Introducing Evolutionary Computing in Regression Analysis

Olcay Akman
Department of Mathematics
Illinois State University
Normal, IL 61790 USA
oakman@ilstu.edu

ABSTRACT
A typical upper-level undergraduate or first-year graduate level regression course syllabus treats "model selection" with various stepwise regression methods. In this paper, we implement the method of *evolutionary computing* for "subset model selection" in order to accomplish two goals: i) to introduce students to the powerful optimization method of genetic algorithms, and ii) to transform a regression analysis course into a regression and modeling course without requiring any additional time or software commitment. Furthermore, we employ the Akaike Information Criterion (AIC) as a measure of model fitness instead of the commonly used measure of R-square. The model selection tool uses Microsoft Excel, which makes the procedure accessible to a very wide spectrum of interdisciplinary students, with no specialized software requirement. An Excel macro, to be used as an instructional tool, is freely available through the author's website.

Keywords
Genetic Algorithm, Model Selection, AIC

1. INTRODUCTION
Predictive model selection can be a difficult procedure for data sets with a large number of explanatory variables. Determining what variables best explain the system by exhaustive search becomes unreasonable as the number of variables increases. For example, over one billion possible models exist for data with 30 explanatory variables.

One area in which model selection is important is multiple linear regression. In ecological studies one of the commonly used methods for selection is step-wise regression, with forward or backward variable selection algorithms. These methods have been criticized for lacking the ability to truly pick the best model for several reasons [2, 12]. One problem is that the choice in which the variables enter the selection algorithm is not justified theoretically. In addition, the probabilities for the selection procedure are chosen arbitrarily, which may lead to a poorly selected model. One of the leading advocates of implementing cutting-edge methods in predictive model selection [11] provides a solid foundation for evolutionary computing. Finally, since these methods employ local search, it is unlikely that the global maximum set of variables will be found [9, 6, 7, 10].

Because of the drawbacks of the current model selection procedures, we propose to use a genetic algorithm to optimally determine the subset of variables for a multiple regression model. Genetic algorithms are a wise choice for this procedure. They are a global search tool and are not prone to the problems associated with stepwise selection being a local search. Genetic algorithms operate by considering many models at the same time; through selection, components of the best models come together to form the maximal model. We will now go through the basics of genetic algorithms. This is a brief explanation; a thorough one can be found in [5].

2. GENETIC ALGORITHMS
Genetic algorithms (GAs) are a set of optimization techniques inspired by biological evolution operating under natural selection. First developed by Holland [8], they have grown in popularity because of the ability of the algorithm to perform well on many different types of problems. In a genetic algorithm, possible solutions are coded using binary strings, which are called chromosomes. Each chromosome has a fitness value associated with it based on how well the string is optimizing the problem. During each generation, the time step of the algorithm, a population of chromosomes compete to have their "genes" passed on to the next generation. The selection step is used to pick the chromosomes for the next generation based on their fitness. Those selected enter the mating pool, where two chromosomes mate using crossover. During this phase, parts of each parent string are swapped to form two new chromosomes that have certain aspects of their parents. After crossover, mutation is applied. Mutation occurs with a small probability, and is defined as a change from 0 to 1 or 1 to 0 at a certain location in the binary string. Mutation allows the introduction of new "genes" that were either lost from the population or were not there to start with. Through successive generations, increasingly better chromosomes come to dominate the population and the optimal (or close enough) solution is realized.

3. AKAIKE INFORMATION CRITERION

A key parameter of a GA is a method to evaluate the fitness of a chromosome. In order to use a GA for model selection in multiple regression, a way to evaluate the chromosomes is needed. In other words, a method is needed to determine how well the subset explains the system.

Akaike introduced AIC in 1973 [1] as a measure of the complexity of a model. It measures the bias due to the estimation of the model from the true distribution based on the data. Additionally, AIC takes into account the number of parameters used to create the model. The formula for AIC is given as

$$AIC(k) = -2\log L(\hat{\theta_k}) + 2m(k), \tag{1}$$

where $L(\hat{\theta_k})$ denotes the maximum likelihood function, $\hat{\theta_k}$ is the maximum likelihood estimate of parameter vector θ_k under the model m_k, and $m(k)$ is the number of parameters in the model. The first term of AIC gives the lack of fit of the model, and the second term is a penalty for the number of parameters in the model. The model with the lowest AIC value is considered the best, because the model best determines the underlying stochastic process with the least number of parameters.

4. A GENETIC ALGORITHM FOR MULTIPLE LINEAR REGRESSION MODEL SELECTION

4.1 Background

The first step to implementing a genetic algorithm for any optimization problem is to determine a way to encode the problem into a binary string. In the case of multiple linear regression, we have q data points with n explanatory variables and one response variable. We wish to fit the data to

$$y = X\beta + \epsilon, \tag{2}$$

where y is an $n\times1$ response vector, X is an $n\times q$ matrix of the data points, β is a $q\times1$ coefficient matrix, and ϵ is an $n\times1$ error vector with entries from independent normal distributions ($N(0,\sigma^2)$ for all components). The dataset contains n explanatory variables. The encoding is done by creating a binary string with $n+1$ bits, where each bit represents a different parameter of the model. The additional parameter in the binary string is the intercept for the linear model. A parameter is included in the model if the value of the bit for that parameter is a 1 and is excluded if it is a 0. A quick example will help explain this procedure.

Suppose that we have a dataset where we are interested in what variables explain the reproductive fitness of a species of trees. The possible explanatory variables will include

1. Age of tree
2. Height of tree
3. Soil pH
4. Density of trees in the surrounding area
5. Average temperature of environment
6. Average rainfall of environment
7. Circumference of trunk
8. Longitude of environment
9. Latitude of environment
10. Prevalence of disease in environment.

In order to use a genetic algorithm in choosing the best model, each binary string will have 11 bits.The first bit is for the intercept and the following 10 correspond to the possible explanatory variables. For example, the string 10010111101 would code for a model which includes the intercept, soil pH, average temperature of environment, average rainfall of environment, circumference of trunk, longitude of environment, and prevalence of disease in environment. To further demonstrate this point, the string 00001000110 is a model that has no intercept, and includes density of trees in the surrounding area, longitude of environment, and latitude of environment (see Table 1).

Chromosome	Variables Included
10010111101	Intercept,3,5,6,7,8,10
00001000110	4,8,9

Table 1: Chromosomes and variables included by the model it represents

Once we have a method of encoding, we need a way to evaluate the binary strings in order to choose the best model. Although several choices for evaluation are available, this paper focuses on AIC. Since the model with the lowest AIC value is considered the best, the genetic algorithm chooses strings biased towards those with the lowest value.

The probability that a string will be chosen for the mating pool is proportional to its transformed fitness. For example, if one string has a value of k for its fitness and a second has $5k$ for its fitness, the second string has 5 times a better chance of being in the mating pool. Additionally, note that the string with the worst AIC value will never be picked for the mating pool, as its fitness will be 0.

Now that we have a method of encoding and a way to evaluate the strings, we will determine some parameters of the genetic algorithm. The first one we consider is the method of creating the initial population and determining its size. Unless previous knowledge about the problem is available, it is commonplace in genetic algorithms to randomly generate binary strings [5]. However, in the case of model selection, a user may want to force a parameter(s) to be included, even if it is not part of the model with the lowest complexity. In this case, the initial population can be generated in such a way that certain parameters are always in the model. In addition to how the population is initially generated, the user must choose the size of the initial population. This process can be difficult. Generally the size should not be too large because it will slow down the algorithm, and should not be so small that genetic drift takes over the course of evolution of the population. In typical genetic algorithms, the size of the population stays the same; however, this may not be an effective use of computation. We will see in the next section that starting with a larger size then reducing it may be more effective.

Finally, we discuss the genetic operators which allow the algorithm to find the optimal model. There are two operators that are generally implemented in genetic algorithms: crossover and mutation. Crossover mimics biological crossover in a simplified manner. First, the probability of crossover (p_c) is chosen. When in the mating pool, a pair of strings are chosen along with a random number from $[0, 1]$. If that number is less than the probability of crossover, crossover occurs. Thus, if $p_c = 1$, then every pair will cross, and if $p_c = 0$, then the strings will not be altered by crossover. After the choice of p_c, the number of crossover points must be chosen. The location of the crossover points is chosen at random. Then the bits from the parent strings are swapped to create two new offspring strings (see Figure 1). The purpose of crossover is to bring together models which have components that reduce complexity. In the previous example about trees, we had two parent strings where Parent 1 coded for a linear model with the intercept, soil pH, average temperature of environment, average rainfall of environment, circumference of trunk, longitude of environment, and prevalence of disease in environment. Parent 2 coded for density of trees in the surrounding area, longitude of environment and latitude of environment. Applying crossover of the two parents created two offspring (see Figure 1), where Offspring 1 coded for a model with an intercept, soil pH, average temperature of environment, longitude of environment, latitude of environment, and prevalence of disease in environment. Offspring 2 is a model that includes density of trees in the surrounding area, average rainfall of environment, circumference of trunk, and longitude of environment. Through successive generations and application of crossover of low complexity models, the algorithm is able to find the least (or close enough) complex model to explain the data.

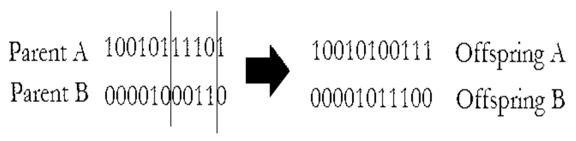

Fig. 1: Diagram of crossover with 2 points

Crossover can only generate models that include parameters that already exists in the population. What if the least complex model includes a parameter that is not present in the population? That is, the position in the string that codes for that parameter is fixed at 0. Mutation alleviates this problem. Mutation in genetic algorithms is similar to mutation that occurs in DNA. First, the probability of mutation (p_m) has to be determined. This value gives the probability that at each location in the string the bit will be flipped. Flipping is defined as the change of a 0 to 1 or a 1 to a 0. Typically, mutation rates are low, on the order of 10^{-3} to 10^{-5}, however, strings are usually longer for other applications of genetic algorithms than they are for determining least complex models.

We conclude this section with pseudo code for a genetic algorithm used to find the least complex model that sufficiently describes the data.

```
1. Generate initial population
2. While (t<max generations OR the maximum number
   of computations have not been executed)
   (a) Calculate AIC for the model each string encodes
   (b) Select strings for the mating pool
   (c) Create a new population using crossover
   (d) Mutate new population
   (e) t=t+1
3. End
```

5. CLASSROOM IMPLEMENTATION

Since the goal of this approach is multifaceted, the classroom implementation has three basic components as a part of a typical regression analysis syllabus.

The first component is a one lecture-hour introduction to genetic algorithm concepts that is best initiated just before the time where a typical syllabus introduces model building via stepwise regression. At this point students are familiar with multiple regression, goodness of fit measures, and their use in model comparison. This component does not need to cover detailed descriptions of crossover options, optimal mutation rates, population sizes, required number of generations, or stopping rule conditions, but rather should be designed to expose the students to the basic notions of evolutionary computing. In fact this is where, for instance, an idea of “numbers mating to create new numbers” fascinates most students, making them eager to learn what comes next.

The second component is also an hour-long demonstration of evolutionary computing. It is made of an exercise where students can witness how an initial guess evolves to be the solution of a problem by following the performances of populations within each generation. One of the best examples of this classroom exercise is obtaining the solution for a Diophantine equation using MS Excel. For this step, using MS Excel, as opposed to an R-code written exclusively to perform GA optimization, a canned GA program, or an online applet, would be highly recommended since this approach allows the students to store each population on a sheet and enables them to compare the improvement in fitness from generation to generation. This step doesn’t have to be very sophisticated. In fact performing the GA only for a few generations (a few Excel worksheets) within a lecture will be enough to convey the underlying message of evolutionary computing. In particular, the recommended exercise is as follows:

1) We are looking for positive integers $a, b, c,$ and d such that $a + 2b + 3c + 4d = 30$. This is the famous Diophantine equation, which has now become a benchmark tool in teaching evolutionary computing or GA based optimization in undergraduate level courses due to its simple but yet non-analytic solution structure. It is a commonly used example

for teaching GA. Another use of it using C++ code can be seen in [3].

2) Generate 4 uniform integers from $(0, 30)$ and save them to columns A2 through D2. Enter the formula $= ABS(A2+2*B2+3*C2+4*D2-30)$ to cell E2 (Use A1 to E1 for labels). Notice that since we are seeking a solution in terms of these four positive integers, none of them can be greater than 30. At this step the user needs to know how to hold the random numbers static, else each time Enter is hit the random numbers will change. This can easily be achieved by setting the Calculation option to Manual in Excel 2010 (Click Formulas ribbon, click Calculation Options,select Manual to disable auto-calculation). In older versions use Click Tools > Options > Calculation tab.

3) Repeat step 2, say 50 times (using the cells A51 to E51), which creates the "1st generation" of 50 "chromosomes" each of which has 4 "genes" (a through d) with "fitness" given in column E. In this case, the smaller the value in column E, the fitter the corresponding chromosome is. Hence, entering the formula $1/fitness$ into column F will be useful during the rest of the process.

4) Normalize fitness of each chromosome saved in column F (by dividing each fitness by the total fitness) to create the probability of selecting a chromosome proportional to its fitness. As a suggestion, the instructor might use conditional formatting in the fitness probability column to help students to visualize the relative magnitudes of the selection probabilities,. In Excel 2010, this can be achieved by using Data Bars-Gradient Fill option.

5) Copy and boldface the "best" solution for these 50 individuals in row 52.

6) Randomly select one pair with probability given in column F and copy them into into the first two rows of column G. These are the two parents of the first offspring. Specifically select two rows using the probabilities in column F.

7) Generate a uniform integer from (1,3) to determine the crossover point. Swap the tails of the two parents to create two children. Copy them to column H.

8) Repeat this process for 25 pairs, forming 50 new children.

9) Randomly choose 5 children (rows). For each selected chromosome generate a uniform integer from (1,4). This will indicate which gene to "mutate". At this point, we explain that 10% of the individuals are subject to mutation for some "divine" reason.

10) Replace the selected genes with newly generated integers from (0,30).

11) Copy these 50 rows onto a new worksheet with their respective fitnesses.

12) The fitness of the best individual in row 52 of the 2nd worksheet can now be compared with that of the first worksheet.

13) Repeat this process for 3 more generations so that a total 5 generations of 50 individuals can be used to to discuss the principles evolutionary computing.

After these steps, the students are now ready to work with GAs for subset model selection, which is the third component of this approach. So far, with the use of the Diophantine equation we demonstrated how chromosomes (variables of the equation) evolve improving the fitness generation after generation. We now make the connection to regression modeling, where the fitness is the quality of the model, say R-square or AIC (another commonly used more robust goodness of fit measure for students of this level). We will see how R-square improves as models evolve. Since the idea of subset model selection requires comparing regression models containing different predictors we will treat models with different set of predictors as different chromosomes. This is exactly what we did for the Diophantine equation where different values of the variables were defined different chromosomes. As an alternative to stepwise regression, we generate a binary string of length equal to the number of possible predictors, and include only. the variables that correspond to 1's. Repeating this process generates a population of regression models each one has its own fitness. We then proceed with the natural selection process as described in Section 4. For this step, it is recommended that the first few steps are demonstrated, again using MS Excel. However, for model selection discussions using real-life data sets, it is recommended that the freely available Excel macro on the author's website[1] should be used to focus on the modeling discussions without allocating any more class time to the process itself. The first few steps can be implemented as follows:

1) Save the response variable to column A, and explanatory variables into columns starting with B.

2) Generate a binary string of length equal to the number of explanatory variables.

3) Form a regression model by including only the explanatory variables that correspond to 1 in the binary string.

4) Run MS Excel regression utility for the model.

5) Repeat from step 2 for another model, pointing out the different models due to the different set of explanatory variables being used.

6) Use the MS Excel macro for a desired number of generations, discussing the model sensitivity aspects of different runs.

The steps given above were implemented as a part of a section where subset model selection was covered in an upper level undergraduate/lower level graduate Regression Analysis class. Since no regression analysis book has a chapter allocated to this approach, the supplementary notes, Excel files were distributed. Although the lectures were also supplemented by Power Point presentations, this is optional and not an essential part of the approach described here.

[1]www.ilstu.edu/∼oakman

6. CONCLUSIONS

Treating predictive model selection via GA in a regression analysis course serves two very useful purposes. First, it introduces students to the notion of evolutionary computing by blending its basic concepts within the very familiar framework of of regression methods. This requires no background beyond some introductory statistics knowledge. Second, it arms students, especially those with diverse interests such as biology, sociology, economics and so on, with a very powerful and cutting-edge method of model building. Additionally, the genetic algorithm approach combined with the use of AIC is better at handling data in which collinearity exist than the traditional selection methods such as forward, backward, and stepwise selection. Although no formal study of student performance was conducted, every student, even the ones who perform less than perfect seem to relate to the material much better than they do to the traditional approaches. Course evaluations consistently indicate this chapter as of the their favorite chapters. In fact several independent Study projects, two M.S. theses were produced on the topic by the students who approach the instructor after this chapter was covered.

7. REFERENCES

[1] Akaike, H. (1973). Information theory and an extension of the maximum likelihood principle. In B.N. Petrov and F. Csaki (Eds.), *Second international symposium on information theory*, Academiai Kiado, Budapest, 267-281.

[2] Boyce, D. E., Farhi, A., and Weischedel, R. (1974). *Optimal Subset Selection: Multiple Regression, Interdependence, and Optimal Network Algorithms.* Springer- Verlag, New York.

[3] http://www.generation5.org/content/1999/gaexample.asp?Print=1

[4] Fisher, R.A. (1930) *The Genetical Theory of Natural Selection* Clarendon Press, Oxford.

[5] Goldberg, D. E.(1989) *Genetic algorithms in search, optimization, and machine learning.* Reading, MA: Addison-Wesley.

[6] Hocking, R. R. (1976). The analysis and selection variables in linear regression, *Biometrics*, 32, 1044.

[7] Hocking, R. R. (1983). Developments in linear regression methodology: 1959-1982, *Technometrics*, 25, 219-230.

[8] J. Holland. (1975) *Adaptation in Natural and Artificial Systems.* The MIT Press.

[9] Mantel, N. (1970). Why stepdown procedures in variables selection, *Technometrics*, 12, 591-612.

[10] Moses, L. E. (1986). *Think and Explain with Statistics*, Addison-Wesley, Reading, MA.

[11] J. Whittingham, Philip A. Stephens, Richard B. Bradbury and Robert P.Freckleton Why Do We Still Use Stepwise Modelling in Ecology and Behaviour? Journal of Animal Ecology, Vol. 75, No. 5 (Sep., 2006), pp. 1182-1189

[12] Wilkinson, L. (1989). *SYSTAT: The System for Statistics*, SYSTAT, Evanston, IL.