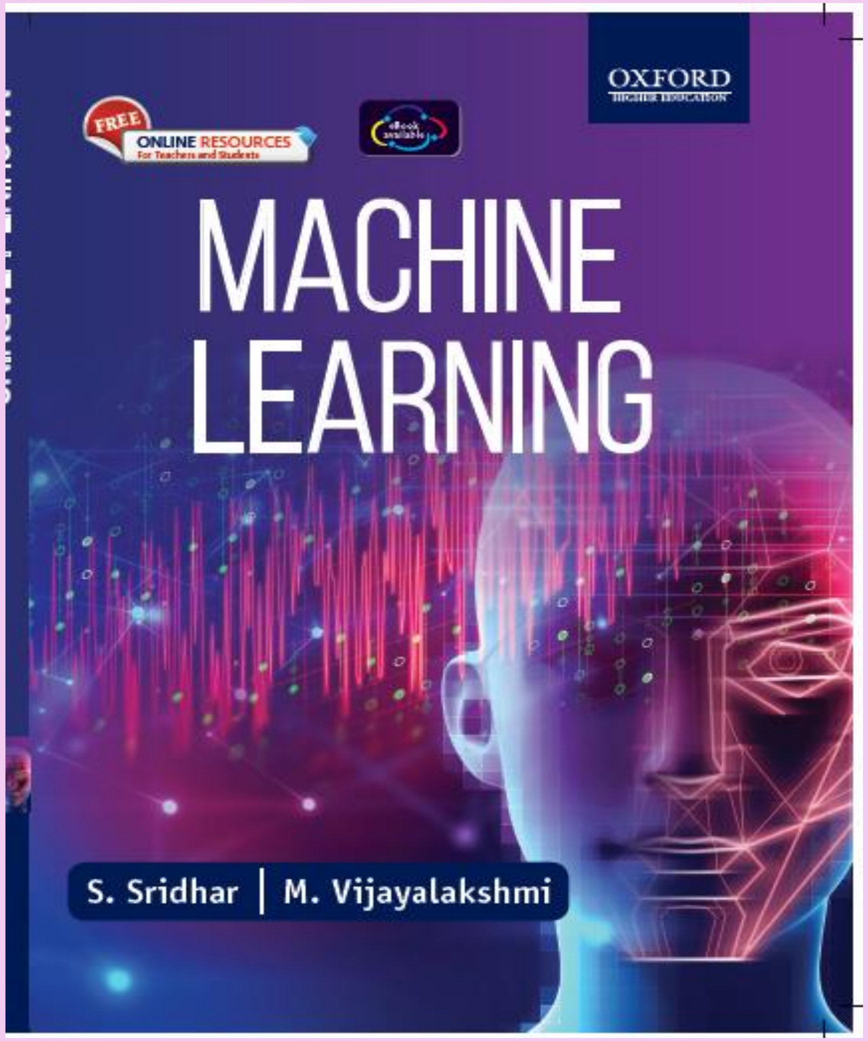

OXFORD
UNIVERSITY PRESS

Machine Learning

S. Sridhar and M. Vijayalakshmi

Chapter 15

Genetic Algorithms

Genetic Algorithms

Genetic algorithm (GA) is one of the important branches of evolutionary computation. It is a family of search-based algorithms that uses the concept of natural selection and principle of evolution. Genetic algorithms too are based on the Charles Darwin principle of natural selection.

Natural Selection and its Equivalents

Table 15.1: Natural Selection and its Equivalents

Concepts in Natural Selections	Equivalent in Genetic Algorithms
Chromosome	String of binary bits or digits or character strings to represent candidate solutions
Population	Set of strings
Fitness of survival	Fitness function
Reproduction and mating	Application of genetic operations
Gene	Parts of the solution
Locus	Position in the string
Allele	Positional value
Genotype	Encoded solution
Phenotype	Decoded solution

Characteristics of GA

1. GA is a population-based technique where the traditional mathematics such as derivatives is used. Also, it is different from a single solution based technique. GA finds many solutions at a time.
2. GA is based on the concept of fitness, which are not used by traditional methods.
3. GA is based on probabilistic concepts and hence all its operations are probability based. So, there is no guarantee that GA will produce answers. This contrasts with traditional methods that are deterministic in nature.

Advantages of GA

Advantages of GA

1. GAs are population-based searches and hence search many points rather than a single point in search space.
2. Can be executed in parallel and hence can deal large problems
3. Can handle continuous as well as discrete optimization problems
4. GA is a metaheuristic algorithm. So, it is a problem independent strategy for solving problems.

Advantages of GA

5. Fast
6. These algorithms find better solutions by avoiding local minima and always get the global maxima.
7. It is useful for solving optimization problems.

Disadvantages of GA

1. GA operates in a different space than the given problem space and hence tuning of hyper-parameters is must.
2. It can handle a limited domain of problems. For example, one cannot use GA for all generic problems apart from optimization problems.
3. Identification of the fitness function in GA is difficult as it depends on the problem.
4. The selection of suitable genetic operators for a given problem is difficult.
5. There is a limit on the number of iterations and population size due to computational requirement.
6. GAs are computationally very intensive programs and require more resources.
7. May terminate prematurely
8. These are probabilistic algorithms and hence there is no guarantee of solutions.

GA Structure

Algorithm 15.1: Genetic Algorithms

1. Create an initial population randomly based on the parameter, population size.
2. Apply fitness function to evaluate population fitness as the ratio of given chromosome and fitness of the population.
3. While (! Termination condition) do.

   Select the parents randomly as per parameter population size.
4. Create a new population by mating the parents through crossover with parameter crossover probability and mutation with parameter mutation rate.
5. Replace the least fit chromosomes with chromosomes of high fitness value and update the population.
6. Evaluate new fitness value of the population for convergence.

Encoding Methods

The first stage of the genetic algorithm is encoding or representation. GA encodes each candidate solution. For example, for the problem $f(x) = x^3$, where x ranges from 0 to 15, any value of x, say 3 or 5 or 9, can be a solution as this remains unknown in the beginning of the problem. These are called potential solutions and hence called candidate solutions. Candidate solutions are different from best solutions, as the best solution maximizes or minimizes the objective function subjected to the constrains and is known only after the problem is solved.

Types of Encoding

1. Binary encoding
2. Permutation encoding
3. Value encoding as real number, Integer/Literal
4. Gray encoding

Binary Encoding

Binary encoding is one of the commonest methods of encoding and is useful for majority of the genetic algorithms. Here, a chromosome is represented as a string of 1's and 0's. For example, chromosomes A and B can be encoded as follows:

Chromosome A:	0	1	1	0	0	1	0	1
Chromosome B:	1	1	1	0	0	0	0	1

The number of bits of a chromosome is called length. The choice of the length of the vector depends on precision. The length is determined as follows:

$$\frac{b - a}{2^m - 1} \leq precision$$

Permutation Encoding

TRAVELLING SALESPERSON PROBLEM

Chromosome A:	1	2	3	4	5
Chromosome B:	1	5	4	3	2

Chromosome A represents a route 1, 2, 3, 4, 5 and chromosome B represents another route 1, 5, 4, 3, 2.

Value Encoding

VALUE ENCODING IS A SET OF CHARACTERS OR REAL VALUES

Chromosome A:

A	G	T	A	A	A

(Character sequence)

Similarly, the chromosome can be encoded as a real number as shown below:

Chromosome B:

12.2	3.56	23.67	31	43

(Real numbers)

Grey Encoding

Gray encoding is also one of the popular methods of encoding a chromosome like binary encoding. It was designed by Frank gray which is like binary coding but with a specific property that the two successive numbers would differ only in one bit. Like binary codes, gray codes are also used in genetic algorithms.

Population Initialization

1. Random initialization
2. Heuristic initialization

Fitness Function

$$F(x_i) = \frac{\mathit{Fitness}\text{ of the } i^{th}\,\mathit{chromosome}(f(x_i))}{\mathit{Fitness}\text{ of the population}\left(\sum_{i=1}^{\mathit{population\ size}} f(x_i)\right)}$$

Characteristics of Fitness Function

1. It should be possible to compute fitness functions faster.
2. It is the quantitative measure of how fit the solution is and should be positive.
3. It should be exact. If exact fitness cannot be determined, then the fitness function should approximate it well.

Fitness Scaling

If the fitness value is not positive, then it can be scaled as:

$$F(x) = a \times f(x) + b$$

Where, a and b are variables that are user controlled. The parameter a is a positive number for maximization problems and the parameter b ensures that the fitness value is positive. This process is called fitness scaling.

Fitness Based Selection

Algorithm 15.2: Fitness-based Methods

Step 1: Calculate the fitness value of each chromosome.

Step 2: Find the total fitness of the population.

Step 3: Normalize the value by diving the fitness value of each chromosome by the total fitness of the population.

Step 4: Find the cumulative probability of the chromosome.

Roulette Wheel Selection

SELECTION OF CHROMOSOMES IS BASED IN PROPOSITION TO FITNESS VALUE

Practically, roulette wheel spin can be done as follows:

- Generate a random number, *s*.
- Select the first chromosome whose cumulative value crosses *s*.
- Repeat this for population size times to select all chromosomes.

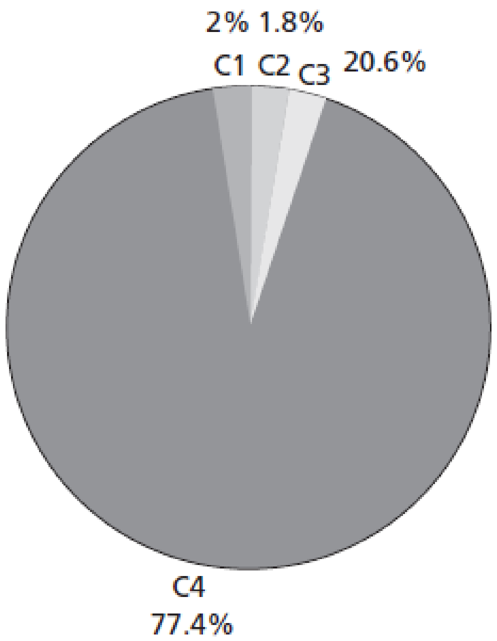

Figure 15.2: Roulette Wheel Selection

Rank Selection

This technique ranks the chromosomes first and then associates rank 1 with the worst chromosome, rank 2 with the next worst and so on till it reaches the best chromosome, which is assigned rank *N*. The same roulette wheel algorithm is used. The advantage of this method is that unlike roulette wheel that allows greater space for the best one and chooses that chromosome for selection, ranking provides an equal opportunity for selection. But the disadvantage is that it leads to slower convergence of genetic algorithms.

Stochastic Universal Sampling

This method is also same as the roulette wheel selection method. But instead of one fixed point, many fixed points are present. So only one spin is required to select multiple parents. The only necessary condition for this technique to be effective is that the fitness value should be positive.

Tournament Selection

In this method, k individuals are selected from the population in random and the winner is determined. This process is repeated till all parent chromosomes are selected. This is like traditional cricket tournament, where the teams selected arbitrarily play and then the winner is selected. Tournament size is an important parameter. If it is large, then weak individuals are removed because of competition. If the tournament size is low, then more chromosomes are selected for the next generation.

Steady State Selection

This selection method chooses the best chromosome with the largest fitness function. Most importantly, this method rejects the chromosomes with the least fitness value. Rest of the chromosomes can survive to the next generation. This is helpful in keeping the diversity of the population.

Elitism

Elitism is a method that retains the best chromosome to the new population while the rest are chosen through conventional methods. This increases performance of the genetic algorithms as it prevents the loss of good chromosomes.

Crossover Methods

MODEL REPRODUCTION PROCESS OF NATURAL SELECTION

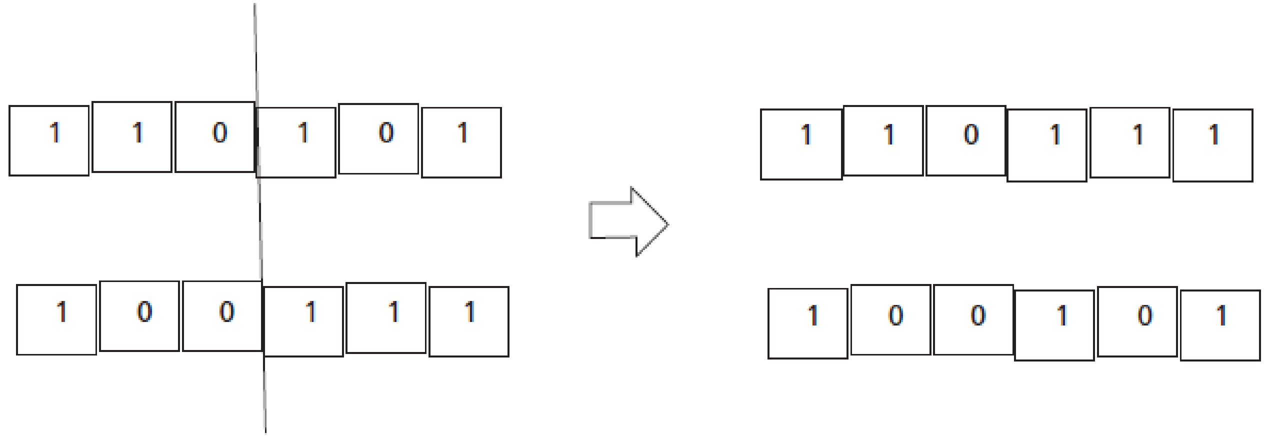

Figure 15.3: One-point Crossover

2-point crossover

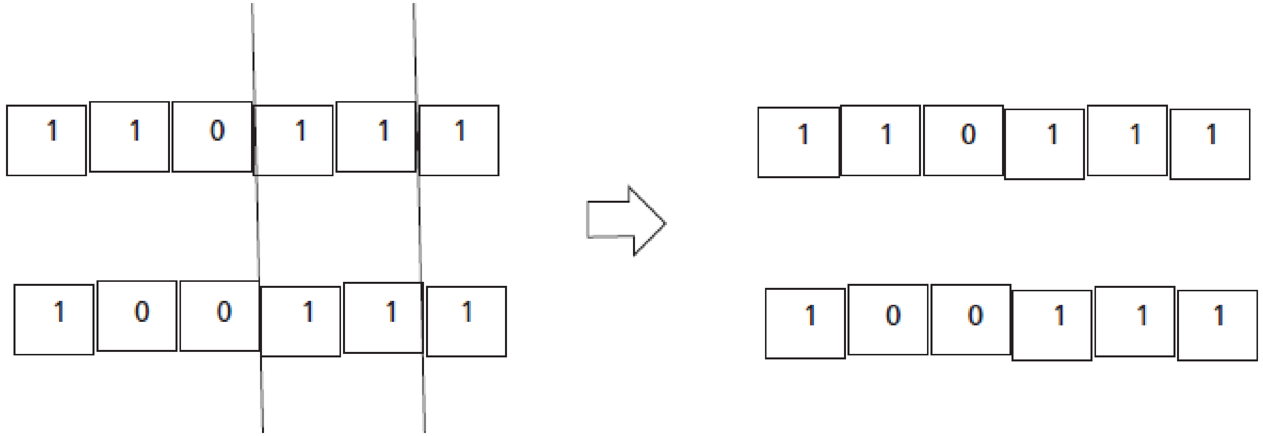

Figure 15.4: Two-point Crossover

This logic can be extended to k-point crossover too with k crossover points.

Uniform Crossover

This is another useful type of crossover where the bits are randomly copied from two parent chromosomes. A coin may be flipped for every bit and based on the outcome either the bit is included or excluded. The coin can be a biased one to ensure that the child chromosome is a better one with more randomness.

Arithmetic Crossover

This type of crossover applies some operations like OR or AND to perform crossover. The following example shows an AND operation as an arithmetic crossover. Another useful arithmetic combination is as shown below:

$$\text{Child 1: } \alpha x + (1 - \alpha)y$$

$$\text{Child 2: } \alpha x + (1 - \alpha)y$$

If $\alpha = 0.5$, then both children are identical.

Permutation Crossover

It can be recollected that the chromosomes are represented as a number in order. In the one-point permutation crossover, one crossover point is selected. The first position to the crossover point is copied to the new chromosome and rest from the second chromosome if it is not the offspring of the resultant.

Mutation

USEFUL WHEN NO PROGRESS IS POSSIBLE

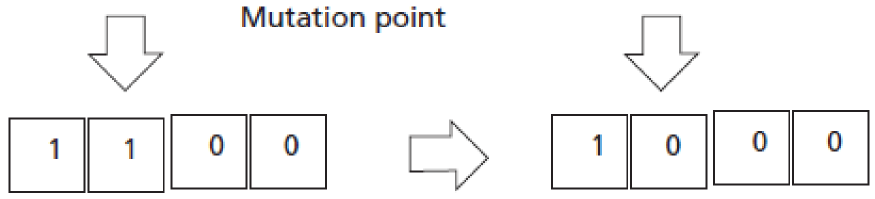

Figure 15.5: Mutation Operation

Swap Mutation

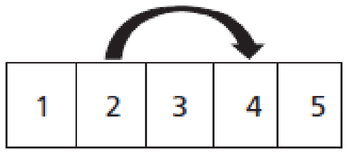

Figure 15.6: Swap Mutation

Scramble Mutation

In scramble mutation, a subset of chromosomes is selected and the values are scrambled or shuffled randomly.

Inverse Mutation

In inversion mutation, the operator takes a set of bits and reverses it. For the given example, 1 0 0 1 0 0, the first three bits can be reversed to yield a new string 0 1 1 1 0 0 as shown in Figure 15.7.

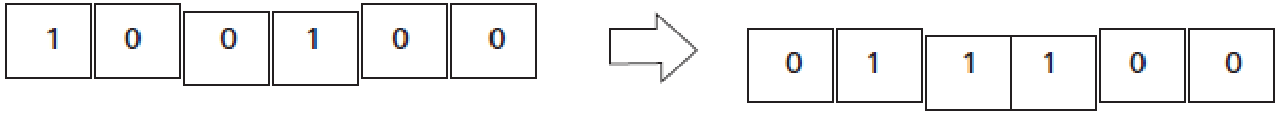

Figure 15.7: Inversion Mutation

Case Study – Maximization of a function

Algorithm 15.4: Maximization of a Function

Step 1: Form a population with population size 4. The length of the chromosome is 4.

Step 2: Let the fitness function be $f(x) = x^3$. If $x = 4$. The fitness function is $f(x) = 4^3 = 64$. Compute the fitness function for the population.

Step 3: Select operation: Select the parent chromosomes from the population if its fitness value is high.

Step 4: Apply genetic operations:

Step 4(a) Apply crossover.

Step 4(b) Apply mutation.

Step 5: Allow chromosomes of step 4 along with the parent chromosomes to form new population.

Step 6: Repeat the steps 2-5 till convergence is achieved. If convergence is achieved, then exit; else go to step 2.

Example

Example 15.2: Solve a problem $f(x) = x^3$, where $0 \le x \le 15$. Find some stages of application of genetic algorithms.

Table 15.3: First Iteration

Chromosomes	Code (randomly chosen)	x	$f(x) = x^3$	$\frac{f(x)}{\sum f(x)}$	$\frac{f(x)}{average \sum f(x)}$	Truncated to
1.	0010	2	8	0.002	0.009	0
2.	0100	4	64	0.018	0.0722	0
3.	1001	9	729	0.206	0.0823	1
4.	1110	14	2744	0.774	3.096	3
Sum			3545			
Average			886.25			
Max			2749			

Example

It can be observed from the Table 15.3, that the unproductive chromosomes like 0010 and 0100 are eliminated. The other chromosomes such as chromosome 1110 (three times) and 1001 (one time) are taken to the next generation. These chromosomes are selected for the genetic operation crossover. The crossover point is selected randomly and shown in Table 15.4.

Table 15.4: Crossover

Chromosomes Selected	Crossover Point (randomly chosen)	Offspring
1110 1110	3	1110 1110
1110 1001	3	1111 1000

Example

Table 15.5: Second Iteration

Chromosomes	Code (randomly chosen)	x	$f(x) = x^3$	$\frac{f(x)}{\sum f(x)}$	$\frac{f(x)}{average \sum f(x)}$	Truncated to
1.	1110	14	2744	0.2927	1.17	1
2.	1110	14	2744	0.2927	1.17	1
3.	1111	15	3375	0.36	1.44	1
4.	1000	8	512	0.055	0.218	0
Sum				9375		
Average				2343.75		
Max				3375		

One can also observe that the sum and average of the population are higher than the first generation. Thus, the fitness of the population is increased.

It can also be observed that the unproductive chromosome 1000 gets eliminated and the chromosome 1111 dominates, that is the solution. Thus, genetic algorithm terminates here. But in real-world problems, many iterations need to be carried out for getting a final solution.

Case Study – Classifier

IF (condition) then Action

These simple production rules can be encoded. For example, a rule of 'IF *A* and NOT *B* THEN *C*' can be encoded as 101. This is called a string and every bit is called a chromosome. A set of such rules in the form of strings constitutes the initial population.

$$\text{Fitness } (h) = [\text{correct } (h)]^2$$

where Correct (h) is the percentage of the correctly classified samples by hypothesis h.

Simulated Annealing

Minimization or maximization of a cost function has many variables and it is a challenging task. Simulated annealing is a metallurgy-inspired idea where an iron rod is heated to a higher temperature, when the impurities are removed by shifting the atoms unpredictably. Then, it can be cooled to get a pure crystal. This idea is simulated for optimizing the algorithm in simulated annealing.

Genetic Programming

Algorithm 15.5: Genetic Programming

Step 1: Choose possible functions and terminals.

Step 2: Generate an initial population of random trees using the set of functions and terminal. A random tree can be of any size.

Step 3: Calculate fitness function of the generated trees.

Step 4: Apply crossover and mutation.

Step 5: Repeat till convergence is achieved.

Block World

The operations required by the problem are listed below:

1. CS — Current stack returns top block of stock, if not NIL
2. TB — Top of the block and returns the name of the topmost block and all. Blocks below it in correct order, if not NIL.
3. NN — Next block needed immediately above top of the block (TB)
4. MOVE (x) — Move to stack, if x is on table
5. MT (x) — Move to table

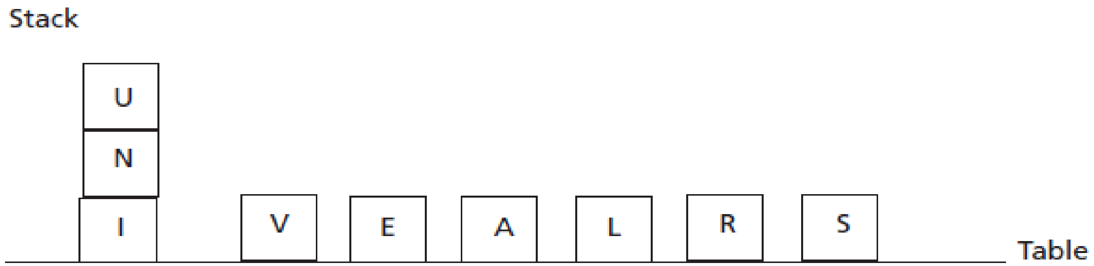

Figure 15.8: A Block World

6. DU (expr1, expr2) — Evaluates two expressions, exp1, exp2 till it is true
7. NOT (expr) — True, if expression expr is NIL
8. EQ (expr1, expr2) — True, if expressions are same, expr1 = expr2

Summary

1. Genetic algorithms are used to solve optimization problems.
2. Genetic algorithms are inspired by the theory of natural selection.
3. The first stage of genetic algorithms is the encoding and representation of a candidate solution.
4. Selection is a genetic operation that selects parent chromosomes based on the value of fitness functions.
5. Selection methods are roulette wheel selection, rank selection, tournament selection, steady state selection and elitism.
6. Crossover operation models reproduce by combining chromosomes.
7. One-point crossover, two-point crossover, uniform crossover and random crossover are some of the crossover operations.
8. The mutation operator randomly changes the bits of the individuals.
9. Feature selection problem is the selection of optimal features for classification problem.
10. Genetic algorithm-based classifiers can be highly effective in classifying instances.