ISSN:0975-9646

Application of Genetic Algorithms in Machine learning

Harsh Bhasin[#], Surbhi Bhatia[*]
[#]*Computergard.com*
Faridabad, India
[*] *Amity University,*
Noida, Uttar Pradesh,India

***Abstract*— This Genetic Algorithms (GAs) are a type of optimization algorithms which combine survival of the fittest and a simplified version of Genetic Process .It has as yet not been proved whether machine learning can be considered as a problem apt for applying GAs. Therefore the work explores the use of GAs in Machine learning. A detailed study on the success of GAs in machine learning was carried out by R. D. King, R. Henery, C. Feng, and A. Sutherland [4] but it was limited to comparison. The paper takes the example of Chess to apply GA and proposes a new technique to apply GA to machine learning which can substitute the existing methodologies .The work proposed is shown to be robust and thus making the learning a natural process rather than an algorithmic one. The paper relies on the randomness of GAs and their ability to make the population converge towards the desired point using a fitness function and combines it with the concept of feedback similar to that of neural networks.**

***Keywords*— Genetic Algorithm, Machine learning, Classifier, Supervised learning.**

I. INTRODUCTION

Genetic algorithms help in heuristic search. It is a contentious point whether GA's can be applied to machine learning. The point has been explored and explained in the following work by taking example of chess playing. The definition and types of classifier systems have been explained in the first section followed by explanation of machine learning. This is followed by the brief analysis of genetic algorithms. The application of GA's to machine learning taking the example of chess has been explained in section IV. It has been found that if there are many rules to be applied for a particular condition then GA's give an effective solution if the rules can be assigned correct fitness values.

II. CLASSIFIER SYSTEM

A classifier system is a system that learns syntactically simple rules called classifiers through credit assignment and rule discovery mechanisms. These systems recognize new information continuously from the environment. They develop assumptions without altering the acquired capabilities. The above can also be used to make an expert system a reality. Classifier systems determine the ranking among the population members via multiple interactions with the environment whereby the strength changes occur via the apportionment of credit subsystem of classifier system [4].
The performance of the classifier system depends on the classifiers length. The performance increases as the length increases. So the plausibility of it devising an action after encountering a new condition increases.
Classifiers match messages and actions generated from them modify the environment. The systems have detectors and effecters. Detectors select certain aspects of environment and translate the input to a binary form to be processed by classifiers [5]. Effectors translate binary into a form to modify the environment. The components of a classifier System are

A. Input interface
It uses detectors and converts the binary form into standard messages.

B. Classifiers
Classifiers process messages using system procedures.

C. Message list
Message list has the list of current messages

D. Output interface
It generates the output in the desired form.

The steps of execution cycle includes as per the literature review are [5]

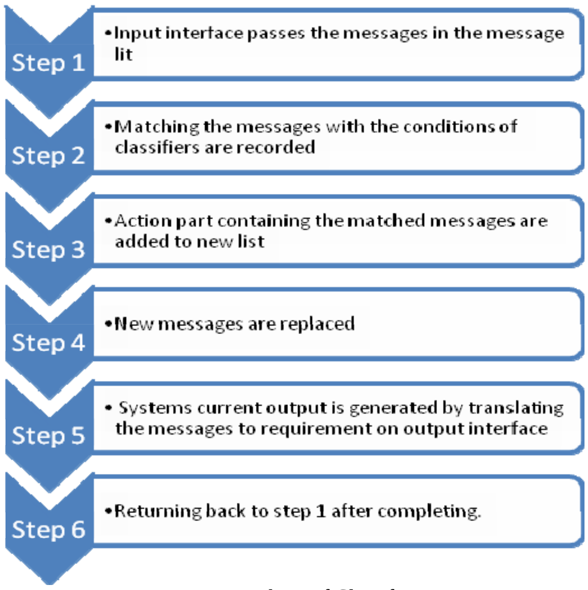
Step 1
•Input interface passes the messages in the message lit
Step 2
•Matching the messages with the conditions of classifiers are recorded
Step 3
•Action part containing the matched messages are added to new list
Step 4
•New messages are replaced
Step 5
• Systems current output is generated by translating the messages to requirement on output interface
Step 6
•Returning back to step 1 after completing.

Fig. 1 Working of Classifier

III. MACHINE LEARNING

Machine learning deals with the design of algorithms that permit computers to develop behaviours based on empirical data. In machine learning the computer learns. A learner can take advantage of examples to incarcerate characteristics of importance of their unfamiliar basic probability distribution. The examples illustrate the relation between the input and desired result. A major task of machine learning research is to automatically learn to identify multifaceted patterns and make intelligent decisions based on examples. The set of all possible behaviours given all possible inputs is too large to be covered by the set of observed examples. Hence the learner must take a broad view from the given examples, so as to be able to fabricate a useful output in new cases [1].

According to Tom M. Mitchell 'A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E'.[2]

The core purpose of a learner is to generalize from its experience [3] .The teaching examples from its experience come from some usually indefinite pattern and the learner has to take out from them something more general that allows it to produce useful answers in new cases.

A. Strategies

There are many techniques and strategies used in machine learning. Some of them are

• Inductive Logic Programming
• Simulated Annealing
• Evolutionary Strategies
• Neural Nets

Each of these methods have been analysed in the previous papers [4] .In this paper only GA will be dealt with.

B. Types of Machine Learning Algorithms

Machine learning algorithms can be classified based on desired outcome of the algorithm. It is of two types: supervised and unsupervised.

- Supervised learning generates a function that maps inputs to desired outputs. For example, in MS Word speech to writing translation is possible. A paragraph is taken and training is provided to the machine i.e. computer in order to learn the pronunciation and ellipses of the speaker.
- Unsupervised learning models a set of inputs, like clustering.

Some more algorithms are based on the above two as described in the following section.

- Semi-supervised learning combines both supervised and unsupervised illustrations to produce a classifier.
- Reinforcement learning learns how to act to a given situation in a particular scenario. Every action has some impact in the environment, and the environment provides feedback in the form of rewards and punishments.
- Transduction tries to predict new outputs based on training inputs.

  Learning to learn learns its own inductive bias based on preceding occurrence.

IV. **GENETIC ALGORITHM**

A genetic algorithm is a search heuristic that mimics the process of natural evolution used to generate useful solutions to optimization and search problems. Genetic algorithms are a subset of what we call evolutionary algorithms which solves optimization problems using techniques inspired by natural evolution, such as inheritance, mutation, selection, and crossover [10].

John Holland, from the University of Michigan started his work on genetic algorithms at the beginning of the 60s. A first achievement was the publication of Adaptation in Natural and Artificial System in 1975. Holland had two aims, first to improve the understanding of natural adaptation process, second to design artificial systems having properties similar to natural systems. Holland method considers the role of mutation and also utilizes genetic recombination that is crossover to find the optimum solution [10].

Crossover and mutation are two basic operators of GA. Performance of GA very depend on them. Type and implementation of operators depends on encoding and also on a problem.

There are many ways how to do crossover and mutation.

A. Crossover

1) Single point crossover -

In this case 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

11001011+11011111 = 11001111

2*) Two point crossover*

Here two crossover points 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

11001011 + 11011111 = 11011111

3) Uniform crossover

In this method bits are randomly copied from the first or from the second parent

11001011 + 11011101 = 11011111

B. Mutation

Mutation is a genetic operator used to maintain genetic diversity from one generation of a population to the next. It is similar to biological mutation [10].

Method given in most of the sources including Wikipedia:

An example of a mutation operator involves a probability that an arbitrary bit in a genetic sequence will be changed from its original state. A common method of implementing the mutation operator involves generating a random variable for each bit in a sequence. This random variable tells whether or not a particular bit will be modified. This mutation procedure, based on the biological point mutation, is called single point mutation. Other types are inversion and floating point mutation. When the gene encoding is restrictive as in permutation problems, mutations are swaps, inversions and scrambles.

The purpose of mutation in GAs is preserving and introducing diversity. Mutation should allow the algorithm to avoid local minima by preventing the population of chromosomes from becoming too similar to each other.

C. Selection

Chromosomes are selected from the population to be parents to crossover. According to Darwin's evolution theory the best ones should survive and create new offspring. There are many methods how to select the best chromosomes, for example roulette wheel selection [10].

1) Roulette Wheel Selection

Parents are selected according to their fitness. The better the chromosomes are, the more chances to be selected they have. Imagine a roulette wheel where are placed all chromosomes in the population, every chromosome has its place big accordingly to its fitness function.

V. PROPOSED WORK

The work proposed intends to find the fitness value of the rule with the help of reinforced learning algorithm .Reinforcement learning algorithms helps an agent to improve its performance by using the feedback it gets from the environment. In reinforcement learning the system receives feedback which makes it closer to supervised learning.

Chess programs suffer problems with positions where the estimate depends primarily on positional features. The point holds as these decisions might not have ramification in the near scenes but have substantial effect in the later stages. The recent programs can look incredibly deep ahead so they are pretty good in scheming tactical lines. Winning material in chess typically occurs within a few moves and therefore most chess programs have a search-depth of at least 8 ply. Deeper search can occur if there are only a few pieces on the board.
Humans are able to recognize patterns in positions and therefore derive important information on what a position is about. Expert players are quite good in grouping pieces together into chunks of information.

Computers analyse a position with the help of their chess knowledge. The more chess knowledge it has, the longer it takes for a single position to be evaluated, here is where genetic can be applied. The playing strength not only depends on the amount of knowledge, it also depends on the time it takes to evaluate a position, because less evaluation-time leads to deeper searches. If to each rule corresponding to a particular condition fitness value can be assigned then it becomes an apt case for applying Genetic Algorithm.

Deep Blue for instance has its strength largely due to a high search-depth. Other programs focus more on chess knowledge and therefore have a relatively lower search-depth. Based on the above approach a strategy of finding out the right move in chess; at the junctions where many moves are possible; has been proposed.

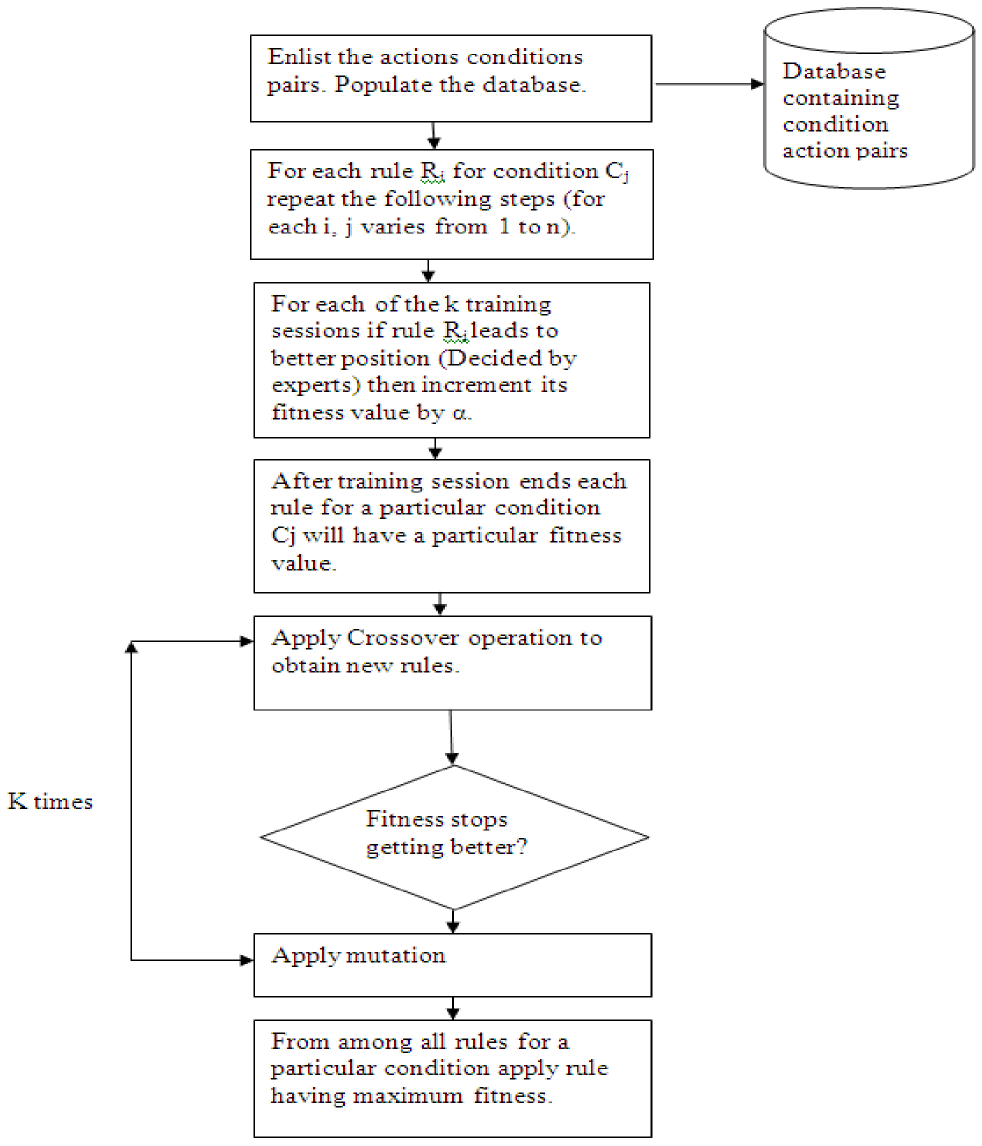
Enlist the actions conditions pairs. Populate the database.
Database containing condition action pairs
For each rule R_i for condition C_j repeat the following steps (for each i, j varies from 1 to n).
For each of the k training sessions if rule R_i leads to better position (Decided by experts) then increment its fitness value by α.
After training session ends each rule for a particular condition Cj will have a particular fitness value.
Apply Crossover operation to obtain new rules.
Fitness stops getting better?
K times
Apply mutation
From among all rules for a particular condition apply rule having maximum fitness.

Fig.2: Process to apply rule Ri for a condition Cj

IV. CONCLUSIONS

The game of chess has following outcomes

White wins

Black wins

No Result: depending on the rules

The following have been implemented so far

A. Board Structure: This is contained in board.cs. The entire board is stored in a 35-byte struct. It is a bit-field which contains 64 1-bit unsigned char variables, each containing the size that the piece at a specific square is on and 64 3-bit unsigned char variables, each containing the type of piece that is on a specific square.
B. Interface: This is in interface.cs. The code being developed directly interacts with the user and specific command .The program is in GUI .The code translates column coordinates, converts numerical char values into actual numerical values and numbers that stand for different types of pieces into strings .
C. Validation: This is in validation.cs. This module returns true or false depending on whether or not a given move could be made on a board with no pieces other than the one moving. This means that a king moving more than two spaces will return false
D. Check Move Validation: This is done via Check () in CMvalidation.cs.
E. Castling: This is handled through if statement in main () and Castle (). It uses variables in boards to make sure that players who have already moved their kings cannot castle.
F. End of Game: This is handled in endofgame.cs. At the end of each round, the program finds out if the game is in stalemate and if the player is in check.
G. Database consisting of rules and conditions.
H. Database consisting of expert opinion and effective fitness values.
I. Classifier program which segregates the rules.

The proposal is being implemented but if completely implemented it will have a better result as the time required for searching the moves will be largely reduced. Moreover it will give a way of incorporating GA in machine learning.

REFERENCES

[1]. Classifier System and Genetic Algorithm , L.B. Booker, D.E. Goldberg and J.H. Holland ,Computer Science and Engineering, 3116 EECS Building, The University of Michigan, Ann Arbor, MI 48109, U.S.
[2]. Mitchell, T. (1997). Machine Learning, McGraw Hill. ISBN 0-07-042807-7, p.2.
[3]. Classifier System and Genetic Algorithm, Robert A. Richards
[4]. R. D. King, R. Henery, C. Feng, and A. Sutherland. A Comparative Study of Classification Algorithms: Statistical, Machine Learning, and Neural Network. In K. Furukawa, D. Michie, and S. Muggleton, editors, Machine Intelligence, volume 13, pages 311–359. Clarendon Press, 1994.
[5]. G. K´okai. GeLog - A System Combining Genetic Algorithm with Inductive Logic Programming. In Proceedings of the International Conference on Computational Intelligence, 7th Fuzzy Days, pages 326–345. Springer-Verlag, 2001.
[6]. William B. Langdon and Riccardo Poli. Foundations of Genetic Programming. Springer, 1998.
[7]. Arthur L. Samuel. Some Studies in Machine Learning Using the Game of Checkers. I. In David Levy, editor, Computer Games, volume 1, pages 335–365. Springer-Verlag, 1988.
[8]. C. Wyman. Using Genetic Algorithms to Learn Weights for Simple King-Pawn Chess Endgames. Technical report, University of Utah, 1999.
[9]. Learning rules from chess databases. World Wide Web. http://web.comlabox.ac.uk/oucl/research/areas/machlearn/chess.html, 1999.
[10]. S. Thrun. Learning to Play the Game of Chess. In G. Tesauro, D. Touretzky, and T. Leen, editors, Advances in Neural Information Processing Systems (NIPS) 7, Cambridge, MA, 1995. MIT Press.
[11]. Alan M. Turing. Chess. In David Levy, editor, Computer Chess Compendium, pages 14–17. Springer-Verlag, 1953.
[12]. Harsh Bhasin, Cryptography using genetic algorithms, ICRITO 2010.
[13].