Predictive Regression Model Selection Using Genetic Algorithms

Matthew Keeran

Abstract

Finding the best regression model out of all possible combinations of predictor variables can be a difficult task for a large number of predictors. Oftentimes you can only try a subset of all possible models. Forward and backward selection methods are biased to particular regions of the search space (of all possible models). Genetic algorithms on the other hand allows one to search from anywhere within the search space, both globally and locally. This paper examines a genetic algorithm for regressors selection with transformation called GARST [1] and extends its implementation to include an additional type of selection, crossover and fitness function. The algorithm is implemented in python and its performance is compared to forward selection in the statistical software R.

Sections

0 Introduction

Predictive regression models use **predictor** variables to predict a **response** variable. In an increasing number of contemporary settings there are a large number predictor variables (> 20) that are thought to be useful in predicting a response variable but it is not known which subset of variables specifically are the most useful out of all possible combinations. This problem is known as **model selection**. This also is an optimization problem.

With **p = the # of predictor variables**, the total search space (of all possible models) for predictors of the form x^1 (with x being an arbitrary predictor variable) is $\mathbf{2^p}$ [2]. With p = 15 predictor variables, thats a total of 2^{15} = 32,768 models to choose from, with p = 20, thats 2^{20} = 1,048,576 possible models to choose from. With a small number of predictors an exhaustive search of all possible numbers is possible (subset selection), however with a sufficiently large p, in practice only a small subset of the total search space is feasible within a reasonable amount of time.

Common methods for traversing the search space of possible models are **forward** and **backward selection**. **Forward selection** begins by fitting all possible models using a single predictor. Once the best model using a single predictor is found, then all possible models adding a single additional predictor are fitted and the best is chosen (thus, each model created by adding an additional variable are nested models of the previous layer). This process is repeated until a certain threshold for an acceptable model has been met. This threshold can take many forms using various **statistical criteria**.

Backward selection is essentially the same process, but in reverse. First a model using all predictors is fitted. Then all possible models created by removing a single predictor are fitted and the best model of (p-1) predictors is chosen. Again this process repeats until an acceptable model has been found.

Both of these methods are **biased towards particular regions of the search space** as they are a nested incremental approach. Which is to say both methods will never try certain combinations of predictors even though their combination may in fact produce a better model. This also makes them susceptible to getting stuck at local optima in multi modal distributions.

Genetic algorithms on the other hand, operate via generating a random initial population of possible solutions. This allows them to theoretically start the search from anywhere within the search space. The **fitness** of each solution (how good of a model it is) is then calculated. **Selection** of members of the population of solution are then chosen to **crossover** parts of their solution to create children that then can under go **mutation**(s) and thus creates the next generation. This is repeated until either a fitness threshold is met or a fixed number of generations has been reached.

The following **statistical criteria** are considered.

$$\mathrm{AIC} = \frac{1}{n\hat{\sigma}^2}\left(\mathrm{RSS} + 2d\hat{\sigma}^2\right)$$

$$\mathrm{BIC} = \frac{1}{n}\left(\mathrm{RSS} + \log(n)d\hat{\sigma}^2\right)$$

$$\text{Adjusted } R^2 = 1 - \frac{\mathrm{RSS}/(n-d-1)}{\mathrm{TSS}/(n-1)}$$

[2]

1 Representation

This genetic algorithm uses a binary string (bitstring) as its representation. **2 bits** are used for whether to include a predictor in the model and if so what mathematical **transformation** of the variable to perform. **3 bits** are used to represent the **exponent** of the transformation.

00	**Not Used**
01	**x^(exp/2)**
10	**(ln(x))^(exp/2)**
11	**(e^x)^(exp/2)**

exp

000	**-6**
001	**-4**
010	**-2**
011	**-1**
100	**1**
101	**2**
110	**4**
111	**6**

That is, every predictor is represented by **5 bits**.

Example: **01 101 = x_1^(2/2)**
$Y = B_0 + B_1 x_1$

Example: **01101 11111 = x_1^(2/2) + (e^(x_2))^(6/2)**
$Y = B_0 + B_1(x_1) + B_2(e^{(x_2)})^3$

This means that the **bitstring length** to be used is len(model/solution) = **5 x p** (the **max** number of possible **predictors** being used).

2 GA Description

The initial population of bitstrings gets generated, the bitstring gets **decoded**, the models get fitted according to the **ordinary least squares** function in the **statsmodels python** library, the **statistical criteria** for each model gets saved and compared using a **fitness function**, **selection** of parents occurs, **crossover** is performed, and **mutation**s are applied.

2.1 Fitness

The fitness functions tested to date include **minimizing** the **AIC and BIC** criteria as well as **maximizing** the **adjusted R^2**. All three were used independently.

2.2 Selection

Genetic algorithms **recombine solutions** (**bitstrings**) iteratively, for a number of **generations**, by first **selecting individuals**/solutions from the **population**.

Both **uniform selection** as well as **proportional selection** have been implemented. With uniform selection, all members of the population have an **equal chance of being chosen** for crossover. Proportional selection is where every member of the population has a probability of being selected that is based on their **relative fitness** of the entire population.

2.3 Crossover

Single point crossover gets performed with a **probability** of **0.8**, which means there is a 20% chance that crossover does not occur, in which case the parents take the place of the children in the next generation. Crossover occurs randomly along the bitstring every time.

Example (**Single Point** after the **5th bit**.)

Parent 1: 01101 11111

Parent 2: 10100 01010

Child 1: 01101 01010

Child 2: 10100 11111

Selection & crossover repeat until the original population size has been reached.

Uniform crossover is also performed, where every bit of 2 parents have the same probability of crossover. Uniform crossover exists to reduce the bias of the tails of the bitstrings having a lower probability of crossover.

2.4 Mutation

Although the original paper [1] uses a mutation rate based on the number of bits used to represent each solution, this seems unjustified especially in the case of a small or large number of predictors. Instead a fixed valued of 0.05 will be used as that has shown to produce good results in a number of different implementations of genetic algorithms. In the future a value of 0.01 will also be tested for the same reason.

The best two models in every generation get a guaranteed spot in the next generation, this is known as **elitism**.

After the next generation gets created, this loop of fitting the models, evaluating their fitness, selecting parents, performing crossover and mutation repeats until a fixed number of generations has been reached

3 Experiments

3.1 Execution Environment

Ubuntu 64 bit running in Virtual Box
on Windows 8.1
Dual-Core Intel i5 2.7 GHz
2 GB RAM
Python 2.7

3.2 Dataset

"Body Fat Measurement" (Johnson 1996)
2 responses 16 predictors, 252 observations (men)

% Body Fat (Brozek)
% Body Fat (Siri)
Density (gm/cm^3)
Age (yrs)
Weight (lbs)
Height (inches)
Adiposity index = Weight/Height^2 (kg/m^2)
Fat Free Weight = (1 – fraction of body fat) * Weight
Neck Circumference (cm)
Chest Circumference (cm)
Abdomen Circumference (cm)
Hip Circumference (cm)
Thigh Circumference (cm)
Knee Circumference (cm)
Ankle Circumference (cm)
Extended Biceps Circumference (cm)
Forearm Circumference (cm)
Wrist Circumference (cm)

Though there are 2 **response** variables relating to different measurements of % body fat (Brozek and Siri), only **Brozek**'s measurement will be used based on [3]. That leaves **16 predictor** variables.

3.3 Scope of Experiments

I first repeat the original experiments described in [1] with one of the original datasets and the original GA parameters. I then test proportional selection and uniform crossover and compare their run times and the best models. I plan in the future to expand my tests to include another dataset with a larger number of predictors and observations as well as with my own fitness function.

3.4 Results

Runs were done with and without printing extensive details of the best model for each generation to show roughly what % improvement can be expected. The best model shown is the one with the details.

Runs 1 &2

Dataset	Body Fat Measurement (Johnson 1996)
Generations	1000
Population	100
Selection Type	Uniform
Crossover Type	Singe Point
Crossover Probability	0.8
Mutation Rate	0.05
Elitism	1
Fitness Function	**AIC**
Elapsed Time (no prints):	10 min 29.35 seconds
Elapsed Time: (w/ prints)	11 min 32.37 seconds

Best model Found Used 12/16 variables:

% Body Fat =

Coefficient	**Variable**
-1028.6544	**Intercept**
-1.5595	**(e^Density)^(3)**
234.8516	**ln(Weight)^(1/2)**
2.541×10^{-6}	**(Height)^(3)**
-0.0564	**ln(Adiposity)^(3)**
1251.4836	**ln(FF weight)^(-1/2)**
-12.3932	**(e^Neck)^(-1)**
-280.3596	**(Chest)^(-1)**
-674.7705	**ln(Abdomen)^(-3)**
-255.7940	**ln(Thigh)^(-3)**
-4.121×10^{7}	**(e^Ankle)^(-1)**
0.4938	**(Forearm)^(1/2)**
0.0001	**(Wrist)^(3)**

AIC	**464.9**
BIC	503.7
Adjusted R^2	0.994

Additionally 6 of the predictors had **p values** > 0.05

ln(Adiposity)^(3)	0.162
ln(FF weight)^(-1/2)	0.126
(Chest)^(-1)	0.690
ln(Abdomen)^(-3)	0.053
(e^Ankle)^(-1)	0.690
(Forearm)^(1/2)	0.076

Runs 3 &4

Dataset	Body Fat Measurement (Johnson 1996)
Generations	1000
Population	100
Selection Type	Uniform
Crossover Type	Singe Point
Crossover Probability	0.8
Mutation Rate	0.05
Elitism	1
Fitness Function	**BIC**
Elapsed Time (no prints):	10 min 09.42 seconds
Elapsed Time: (w/ prints)	11 min 36.25 seconds

Best model Found Used 8/16 variables:

% Body Fat =

Coefficient	**Variable**
-1044.8943	**Intercept**
-1.5872	**(e^Density)^(3)**
243.9165	**ln(Weight)^(1/2)**
1.8×10^{-12}	**(e^Height)^(-2)**
-2.727×10^{-09}	**(e^Adiposity)^(-2)**

1222.7885	**ln(FF weight)^(-1/2)**
-6.507 x 10^{-12}	**(e^Neck)^(1/2)**
-1.769 x 10^{4}	**(Abdomen)^(-2)**
1767.8647	**(Ankle)^(-3)**

AIC	468.5
BIC	**486.2**
Adjusted R^2	0.994

Additionally 3 of the predictors had **p values** > 0.05

(e^Height)^(-2)	0.217
(e^Neck)^(1/2)	0.195
(Ankle)^(-3)	0.622

Runs 5 & 6

Generations	1000
Population	100
Selection Type	Uniform
Crossover Type	Singe Point
Crossover Probability	0.8
Mutation Rate	0.05
Elitism	1
Fitness Function	**Adjusted R^2**
Elapsed Time (no prints):	10 min 30.32 seconds
Elapsed Time (w/ prints):	11 min 45.49 seconds

Best model Found Used 12/16 variables:
% Body Fat =

Coefficient	**Variable**
332.9023	**Intercept**
-1.4931	**(e^Density)^(3)**
2.105 x 10^{-25}	**(e^Age)^(-3)**
54.4051	**ln(Weight)^(1)**
-6.453 x 10^{-5}	**(Adiposity)^(3)**
-256.4292	**ln(FF weight)^(1/2)**
2.161 x 10^{-11}	**(e^Neck)^(1/2)**
-311.3412	**(Chest)^(-1)**
-657.9216	**ln(Abdomen)^(-3)**

9.8886	**ln(Thigh)^(1/2)**
0.6878	**(Bicep)^(-3)**
0.2064	**ln(Forearm)^(2)**
7.529 x 10^{-10}	**(e^Wrist)^(1)**

AIC	453.9
BIC	492.7
Adjusted R^2	**0.9943443523**

Additionally 3 of the predictors had **p values** > 0.05

(e^Age)^(-3)	0.093
(Bicep)^(-3)	0.285
ln(Forearm)^(2)	0.065

Run 7

Generations	1000
Population	100
Selection Type	**Proportional**
Crossover Type	Singe Point
Crossover Probability	0.8
Mutation Rate	0.05
Elitism	1
Fitness Function	**AIC**
Elapsed Time (w/ prints):	10 min 2.26 seconds

Best model Found Used 10/16 variables:
% Body Fat =

Coefficient	**Variable**
125.2081	**Intercept**
-39.9172	**(e^Density)^(1)**
-1231.8989	**ln(Weight)^(-0.5)**
1220.1265	**ln(FF Weight)^(-0.5)**
-0.0075	**(e^Neck)^(-0.5)**
-592.5908	**ln(Abdomen)^(-3)**
51.2522	**ln(Hip)^(-1)**
-41.6573	**ln(Thigh)^(-2)**

0.0009	**(Knee)^(2)**
-15.3339	**ln(Forearm)^(-2)**
7.22×10^{-10}	**(e^Wrist)^(-1)**

AIC	**477.9**
BIC	513.2
Adjusted R^2	0.994

Additionally 3 of the predictors had **p values** > 0.05

ln(Hip)^(-1)	0.227
ln(Thigh)^(-2)	0.277
ln(Forearm)^(-2)	0.291

Run 8

Generations	1000
Population	100
Selection Type	Uniform
Crossover Type	**Uniform**
Crossover Probability	0.8
Mutation Rate	0.05
Elitism	1
Fitness Function	**AIC**
Elapsed Time (w/ prints):	10 min 3.69 seconds

Best model Found Used 11/16 variables:

% Body Fat =

Coefficient	**Variable**
-764.7337	**Intercept**
-1.5431	**(e^Density)^(3)**
0.7107	**ln(Age)^(0.5)**
52.6389	**ln(Weight)^(1)**
2.185×10^{-6}	**(Height)^(3)**
-3.097×10^{-5}	**(Adiposity)^(3)**
1256.4456	**ln(FF Weight)^(-0.5)**
7.246×10^{-7}	**(Chest)^(3)**
-1.053×10^{6}	**(Abdomen)^(-3)**
-46.1505	**ln(Thigh)^-0.5)**
-466.4869	**(Forearm)^(-2)**
2.1168	**ln(Wrist)^(1)**

AIC	**460.9**
BIC	503.3
Adjusted R^2	0.994

Additionally 5 of the predictors had **p values** > 0.05

ln(Age)^(0.5)	0.311
(Height)^(3)	0.125
(Chest)^(3)	0.084
(Forearm)^(-2)	0.084
ln(Wrist)^(1)	0.112

4 Conclusions

There are a number of important things to observe regarding this implementation. First of all, it tests a number of different transformations of predictor variables. Forward, backward and subset selection methods in R only examine all models with the predictor variable $x = x^1$, not all possible combinations of ln(x), e^x and x^y. Additionally, 8 different possible exponents were used for each of the 3 transformations implemented. This makes the search space for this algorithm immensely larger by orders of magnitude than the traditional search methods. That being said it on the other hand is potentially far more expressive mathematically.

Initially I was curious as to whether or not the genetic algorithm would have a preference towards simplistic mathematical transformations of the variables, and assumed that it might. This however is most certainly not the case.

(e^Density)^(exp) was included in all best models **ln(Weight)**, **Adiposity** occurred in all but one with different transformations and exponents, the **ln(FF Weight)^(+-1/2)** occurred in all runs, **Abdomen** also appeared in all models, though with no clear preference as to transformation.

It is interesting to note that of the best models found based on different criteria, 25 to 50% of the variables being used were had insignificant p values (> 0.05).

The algorithm also ran on average a 1 minute and 15 seconds slower with printing out the full summary of the best model for each generation, using 16 predictors and 252 observations.

Additionally, there was no significant difference in run time using proportional selection and uniform crossover, although I thought there might be.

I was expecting the GA to converge after 500 or so generations and not improve much further after that. However even though the rate at which it continued to find a better model slowed significantly after generation 200, it still managed to improve throughout all 1000 generations, surely a testament to the sheer scope of the search space.

It is important to notice, the run that used adjusted R^2 as the fitness function also found a model with a smaller AIC than the best model found when using the AIC as the fitness function. This implies correlation between AIC and adjusted R^2 and might mean that optimizing for one will essentially optimize for the other. Furthermore there is a significant degree of variability among runs even using the same parameter settings. In order to investigate this further I will in the future have the program print out more extensive generational details into a csv that will streamline analysis.

5 Extensions

After implementing and testing everything I initially set out to test at this point, the first extension to this work will be implementing muti-threading. There are a number of instances in which the same operation is being performed on every member of the population which is ripe for parallelization. This will likely provide the most significant improvement in terms of the runtime of the algorithm

In the future it will be interesting to implement a much more simplified representation than currently being used. The new representation would use one bit to represent whether a predictor would be included in the model, and if so it would take the form x^(exp), with 2 bits representing the possible exponents [1,2,3,4], for a total of 3 bits per predictor variable.

Additionally, 5 fold cross-validation will be implemented and a test mean square error will be calculated. This will serve as an alternative fitness function and although will require more computation and thus take longer per generation, will ideally produce even better models than with the current method.

Appendix

Python Libraries used:
sys
csv
math
time
heapq
random
datetime
numpy
statsmodels.api

References

1. Sandra Paterlini and Tommaso Minerva. Regression Model Selection Using Genetic Algorithms. In Recent Advances in Neural Networks, Fuzzy Systems & Evolutionary Computing, pages 19 - 27. Italy, 2010. ACM.

2. Gareth James, Daniela Witten, Trevor Hastie, Robert Tibshirani. An Introduction to Statistical Learning. 2013.

3. Guerra RS, Amaral TF, Marques E, Mota J, Restivo MT Accuracy of Siri and Brozek equations in the percent body fat estimation in older adults. In Journal of Nutrition Health and Aging, November 2010.