University of Houston

Final Project

Applying genetic algorithms to solve regression problems

COSC 6342

Machine Learning

Ricardo Vilalta

Nirupam Bidikar

ID: 1878058

Pranav Saineni

ID: 1884587

Shoumik Sharar Chowdhury

ID: 1878598

December 9, 2020

This page intentionally left blank.

Survey

Evolutionary Algorithms

Evolutionary algorithms are a family of biologically inspired algorithms that are generally used for optimization. Using this algorithm, we look for the best or optimal solution to a problem. This is heavily inspired from Darwin's "survival of the fittest" idea.

All evolutionary algorithms have an initialization step, an evaluation step, a termination step, a selection step and a variation step.

There are a few different evolutionary algorithms like genetic algorithm, simulated annealing, steady state genetic algorithm, etc.

Genetic Algorithms (GA)

Genetic algorithms [2] are the most popular type of evolutionary algorithms which are generally used for search techniques or optimization.

The paper by Messa and Lybanon [3] concluded that genetic algorithms form a basis for another method for curve fitting and minor changes in the parameter and strategies could help achieve a fair degree of accuracy. It also stated that when we have little to no knowledge, experimentation with genetic algorithm is required to achieve high accuracy

The paper by M. Gulsen et al. [4] concluded that genetic algorithms are robust, search and problem parameters can easily be altered to make GA approach to curve fitting viable and versatile. They also say that the computational effort to reach the accurate solution is dependent on the complexity of the function that is fitted to the data

A population of candidate solutions are generated by GA instead of generating a sequence of candidate solutions. A population is a group of solutions to a problem at any step. After that, the five steps of evolution are carried out to find the desired solutions to a problem.

Simple Genetic Algorithm (SGA)

The simple genetic algorithm works in the following way:

i. In the initialization step, a population of random solutions are generated. Each candidate solution has a set of parameters, called chromosomes, that define the solution to the problem.

ii. In the evaluation step, a fitness of the population is calculated. The fitness value (or score) is a number that measures how good a solution is based on the problem that is being solved.

iii. Once we obtain the fitness value, we check if any of the termination criteria is met. A termination criteria could be a number of things - if we have achieved our goal, if our solution isn't improving anymore, if we have reached a maximum number of generations. If the termination criteria is not met, we move on to the selection step.

iv. In the selection step, we choose a sample from the population to create new solutions which are called offsprings. This selection is generally based on the fitness score of the population.

v. In the variation step, the offsprings are created from the parent population using several operators which are discussed in detail later. The least fit populations in the evaluation step are replaced with these new offsprings and this loop continues until the termination criteria is met.

There are several genetic operators like crossover, mutation and replication.

- **Crossover:** Crossover is a technique to form an offspring using genetic information from two parents. The choice of which genetic information comes from which parent is based on a crossover mask.

  For example, if the parent A has the chromosome 11101001000 and parent B has the chromosome 00001010101 and our crossover mask is 11111000000, the two offsprings will have chromosomes 11101010101 and 00001001000. In the example, the first 5 bits from parent A remains the same for the first offspring while the last 6 bits are taken from parent B, according to the crossover mask.

- **Mutation:** In mutation, a bit is randomly chosen in the solution (assuming a uniform distribution in the chromosome) and flipped.
- **Replication:** Replication is the process of replicating offsprings from parents without changing anything.

Steady State Genetic Algorithm (SSGA)

Another type of genetic algorithm is the steady state genetic algorithm also known as SSGA [7]. Unlike a regular genetic algorithm, selection does not replace the individuals in the population. Instead of adding the children of the selected parents into the next generation, we select the best individuals out of the two parents and two children are added back into the population. The population size in SSGA remains constant.

Linear Regression

Regression is a method of modelling a target value based on predictors. Regression differs from case to case based on the number of independent variables and how each of those variables correlate to each other. Simple linear regressions is a type of regression where the number of independent variables is one and there is a linear relationship between the independent and dependent variable.

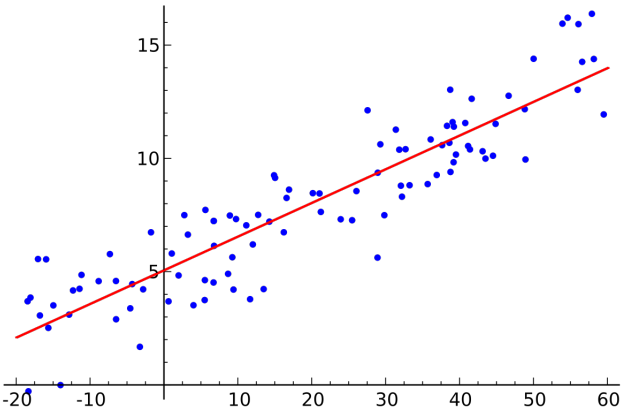

Figure 1: Linear regression

Source: Wikipedia

The red line in the above graph is referred to as the best fit straight line. The line

can be modelled based on the linear equation shown below:

$$y = c + mx$$

Cost function: The cost function helps us to figure out the best value for c and m which provides the best fit line for our data points.

Since we want the best or optimal value for m and c, we convert this search problem into a minimization problem where we would like to minimize the error between the predicted value and the actual value.

$$minimize\ \frac{1}{n}\sum_{i=1}^{n}(pred_i - y_i)^2$$
$$J = \frac{1}{n}\sum_{i=1}^{n}(pred_i - y_i)^2$$

The difference between the predicted values and ground truth measures the error difference. We square the error difference and sum over all data points and divide that value by the total number of data points. This provides the average squared error over all the data points also known as the Mean Squared Error(MSE) function.

Gradient Descent: Gradient descent is a method to update c and m to reduce the cost function. The concept behind is that we start with some value for c and m and iteratively change these values to reduce the cost. In order to update c and m we take gradient from cost function. To find these we take partial derivatives with respect to c and m.

$$J = \frac{1}{n}\sum_{i=1}^{n}(pred_i - y_i)^2$$
$$J = \frac{1}{n}\sum_{i=1}^{n}(c + mx_i - y_i)^2$$

$$\frac{\delta J}{\delta c} = \frac{2}{n}\sum_{i=1}^{n}(c + mx_i - y_i) \Rightarrow \frac{\delta J}{\delta c} = \frac{2}{n}\sum_{i=1}^{n}(pred_i - y_i)$$
$$\frac{\delta J}{\delta m} = \frac{2}{n}\sum_{i=1}^{n}(c + mx_i - y_i)\cdot x_i \Rightarrow \frac{\delta J}{\delta m} = \frac{2}{n}\sum_{i=1}^{n}(pred_i - y_i)\cdot x_i$$

$$c = c - \alpha \cdot \frac{2}{n} \sum_{i=1}^{n} (pred_i - y_i)$$
$$m = m - \alpha \cdot \frac{2}{n} \sum_{i=1}^{n} (pred_i - y_i) \cdot x_i$$

α is the learning rate which we specify. Smaller value for α gets you closer to minima but takes more time to reach while larger value for alpha can lessen the time but could overshoot the minima.

Simulated Annealing

Simulated annealing is a local search meta-heuristic which provides us a mean to escape local optima by using hill climbing moves to find the global optima. Basically it is a combination of hill climbing and random walk. We use hill climb to find the global maximum and random walk to increase the efficiency to find the global optimum value.

$$P = exp\left(-\frac{E_1 - E_0}{T}\right)$$

where E_1 is new cost, E_0 is old cost and T is temperature.

The acceptance probability helps us to compare the new cost with old cost. It gets smaller if the new solution gets worse than the old one. If the randomly generated value is greater than the acceptance probability we will change our old cost to new one.

Tsoukalas and Fragiadakis [8] applied multiple linear regression and genetic algorithm model to predict occupational risk in the shipbuilding industry. The result from the LR model was fed to GA and possible solutions were generated and evaluated using their fitness functions. Their model proved to be a feasible way to estimate the risk factor and was an inspiration for this project idea.

Datasets

The datasets used in this paper are the Boston House prices [1] dataset and the diabetes dataset taken from Scikit-Learn [6]. For preprocessing, the data was normalized using the *MinMaxScaler* function.

The Idea and Experimentation

Linear regression can be a good tool to solve estimation problems by trying to fit a line through given data. It does this by minimizing residual error. Since we can express a linear regression as an optimization problem, we decided to see if genetic algorithms can solve this problem too.

Using the principles of genetic algorithms, we generate a population of solutions. Here each solution is a list of coefficients matching the input size of the dataset. Next we evaluate this population using our fitness function which is a formula based implementation of regression using Least squares method. Best individuals are selected and crossover, mutation operations are applied to generate a new population.

We tried 4 methods to boost performance of our model which are described below. Comparison with regular linear regression is also shown. All the graphs shown below have been plotted with the validation set.

Simple Genetic Algorithm

This idea follows the basic principles of GA i.e. initialization, evaluation, crossover, mutation. We define the regression problem and describe it within the confines of the domain of GA.

The base model performs close to the linear model. We tune the parameters a lot to get this amount of performance. Individual mutation probability were bumped to 0.35 to get these results. Higher mutation probability values are risky as we might end up altering good individuals in the population.

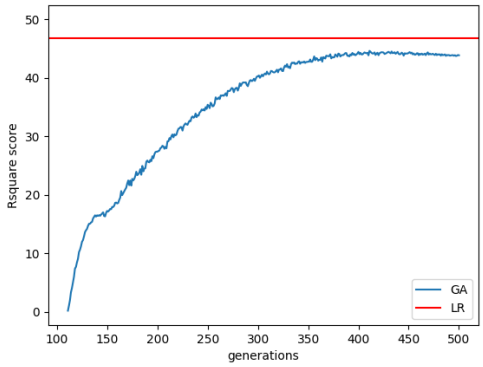

Figure 2: LR vs GA using simple GA - Diabetes

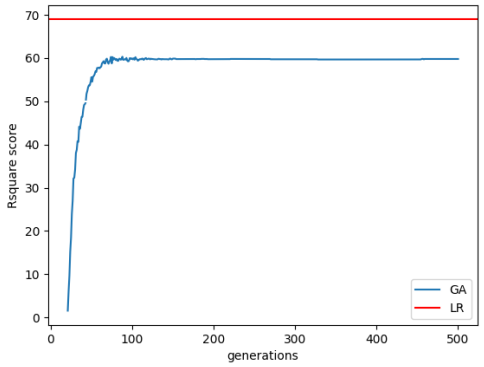

Figure 3: LR vs GA using simple GA - Boston

Variable mutation probability

Depending on the fitness difference between generations, we alter the mutation probability. As the difference decreases we increase the probability to bring back some randomness and get out of the local minima.

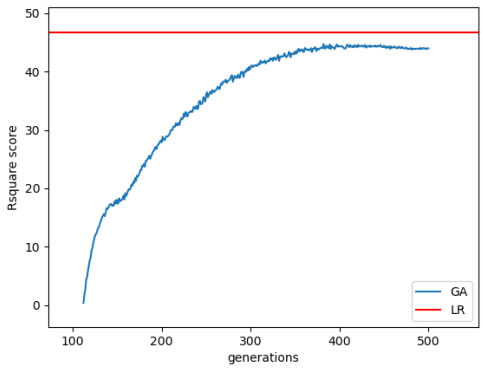

Figure 4: LR vs GA using variable mutation - Diabetes

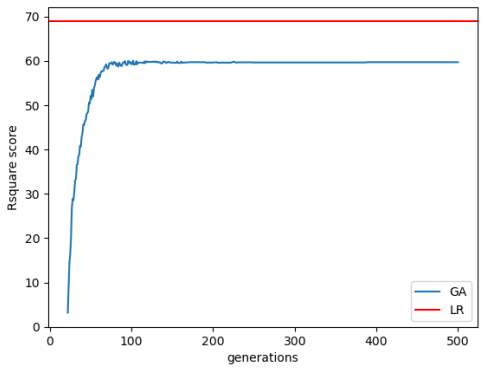

Figure 5: LR vs GA using variable mutation - Boston

This method reaches close to linear regression in performance as seen from figure 4 and figure 5. The mutation certainly helps improve performance but in the end converges to nearly the same solution. Mutation probability used was 0.35.

Crossover-Mutation Split Population

After evaluation a population, parent pairs are selected and sent to crossover. The top 50% of the offsprings are retained and the remaining half undergo mutation.

This was done to preserve good individuals and attempt to improve the others.

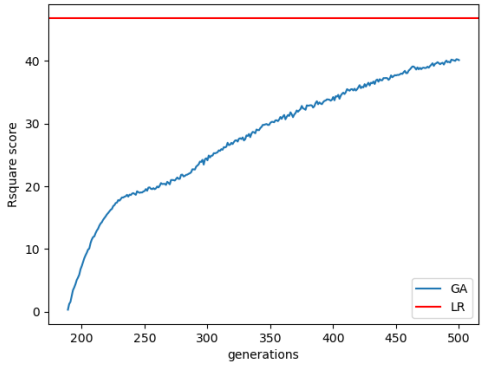

Figure 6: LR vs GA using split population - Diabetes

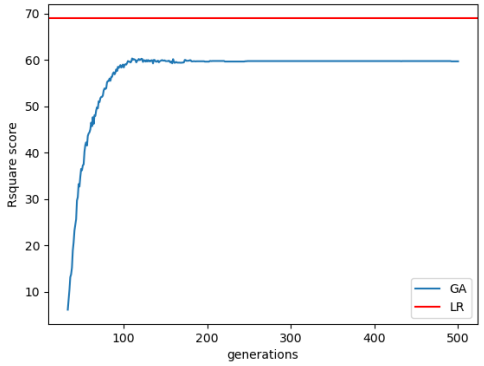

Figure 7: LR vs GA using split population - Boston

From the results, it is evident that the performance is not optimal. We can see in figure 7 that the algorithm converges to a solution and attain equilibrium. In 6, we observe an increasing trend but it would require more generations to converge to a solution at which point it is not worth trying. This algorithm takes the longest to converge among the mentioned methods. Even increasing mutation probability resulted in no improvement.

Simulated Annealing Optimization

Liu et al. [5] used the idea of simulated annealing with GA to develop bus routes. Their results showed that this model was able to converge to an optimal solution. We wanted to use simulated annealing to help our model get out of a local minima and converge to the best possible solution. We tweaked this approach slightly by using the validation set in the evaluation function to impose stricter norms on the best individuals selected.

We chose coefficient of determination (COD) value for evaluation in the optimization function rather than choosing error and it showed slightly better performance. This method converges to a solution the fastest among the mentioned methods.

Given the randomness factor and the properties of simulated annealing optimization, we expected this model to converge to a significantly better solution and even slightly beat the regression model. The results indicate that SA model marginally outperforms simple GA with both getting an average squared error of 3115.742 and

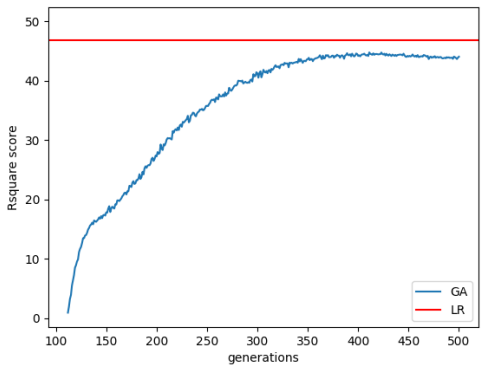

Figure 8: LR vs GA using simulated annealing - Diabetes

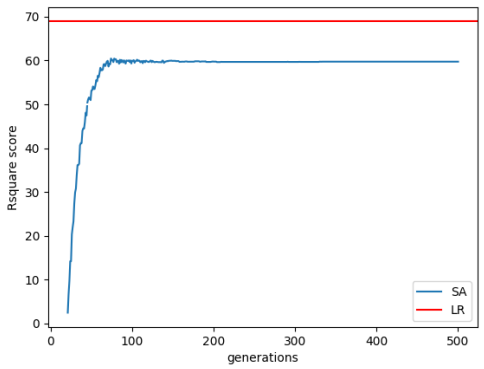

Figure 9: LR vs GA using simulated annealing - Boston

3118.398 respectively.

We also see that simple genetic algorithm and simulated annealing optimization perform similarly where simulated annealing working slightly better than genetic algorithm in some cases. It does not suffer from problems of local optima as it can get unstuck from cases where neighbour solutions are worse than the current one.

Future Work

In the future, we would like to try implementing complex crossover masks and selection techniques to help boost the performance of our model. A good and efficient fitness function is essential and we would like to test our more complex functions which are well suited to our problem. We would also like to extend the idea of solving optimization problems using GA in other domains.

Conclusion

This project started out with an idea about solving regression problem using genetic algorithms by describing it as an optimization problem and trying to search for an optimal solution. As we started implementing different methods to solve this problem, we saw that no matter the method, the algorithm converges to a similar enough solution with the only difference being the time taken to get there. We test with different parameters for the genetic algorithms and for the optimization function which resulted in a marginal improvement in performance. We still have

some work to do in developing a good enough model but with more research it is plausible that we might find a model which can beat traditional linear regression models.

References

[1] D. Harrison and D. L. Rubinfeld, "Hedonic housing prices and the demand for clean air," *Journal of Environmental Economics and Management*, vol. 5, no. 1, pp. 81–102, 1978. DOI: 10.1016/0095-0696(78)90006-2.

[2] J. H. Holland, "Genetic algorithms and adaptation," in *Adaptive Control of Ill-Defined Systems*, O. G. Selfridge, E. L. Rissland, and M. A. Arbib, Eds. Boston, MA: Springer US, 1984, pp. 317–333, ISBN: 978-1-4684-8941-5. DOI: 10.1007/978-1-4684-8941-5_21. [Online]. Available: https://doi.org/10.1007/978-1-4684-8941-5_21.

[3] K. Messa and M. Lybanon, *Curve Fitting Using Genetic Algorithms*. 1991.

[4] M. Gulsen, A. E. Smith, and D. M. Tate, "A genetic algorithm approach to curve fitting," *International Journal of Production Research*, vol. 33, no. 7, pp. 1911–1923, 1995. DOI: 10.1080/00207549508904789. eprint: https://doi.org/10.1080/00207549508904789. [Online]. Available: https://doi.org/10.1080/00207549508904789.

[5] L. Liu, P. Olszewski, and P.-C. Goh, "Combined simulated annealing and genetic algorithm approach to bus network design," in *International Conference on Transport Systems Telematics*, Springer, 2010, pp. 335–346.

[6] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[7] A. Agapie and A. H. Wright, "Theoretical analysis of steady state genetic algorithms," *Applications of Mathematics*, vol. 59, no. 5, pp. 509–525, Oct. 2014, ISSN: 1572-9109. DOI: 10.1007/s10492-014-0069-z. [Online]. Available: https://doi.org/10.1007/s10492-014-0069-z.

[8] V. Tsoukalas and N. Fragiadakis, "Prediction of occupational risk in the shipbuilding industry using multivariable linear regression and genetic algorithm analysis," *Safety Science*, vol. 83, pp. 12–22, 2016.