Fundamentals of genetic algorithm pdf

A genetic algorithm is a susururistic inspired by Charles Darwin's theory of natural evolution. This algorithm reflects the process of natural selection, in which the fittest individuals are selected for reproduction to produce next-generation offspring. Concept of natural selectionThe process of natural selection begins with the selection of the fittest individuals from a population. They produce offspring that inherit the characteristics of their parents and are added to the next generation. If parents have better fitness, their offspring are better than their parents and have better chances of survival. This process continues to iterate and in the end a generation with the fittest individuals is found. This term can be used for a search problem. We consider a number of solutions to a problem and choose the best from them. Five phases are considered in a genetic algorithm. InitialPopulationFitness functionSelectionCrossoverMutationInitial PopulationThe process begins with a group of individuals called a population. Each individual is a solution to the problem you want to solve. A person is identified by a set of parameters (variables) called genes. Genes are joined to form a string to form a chromosome (solution). In a genetic algorithm, the set of genes of an individual is represented with a string in the form of an alphabet. Typically, binary values are used (string of 1s and 0s). We say that we encode the genes in a chromosome. Population, chromosomes and GenesFitness functionFitness fitness function determines how fit an individual is (the ability of an individual to compete with other individuals). It gives each individual a fitness score. The probability that a person will be selected for reproduction is based on their fitness rating. SelectionThe idea of the selection phase is to select the fittest individuals and let them pass on their genes to the next generation. Two pairs of individuals (parents) are selected based on their fitness scores. People with high fitness have more chances of being selected for reproduction. Crossover Crossover is the most significant phase in a genetic algorithm. For each pair of parents to be paired, a crossover point is randomly selected from the genes. Consider, for example.B the crossover point as 3, as shown below. Crossover-PointOffspring is created by exchanging the genes of the parents with each other until the crossover point is reached. Gene exchange between parentsThe new offspring are added to the population. New progenymutation new offspring that have been formed, some of their genes may be subjected to a mutation with a low chance probability. This implies that some of the bits in the bit string can be flipped. Mutation: Before and after the mutation occurs to maintain diversity within the population and prevent premature convergence. TerminationAlgorithm Algorithm if the population has converged (does not produce offspring that are significantly different from the previous generation). Then it is said that the genetic algorithm has provided a number of solutions to our problem. CommentsThe population has a fixed size. With the formation of new generations, individuals with the lowest fitness die and offer space for new offspring. The sequence of phases is repeated to produce individuals in each new generation that are better than the previous generation. PseudocodeGiven below is an example of implementing a genetic algorithm in Java. Feel free to play with the code. For a set of 5 genes, each gene can contain one of the binary values 0 and 1. The fitness value is calculated as the number of 1s present in the genome. If there are five 1s, then it has maximum fitness. If there are no 1s, then it has minimal fitness. This genetic algorithm tries to maximize fitness function to create a population consisting of the fittest individual, i.e. individuals with five 1s. Note: In this example, after crossover and mutation, the least fit individual is replaced from the new fittest offspring. Example edition finding the fittest solution in the 32nd generation Including research from science, government laboratories and industry Contains high-profile papers that have been extensively reviewed. , Government laboratories, and industry Contains high-profile papers that have been extensively reviewed continuing the tradition of presenting not only current theoretical work, but also topics that will make future research in the field of ideal for researchers in machine learning, especially those involved with evolutionary calculation currently not access to this book, but you can buy separate chapters directly from the table of contents or buy the full version. Buy the book Volume 45, Issue 1, March 2009, Pages 16-20View full text. In addition, a generic structure of GAs is represented in both pseudocode and graphical form. Readers are advised to understand all the concepts presented in this section correctly and keep an eye on them when reading other sections of this tutorial. Basic terminology Before you start a discussion about genetic algorithms, it is important to be familiar with some of the basic terminology used during this tutorial. Population - It is a subset of all possible (coded) solutions to the given problem. The population for a GA is analogous to the population for people, except that instead of people we Solutions that represent man. Chromosomes - A chromosome is such a solution to the given problem. Gene - A gene is an elemental position of a chromosome. Allele - It is the value that a gene takes for a particular chromosome. Genotype - Genotype is the population in the calculation space. In the computing room, the solutions are presented in such a way that they can be easily understood and manipulated with a computing system. Phenotype - Phenotype is the population in the real solution space, in which solutions are represented in a way in which they are represented in real situations. Decoding and encoding - For simple problems, phenotype and genotypic spaces are the same. In most cases, however, phenotytype and genotypic spaces differ. Decoding is a process of converting a solution from genotype to phenotype space, while encoding is a process of transformation from phenotype to genotype space. Decoding should be done quickly, as it is performed repeatedly in a GA during the calculation of the fitness value. Consider, for example, the 0/1 Knapsack problem. The phenotype space consists of solutions that contain only the item numbers of the items to be picked. In genotype space, however, it can be represented as a binary string of length n (where n is the number of elements). A 0 at position x indicates that the xth element is selected, while a 1 represents the back. This is a case where genotype and phenotype spaces are different. Fitness Function - A simple defined fitness function is a function that takes the solution as input and generates the suitability of the solution as an output. In some cases, the fitness function and the objective function may be the same, while in others it may be different due to the problem. Genetic Operators - These change the genetic composition of offspring. These include crossover, mutation, selection, etc. Basic structure The basic structure of a GA is as follows: We start with an initial population (which can be randomly generated or sown by other heuristics), select parents from that population for mating. Apply crossover and mutation operators to parents to generate new off-springs. Finally, these off-springs replace the existing individuals in the population and the process repeats itself. In this way, genetic algorithms actually try to mimic human evolution to some extent. Each of the following steps is treated as a separate chapter later in this tutorial. A generalized pseudo-code for a GA is explained in the following program: GA() initialize population find fitness of the population, while Done Parent Selection Crossover With Probability PC Mutation With Probability pm Decode and Fitness Calculation Survivor Selection Find Best Return Best