International Journal of Scientific Research in Science, Engineering and Technology
Print ISSN: 2395-1990 | Online ISSN : 2394-4099 (www.ijsrset.com)
doi : https://doi.org/10.32628/IJSRSET1218226

A Comparative Study on the Performance of Gene Expression Programming and Machine Learning Methods

Mohammad Anas, Mohiuddeen Khan, Hammad Basit
Department of Computer Engineering, Aligarh Muslim University, Aligarh, Uttar Pradesh, India

Article Info
Volume 8 Issue 2
Page Number: 140-147

Publication Issue :
March-April-2021

Article History
Accepted : 20 March 2021
Published : 29 March 2021

ABSTRACT

Usually, evolutionary algorithms are used to provide strong approximations to problems that are difficult to solve with other methods. Gene expression programming (GEP) is a type of evolutionary algorithm used in computer programming to generate computer programs or models. These computer programs are complex tree structures that, like a living organism, learn and adapt by modifying their sizes, shapes, and composition. In the present work, a comparison study was made among GEP and the standard prediction techniques to find the best predicting model on the BOSTON HOUSING dataset. Three approaches viz. GEP, ANN and polynomial regression were implemented on the dataset. The study showed how the three methods solve the problem of high bias and high variance and which one outperforms the other. The research work, however, gave a glimpse of the actual limitations and advantages of the methods on one another indicating the dependency of method on the type of data used. The results conclude the comparison of different methods on different performance metrics. The GEP model however reduced the problem of high bias and high variance by giving a slight difference between the train and test accuracy but was not able to outperform ANN and polynomial regression in terms of performance metrics.

Keywords: Evolutionary Algorithms, Gene Expression Programming, Machine Learning, Artificial Neural Network, Polynomial Regression

I. INTRODUCTION

In Evolutionary algorithms, individuals are selected based on their fitness, and genetic variation is introduced using one or more genetic operators. They have been used in artificial computational systems since the 1950s to solve optimization problems. Reichenberg's[1] implementation of evolution methods in 1965 cemented the success of evolutionary algorithms.

Gene expression programming is similar to genetic algorithms and genetic programming and belongs to the class of evolutionary algorithms. It inherits linear chromosomes of fixed length from genetic algorithms

and expressive parse trees of various sizes and shapes from genetic programming.

Many studies have been conducted to show a comparison of evolutionary algorithms and neural networks. Most results showed GEP and evolutionary algorithms performing better than the standard prediction models[2] such as neural networks, decision trees, etc. This research work implements three prediction techniques viz. gene expression programming (GEP), artificial neural networks (ANNs) and polynomial regression on the BOSTON HOUSING dataset to perform the comparison among the results obtained by these methods.

The Boston Housing Dataset is sourced from UCI datasets. It was collected by the United States Census Service on housing in the Boston MA area. It has 13 training features. The features are per capita crime rate by town (CRIM), the proportion of residential land zoned for lots over 25,000 sq. ft (ZN), the proportion of non-retail business acres per town (INDUS), Charles River dummy variable (CHAS), nitric oxides concentration (NOX), the average number of rooms per dwelling (RM), the proportion of owner-occupied units built prior to 1940 (AGE), weighted distances to five Boston employment centres (DIS), index of accessibility to radial highways (RAD), the full-value property-tax rate per $10,000 (TAX), the pupil-teacher ratio by town (PTRATIO), $1000(\text{Bk} - 0.63)^2$ where Bk is the proportion of blacks by town (B), % lower status of the population (LSTAT). The target was the median value of owner-occupied homes in $1000's (MEDV).

It is known that GEP and genetic algorithms are best for the tasks involving discrete data whereas ANNs are used for the tasks containing continuous data. This work shows how ANN performs better than the GEP on continuous data. However, the best part of GEP or genetic algorithms is that they provide a visual experience to the user. In GEP, the expression trees give an in-depth knowledge of the method and the working process. ANNs and regression models on the other side don't provide any information about the inside process. GEP being inspired by the natural evolution make the user understand the process even better which is very good for new researchers.

II. Gene expression programming (GEP)

Gene Expression Programming (GEP), proposed by Ferreira[4], is an evolutionary algorithm that automatically generates the finest solution to a particular problem. GEP has been used in variety of different purposes ranging from statistical models, classification, and symbolic regression.

A. GEP Model

In the GEP model, solutions exist in the form of expression trees comprising of mathematical functions, constants, and input variables. Like genes in a living organism, these tree structures are encoded in plain linear chromosomes of fixed length, and they learn and adapt by modifying their sizes, shapes, and composition.

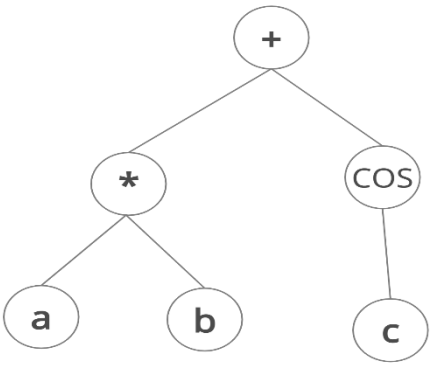

Figure 1: An example of expression tree for equation for equation $(a * b) + \cos(c)$

Genetic Programming generally begins with a population of initial population that is created at random. It then applies genetic operators like

crossover and mutation to iteratively turn the initial population of possible solutions into a new generation of the population. A fitness function is used to pick the individuals that will survive the next iteration. The fitness function is an objective function that defines how close an entity is to achieve a goal based on a set of standards. The process is replicated in this manner until the termination condition, which is the maximum number of generations, is reached.

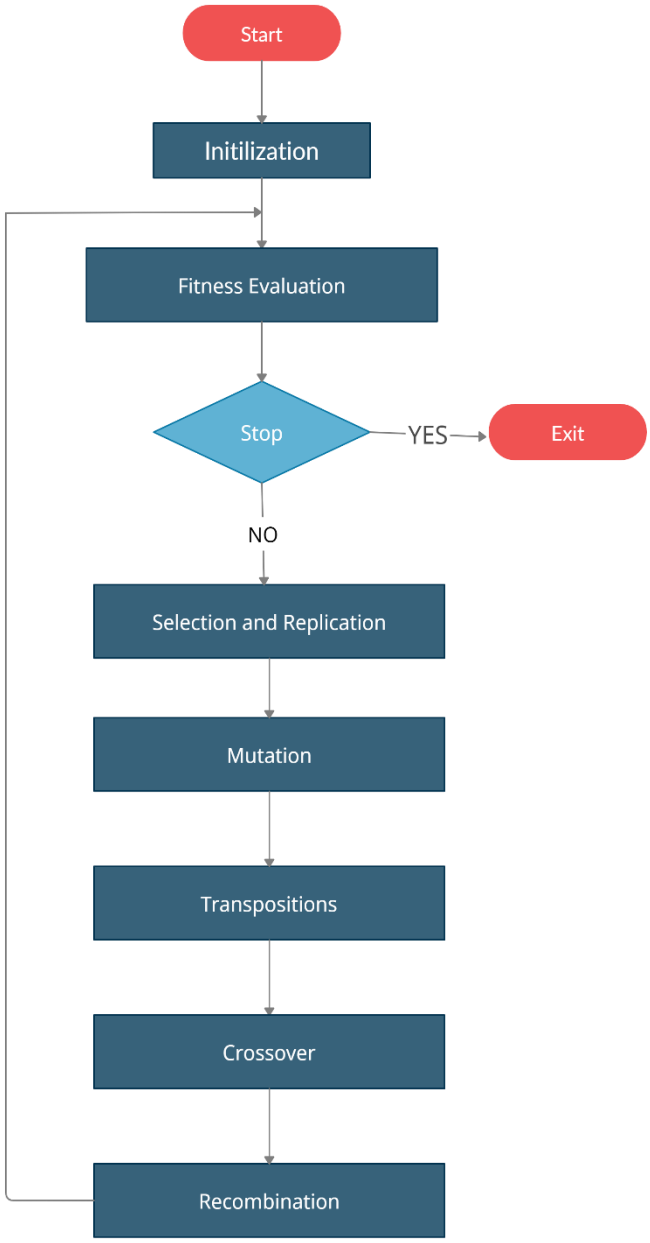

Figure 2: Flow Diagram of GEP Model

In the end, we select the single best individual in the ultimate population as the final solution. Figure 2 shows the basic steps in the GEP algorithm.

B. Implementation of GEP RNC

GEP with real random numerical constants (GEP-RNC) have chromosomes caring an extra domain for encoding the random numerical constants. As determining real constants is significant in regression problems, an extra gene domain for random numerical constants (RNC) was used. A richly expressive framework was developed by combining this domain with a special terminal placeholder for the RNCs[7].

This GEP-RNC algorithm was used for symbolic regression on the BOSTON HOUSING dataset. The dataset was divided into two sections: train (80%) and test (20%). The Geppy framework was used to evaluate the dataset[5]. The work began with the definition of a primitive set that included all of the building blocks for the GEP. A Mean Squared Error based fitness function was defined to assess an individual's fitness. Linear fitness scaling was added to it to allow fitness function to narrow down to the best result results, especially when there is a very close difference between them[6].

Various operators as defined in table I, are used to form our final symbolic regression function by evolving the model. The number of programs in each generation is determined by the size of population. In every iteration, we chose the best 3 individuals known as champs and pass them to the next iteration until we exhaust all the generations and iteration is stopped. The results obtained are shown in table 3.

TABLE I

DIFFERENT PARAMETERS OF GEP RNC MODEL

Parameter	Values
Operators	+, -, *, /, sin, cos, tan
Head Size	20
Number of genes	2
Length of the RNC array	13
Linking function	Addition
Mutation rate	0.05
Inverse rate	0.1
One-point recombination rate	0.3
Two-point recombination rate	0.2
Size of population	180
Number of generations	100
No of Champs	3

From the above model the following eq(1) was obtained as final result. Figure 3 shows the Expression tree for eq(1).

$$\begin{aligned} medv = & -0.289835688960316 * lstat * rm \\ & - 0.579671377920632 * lstat \\ & - 0.289835688960316 * ptratio \\ & + 0.289835688960316 * rm \\ & + 0.289835688960316 \\ & * sin\,(tratio) \\ & - 0.208269336281887 \end{aligned} \quad (1)$$

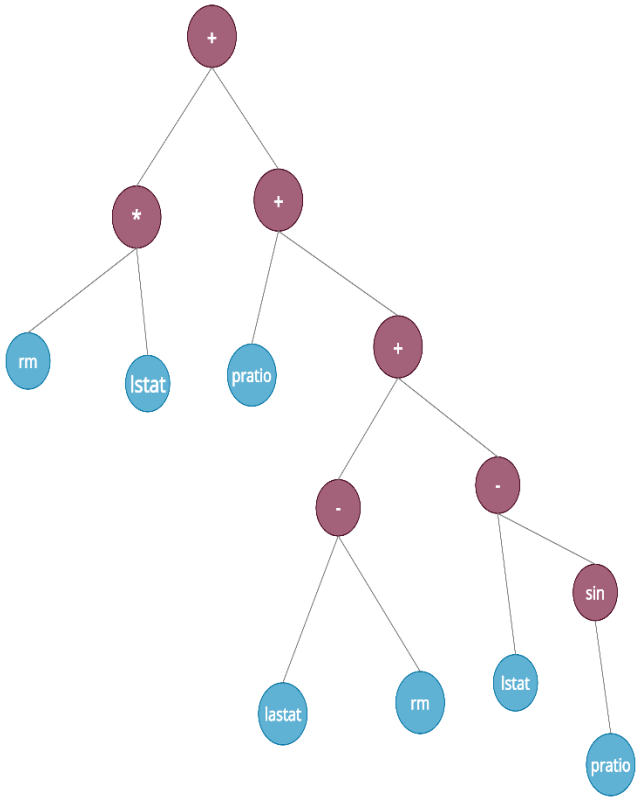

Figure 3: Expression tree of the final solution. The constants are hidden for clarity.

III. Artificial Neural Network (ANN)

A conceptual perspective focused on the structure and functions of a biological neural network is known as an artificial neuron network (ANN). Many research works showed that ANN provides more accurate results than standard statistical methods of prediction. We deployed an artificial neural network on the BOSTON HOUSING dataset to compare its performance with the other methods of prediction.

A. ANN model architecture

An artificial neural network (ANN) is made up of neurons, which are small and well-organized units. They're connected by elements that pass different weights and allow signals to pass through them. A feed-forward neural network model is made up of

several layers of neurons. Figure 4 shows a simple neural network with 3 sequential layers having (3-4-1) neurons. Hidden layers link the first layer, also known as the input layer, to the output layer. The mathematical functioning of a single neuron can be represented by eq(2).

$$y(x) = f(\sum_{i=0}^{n} w_i x_i + b) \quad (2)$$

$where: x\ is\ a\ neuron$
$n = input\ from\ x_0\ to\ x_n$
$y(x) = single\ output, w_1 = weights$
$b = bias, f = activation\ function$

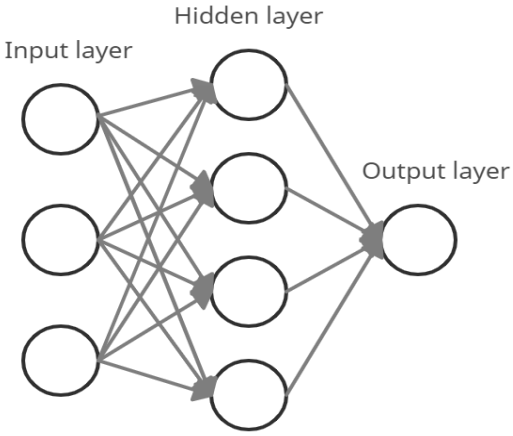

Figure 4: A neural network

In order to approximate the output data, the basic feed-forward network implements a nonlinear revolution of input data. By measuring a weighted sum and applying a bias to it, the activation function (*f*) determines whether a neuron in the neural network should be triggered or not. The main purpose of activation functions in a neural network is to introduce non-linearity in the output of the neural network. For our network, we used the *tanh* activation function[8] denoted by eq(3). The plot of the *tanh* function is shown in Figure 5.

$$tanhx = \frac{e^x - e^{-x}}{e^x + e^{-x}} \quad (3)$$

B. Training ANN

For better comparison among different methods, the train and test data were split in the same ratio i.e. 80%(training)/20%(test). The neural network had 5 sequential layers including the input and the output layer. *tanh* activation function was enabled for the layers to produce non-linearity to the output. The above model was deployed on the training dataset and was trained for 30 epochs with the implementation of Adam optimizer. Following results were obtained given in table 2.

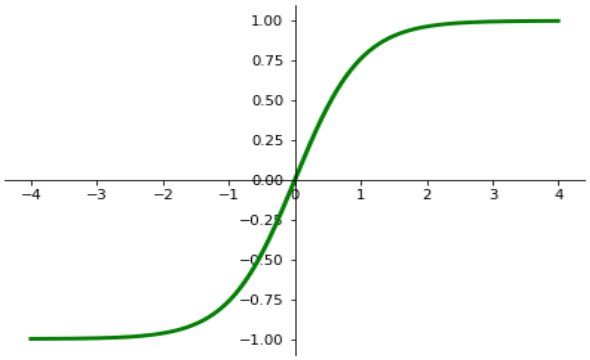

Figure 5: *tanh* function

IV. Polynomial regression

Polynomial regression has been widely used to predict non-linear values and has proved to be very good in various scenarios. We will now see the efficacy of this model on the Boston housing dataset.

A. Polynomial regression model

Polynomial regression is a form of regression analysis that models the relationship between the independent variable x and the dependent variable y is an nth degree polynomial in x. Polynomial regression fits a nonlinear relationship between the value of x and the corresponding conditional mean of y. Although polynomial regression fits a nonlinear

model to the data, as a statistical estimation problem it is linear, in the sense that the regression function E (y | x) is linear in the unknown parameters that are estimated from the data[9]. The general equation of polynomial regression of degree n is given by eq(4).

$$y = \beta_0 + \beta_1 x_i + \beta_2 x_i^2 + \cdots + \beta_n x_i^n + \varepsilon \quad (4)$$

Simple linear regression provides us with a good result on a linear dataset, but if we apply the same model without any modification on a non-linear dataset, the results produced can be very disappointing, the accuracy may be decreased and the error rate may be very high. So, for cases of a non-linear dataset, we use polynomial regression models[3] which can deal with nth degree curves and fit on the data appropriately giving us better accuracy and a smaller error rate.

B. Polynomial Regression Implementation

In implementing polynomial regression (PR), the most important factor is to determine the nth degree which would appropriately adapt to the dataset without overfitting. To train the polynomial regression model, we divided the dataset into train and test sets, each comprising 80% and 20% of data respectively. After a hit and trial method, it was found that n=2 is the most appropriate degree for our polynomial regression model.

On looking through data, it was observed that the scale of various features is very different and hence, directly applying a regression model on these features will give very abrupt results. It must be normalized. We used Z score normalization for standardizing the dataset.

Another problem found in the dataset was that it suffered from the problem of multicollinearity, that is various features are dependent on each other. In such cases, the impact of one's variable on dependent variable X tends to be less precise than when all the features are independent. Thus, we need to reduce collinearity without losing any information. The number of features is also big and may adversely affect the training of our regression model.

To eradicate these problems, we implemented the Principal Component Analysis, as discussed here[10]. Principal component analysis (PCA) is a statistical procedure that uses an orthogonal transformation to convert a set of observations of possibly correlated variables (entities each of which takes on various numerical values) into a set of values of linearly uncorrelated variables called principal components. This transformation is defined in such a way that the first principal component has the largest possible variance (that is, accounts for as much of the variability in the data as possible), and each succeeding component, in turn, has the highest variance possible under the constraint that it is orthogonal to the preceding components.

On implementing Principal Component Analysis, we managed to reduce the number of features from thirteen to five. It also removed the collinearity. Figure 6 shows the distribution of features after applying PCA. The results are shown in table 4.

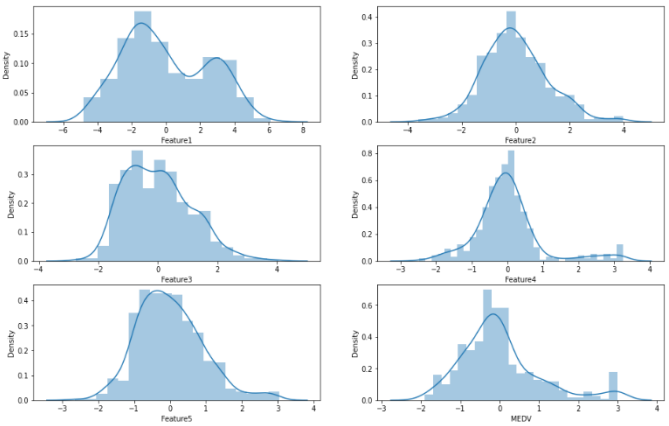

Figure 6: Distribution of features after applying PCA

V. RESULTS

The models described above were deployed and the following results were obtained on different performance metrics given in the tables below: -

TABLE II
PERFORMANCE OF ANN

Performance metric	Train	Test
R2 score	.9276	.8902
MAE	.2042	.2470
MSE	.0733	.1018

TABLE III
PERFORMANCE OF GEP

Performance metric	Train	Test
R2 score	.7545	.7340
MAE	.3240	.3274
MSE	.02402	.2843

TABLE IV
PERFORMANCE OF Polynomial Regression

Performance metric	Train	Test
R2 score	.9364	.8242
MAE	.1845	.2895
MSE	.0629	.1765

VI. CONCLUSION

This research showed a comparative study among three prediction techniques and was able to show the dominance of neural networks on the other two. We conclude that evolutionary algorithms do not always beat the neural networks and regression methods instead it depends on the nature of the data of the problem. Continuous data like the one we used here showed ANN and regression outperforming the GEP-RNC method. A problem involving discrete data may show GEP and other evolutionary algorithms outperforming standard statistical prediction techniques.

VII. REFERENCES

[1]. Rechenberg, Ingo (1973). Evolutionsstrategie. Stuttgart: Holzmann-Froboog. ISBN 3-7728-0373-3.

[2]. Roy, Sumit & Ghosh, Ashmita & Das, Ajoy & Banerjee, Rahul. (2014). A comparative study of GEP and an ANN strategy to model engine performance and emission characteristics of a CRDI assisted single cylinder diesel engine under CNG dual-fuel operation. Journal of Natural Gas Science and Engineering. 21. 814-828. 10.1016/j.jngse.2014.10.024.

[3]. Ostertagova, Eva. (2012). Modelling Using Polynomial Regression. Procedia Engineering. 48. 500–506. 10.1016/j.proeng.2012.09.545.

[4]. Ferreira, Candida. (2001). Gene Expression Programming: A New Adaptive Algorithm for Solving Problems. Complex Syst. 13.

[5]. Shuhua Gao. (2020). geppy: a Python framework for gene expression programming.

[6]. Sadjadi, Farzad. (2004). Comparison of fitness scaling functions in genetic algorithms with applications to optical processing. Proceedings of SPIE - The International Society for Optical Engineering. 5557. 10.1117/12.563910.

[7]. Peng, Yuzhong & Yuan, ChangAn & Qin, Xiao & Huang, Jiangtao & Shi, YaBing. (2014). An improved Gene Expression Programming approach for symbolic regression problems. Neurocomputing. 137. 293–301. 10.1016/j.neucom.2013.05.062.

[8]. Nwankpa, Chigozie & Ijomah, Winifred & Gachagan, Anthony & Marshall, Stephen. (2020). Activation Functions: Comparison of trends in Practice and Research for Deep Learning.
[9]. Ruppert, David. (1996). Local polynomial regression and its applications in environmental statistics. Statist Environ. 3.
[10]. Jolliffe, Ian. (1986). Principal Component Analysis. Springer: Berlin, Germany. 87. 41-64. 10.1007/b98835.

Cite this article as :

Mohammad Anas, Mohiuddeen Khan, Hammad Basit, "A Comparative Study on the Performance of Gene Expression Programming and Machine Learning Methods ", International Journal of Scientific Research in Science, Engineering and Technology (IJSRSET), Online ISSN : 2394-4099, Print ISSN : 2395-1990, Volume 8 Issue 2, pp. 140-147, March-April 2021. Available at
doi : https://doi.org/10.32628/IJSRSET1218226
Journal URL : https://ijsrset.com/IJSRSET1218226