Genetic Programming for Classification of Remote Sensing Data

Jihn-Fa Jan[1,2]

【Summary】

The overall objective of this research was to develop an adaptive machine learning technique for the classification of remote sensing data. The genetic programming paradigm was implemented to classify vegetation, bare soil, and burnt-over areas using SPOT multispectral data. Two SPOT imageries obtained on 31 Dec. 1986 and 15 Jan. 1988 were used in this study. The results show that the genetic programming paradigm was very effective in classifying the data set (e.g., the best classification accuracy obtained was 99% for the training samples). Moreover, the computer programs derived from genetic programming allowed important variables for classification to be identified. Classification results for the machine learning approach were then compared to the results obtained using a conventional statistical approach (i.e., the Gaussian maximum likelihood classifier). The comparison shows that the classification results for both approaches are similar.

Key words: genetic programming, machine learning, image classification.

Jan, J. F. 1998. Genetic programming for classification of remote sensing data. Taiwan J. For. Sci. 13(2): 109-118.

遺傳程式設計法於遙測影像分類之研究

詹進發[1,2]

摘　要

本研究之目的是在探討機器學習方法應用於遙測影像分類之可行性，並利用SPOT衛星資料以遺傳程式設計法進行分類，以區分植生、裸土及火災跡地。分類結果顯示遺傳程式設計法可以有效分類遙測影像，以訓練樣本進行分類之精確度可達99%，遺傳程式設計法所自動產生之電腦程式並可用於選取分類所需之重要變數。機器學習方法分類結果並與傳統之統計方法分類結果相互比較，結果顯示二者之分類效果相似。

關鍵詞：遺傳程式設計法、機器學習、影像分類。

詹進發。1998。遺傳程式設計法於遙測影像分類之研究。台灣林業科學13(2): 109-118。

---

[1] 國立台灣大學森林學系，台北市羅斯福路四段 1 號 Department of Forestry, National Taiwan University. 1 Sec. 4, Roosevelt Rd., Taipei, Taiwan, ROC.

[2] 通訊作者 Corresponding author

1997 年 8 月送審　1998 年 4 月通過 Received August 1997, Accepted April 1998.

C.E.P.S.

INTRODUCTION

Background

With the launch of Landsat 1 in 1972, the acquisition of data concerning natural resources entered a new era. During the past 2 decades, remote sensing technology has evolved very rapidly, and it has been applied in various disciplines with various satisfactory levels of success (Avery and Berlin, 1992). Major advantages of remote sensing include its ability to obtain digital data over large geographic areas very rapidly and its cost effectiveness. The data contain spectral reflectance of different wavelengths for surface features which form spectral response patterns that can be utilized for computer categorization of land cover types (Hoffer, 1979; 1986).

Image classification is one of the most common applications of data analyses using remote sensing data; however, the task of extracting features from remote sensing data can be very difficult because of the complexity of spectral patterns within the data (Avery and Berlin, 1992; Hoffer, 1986). In the past, many computer-assisted techniques have been developed for image classification using remote sensing data. Most of these techniques use statistical approaches to perform data analysis (Hoffer, 1986). However, these techniques often require the analyst to have a high level of expertise with regard to variable selection, data transformation, selection of appropriate analysis models, and interpretation of results. In some cases, the data analysis procedures require a large amount of human interactions that make the implementation more difficult and inefficient (Hoffer, 1986; Swain, 1978; 1982).

High resolution sensors aboard satellites are capable of producing imagery on a daily basis that contains trillions of bytes of data. The demand for techniques that can discern spectral patterns within image data more accurately and efficiently have increased as the quantity and complexity of image data grow. The quest for more advanced techniques has drawn much interest among researchers (Lillesand and Kiefer, 1994).

Genetic algorithm and genetic programming

Machine learning techniques have been the focus of attention for a wide range of research over the past 30 yr (Hertz *et al.*, 1991; Levy, 1992). However, slow learning rates and requirements of more advanced computing equipment make them impractical for many applications (Hertz *et al.*, 1991). With the advance of computer technologies, the availability of inexpensive high performance computers results in more practical use of these techniques for various applications (Levy, 1992).

The genetic algorithm is an adaptive machine learning model derived from simulating the mechanisms of natural evolution, which has been used as a global optimization algorithm in a wide variety of applications since its advent. This algorithm can also be used for constructing classifiers for pattern recognition (Goldberg, 1989; Booker *et al.*, 1989). The genetic programming extends the genetic algorithm paradigm to automatically generate computer programs that can solve various problems (Koza, 1992).

This research focused on applying the genetic programming paradigm to develop a classifier for image classification using remote sensing data. The classification results of genetic programming were compared to the results of image classification using a conventional statistical approach. The overall objective of this research was to develop adaptive machine learning techniques and to define their effectiveness for the classification of remote sensing data.

MATERIALS AND METHODS

Materials

The study area of this research is located in compartment 66 of the Moogua Forest District managed by the Taiwan Forestry Bureau. Two SPOT multispectral data (210 x 200 pixels), as shown in Figs. 1 and 2, obtained on 31 Dec. 1986 and 15 Jan. 1988 were used in this study. A forest fire occurred in this area on 6 Feb. 1987. The burnt-over area is shown in gray-bluish tone in Fig. 2 as indicated by the white square block. Fig. 3 depicts the drawing for the burnt-over area based on a ground investigation conducted by the staff of the forest district.

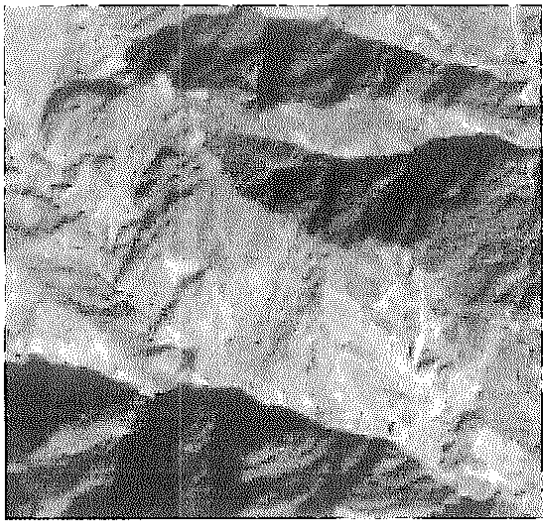


Fig. 1. SPOT image obtained before fire occurred (31 Dec. 1986). The shadow area is shown in dark tone (image size: 210 x 200 pixels).

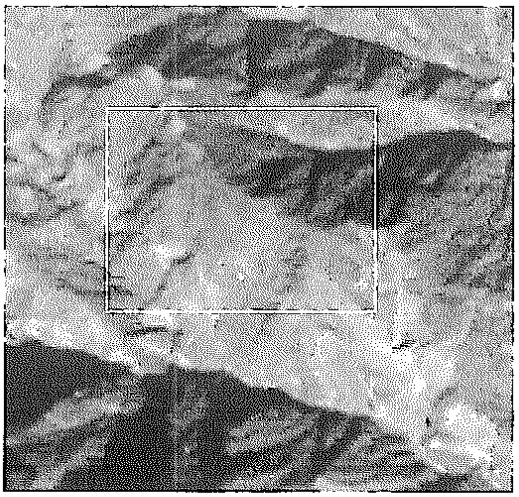


Fig. 2. SPOT image obtained after fire occurred (15 Jan. 1988). The burnt-over area is shown in gray-bluish tone in the white square block (image size: 210 x 200 pixels).

SPOT imageries were provided by the Center for Space and Remote Sensing Research (CSRSR), National Central University. Geometric rectification for the imageries was done by CSRSR using a digital terrain model and ground control points. A forest cover map provided by the Taiwan Forestry Bureau was used as ancillary data for selection of training samples. The study area was thoroughly studied by Chou and Cheng

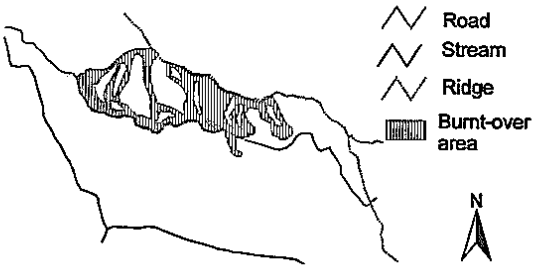


Fig. 3. Drawing of burnt-over area.

(1989), and these data sets and the associated reference data were generously made available for the current study.

Processing and analysis of data was performed on an IBM compatible personal computer. ERDAS/IMAGINE was used to process the image data, and to export the image data to text format that can be interpreted by other software. For the machine learning model, a genetic programming core system called lil-gp, designed by Zongker and Punch (1996) at Michigan State University, was used to develop the classifier for the classification of remote sensing data.

Genetic algorithm

The genetic algorithm (GA) is an adaptive machine learning model derived from simulating the mechanisms of natural evolution. Inspired by the mechanisms of biological organisms adapting to the environment through evolution, John H. Holland, and his students at the University of Michigan, developed the algorithm in the 1960s. A complete description of the GA was first published in 1975. During the 1980s, a rapid increase in interest of applying GA in solving various problems resulted from an increasing recognition of GA's learning capability and knowledge that GA provides a tool in areas that do not yield readily to standard approaches (Holland, 1992; Levy, 1992).

Natural populations evolve through many generations according to the principles of natural selection and "survival of the fittest", which was demonstrated clearly by Charles Darwin in *The Origin of Species*. By mimicking this process, a

genetic algorithm is able to "evolve" solutions to real world problems, given that the genetic algorithm is suitably encoded (Beasley *et al.*, 1993). In the genetic algorithm paradigm, a population is composed of fixed-length strings (chromosomes) that represent possible solutions to a given problem. Through a simulated evolution process, the individuals compete to breed new, and hopefully better, solutions to the problem. After a sufficient number of generations, optimal or near optimal solutions are found for the problem (Goldberg, 1989).

In the genetic algorithm, the individuals that comprise a population are generated in a stochastic manner. The initial population is created by a large number of randomly generated strings. There is a fitness measure associated with each string in the population. The fitness measure is determined by a fitness function, or objective function, that we want to maximize. Then the population evolves to hopefully find better solutions in successive generations through 3 operators: reproduction, crossover, and mutation.

Reproduction is a process of copying strings, however, it is biased according to their fitness measure. This is an artificial version of natural selection, by which the strings with a higher fitness measure are reproduced with higher probabilities, thereby more of their offspring enter into the next generation. There are a number of different selection schemes, of which the following 2 methods are commonly used:

1. Fitness-proportionate selection: The individuals are selected with probabilities in proportion to their fitness measure:

$$p(s_i) = \frac{f(s_i)}{\sum_{j=1}^{n} f(s_j)}$$

where n is the population size, and $f(s_i)$ is the fitness measure of the ith individual string.

2. Tournament selection: A number of individuals are selected at random from the population with uniform probability, and only the best individual within this subset is reproduced.

After reproduction, the individuals within the newly created population are randomly selected for mating--crossover. A crossover point is randomly chosen with uniform probability between 1 and the string length less 1. The pair of strings involved in crossover then exchange substrings divided at the crossover point and create 2 new strings. For example, consider 2 strings S_1 and S_2 randomly selected from the population:

original strings before crossover:

Father: $S_1 = 11010||1101010$

Mother: $S_2 = 10001||1100011$

resulting strings after crossover:

Child 1: $S_1 = 11010||1100011$

Child 2: $S_2 = 10001||1101010$,

in which the crossover point is chosen as between positions 5 and 6. As shown above, the new strings inherit some features from both of their parents. These 2 new strings become part of the population in the next generation.

By using reproduction and crossover based on the fitness measure, the population is able to improve over time, i.e., more "fitter" strings appear in successive generations. Although these 2 operators seem to be quite simple, it is the combination of these 2 operators that allows the genetic algorithm to gain its power in solving a host of different problems.

Occasionally, after crossover, a string undergoes a process called mutation, which is simply flipping the binary value at a randomly chosen string position. For example,

original string before mutation:

S = 11010**1**101010

resulting string after mutation:

S = 11010**0**101010.

Mutation is applied with a small frequency (typically 0.001), and thus plays a secondary role in the operations of a genetic algorithm. The purpose of mutation is to maintain diversity in the population. Though the reproduction and crossover operators can find useful genetic information effectively, potentially useful genetic materials may be lost in the process. Therefore the mutation operator is applied to prevent loss of potentially useful genetic information.

In summary, a typical genetic algorithm includes the following aspects:

1. Encoding: The parameters to a potential

solution for a given problem must be encoded as a string. Typically a string is composed of fixed-length bits array or characters; however, there is a substantial amount of research that has implemented variable-length strings or different structures.

2. Fitness function: A fitness function has to be designed for the given problem. When a set of parameters is presented in the form of a string, the associated fitness measure is evaluated using the fitness function. The relative fitness level, or goodness, of a string is determined by its fitness measure.

3. Evolution: A population is initialized with a large number of randomly generated strings, which represent potential solutions to a given problem. This is analogous to searching for a solution to a given function with many different starting points. The population then undergoes an artificial natural selection process to find better solutions in successive generations until a termination criterion is satisfied. Fig. 4 shows the general procedures for a typical genetic algorithm.

Genetic programming

The genetic algorithm paradigm has been applied in many applications with satisfactory results. As shown in the previous section, a conventional genetic algorithm requires the solutions for a given problem to be represented in fixed-length strings. For many problems, however, this representation scheme may not be appropriate because the size and components of solutions are not known in advance (Koza, 1992).

```
// A typical genetic algorithm
InitializePopulation;
EvaluateFitnessMeasure;

while TerminationCriterionNotMet do
{
   Reproduction;  // reproduction biased in favor of fitter ones
   i := 1;
   while (i <= PopulationSize / 2) do
   {
      SelectTwoIndividuals;
      Crossover;  // crossover to create two offspring
      ReplaceParentsWithOffspring;
      i := i + 1;
   }
   Mutation;      // this is done in small frequency
   EvaluateFitnessMeasure;
}
SelectSurvivors;  // select the survivors as the results
```

Fig. 4. Pseudo code for a typical genetic algorithm.

The genetic programming (GP) extends the genetic algorithm paradigm to breed computer programs that can solve various problems. Instead of fixed-length strings that encode possible solutions to a given problem, the objects that constitute the population are computer programs composed of functions and terminals appropriate to the problem. These computer programs, when executed, are the potential solutions to the problem.

In genetic programming, a computer program is an S-expression (symbolic expression) of the LISP programming language. A LISP S-expression is a type of prefix notation. For example, (* 3 5) is a LISP S-expression. This S-expression calls the multiplication function with 2 arguments, and the value returned by evaluation of this function is 15. An S-expression can also be represented as the parse tree of a program. Fig. 5 depicts 2 individual computer programs (S_1 and S_2) in a population. Equivalently, these 2 tree-structured computer programs can be represented in terms of the following S-expressions:

S_1 = (* A (+ B C))

S_2 = (- (/ C D) (IF (> B 1) 3 4)).

Genetic programming starts with an initial population of randomly generated computer programs in the form of S-expressions as shown in Fig. 5. Each computer program is composed of functions and terminals. The functions can be arithmetic operators, logical functions, mathematical functions, or domain-specific functions. The terminals are usually input values or randomly generated constants. When these computer programs are executed, the results are potential solutions to a given problem. Similar to the genetic

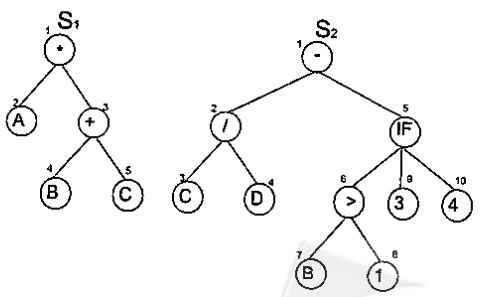


Fig. 5. Two example computer programs.

algorithm, this initial population is actually a random search of the search space of the problem.

Each individual computer program within the population is associated with a fitness measure that is determined by how well it can perform a problem-specific task. Typically, each computer program is run over a number of different fitness cases and the fitness is measured as a sum or an average performance over these cases. For example, in the classification problem, the fitness of a computer program is a measure of the number of input patterns that it can correctly classify. Unless the given problem is very simple, the computer programs in the initial population (i.e., generation 0) generally do not yield good results. Similar to the genetic algorithm, an artificial version of natural selection is used to breed offspring population that hopefully will contain more "fitter" individual computer programs.

The reproduction operation involves 2 steps, i.e., selection (biased in favor of fitter ones), and duplication. First, a computer program is selected from the current population of computer programs using some selection scheme based on fitness (e.g., fitness-proportionate selection, and tournament selection). Second, the selected program is copied, without alteration, into the new population of successive generation. This is asexual reproduction, because only 1 parental S-expression is involved, and 1 offspring S-expression is produced.

The crossover operation is sexual reproduction in that 2 new offspring computer programs are created from 2 parental programs selected in proportion to fitness. Typically, the parental programs have different sizes and shapes. The operation begins by independently selecting a random point in each parental program as the crossover point for that parent. Then the 2 parental programs exchange subexpressions divided by the crossover points to create 2 new offspring programs. For example, let us consider the S-expressions S_1 and S_2 in Fig. 5 as the parental programs. Further assume that the crossover points for S_1 and S_2 are randomly selected as 5 and 2, respectively. As shown in Fig. 6, the resulting offspring programs are 2 new programs with different shapes and sizes than their parents, which are composed of subexpressions from the parental programs.

After reproduction and crossover, the population of offspring programs replaces the old population from the previous generation. The fitness of each computer program in the new population is evaluated over all the fitness cases, and the process is repeated for many generations until a termination criterion is met. Typically, the best individual program that appeared in any generation during the process is designated as the result of genetic programming.

It is worth noting that genetic programming usually does not use the mutation operator. In the genetic algorithm, mutation serves as a secondary operator to restore lost diversity in a population. In genetic programming, the functions and terminals usually do not disappear entirely from a population. Moreover, when the crossover points of both parents happen to be endpoints of trees, the crossover operator works in a manner similar to mutation. Therefore, mutation is not needed in genetic programming.

In contrast to the genetic algorithm and other techniques, the genetic programming paradigm has several important features:

1. Hierarchical characteristic: The computer programs that undergo adaptation are tree-like hierarchical structures composed of functions and terminals that are appropriate to a problem. The results of genetic programming are programs of hierarchical structure, which allow hierarchical interpretation of different parameters for a problem.

2. Dynamic variability: The sizes and shapes of potential solutions change as the evolution process proceeds. If the size and shape of a

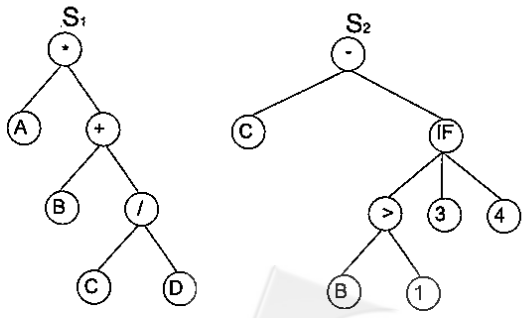


Fig. 6. Two offspring produced by crossover.

solution is fixed in advance, it may limit what the system can solve. This feature makes genetic programming more suitable in solving some problems.

3. Data representation: In genetic programming, the inputs and outputs are usually in their natural forms. Preprocessing of data or post-processing of outputs are either minor or unnecessary.

4. Interpretation of results: Unlike some techniques that use encodings for a solution, the results of genetic programming are computer programs that can be readily used to solve a given problem. Moreover, the structure of the resulting programs may provide better insight into the problems and related parameters.

In summary, the general procedures for the genetic programming paradigm follow these 5 steps (Koza, 1992):

1. Initialization: Create an initial population of computer programs by randomly composing functions and terminals of the given problem.

2. Fitness evaluation: Run each computer program within the population to determine its fitness according to how well it can perform a problem-specific task.

3. Decision of termination: If the termination criterion has been satisfied, perform the results designation step; otherwise, perform the creating new population step, and then return to the fitness evaluation step.

4. Creating new population: Create a new population of computer programs by using the reproduction and crossover operations. These 2 operations are applied to computer programs chosen from the population using some selection scheme based on fitness.

(a) Reproduction: Duplicate the selected existing computer programs to the new population (i.e., asexual reproduction).

(b) Crossover: Create 2 new offspring computer programs from 2 existing programs by exchanging randomly chosen parts of these 2 programs (i.e., sexual reproduction).

5. Results designation: Designate the best computer program that appeared in any generation as the result of genetic programming and terminate the process. The resulting program may be a solution to the given problem.

In the study, implementation of the genetic programming paradigm follows these 5 major steps:

1. Determining the terminal set: The terminals are usually input values or randomly generated constants. In this study, the input values were the spectral reflectance measurements of each pixel, i.e., the pixel values of each spectral band.

2. Determining the function set: The 2nd step is to identify a set of functions that are used to generate mathematical expressions that can be used as discriminant functions for classification of the input data. The functions can be arithmetic functions, logical functions, mathematical functions, Boolean functions, or domain-specific functions. Each function may take 1 or more arguments.

3. Determining the fitness measure: In this study, genetic programming was used to construct classifiers for classification purposes. Each individual computer program within a population was associated with a fitness measure that was measured by how well it could perform the classification task.

4. Determining the controlling parameters: The 2 major parameters that control the genetic programming paradigm are the population size and the maximum number of generations to be run. Experiments with different sets of controlling parameters were done to determine appropriate parameters for the classification task in this study.

5. Determining the termination criterion and methods for results designation: Genetic programming starts with an initial population of randomly generated computer programs in the form of S-expressions as shown in Fig. 5. An artificial version of natural selection using the reproduction and crossover operations is used to breed offspring populations that hopefully will contain more "fitter" individual computer programs. In this study, numerous runs were executed to find computer programs that were used as classifiers for the given data set. The termination criterion was set as (1) terminating the process when all input patterns were correctly classified, or (2) terminating the process if the maximum number of generations was reached.

C.E.P.S.

For each run, after termination of the learning process, the "fittest" computer program (i.e., the computer program having the highest overall classification accuracy) was designated as the result of that run. The computer program with the best performance among these runs was considered as the best classifier for the given data set.

RESULTS AND DISCUSSIONS

The original SPOT data contain a large proportion of shadow areas, which were removed from the image data before classification. The shadow-free SPOT data were then categorized into 3 land cover types: vegetation, bare soil, and burnt-over area. In order to adequately represent the variations in spectral patterns of each cover type, 2 to 3 training areas were selected from spectrally homogeneous regions in each cover type. A total of 1311 pixels were selected from these cover types as the training samples, which comprised more than 3% of the original data and 5% of the shadow-free data, respectively.

In this study, a hierarchical classification approach was applied to classify the data, i.e., the shadow-free data were classified into soil and non-soil cover types first, and then the non-soil cover type was further classified into vegetation and burnt-over area. Two classification approaches were implemented in this study, i.e., the genetic programming approach, and a conventional statistical approach (Gaussian maximum likelihood classifier). Evaluation of the classification accuracy was done based on the classification results of the training samples.

Since the genetic programming starts with a set of randomly generated potential solutions to a given problem, it usually yields different results for different runs. For 1 run, genetic programming resulted in a computer program that was able to classify soil and non-soil with 100% accuracy. The program was simply a logical expression of the form (LT 17 B2). This program indicated that if the spectral value of band 2 for a pixel was less than 17, it must be non-soil, otherwise, it was soil. For classifying vegetation and burnt-over area, genetic programming resulted in a computer program that achieved 99% accuracy by using spectral values of bands 2 and 3. The results show that genetic programming is capable of selecting important variables, without human interactions, for the classification task. Alternatively, the same results may also be achieved using conventional statistical approaches such as step-wise discriminant analysis; however, more effort is usually required from the analyst. Furthermore, these statistical approaches often require the analyst to have a high level of knowledge regarding the nature of the given data, interpretation of the results, and selection of appropriate statistical models for analysis.

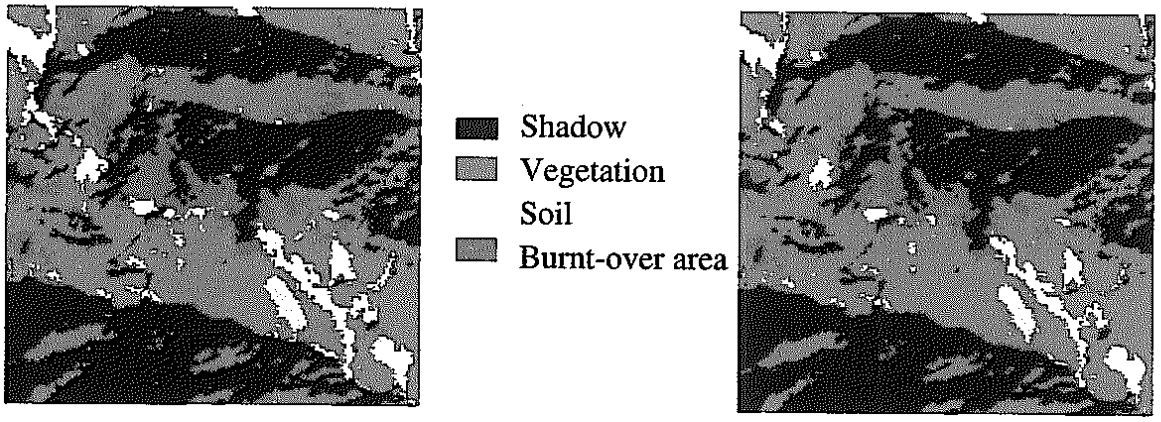
Table 1 depicts the classification results of the training samples for both the genetic programming approach and the statistical approach. Classification of the training samples shows that the overall classification accuracies for the genetic programming approach and the statistical approach were 99.3% and 98.9%, respectively. The resulting classifiers for both approaches were used to classify the entire shadow-free data, and the classification results are shown in Table 2. The cover type maps resulting from classification of the entire shadow-free data are depicted in Fig. 7. The results indicate that more pixels were classified

Table 1. Comparison of classification results of training samples

Cover type	Total pixels	Number of pixels correctly classified	
		GP[1)]	GML[2)]
Soil	198	198	198
Vegetation	899	896	888
Burnt-over	214	208	210
Total	1311	1302	1296
Overall accuracy (%)		99.3	98.9

[1)] GP: genetic programming.
[2)] GML: Gaussian maximum likelihood.

C.E.P.S.



Result of statistical approach **Result of genetic programming**

Fig. 7. Classification results of genetic programming approach and statistical approach.

Table 2. Classification results of the entire shadow-free data

Cover type	GP[1)]		GML[2)]	
	Number of pixels	%	Number of pixels	%
Soil	2085	8.2	2906	11.4
Vegetation	19305	75.9	17014	66.9
Burnt-over	4035	15.9	5505	21.7
Total	25425	100.0	25425	100.0

[1)] GP: genetic programming.
[2)] GML: Gaussian maximum likelihood.

as vegetation, and less pixels were classified as burnt-over and soil using the genetic programming approach, as compared to the results using the statistical approach. Though it is desirable to evaluate the classification accuracy using ground truth reference data, the accuracy assessment was done by visually examining the resulting cover type maps due to lack of geometric reference data in the original illustration (Fig. 3) showing the burnt-over area. From visual comparison of the cover type maps shown in Fig. 7, it appears that these 2 approaches achieved similar results. However, the resulting cover type map of the genetic programming approach shows that there were a few patches of vegetation surrounded by burnt-over areas, which was more accordant with the ground investigation as shown by Fig. 3 as compared to the map obtained from the statistical approach.

CONCLUSIONS

In this study, a classifier using machine learning techniques (i.e., the genetic programming paradigm) was developed to aid in land cover type classification using remote sensing data. The classification results of the genetic programming were compared to the results of a conventional statistical classification approach (i.e., the Gaussian maximum likelihood classifier).

The following conclusions can be drawn based on the results as described in the previous section:

1. The genetic programming approach was able to classify vegetation, bare soil, and burnt-over area within the SPOT data effectively. Classification results of the training samples show that the performance of the genetic programming approach was similar to the statistical approach (99.3% vs. 98.9%).

2. The computer programs generated by genetic programming can provide information about important variables for the classification task. For example, (LT 17 B2) was generated by genetic programming to identify soil and non-soil, which indicates that only 1 band (i.e., band 2) is needed to identify bare soil within the given data set. Similarly, the genetic programming approach

resulted in a computer program that used only 2 bands (i.e., bands 2 and 3) to classify vegetation and burnt-over area. More importantly, this was achieved by machine learning without human interactions.

3. Classification of the entire shadow-free data for both the genetic programming approach and the statistical approach achieved similar results. Visual comparison of the resulting cover type maps produced by these 2 approaches indicates that the genetic programming approach had better performance than did the statistical approach in delineating the burnt-over area.

ACKNOWLEDGEMENTS

The author is very grateful to Dr. Chi-Chuan Cheng, who provided the data and support for this study.

LITERATURE CITED

Avery, T. E., and G. L. Berlin. 1992. Fundamentals of remote sensing and air-photo interpretation. 5th ed., Macmillan Publishing, New York. 472 pp.

Beasley, D., D. R. Bull, and R. R. Martin. 1993. An overview of genetic algorithms: part 1, fundamentals. Univ. Computing 15(2): 58-69.

Booker, L. B., D. E. Goldberg, and J. H. Holland. 1989. Classifier systems and genetic algorithms. Artificial Intelligence 40: 235-282.

Chou, C. F., and C. C. Cheng. 1989. A study on change detection of forest land-cover using SPOT data. Bull. Taiwan For. Res. Inst., New Series 7(4): 305-317.

Goldberg, D. E. 1989. Genetic algorithms in search, optimization, and machine learning. Addison-Wesley Publishing, Reading, Massachusetts. 412 pp.

Hertz, J., A. Krogh, and R. G. Palmer. 1991. Introduction to the theory of neural computation. Addison-Wesley Publishing, Read-ing, Massachusetts. 327 pp.

Hoffer, R. M. 1979. Computer-aided analysis of remote sensor data magic, mystery, or myth? Pages 156-179 *in* Proceedings of the Symposium on Remote Sensing for Natural Resources: An International View of Problems, Promises, and Accomplishments. University of Idaho, Moscow, Idaho.

Hoffer, R. M. 1986. Digital analysis techniques for forestry applications. Remote Sensing Rev. 2: 61-110.

Holland, J. H. 1992. Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence. 1st MIT Press ed., MIT Press, Cambridge, Massachusetts. 211 pp.

Koza, J. R. 1992. Genetic programming: on the programming of computers by means of natural selection. MIT Press, Cambridge, Massachusetts. 819 pp.

Levy, S. 1992. Artificial life: the quest for a new creation. Pantheon Books, New York. pp. 153-187.

Lillesand, T. M., and R. W. Kiefer. 1994. Remote sensing and image interpretation. 3rd ed., J. Wiley, New York. pp. 413-422.

Swain, P. H. 1978. Fundamentals of pattern recognition in remote sensing. Pages 136-187 *in* P. H. Swain, and S. M. Davis, eds. Remote sensing: the quantitative approach. McGraw-Hill, Inc.

Swain, P. H. 1982. Pattern recognition techniques for remote sensing applications. Pages 609-620 *in* P. R. Krishnaiah, and L. N. Kanal, eds. Handbook of statistics 2: classification, pattern recognition and reduction of dimensionality. North-Holland Publishing, Amsterdam.

Zongker, D., and B. Punch. 1996. lil-gp 1.01 User's Manual. Michigan State University. 62 pp.

C.E.P.S.