Study of Genetic Algorithm, an Evolutionary Approach

Mrs.K.Jayavani
Research Scholar,
Manonmaniam Sundaranar University,
Tirunelveli
Vanigopinath@gmail.com

Dr.G.M.Kadhar Nawaz, M.C.A., Ph.D.,
Director,
Dept.Of.Computer Application,
Sona College of Technology.
Salem.

***Abstract*--**Data mining is the process of discovering interesting knowledge, such as patterns, associations, changes, anomalies and significant structures, from large amount of data stored in databases, data warehouses, or other information repositories. To do this process, data mining uses a variety of algorithms according to the specifications of measures and threshold. The results of this analysis are then used to build models based on real world behavior, which are in turn used to analyze incoming data and make predictions about future behavior. Here, we are focusing on one of the efficient evolutionary algorithm called genetic algorithm. This is a search technique used in computing to find exact or approximate solutions to optimization and search problems. Genetic algorithms are categorized as global search heuristics that use techniques inspired by evolutionary biology such as inheritance, mutation, selection, and crossover. Genetic algorithms are numerical optimization algorithms inspired by both natural selection and natural genetics. This method is a general one, capable of being applied to an extremely wide range of problems. In this paper we will discuss the Genetic algorithm techniques and its application in data mining in detail.

Keywords *--Evolutionary algorithm, Genetic algorithm, Optimization, Selection, Crossover, Mutation, Natural selection, Natural Genetics*.

*****

I. INTRODUCTION

Genetic algorithms (GAs), first proposed by Holland in 1975(Adaptation in Natural and Artificial Systems, 1975) are computational models for finding a solution to a problem, mostly optimisation, modelled loosely on the principles of evolution via natural selection. They are useful when the search space for solution is large and complex, and no mathematical analysis exists to narrow down the search space.

A genetic algorithm emulates biological evolution by first considering a set of individuals (the population, characterized by a set of chromosomes) in the current generation and a set of biologically-inspired operators that can change these individuals to a newer generation of population. Chromosomes, which carry genetic information, is comprised of genes, each encoding a particular trait of an individual, for example the colour of eyes. The complete set of genetic material (all chromosomes) of a particular species is called its genome or the gene pool. An individual of a species is characterized by its own settings for genes; the particular set of genes distinguishing the individual is called its genotype. According to evolutionary theory, only the fittest of individuals are likely to survive and generate off-springs by transmitting their biological heredity to new generations.

In computing terms, the chromosome is essentially a string of bits, and Gas map these strings of bits to each potential solution. Each solution becomes an individual in the population, and each string becomes the representation of an individual. The genetic algorithm then manipulates the most promising strings in the set of strings, or the population, in its search for an improved set of solutions. It eliminates weak elements by favoring the retention of optimal individual, and recombines features of good individuals (through reproduction process) to perhaps generate better individuals. The algorithm performs in cycles of generations. In each generation, the GA creates a set of new solutions (or chromosomes) by different genetic operations that correlate to the processes of natural reproduction.

Genetic algorithms have been shown to be an effective tool in datamining.Essentially, when a problem in data mining is viewed as a search problem or an optimization problem, there is a scope of applying GA.It has been shown to be successful in association rule mining, clustering and classification.

II. GENETIC ALGORITHM

Wherever Times is specified, Times Roman or Times New Roman may be used. If neither is available on your word processor, please use the font closest in appearance to Times. Avoid using bit-mapped fonts if possible. True-Type 1 or Open Type fonts are preferred. Please embed symbol fonts, as well, for math, etc.

A. Process

Genetic algorithms are a part of evolutionary computing and they are inspired by Darwin's theory of evolution. When GAs are applied to solve a problem, the first step is to define a representation that describes the problem states. An initial population is then defined, and genetic operators are used to generate the next generation. This procedure is repeated until the termination criteria is satisfied. This basic principle of genetic algorithm is outlined below.

1. [Start] Generate random population of n chromosomes(suitable solution)
2. [Fitness] Evaluate the fitness f(x) of each chromosome x in the population
3. [New population] Create a new population by repeating the following steps until the new population is complete.

(a) [Selection] Select two parent chromosomes from a population according to their fitness(the better fitness, the better chance to be selected)
(b) [Crossover] With a crossover probability cross over the parents to form a new offspring. If no crossover was performed, offspring (children) is the exact copy of parents.
(c) [Mutation] With a mutation probability mutate new offspring's at each locus (position in chromosome)...
(d) [Accepting] Place new offspring's in the new population.

4. [Replace] Use new generated population for the further run of the algorithm.
5. [Test] If the end condition is satisfied, stop, and return the best solution in current population.
6. [Loop] Go to step 2.

There are many parameters and settings that can be implemented differently in various problems:

- Identify good (above-average) solution in a population.
- Make multiple copies of the good solutions.
- Eliminate bad solutions from the population so that multiple copies of good solutions can be placed in the population.

These are the main key issues when using genetic algorithms. This will be explained in the upcoming sections.

B. *Representing a solution*

Before genetic algorithms are applied to solve a problem, the first step is to define a representation that describes the problem states. The most commonly used representation is the binary string, which is sequence of 1's and 0's.This process of coding is intended to achieve a pseudo chromosomal representation of the problem. Besides binary string representation, there are many other possible ways of encoding in GA.The suitability of the encoding methods chosen depends on the problem at hand. The type of encoding system is discussed below.

Binary encoding: This is the most common representation and popular encoding because of its simplicity. In this encoding, every chromosome is a string of bits, e.g.
Chromosome A: 0101001011001000
Chromosome B: 1100001101011011

Each bit representing some characteristic of the solution. This encoding, however, is not natural for many problems and sometimes corrections must be made after crossover and/or mutation. This is generally used in knapsack problems.

Value encoding: In value encoding, each chromosome is represented as the string of some value. Value can be integer, real number, character or some object. E.g.,

Chromosome A: 1.231, 1.324, 0.231, 2.326, 2.243
Chromosome B: ABDJEIFJDHERDLDFLIGTEF
Chromosome C: (red), (red), (right), (forward), (right)

The traditional genetic operators cannot be used in value encoding, and it becomes necessary to develop new types of crossover and mutation operations. Value encoding can be generally used for finding weighs in neural networks.

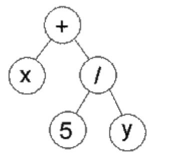
Permutation Encoding: Permutation encoding is used in ordering problems. In this, each chromosome represents position in a sequence. Example for this is travelling salesman problem, the string of numbers represent the sequence of cities visited by salesman.

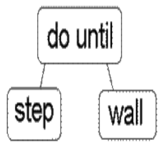
Chromosome A: 1 5 6 4 5 3 7 8 2
Chromosome B: 7 6 5 3 8 4 2 1 9

Permutation encoding is only useful for specific order problems. Here some types of crossover and mutation corrections must be made to leave the chromosome consistent (i.e. have real sequence in it).

Tree Encoding: Tree encoding is used mainly for evolving programs or expressions for genetic programming. In tree encoding every chromosome is a tree of some objects, such as functions or commands in programming language.

Chromosome A Chromosome B





(+ x (/ 5 y)) (do_until step wall)

Programing language LISP is often used to this, because programs in it are represented in this form and can be easily parsed as a tree, so the crossover and mutation can be done relatively easily.

Fitness

In order to determine which chromosomes are good for retention and which are weak, it becomes necessary to ascertain the fitness of a chromosome. Thus in genetic algorithm, fitness is used to allocate reproductive traits to the individuals in the population and thus act as some measure of goodness to be maximized. This means that individuals with higher fitness value will have higher probability of being selected as candidates for further examination. Certain genetic operators require that the fitness function be non-negative, although certain operators need not have this requirement. The fitness function is always problem dependent.

III. GENETIC OPERATORS

Genetic operators used in GAs maintain genetic diversity, is a necessity for the process of evolution.

Genetic operators are analogous to those which occur in the natural world:

- Selection(or Reproduction)
- Crossover(or Recombination)
- Mutation

A. *Selection Mechanism*.

The selection operators are also called as reproduction operators. Selection means extract a subset of genes from an existing population according to any definition of quality. That is, selection is used to determine which individuals survive to the next generation by taking part in the process of reproduction. Individuals solutions are selected a fitness based process, where fitter solutions (as measured by a fitness function) are typically more likely to be selected. Thus, the primary objective of the selection operator is to make duplicates of good solutions and eliminate bad solutions while keeping the population size constant. This is done in the following way.

- Identify good (above-average) solution in a population.
- Make multiple copies of the good solutions.
- Eliminate bad solutions from the population so that multiple copies of good solutions can be placed in the population.

According to Darwin's theory of evolution the best chromosome survive to create new offspring. There are many methods to select the best chromosomes. Certain selection methods rate the fitness of each solution and preferentially select the best solutions. Other methods rate only a random sample of the population, as this process may be time consuming. The popular and well-studied selection methods include roulette wheel selection and tournament selection.

In roulette wheel selection, individuals are given a probability of being selected that is directly proportional to their fitness. Two individuals are then chosen randomly based on these probabilities and produce offspring. In tournament selection, parents are pooled and a tournament is held with in the pool(s).

B. Crossover

Crossover operator combines (mates) two chromosomes (parents) to produce a new chromosome (offspring). The idea behind crossover is that the new chromosome may be better than both of the parents if it takes the best characteristics from each of the parents. Crossover occurs during evolution according to a user-definable crossover probability. Crossover selects gene from parent chromosomes and create a new offspring.
The crossover operator are of many types:

- One point crossover
- Two point crossover
- Uniform crossover
- Arithmetic crossover and
- Heuristic crossover

The operators are selected based on the way chromosomes are encoded.

Single point crossover: One crossover point is selected, binary string from beginning of chromosome to the crossover point is copied from one parent, and the rest is copied from the second parent. This is the simple and most commonly used method.

11001011+11011**111** = **11001111**
Parent A Parent B Offspring

Two point crossover: Two crossover point are selected, binary string from beginning of chromosome to the first crossover point is copied from one parent, the part from the first to the second crossover point is copied from the second parent and the rest is copied from the first parent.

110010**11** + 11**0111**11 = **11011111**
Parent A Parent B Offspring

Uniform crossover: Bits are randomly copied from the first or from the second parent.

1**10**010**11** + **1**10**111**01 = 11011111
Parent A Parent B Offspring

Arithmetic crossover: Some arithmetic operation is performed to make a new offspring

11001011 + 11011111 = 11001001 (AND)
Parent A Parent B Offspring

C. *Mutation*

After selection and crossover, we now have a population full of individuals. Inorder to ensure that the individuals are not all exactly the same, we go for mutation. This operator is used to maintain genetic diversity from one generation of a population of chromosomes to the next. It introduces new genetic structures in the population by randomly changing some of the string bits. In other words, mutation is implemented by occasionally altering a random bit in a chromosome. With the new gene values, the genetic algorithm may be able to arrive at better solution than was previously possible. Mutation is an important part of the genetic search, helps to prevent the population from stagnating at any local optima. Moreover, some important regions of the search space may never be explored if this operator is not introduced. The mutation operators are of many types which are given below.

Flip bit: This operator simply inverts the value of the chosen gene. That is 0 goes to 1 and 1 goes to 0. This mutation operator can only be used for binary genes

Original offspring 1	1101111000011110
Original offspring 2	1101100100110110
Mutated offspring 1	1100111000011110
Mutated offspring 2	1101101100110110

Boundary*:* A mutation operator that replaces the value of the chosen gene with either the upper or lower bound for that gene (chosen randomly). This mutation operator can only be used for float and integer genes.

Non-Uniform: A mutation operator that increases the probability that the amount of the mutation will be close to 0 as the generation number increases. This mutation operator keeps the population from stagnating in the early stages of the evolution then allows the genetic algorithm to fine tune the solution in the later stages of evolution. This mutation operator can only be used for integer and float genes.

Uniform: A mutation operator that replaces the value of the chosen gene with a uniform random value selected between the user-specified upper and lower bounds for that gene. This mutation operator can only be used for integer and float genes

Gaussian: A mutation operator that adds a unit Gaussian distributed random value to the chosen gene. The new gene value is clipped if it falls outside of the user-specified lower or upper bounds for that gene. This mutation operator can only be used for integer and float genes.

III. APPLICATION OF GENETIC ALGORITHM IN DATA MINING

The application of genetic algorithm in the context of data mining is generally on tasks of hypothesis testing and refinement, where the user poses some hypothesis and the system first evaluates the hypothesis and then seeks to refine it. Hypothesis refinement is achieved by starting with some initial hypothesis and then allowing some or all parts of it to vary. The important aspect of genetic algorithm application is in the encoding of the hypothesis and in the formulation of the evaluation function for fitness. Another way to use genetic algorithm for data mining is to design hybrid techniques by blending one of the known data mining techniques with the algorithm. For example it is possible to use genetic algorithm for optimal decision induction. We can build many decision trees using any of the traditional techniques by randomly generating different samples. GA's have been used in wide variety of optimization tasks, prediction and classification. It supports automatic programming such as building computational structures like cellular automata and sorting networks.

Advantages

- Concepts are easy to understands due techniques similar to the natural processes like inheritance, mutation, etc.
- Can be used where traditional search methods fail.
- Useful where search space is large, complex or poorly understood.
- Provides us with several local optimums as well as the global optimum
- Solves problems with multiple solutions.
- Genetic algorithms are easily transferred to existing simulations and problems.

Limitations

- Due to poorly known fitness function, some optimization problems cannot be solved by genetic algorithms. These are called variant problems.
- There is no assurance of finding a global optimum. It happens very often when the population has a lot of individuals.
- Like other artificial intelligence techniques, the genetic algorithm cannot assure constant optimization response time.
- While using genetic algorithms, it is true that the entire population is improving but this could not be said for an individual within this population.
- Writing of fitness function must be accurate.

V. CONCLUSION

Genetic algorithm approaches to data mining are increasingly becoming popular and there are several extensions of the basic GA for different applications. Results can be good on some problems while rather poor on others. If we use mutation only, it makes the algorithm very slow, crossover makes it significantly faster. They have applications in commercial, educational and scientific areas. In addition there are other population –based techniques such as Ant Colony Algorithm and Particle Swarm Techniques that are being developed, which share some foundational concepts with GA.

REFERENCES

[1] Mitchell, Melanie (1996). An Introduction to Genetic Algorithms. Cambridge, MA: MIT Press
[2] Eiben, Agoston; Smith, James (2003). Introduction to Evolutionary Computing. Springer
[3] Coley, A.D. An Introduction to genetic Algorithms for Scientists and Engineers, World Scientific, Singapore, 188p, 1999.
[4] Goldberg, D.E. Genetic Algorithm in Search Optimization and Machine Learning .Addison Wesley Publishing Compnay,1989
[5] D.Simon, Evolutionary Optimization Algorithm, Wiley, 2013
[6] M.Gen, R. Cheng, "Genetic Algorithms and Engineering Optimization," John Wiley & Sons, Inc., New York, 2000.
[7] Anshul Sharma, Anuj Mehta 2013. "Review Paper of Various Selection Methods in Genetic Algorithm", International Journal of Advanced Research in Computer Science and Software Engineering Volume-03, No-07, July 2013, (1476-1479).
[8] Dr.Rajib Kumar Bhattacharyya, "Introduction to Genetic Algorithms" 2013.
[9] Darwin. C, "The origin of species by means of natural selection", 1859.
[10] Freitas, "A survey of evolutionary algorithms for data mining and knowledge discovery", In: A. Ghosh, and S. Tsutsui (Eds.) Advances in Evolutionary Computation. Springer-Verlag, 2002.
[11] J. Han, M. Kamber, and J. Pei, Data Mining: Concepts and Techniques, 3rd ed. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2011.
[12] Fogel D.B. (Ed.).Evolutionary Computation, IEEE Press, New York, 1998.
[13] Freitas A.A. Data Mining and Knowledge Discovery with Evolutionary Algorithms, Springer, Berlin, 2002, p. 956.
[14] Keenan M. Novel and combined generation, selection, recombination and mutation operators for evolutionary computing. In proceedings of the Ninth International Conference On Computer Applications in Industry. ISCA Publisher, 1996.
[15] Schmitt Lothar M. Fundamental study theory of genetic algorithms, Theoretical Computer Science, 259, 2001, p 1-61.
[16] Pham, D.T., and Karaboga, D., Intelligent Optimisation Techniques. Springer, London, Great Britain, 2000, 261p.
[17] Rothlauf, F., Representations for Genetic and Evolutionary Algorithms. Springer, Netherlands, 2006, 314p.