Day 18

30 Days of Machine Learning

Genetic Operators & Hypothesis Space Search

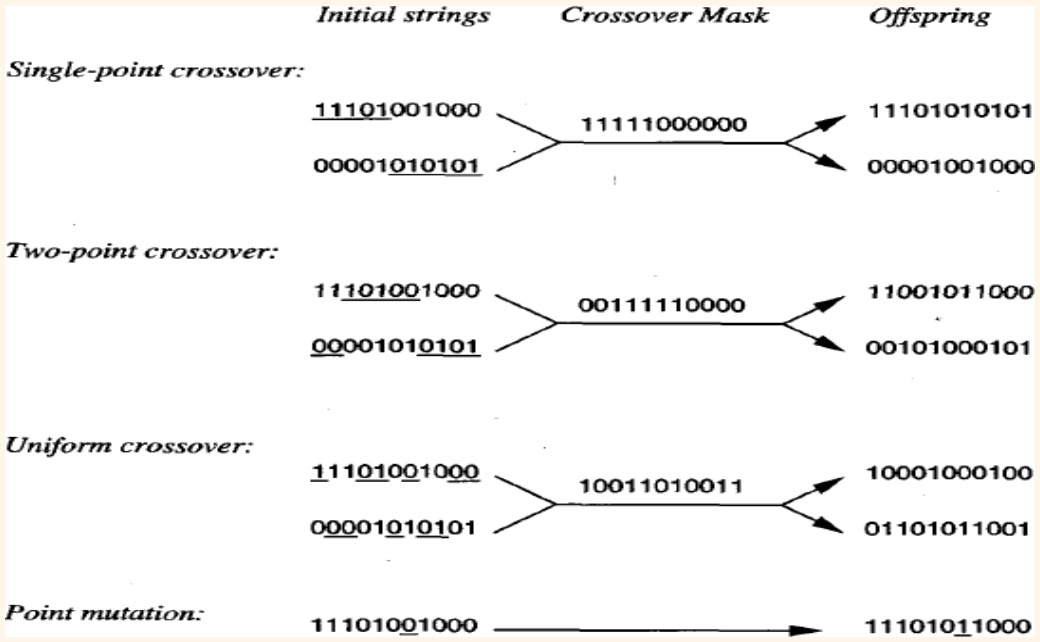
Prepared By

Quaid-e-Awam University of Engineering Science & Technology Nawabshah

Genetic Operators:

Genetic Operators The generation of successors in a GA is determined by a set of operators that recombine and mutate selected members of the current population. These operators correspond to idealized versions of the genetic operations found in biological evolution. The two most common operators are crossover and ***mutation***.

The ***crossover operator*** produces two new offspring from two parent strings, by copying selected bits from each parent. The bit at position i in each offspring is copied from the bit at position i in one of the two parents. The choice of which parent contributes the bit for position i is determined by an additional string called the ***crossover mask***. To illustrate, consider the ***single-point crossover*** operator at the top of Table Consider the topmost of the two offspring in this case. This offspring takes its first five bits from the first parent and its remaining six bits from the second parent, because the crossover mask 11 11 1000000 specifies these choices for each of the bit positions. The second offspring uses the same crossover mask but switches the roles of the two parents. Therefore, it contains the bits that were not used by the first offspring. In single-point crossover, the crossover mask is always constructed so that it begins with a string containing ***n*** contiguous Is, followed by the necessary number of 0s to complete the string. This results in offspring in which the first ***n*** bits are contributed by one parent and the remaining bits by the second parent. Each time the single-point crossover operator is applied the crossover point ***n*** is chosen at random, and the crossover mask is then created and applied.



In ***two-point crossover***, offspring are created by substituting intermediate segments of one parent into the middle of the second parent string. Put another way, the crossover mask is a string beginning with ***no*** zeros, followed by a contiguous string of n_l ones, followed by the necessary number of zeros to complete the string. Each time the two-point crossover operator is applied, a mask is generated by randomly choosing the integers n_o and n_l.

Ahmed Ali

Fitness Function and Selection

The fitness function defines the criterion for ranking potential hypotheses and for probabilistically selecting them for inclusion in the next generation population. If the task is to learn classification rules, then the fitness function typically has a component that scores the classification accuracy of the rule over a set of provided training examples. Often other criteria may be included as well, such as the complexity or generality of the rule. More generally, when the bit-string hypothesis is interpreted as a complex procedure (e.g., when the bit string represents a collection of if-then rules that will be chained together to control a robotic device), the fitness function may measure the overall performance of the resulting procedure rather than performance of individual rules.

In our prototypical GA shown in above Table, the probability that a hypothesis will be selected is given by the ratio of its fitness to the fitness of other members of the current population as seen in Equation above. This method is sometimes called ***fitness proportionate selection***, or roulette wheel selection. Other methods for using fitness to select hypotheses have also been proposed. For example, in ***tournament selection***, two hypotheses are first chosen at random from the current population. With some predefined probability p the more fit of these two is then selected, and with probability (1 - p) the less fit hypothesis is selected. Tournament selection often yields a more diverse population than fitness proportionate selection. In another method called ***rank selection***, the hypotheses in the current population are first sorted by fitness. The probability that a hypothesis will be selected is then proportional to its rank in this sorted list, rather than its fitness.

An Illustrative Example

A genetic algorithm can be viewed as a general optimization method that searches a large space of candidate objects seeking one that performs best according to the fitness function. Although not guaranteed to find an optimal object, GAS often succeed in finding an object with high fitness. GAS have been applied to several optimization problems outside machine learning, including problems such as circuit layout and job-shop scheduling. Within machine learning, they have been applied both to function-approximation problems and to tasks such as choosing the network topology for artificial neural network learning systems.

To illustrate the use of GAS for concept learning, we briefly summarize the GABIL system described by DeJong et al. (1993). GABIL uses a GA to learn Boolean concepts represented by a disjunctive set of propositional rules. In experiments over several concept learning problems, GABIL was found to be roughly comparable in generalization accuracy to other learning algorithms such as the decision tree learning algorithm C4.5 and the rule learning system AQ14. The learning tasks in this study included both artificial learning tasks designed to explore the systems' generalization accuracy and the real-world problem of breast cancer diagnosis.

The specific instantiation of the GA algorithm in GABIL can be summarized as follows: ***Representation***. Each hypothesis in GABIL corresponds to a disjunctive set of propositional rules. In particular, the hypothesis space of rule preconditions consists of a conjunction of constraints on a fixed set of attributes, as described in that earlier section. To represent a set of rules, the bit-string representations of individual rules are concatenated.

To illustrate, consider a hypothesis space in which rule preconditions are conjunctions of constraints over two Boolean attributes, ***a1*** and ***a2***.The rule postcondition is described by a single bit that indicates the predicted value of the target attribute c. Thus, the hypothesis consisting of the two rules

IF al=T^a2=F THEN c=T; IF a2=T THEN c=F

would be represented by the string

a1	***a2***	***c***	***a1***	***a2***	***c***
10	***01***	***1***	***11***	***01***	***0***

Note the length of the bit string grows with the number of rules in the hypothesis. This variable bitstring length requires a slight modification to the crossover operator, as described below.

Genetic operators. GABIL uses the standard mutation operator of above Table in which a single bit is chosen at random and replaced by its complement. To accommodate the variable-length bit strings that encode rule sets, and to constrain the system so that crossover occurs only between like sections of the bit strings that encode rules, the following approach is taken. To perform a crossover operation on two parents, two crossover points are first chosen at random in the first parent string. Let dl (dz) denote the distance from the leftmost (rightmost) of these two crossover points to the rule boundary immediately to its left. The crossover points in the second parent are now randomly chosen, subject to the constraint that they must have the same d_1 and d_2 value. For example, if the two parent strings are

	a_1	a_2	c	a_1	a_2	c
h_1 :	10	01	1	11	10	0

and

	a_1	a_2	c	a_1	a_2	c
h_2 :	01	11	0	10	01	0

and the crossover points chosen for the first parent are the points following bit positions 1 and 8,

	a_1	a_2	c	a_1	a_2	c
h_1 :	1[0	01	1	11	1]0	0

where "[" and "1" indicate crossover points, then dl = 1 and dz = 3. Hence the allowed pairs of crossover points for the second parent include the pairs of bit positions (1,3), (1,8), and (6,8). If the pair (1,3) happens to be chosen,

	a_1	a_2	c	a_1	a_2	c
h_2 :	0[1	1]1	0	10	01	0

then the two resulting offspring will be

		a_1	a_2	c		a_1	a_2	c		a_1	a_2	c
	h_3:					11	10	0				
and												
	h_4:	00	01	1		11	11	0		10	01	0

As this example illustrates, this crossover operation enables offspring to contain a different number of rules than their parents, while assuring that all bit strings generated in this fashion represent well- defined rule sets.
As this example illustrates, this crossover operation enables offspring to contain a different number of rules than their parents, while assuring that all bit strings generated in this fashion represent well defined rule sets.

Fitness function. The fitness of each hypothesized rule set is based on its classification accuracy over the training data. In particular, the function used to measure fitness is

$$Fitness(h) = (correct(h))^2$$

where ***correct (h)*** is the percent of all training examples correctly classified by hypothesis h. In experiments comparing the behavior of GABIL to decision tree learning algorithms such as C4.5 and ID5R, and to the rule learning algorithm AQ14report roughly comparable performance among these systems, tested on a variety of learning problems. For example, over a set of 12 synthetic problems, GABIL achieved an average generalization accuracy of 92.1 %, whereas the performance of the other systems ranged from 91.2 % to 96.6 %.

Extensions:

In one set of experiments, they explored the addition of two new genetic operators that were motivated by the generalization operators common in many symbolic learning methods. The first of these operators, ***AddAlternative,*** generalizes the constraint on a specific attribute by changing a 0 to a 1 in the substring corresponding to the attribute. For example, if the constraint on an attribute is represented by the string 10010, this operator might change it to 101 10. This operator was applied with probability. O1 to selected members of the population on each generation. The second operator, ***dropcondition*** performs a more drastic generalization step, by replacing all bits for a particular attribute by a 1. This operator corresponds to generalizing the rule by completely dropping the constraint on the attribute and was applied on each generation with probability .60. The authors report this revised system achieved an average performance of 95.2% over the above set of synthetic learning tasks, compared to 92.1% for the basic GA algorithm.
In the above experiment, the two new operators were applied with the same probability to each hypothesis in the population on each generation. In a second experiment, the bit-string representation for hypotheses was extended to include two bits that determine which of these operators may be applied to the hypothesis. In this extended representation, the bit string for a typical rule set hypothesis would be

a_1	a_2	c	a_1	a_2	c	AA	DC
01	11	0	10	01	0	1	0

where the final two bits indicate in this case that the ***AddAlternative*** operator may be applied to this bit string, but that the ***Dropcondition*** operator may not. These two new bits define part of the search strategy used by the GA and are themselves altered and evolved using the same crossover and mutation operators that operate on other bits in the string. While the authors report mixed results with this approach (i.e., improved performance on some problems, decreased performance on others), it provides an interesting illustration of how GAS might in principle be used to evolve their own hypothesis search methods.

Hypothesis Space Search

As illustrated above, GAS employ a randomized beam search method to seek a maximally fit hypothesis. This search is quite different from that of other learning methods we have considered in this book. To contrast the hypothesis space search of GAS with that of neural network BACKPROPAGATION, for example, the radiant descent search in ***BACKPROPAGATION*** moves smoothly from one hypothesis to a new hypothesis that is very similar. In contrast, the GA search can move much more abruptly, replacing a parent hypothesis by an offspring that may be radically different from the parent. Note the GA search is therefore less likely to fall into the same kind of local minima that can plague gradient descent methods. One practical difficulty in some GA applications is the problem of crowding. Crowding is a phenomenon in which some individual that is more highly fit than others in the population quickly reproduces, so that copies of this individual and very similar individuals take over a large fraction of the population. The negative impact of crowding is that it reduces the diversity of the population, thereby slowing further progress by the GA. Several strategies have been explored for reducing crowding. One approach is to alter the selection function, using criteria such as tournament selection or rank selection in place of fitness proportionate roulette wheel selection. A related strategy is "fitness sharing," in which the measured fitness of an individual is reduced by the presence of other, similar individuals in the population. A third approach is to restrict the kinds of individuals allowed to recombine to form offspring. For example, by allowing only the most similar individuals to recombine, we can encourage the formation of clusters of similar individuals, or multiple "subspecies" within the population. A related approach is to spatially distribute individuals and allow only nearby individuals to recombine. Many of these techniques are inspired by the analogy to biological evolution.

Up Next

Population Evolution and the Schema Theorem