \exp(2)$, $X_5 \sim \mathrm{Gamma}(10, 1)$. The responses Y are generated by simply averaging X_1 and X_3, i.e. $Y = \frac{X_1+X_3}{2}$. Apart from finding optimal genotype via Genetic Algorithm, we also compute all the fitness values for all 32 genotypes to see what the exact global optimum is and thus the "best" genotype. Since the reponse Y is only related to variables X_1 and X_3, we hope that both the global search approach and the GA algorithm can only select these two features but not others. Additionally, we also hope that the results returned by these two methods are consistent. Comparison results are presented below in Table 1.

Table 1: Comparison of optimal fitness values with respect to global search and GA algorithm

Table 2: Global Search

	AIC	BIC
lm	19963.50	19945.41
glm	20175.50	20160.68

Table 3: GA Algorithm

	AIC	BIC
lm	19963.50	19945.41
glm	20175.50	20160.68

One can see from the results in Table 1 that for each combination of fitted model and fitness criteria, the GA algorithm and global search method give exactly the same results. In other words, our GA algorithm does find the global maximum fitness value, i.e global minimum value for AIC or BIC.

Next, we present the genotypes returned by both global search and GA algorithm in Table 4.

Table 4: Comparison of optimal fitness values with respect to global search and GA algorithm

Table 5: Global Search

	AIC	BIC
lm	11100	10100
glm	10100	10100

Table 6: GA Algorithm

	AIC	BIC
lm	11100	10100
glm	10100	10100

In summary, our GA algorithm can find the global optima, results of which are consistent with that from global search.

3 Application

In this section, we considered two applications of our genetic algorithm, one on the dataset generated from a linear regression model, which aims to evaluate whether the designed algorithm can successfully select those important features, given known relevant variables combined with some noise terms. Another application is based on the baseball data collected from the textbook "Computational Statistics" Givens and Hoeting (2013). Details will be illustrated as follows.

3.1 Application on Data Generated from Linear Regression Model

In this section, we will first introduce how we generate the covariates and responses for evaluating our genetic algorithm on feature selection. We first generate covariates from the multivariate normal distribution $N_p(\mathbf{0}, \Sigma_x)$, where the (j, j') entry of Σ_x satisfies

$$\Sigma_{x(j,j')} = 0.5^{|j-j'|}, \text{ for } 1 \leq j, j' \leq p.$$

Setting sample size $n = 300$ and feature dimension $p = 10$, we then fit the linear regression model

$$Y = X\beta,$$

where $\beta = (\beta_1, \cdots, \beta_6)^{\mathrm{T}}$ is generated from univariate normal distribution $N(3, 25)$. Moreover, we add some noise $\epsilon = (\epsilon_1, \cdots, \epsilon_n)$ to the covariates generated as above. ϵ_i, for $1 \leq i \leq n$, is generated from $N_{20}(\mathbf{0}, \Sigma_x)$, which has the same parameter setting as X except the dimension. We then combine X and the noise terms together. The resulting covariates matrix is given by

$$\tilde{X} = (X|\epsilon).$$

Apply our genetic algorithm to fit a linear model regressing $Y \in \mathbb{R}^n$ on $\tilde{X} \in \mathbb{R}^{n \times 30}$. Given 10 relevant covariates and 20 noise terms, the designed GA algorithm gives the following results for feature selection. Results are presentesd in Table: 7

Table 7: Selected Genotypes

Var	x1	x2	x3	x4	x5	x6	x7	x8	x9	x10	n1	n2	n3	n4	n5	n6	n7	n8	n9	n10	n11	n12	n13	n14	n15	n16	n17	n18	n19	n20
Gene	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1	0	1	0	1	0	1	0	1	0	1	1	1	1	1	1

One can see that our GA algorithm can select all the relevant features, but will also include some irrelevant noise terms denoted in red. Next, we will take a look at the coefficients of those noise terms from the regression model and further investigate whether the "noise" terms are significant or not. Results are presented in Table 8 and Table 9.

Coefficients of Relevant Variables:

Table 8: x-variables

x1	x2	x3	x4	x5	x6	x7	x8	x9	x10
8.70	0.98	8.46	-0.74	1.87	-1.31	5.47	5.07	0.61	4.99

Coefficients of Noise terms:

Table 9: n-Noise terms

n1	n2	n3	n4	n5	n7	n9	n11	n13	n15	n16	n17	n18	n19	n20
3e-15	-4e-15	-2e-15	4e-15	-2e-15	-1e-15	-5e-15	5e-15	-1e-15	1e-15	5e-15	-1e-14	4e-15	-5e-15	3e-16

One can see from the table that although we have included some noise terms, but values of those coefficients are quite small, which indicates that these noise terms are not significant.

We then repeated the previously described variable selection 50 times. Figure 1 plots the summed absolute value of the coefficients for all selected variables. One can clearly see the same pattern described previously, where the weights of the noise terms are negligible compared to those of the valid x terms.

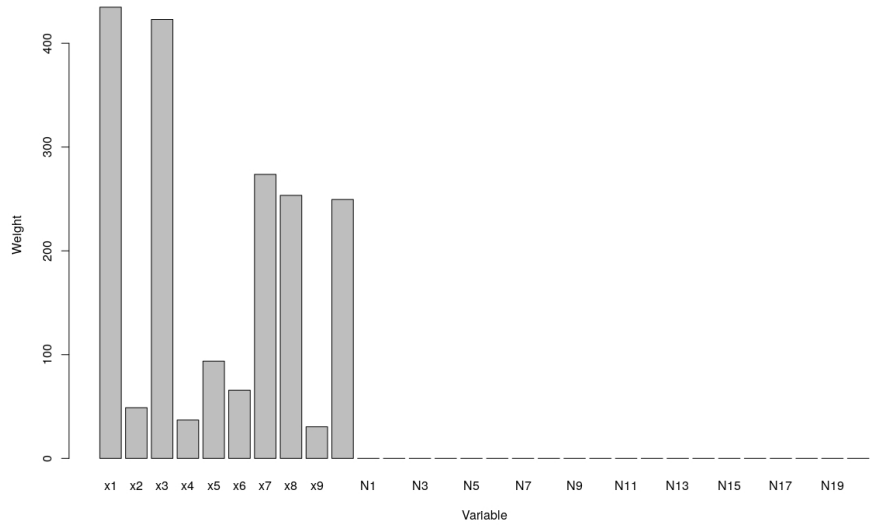


Figure 1: Summed Weights of Regression Variables

3.2 Application on Baseball Data

We next test our genetic algorithm on a real data set to demonstrate the ability to select an optimal variable subset. A data set of baseball player statistics and salary numbers was obtained via the website for Givens and Hoeting (2013).

The data set contains 27 different statistics (such as hits and on-base percentage) for 337 players in the 1991 baseball season. Additionally, the data set contains the salaries for the same 337 players in the 1992 season. We used our genetic algorithm to select player statistic(s) that most influence that players salary in the following year.

We tested our algoritms performance on combinations of different fitness criteria (AIC, BIC) and selection method (tournament, exponential ranking, linear ranking, roulette wheel). All other input parameters were held constant at maxGen = 500, minGen = 50, population = 500, pMutate = 0.1, crossParams = c(0.8, 1), and eliteRate = 0.1. The fitness values are shown in Table 10.

Table 10: Maximum Fitness Value

Selection Method	LM, AIC	LM, BIC	GLM, AIC	GLM, BIC
Tournament	5376.012	5409.318	5375.850	5412.395
Linear	5376.354	5409.318	5375.850	5409.318
Exponential	5378.926	5414.315	5379.316	5422.321
Roulette	5376.365	5417.421	5376.353	5417.723

In this simulation, the minimum (i.e., best) fitness value was produced using GLM as the model, AIC as the fitness criteria, and tournament or linear ranking as the selection method. Generally, the difference between AIC and BIC was greater than the variance across selection methods.

The exponential ranking selection method converged the fastest, followed by tournament selection, as shown in Table 11.

Table 11: Iterations to Reach Maximum Fitness Value

Selection Method	LM, AIC	LM, BIC	GLM, AIC	GLM, BIC
Tournament	64	69	62	63
Linear	75	96	80	88
Exponential	53	52	53	53
Roulette	84	85	94	111

In general, the top genotype returned when using BIC fitness criteria had fewer variables than those using AIC. This makes sense since BIC has a greater penalty for higher numbers of variables: with AIC, the penalty is $2p$, whereas with BIC the penalty is $ln(n)p$.

The number of variables returned in the best genotype of each combination is shown in Table 12.

Table 12: Number of Variables in Best Genotype

Selection Method	LM, AIC	LM, BIC	GLM, AIC	GLM, BIC
Tournament	10	6	12	6
Linear	10	7	15	6
Exponential	11	7	16	8
Roulette	14	7	14	7

Regardless of method, model, or criteria, Strength of Schedule (*sos*), Runs Batted In (*rbis*), Free Agency (*freeagent*), and Arbitration (*arbitration*) are all included in the top genotypes. Free Agency and Arbitration had very high regression coefficients, hovering around 1300 and 850, respectively; conversely, the coefficients of RBIs and SOS were much lower, about 25 and -12. We hypothesize that this is because a player that becomes a free agent and gets signed to a new team will likely negotiate a large salary contract with their new team; players that enter free agency and don't get signed aren't included in this data set.

4 Group Member Contributions

All members worked together to integrate functions, debug/execute testing, and assemble the report document (i.e., fought with Latex as a team).

David Chen: Created and set up package scaffold; wrote the *select()* function; conducted analysis included in Section 3.1.

Qi Chen: Wrote the *evolve()* function (mutation and crossover) and the corresponding test functions.

Emily Suter: Wrote the *regress()* function and the corresponding test functions; conducted baseball analysis in Section 3.2.

Xinyi(Cindy) Zhang: Wrote the *mate()* function (including the selection sub-functions) and the corresponding test functions; wrote global search tests included in Section 2; conducted analysis included in Section 3.1; created report scaffold.

References

Anupriya Shukla, H. M. P. and Mehrotra, D. (2015). Comparative review of selection techniques in genetic algorithm .

Givens, G. H. and Hoeting, J. A. (2013). *Computational Statistics*. Wiley.