Pattern Discovery and Model Construction: an Evolutionary Learning and Data Mining Approach

Harry Zhou
Department of Computer and Information Sciences, Towson University.
Baltimore, MD, USA
WWW home page: http://www.towson.edu/~zhou

Abstract. In the information age, knowledge leads to profits, power and success. As an ancestor of data mining, machine learning has concerned itself with discovery of new knowledge on its own. This paper presents experiment results produced by genetic algorithms in the domains of model construction and event predictions, the areas where data mining systems have been focusing on. The experiment results have shown that genetic algorithms are able to discover useful patterns and regularities in large sets of data, and to construct models that conceptualize input data. It demonstrates that genetic algorithms are a powerful and useful learning algorithm for solving fundamental tasks data mining systems are facing today.

1 Introduction

In the information age, knowledge leads to profits, power and success. Thus, the demand for information in our society makes discovery of useful patterns and regularities in large sets of data a necessity, while the fast growth of CPU speeds and memory sizes makes it possibility. In the past decade, it gave a birth to data mining, a technology that discovers meaningful new correlations and trends by shifting through large amounts of data[4]. As an ancestor of data mining, machine learning has concerned itself with making computers that learn things for themselves. Since the birth of data mining, it provides many algorithms and techniques for data mining systems to explore large sets of data and discover new knowledge[2]. This paper presents experiment results produced by genetic algorithms in the domains of model building and event predictions, the areas where data mining systems have been focusing on. The purpose of this paper is to demonstrate that genetic algorithms are a powerful and useful tool for solving fundamental tasks data mining systems are facing today. In what follows, a brief description of genetic algorithms is described, followed by a discussion of genetic learning and data mining. Genetic learning

experiments conducted in two different domains along with their results are presented. It concludes with future research topics.

2 Outline of Genetic Algorithms

Genetic algorithms are known as an evolutionary approach designed to reflect the idea of "Survival of the fittest". These algorithms encode a potential solution to a specific problem on a simple chromosome-like data segment and apply genetic operators, such as mutation and crossover, to these segments so as to preserve critical information.

A general sketch of a genetic algorithm is given below:

```
t = 1
Initialization P(t) // P(t) is the population at time t
Evaluation P(t);
While (termination condition is not true)
  t = t + 1;
     Selection P(t)
  Offspring Generation P(t)
  Replacement P(t)
  Evaluation P(t)
```

It begins with a population of candidate solutions, usually generated randomly in the Initialization phase. In the Evaluation phase each candidate solution is evaluated and assigned a score indicating how well it performed with respect to the target problem. In the Selection phase better candidate solutions are selected and given more chances to "reproduce" than poorer candidate solutions. The offspring generation is accomplished by applying the genetic operators Mutation and Crossover. The idea is that some good genes from their parents should be inherited by their offspring. The genetic operators produce new instances in such a way that offspring still resemble, but not identical to, their parents. Since the population is limited in size, some poor candidate solutions are replaced by the newly produced children. This process is repeated until termination conditions are met.

3 Descriptive Genetic Learning

A model is a high-level, global description of a data set summarizing the main features. The goal of descriptive data mining is to construct a model that describes all of the data from limited observations. The fundamental objective is to produce insight and understanding about the structure of the data, and to enable us to see its important characteristics. Examples are models(segmentation) used in marketing to divide customers into groups based on purchasing patterns, and cluster analysis used in psychiatric research to describe illness. In this section, a genetic learning approach is presented along with its experimental results in the domain of model construction

and descriptive learning. More specifically, a set of training instances are given to the genetic algorithm, and the expected output from the genetic learning system is a model that describes all positive instances and rejects all negative instances in the domain of regular expressions and finite automata[1,5].

A finite automaton can be described by a 5-tuple (Q, S, δ, q_0, F), where Q is a finite set of states, S is a finite input alphabet, q_0 in Q is the initial state, $F \subseteq Q$ is the set of final states, and δ is the transition function mapping the cartesian product of Q and S into Q. That is, δ(q, a) is a state for each state q and an input symbol a. An example X is said to be accepted by a finite automaton M = (Q, S, δ, q_0, F) if δ(q_0, X) = p for some p in F. The language accepted by M, designated L(M), is the set {x | δ(q_0, X) is in F}. A language is a regular expression if it is a set of input symbols accepted by some finite automata.

In out experiments, a concept to be learned is a language over {0,1} where the grammar is described either by a finite automaton, English, or a regular expression. It is assumed that the teacher uses either English or a regular expression to specify a concept while the genetic algorithm, based on a set of examples, searches for a finite automaton that represents the concept. In other words, the learning task is not simply to construct a finite automaton but rather to find a finite automaton that must be equivalent to the regular expression the teacher has in mind. The input to the genetic algorithm consists of one set of positive examples that must be accepted by the finite automaton and one set of negative examples that must be rejected by the finite automaton. The output from the genetic algorithm is a finite automaton which is expected to be equivalent to the described concept.

In the following experiments, the process of constructing a finite automaton based on a set of data can be viewed as a search in the space of all possible finite automata. With the maximum number of states limited to 8, each candidate solution in the population is denoted by the following form:

$$((X_1, Y_1, F_1)\ (X_2, Y_2, F_2)\ \ldots\ (X_8, Y_8, F_8))$$

where each (X_i, Y_i, Z_i) represents the state i. X_i and Y_i correspond to the destination states of the 0 arrow and the 1 arrow respectively. F_i is represented by three bits. The first two bits of Fi are used to indicate whether there exists an arrow coming from state i. The third bit of Fi shows whether the state i is a final state.

3.1 Finite Automata Construction

The learning task can be summarized as follows. Given a list of positive examples and a list of negative examples, the learning system is expected to construct an abstract description in the form of a model. The produced model should not only accept all the positive examples, and reject all the negative examples, it also is expected to characterize these examples conceptually. It is known that the problem of finite automata construction based on examples is a NP-complete problem, and the hill-climbing has been applied to it with limited success [1,5]. As a global search

algorithm by nature, the capabilities of genetic algorithms are tested in the following experiments.

3.2 Experiment Design

Good examples must characterize the main features of the concept the teacher has in mind. It is reasonable to assume that neither humans nor learning algorithms are able to abstract a concept from a set of poorly selected examples. The following experiment shows how a set of examples are selected and how an experiment is designed. The concept the teacher has in mind is: (10)*, which is a sequence consisting of any number pairs of 1 and 0.

The positive example set:

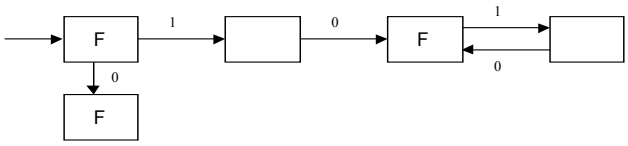
(10) (1010) (101010) (10101010101010) (1010101010101010)

The negative example set:

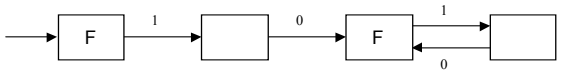
(101) (000) (1001010) (10110) (1) (0)

The output produced by the genetic algorithm based on the above examples is shown below:

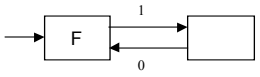
where the letter F in a box indicates the state is a final state.



The finite automaton produced by the genetic algorithm agrees with the given positive examples and negative examples. It is, however, not the one specified by the regular expression (10)*. A close look reveals that the learning algorithm did not understand that an input of 0 should be rejected. Once the training set is modified with a 0 added to the negative example set, the genetic algorithm was able to produce the following two finite automata:



Finite automaton 1



Finite automaton 2

As shown in this experiment, there are many finite automata which are a valid generalization of a given training set. A typical way of solving this problem is to require that the hypothesis be the shortest or the most economical description consistent with all examples[1]. In the context of constructing finite automata, the goal would be to find a finite automaton with the fewest states which is consistent with all positive examples while rejecting all negative examples. In terms of the implementation, every candidate solution in the population is assigned a score indicating how well it matches the input data. Given two equally performing finite automata, the one with fewer states would receive a higher score. This would increase its chance of survival and of generating offspring in the next generation.

3.3 Experiment Results

This section lists the experiments along with their results produced by the genetic algorithm

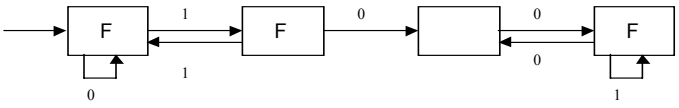
Experiment 1

Positive examples:

(0) (01) (11) (00) (100) (110) (111) (000) (100100)
(110000011100001) (11011100100)

Negative examples:

(101) (010) (1010) (1110) (1011) (10001) (111010)
(1001000) (11111000) (01110011101) (1101110010) (10)



Output: (after 2532 generations)

The regular expression described by English and represented by the above finite automaton is:

any string without an odd number of consecutive 0's following an odd number of consecutive 1's.

Experiment 2

Positive examples:

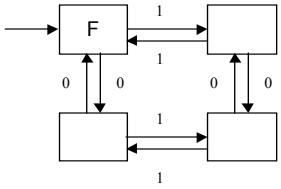
(00) (11) (00) (1111) (1001) (0110) (1000001011100010)
(0000000011) (111111001010) (01010011)

Negative Examples:

(0) (111) (000) (101) (010) (0000000) (11111) (111000000)

(1111110) (000001111) (11001) (00000011100)

Output (after 2820 generations):



The regular expression represented by the above finite automaton is:

Any string consists of an even number of 0s and an even number of 1s

Experiment 3

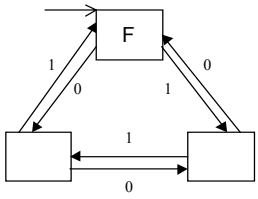
Positive examples:

(10) (01) (1100) (101010) (111) (000) (10000) (01000)
(1100000) (100100100) (0000000011) (11111111110)

Negative examples:

(0) (1) (11) (00) (101) (011) (1101) (1111) (00000)
(00000000) (1111111) (100100) (00000111) (1)

Output: (after 2971 generations)



The regular expression represented by the above finite automaton is:

Any string such that the difference between the number of 0s and 1s is 3 times n (n is an integer).

The above experiments demonstrate the power of genetic algorithm in descriptive learning. It is able to construct a model based on incomplete and imprecise data. By looking at a small set of examples, the genetic algorithm impressively conceptualizes the input data and produces a model that summarizes the important common properties shared by the input examples.

4 Predicative Genetic Learning

In contrast to the global nature of descriptive data mining, predictive data mining searches for local patterns and relationships in the data set. It makes statements only about restricted regions. For example, a search through a database of mail order purchases may reveal that people who buy certain combinations of items are also likely to buy others. The key distinction between prediction and description is that prediction has a unique variable as its objective, such as the value of the a stock at some future date, or which horse will win a race. Letter prediction learning is one form of predicative learning.

4.1 Letter Prediction Learning

This section describes letter prediction learning, a research topic both in psychology research and machine learning research. Letter prediction learning is an instance of concept learning, which is concerned with forming a concept characterizing given facts. It involves observing a collection of examples of some regularity, identifying the essential properties common to these examples, and then making a predication. The underlying methodology of concept learning is an inductive inference, a process whereby one acquires general and high-level knowledge from specific observations of a given phenomenon.

4.2 Human Performance

Simon and Kotovsky used the following 12 learning tasks on two groups of human subjects [3].

Letter Sequence	Expected Prediction
(1) cdcdcd ...	cd
(2) aaabbbccc ...	ddd
(3) abmcdmefmghm ...	ijm
(4) defgefghfghi ...	ghij
(5) mabmbcmcd ...	mde
(6) urtustutt ...	uut
(7) abyabxabw ...	abv
(8) rscdstdetuef ...	uvfg
(9) wxaxybyzczad ...	abe
(10) jkqrklrslmst ...	mntu
(11) pononmnmlmlk ...	lkj
(12) npaoqapraqsa ...	rta

The first group consists of 12 people from different walks of life. The second group consists of 67 subjects comprising an entire class of high school seniors. In an experiment a tested human subject is given a sequence of letters, called a training sequence, and then is asked to make a prediction for a plausible continuation of the sequence. In their experiments, a total of 79 human subjects were given these tasks.

Surprisingly, none of these tasks was solved by all human subjects. The performance of these two groups is summarized below[3]:

	Correct Answers	
Task number	Group of 12	Group of 67
1	12	65
2	12	61
3	12	57
4	2	45
5	9	49
6	5	43
7	9	51
8	4	39
9	7	43
10	6	48
11	5	34
12	5	42

4.3 Experiment Design and Results

Initially, the genetic algorithm learns the English alphabets and their relationship from scratch, produces a list of rules, and then stores them in its knowledge base. Some of the constructed rules are listed below:

The concept of "NEXT":

if <var> A then <var> B
if <var> B then <var> C
.
.
if <var> Y then <var> Z

The concept of "PRIOR":

if <var> B then <var> A
if <var> C then <var> B
.
.
if <var> Z then <var> Y
.

where <var> stands for a variable that can take any value

Similarly, the genetic algorithm learned and constructed other concepts (rules) such as "double next letter", "double prior letter", "vowel", and "consonant". In the learning phrase, the genetic algorithm applies these building blocks to make a new rule based on the letter sequence, and therefore, is able to make prediction .

For each concept, the genetic is given a set of randomly generated letters. Whenever the genetic algorithm constructs a rule generating a correct prediction with respect to the concept, it is rewarded. Otherwise, the genetic algorithm is punished by given a negative reward. Other than these rewards, the entire learning process is conducted by the genetic algorithm without any human assistance.

A letter learning task consists of two phases: a training phase and a predicting phase. In the training phase, a sequence of letters are provided. The genetic algorithm conducts predicative learning by detecting the periodicity in the letter sequence and then applies the rules it has already constructed to make predictions. At the end of a training phase, the genetic algorithm is able to construct a set of rules specifically tailored to the given learning sequence, and then uses it to make a prediction for the continuation of the letter sequence.

Given the 12 learning tasks designed by Simon and Kotovsky[3], the genetic algorithm tested under the same experimental conditions was able to predict the logical continuation of the these sequence correctly without a single error. It is worth repeating that the genetic algorithm was not programmed. Instead, it learned the basic concepts of English from scratch and applied the knowledge to solve these letter prediction learning tasks.

To make learning more challenging, another kind of sequence learning, known as "regularity learning" is also conducted. In regularity learning, the genetic algorithm is given several groups of letters separately. These sequences appeared quite different at first, but a closer look reveals that they are governed by same regularities.

To illustrate, consider the following example:

Group 1: a c e
Group 2: i k m
Group 3: u ? ?

Given the first two groups, the group 3 is expected to be predicted and completed. The following regularity learning tasks have been used in our experiments.

	Group 1	Group 2	Group 3	Predicted
(1)	cc	tt	s?	s
(2)	rrs	aab	x??	xy
(3)	afeh	osuu	id??	of
(4)	abc	stu	m??	no

(5) ae	ou	i?	o
(6) cdf	rst	t??	vw
(7) dfh	bdf	u??	wy
(8) cade	vowu	je??	ki
(9) xghyig	vrnwtm	aab???	bca
(10) ihbojc	atrevs	omm???	unn

In our experiments, the genetic algorithm is able to solve all the above problems correctly using the knowledge it acquired. The results show the potentials of genetic algorithms in the field of predicative learning.

5 Conclusion

This research shows the feasibility and effectiveness of genetic algorithms in both pattern discovery and model construction. Experimental evidence presented in this paper show that the genetic algorithm exhibits forms of intelligent behaviors and learning capabilities, such as knowledge adaptation, rule construction, and conceptual reasoning, which are not yet seen in many other learning systems and data mining systems. As a powerful learning algorithm, genetic algorithms promise potential for many challenging descriptive and predicative learning many data mining systems are facing today.

References

1. A. Birkendorf, Boker and Simon. Learning Deterministic Finite Automata from Smallest Counterexamples. In SIAM Journal on Discrete Mathematics Vol. 13, Number 4:465-491. 2000.

2. J. Han and M. Kamber, Data Mining: Concepts and Techniques. Morgan Kaufmann Publishers. 2001.

3. H. A. Simon and K. Kotovsky, Human Acquisition of Concepts for Sequential Patterns, Psychological Review, 70: 534-536, 1963.

4. Ryszad S. Michalski, "Machine Learning and Data Mining: Methods and Applications". 1998

5. Tomita, M, Learning of construction of finite automata from examples using hill-climbing. Thesis. Carnegie-Mellon University, 1982.