The Study of Crop Pest Forecasting Using GP

Miao Li, Tao Sun, Jian Zhang

Institute of Intelligent Machines, Chinese Academy of Sciences, Science Road #10, 230031 Hefei, Anhui, China

suntao2000st@ustc.edu

Abstract

In this paper, two models for insect pests forecasting have been developed. A traditional linear regression model and a Genetic Programming (GP) based model are proposed. Two of the key issues in the improvement of GP are: 1) set a logical variable *isOld* for the attribute of each individual in the population; 2) take into account not only the performance of individuals, but also the depth of individuals. Using statistical and economical measures to estimate the results, we show that the GP model could give more reliable results.
Keyword: Genetic Programming, Linear Regression, Pest Forecasting

I. Introduction

Crop plant diseases and insect pests predict is the foundation of integrated control in advance, the important prerequisite of a bumper harvest as a result of effective control. However, such predict is affected not only by the characteristic of insect pests but also the restriction of the meteorological factor. Traditional forecasting methods, including data set fitting, regression and approximation theory and so on, are based on the data analysis and model construction manually. Then this problem is transferred to fix the variables $w= (w_1, w_2 \ldots w_m)$ of the function $f(x, w)$ to fit, interpolate and approximate data sets. These methods are very difficult to set up the accurate model and the reliability is low for the error of the experimental data.

To solve these problems, we use Genetic Programming (GP), which has the characteristic of self-organization, self-learning and self-adaptation, to get the mathematics model from history data of insect pests for forecasting system. The natural selection dispels the algorithms design obstacle that needs to describe all the features of history data set and the measures that algorithm of the question should be taken for different characteristics [1][2][3]. This paper mainly uses GP to get model structure which is more intelligent and atomizing for the characteristic of GP. And two improvements for the question of time-consuming and pre-maturing are developed at the same time.

II. Problem Descriptions

The history data is from [4], see Table 1. The purpose is to construct an optimum model and predict the wheat stripe rust grade of Tianshui, Gansu province in 1996 according to the data between 1979 and 1991. The parameters are listed as below:

x-the proportion of infected wheat area;
y-the rate of wheat in Tianshui last autumn;
z-the snow covering days in winter;
T-the actual grade;

Predict index is graded according to the national plant protection standard: serious to light which are expressed by the number of 5 to 1.

Table 1. Tianshui Insect Pests History Data

Sn	YEAR	*x*	*y*	*z*	*T*
1	1979	61.0	0.405	12	3
2	1980	64.8	0.397	16	3
3	1981	50.8	0.002	10	1
4	1982	52.9	0.317	18	3
5	1983	61.2	0.111	18	3
6	1984	76.0	0.521	22	4
7	1985	80.4	0.887	39	5
8	1986	50.9	0.391	6	1
9	1987	50.4	0.082	10	1
10	1988	41.2	0.081	9	1
11	1989	60.0	0.900	25	3
12	1990	80.0	1.910	27	5
13	1991	70.0	0.750	9	3
14	1996	80.0	1.020	36	?

III. Regression Model

Traditional modeling and identification approaches provide us with many models which can fit many forecasting applications. Among them, method of least square (LSM) is the most popular one. Method of least square is the one extensively used in the multi-disciplinary field, which can be applied in data analysis problems such as parameters' reliability estimate, data processing of combination forecasting, experiential formula fitting by experiments and regression and so on[1][5].As for the problem in section 2, we need to construct the approximate expression according to the experimental data sets (x_1, y_1, z_1, T_1), (x_2, y_2, z_2, T_2),…, (x_n, y_n, z_n, T_n), where x, y, z are the factors , T is the grade and n is the number of data sets. The variables(x, y, and z) and the grade mainly have such function relationship: T=f(x, y, z, a_1, a_2, a_3, a_4), in which the fix of parameters a_1, a_2, a_3 and a_4 needs n experimental data sets. To fix these parameters, we get the square of sum of the deviation as below:

$$D(a_1,a_2,a_3,a_4)=\sum_{i=1}^{n}[f(x_i,y_i,z_i,a_1,a_2,a_3,a_4)-Ti]^2 \qquad (1)$$

In order to make value of D the least, we use formula 2 to get the parameters a1 to a4:

$$\frac{\partial D}{\partial a_i} = 0 \, (i = 1, 2, 3, 4) \tag{2}$$

Then through solving the upper equations we get the model structure as the following formula:

$$T = a_1 + a_2 * x + a_3 * y + a_4 * z \tag{3}$$

In this paper we are using an evaluation criterion: root mean-squared error (RMSE) to measure the performance of the developed models, which was defined by the following formula:

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(f(x_i, y_i, z_i) - T_i)^2} \tag{4}$$

where Ti means the i-th actual insect pests grade and $f(x_i, y_i, z_i)$ is the i-th value by the output model. Usually, the model is not perfect. The smaller the RMSE, the better the performance of the model is.

Through solving 4 equations using formula (2), we get the regressor and the optimum model that describes the dynamics of the trend of insect pests in Tianshui using LSM is presented below:

$$T = -3.0748 + 0.08 * x + 0.2511 * y + 0.042 * z \tag{5}$$

IV. GP Model

The computational process of linear regression method is simple. But its shortcoming is to assume the trend of the variables as a straight line. The fit result is usually not good as for fluctuant data sets. GP is good at solving problems of function approximating and curve fitting and so on and is more effective for solving model of unknown structure and parameters. In this section, GP is used to build a suitable model structure for insect pests forecasting.

4.1 Brief introduction of GP

Genetic Programming (GP), proposed on the basis of Genetic Algorithm (GA) by Koza in 1992, is an automated method for creating a working computer program from a high-level problem statement of a problem. Genetic programming creates computer programs in the lisp or scheme computer languages as the solution. [6] [7] [8] [9].

Genetic programming starts from a high-level statement of "what needs to be done" and automatically creates a computer program to solve the problem. Genetic programming starts with a primordial ooze of thousands of randomly created computer programs. This population of programs is progressively evolved over a series of generations. The evolutionary search uses the Darwinian principle of natural selection (survival of the fittest) and analogs of various naturally occurring operations, including crossover (sexual recombination), mutation and gene duplication.
The main steps of GP are as below [6]:

（1） Select of terminals and functions, set certain parameters for controlling the run, and the termination criterion and method for designating the result of the run.

（2） Randomly create an initial population (generation 0) of individual computer programs composed of the available functions and terminals.

（3） Execute each program in the population and ascertain its fitness (explicitly or implicitly) using the problem's fitness measure. Create new individual program(s) for

the population by applying genetic operations (reproduction, crossover, mutation, architecture-altering operations) with specified probabilities.

（4） After the termination criterion is satisfied, the single best program in the population produced during the run (the best-so-far individual) is harvested and designated as the result of the run. If the run is successful, the result may be a solution (or approximate solution) to the problem.

4.2 GP problems and Improvements

4.2.1 GP Problems

GP is a kind of slow and time-consuming method which takes lots of computer resources [10] [11]. First, each individual of the population should be run for several times in order to get their fitness values; second, the genetic process of each individual also takes a lot of time. What is more, GP needs many generations for a good solution. There are mainly two problems in the process of GP application: time-consuming and pre-maturing.

4.2.2 Improvements

To alleviate the above problems, three measures are taken in GP applications:

1) Shorten the calculating time

Leonardo VANNESCHI has done an experiment on FPGA (Field Programmable Gate Arrays) with a single PC (Intel Pentium III, Memory: 128M) [11] .The flowing GP parameters have been used in this experiment: population size of 500 individuals, ramped half-and-half initialization, crossover rate equal to 95%, mutation rate equal to 0.1%, maximum tree depth for the initialization phase equal to 6, maximum depth for the crossover and mutation phases equal to 50.

The total CPU time needed to execute the totality of all FPGA problems until generation 500 for 20 independent runs is 1448 minutes and 31 seconds (mean consuming time: 72 minutes and 25 seconds). The most time-consuming operation is fitness evaluation. We suppose that the population size is 5000, number of individuals for test is 30 and GP runs for 100 times with 300 generations each time, then the operation of fitness evaluation should be run for 5000*30*300*100=4500000000 times!

To shorten the calculating time for function fitness, we set a logical variable *isOld* for the attribute of each individual in the population. If an individual is from the previous generation without changes, this attribute value is logically true (*isOld*=true); on the contrary, if the individual is generated by crossover or mutation etc., this attribute value is logically false (*isOld*=false). So when evaluating fitness of the population, we first estimate the value of *isOld*. If its value is true, the program skips over it and go on to test the *isOld* value of the next generation; On the other hand, calculate the fitness. Loop until finishing all the population. The basic program structure is as below:

```
class Individual {
          //Individual Syntax Tree
          public Program program;
          //Individual Standard Fitness
          double standardizedFitness;
```

```
            //Individual Adjusted Fitness
            double adjustedFitness;
            //Individual Attribute
            boolean isOld;
        .
        .
        .
}
///// Evaluate Individual Fitness
void evaluateFitnessOfPopulation() {
            for (int i = 0; i < population.length; i++) {
   if( population[i].isOld)
            //do not evaluate old individual's fitness
            continue;
   else
            // evaluate new individual's fitness
            fitnessFunction(population[i], i);
              }
            }
```

2) Restrain pre-maturing

After several generations, there are many similar individuals in the GP population. To quantify the diversity of s population, three measures are taken in to account. They are: genotypic entropy, genotypic variance and phenotypic variance (see in [11] for the formulae).

Experiments show that the value of genotypic entropy is small in the beginning but grows big later. However, genotypic variance and phenotypic variance have the opposite trends. This phenomenon shows the GP has the premature trend. Many researchers have applied various methods (e.g. Phenotype and Genotype) to restrain premature and retain varieties of GP population. Though these methods are useful to a certain extent, their main shortcoming is consuming even more time than original GP system. Leonardo VANNESCHI uses the method named plagues -the idea of destroying the natural tendency of populations to maintain equilibrium between births and decrease rates. Therefore, in the method of plagues, a fixed number of individuals is suppressed from the population at each generation to reduce the computational effort individuals, thus to reduce the fitness evaluating time.

Instead of deleting fixed number of bad individuals, we select a set of individuals whose fitness are the worst and the depth of syntax tree are the deepest and mutate them as to maintain population varieties and restrain early-maturing. This method is easy to carry out and don't add extra codes to the original system.

Compared to plague method, we do not delete bad individuals from the population but sort them according to their depth. This method also retrains code bloating to some extent. The following is an experiment that shows the method shortens the stagnation time (stagnation generation is shortened from original 26 to 18):

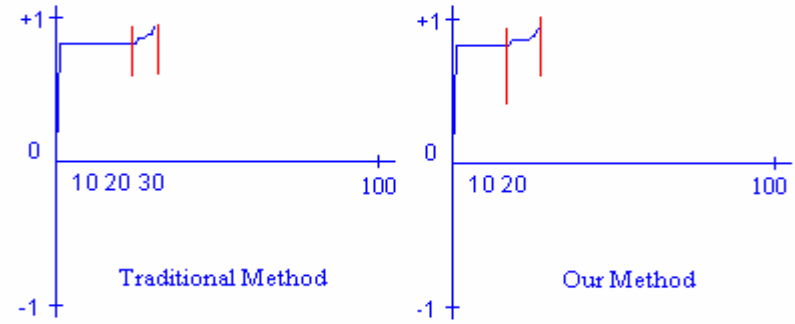

Fig.1. Our method shortens the stagnation time compared to traditional method

4.3 Insect Pests Forecasting Model Construction

Figure 2 shows the program flow chart. Gen is the number of generations, Max is the maxim generation, M means the population size and Pt, Pc, Pm represents probability of duplication, crossover and mutation. J is the individual index.

The function set F= {+, -, *, /, sin, cos}; Terminal set S={x, y, z}. We use error of forecast result as the standard fitness function represented by f (f=|Ti-Ti′|, Ti is i-th solution and Ti′ is the actual value). And the Adjusted fitness function is: Af =1/ (1+f). The following GP parameters used in this system are: population size of 5000 individuals, number of generation is 1000, ramped half-and-half initialization, tournament selection of size 10, crossover rate equal to 95%, mutation rate equal to 0.1%,maximum tree depth for the initialization phase equal to 4, maximum depth for the crossover and mutation phases equal to 50.

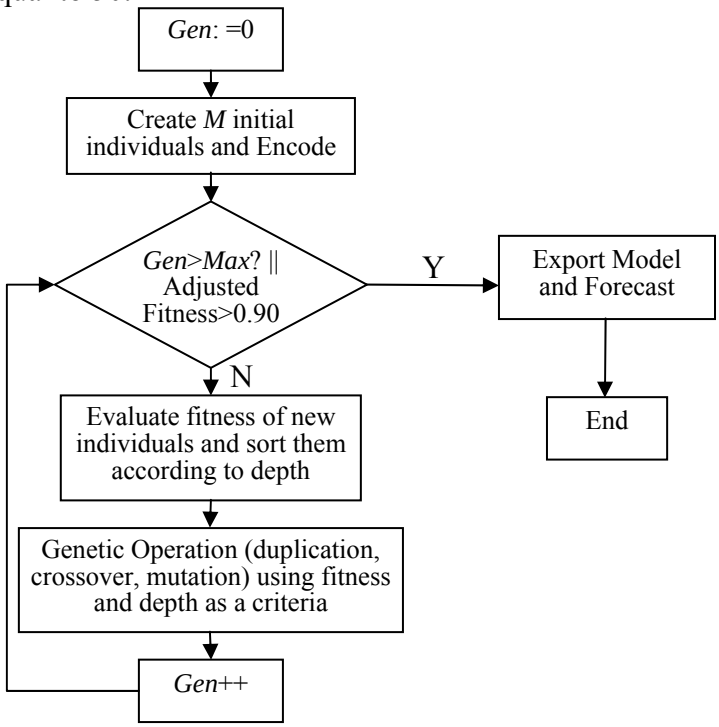

Fig.2. the Program Flow Chart

GP system finds the optimum model for Tianshui historical data sets at generation 340, the adjusted fitness value of which is 0.900025.

4.4 Comparison between GP and linear regression model

The model performance can be evaluated by the Root-Mean-Square Error (RMSE) using formula (4). Table 2 shows the comparison between GP and regression model.

Table 2. Comparison between GP Model and Regression Model. A-Grade: Actual Grade; RM: Regression Model; GPM: GP Model; H-Year: Year of History; F-Year: The Year to Forecast

Year		A-Grade	Fit Value		Error	
			RM	GPM	RM	GPM
H-Year	1979	3	2.312092	3.000896	-0.229303	0.000299
	1980	3	2.635497	3.010948	-0.121501	0.003649
	1981	1	1.605949	0.992404	0.605949	-0.007596
	1982	3	2.465233	2.007322	-0.178256	-0.330893
	1983	3	2.424687	3.003870	-0.191771	0.001290
	1984	4	3.353790	3.956979	-0.161552	-0.010755
	1985	5	4.889084	4.990460	-0.022183	-0.001908
	1986	1	1.719451	0.995308	0.719451	-0.004692
	1987	1	1.674773	1.016126	0.674773	0.016126
	1988	1	1.435958	0.995193	0.435958	-0.004807
	1989	3	3.609370	2.995891	0.203123	-0.001370
	1990	5	5.079013	4.967390	0.015803	-0.006522
	1991	3	2.615825	2.976454	-0.128058	-0.007849
F-Year	1996	5	4.813786	5.016249	-0.037243	0.003250
Root Mean Square Error					0.534089	0.275888

The RMSE of regression model is 0.534089, while RMSE of GP model is 0.27588. The maxim forecast error of GP model is 0.330893(1982), while the maxim error of regression model is 0.719451. Obviously, GP model performs well and the results are more precise.

Ⅴ Conclusions

In this paper, we presented the results of applying two modeling techniques to the Insect Pest Forecasting problem in Tianshui: the traditional regression model and the GP model. GP is capable of discovering a simple model structure with less error and has an advantage over regression model from the comparison in the paper. Thus, simpler and accurate models can be automatically discovered using GP to the needs of the application.

At the same time, we proposed two measures to improve GP efficiency. First, we set a logical variable *isOld* for the attribute of each individual in the population. GP doesn't evaluate those individuals whose attribute *isOld* is true. So it shortens the fitness evaluating time. Second, we take into account not only the performance of individuals (fitness), but also the depth of individuals thus to retain the population varieties and shorten the stagnation time.

References

[1]. Alaa F.Sheta, Ahmed Mahmoud: Forecasting using genetic programming. Southeastern Symposium on System Theory.pp.343-347, 2001
[2]. Lin Guo, De-Shuang Huang and Wenbo Zhao, "Combing genetic optimization with hybrid learning algorithm for radial basis function neural networks", Electronics Letters Online No: 20031021, IEEE 2003, 2 July 2003.
[3]. Xu Guanghu: Mid-long term load forecasting in power system by genetic programming. Relay, pp.21-24, Vol.32 No.12, June 16, 2004
[4]. Miao Li, Technical report of Gansu. Hefei, 2000
[5]. Hu Qingya, Wei Yiming: Linear Programming and Its Application. Science Press, March, 2004.
[6]. John R. Koza: Genetic Programming: On the Programming of Computers by Means of Natural Selection. The MIT Press, (1992)
[7]. Hitoshi Iba, Takashi Sasaki: Using Genetic Programming to Predict Financial Data. IEEE, pp.244-250, 1999
[8]. Wolfgang Banzhaf, Peter Nordin, etc.: Genetic Programming: An Introduction: On the Automatic Evolution of Computer Programs and Its Applications. Morgan Kaufmann Publishers, Inc. December 1997
[9]. Nao Tokui & Hitoshi lba, Empirical and Statistical Analysis of Genetic Programming with Linear Genome. IEEE, 1999
[10]. Leonardo Vanneschi and Marco Tomassini: A Study on Fitness Distance Correlation and Problem Difficulty for Genetic Programming. New York: AAAI, pp. 307—310, 2002
[11]. Leonardo VANNESCHI: Theory and Practice for Efficient Genetic Programming. Leonardo VANNESCHI PhD thesis, Switzerland, 2004