Using Genetic Algorithm to Select Features for a Pattern Recognizing Neural Network Classifying a Tic-Tac-Toe End Game.

Sukrit Gupta

Research School of Computer Science, Australian National University,
Canberra, Australia
u5900600@anu.edu.au

Abstract. This paper describes an exploratory experiment to select the most important features from a data set containing the results of a tic-tac-toe end game. This approach involves using genetic algorithm as a feature selector and passing the selected features as input to a pattern recognizing neural network that classifies the data into a binary classification of "Win for x" or "not" [1]. The neural network is using K-Fold cross validation to keep results accurate. Several tests were performed with varying population size, stall generation, different crossover and selection functions. The evaluation is based on the resultant best and mean fitness values (error value) and the final chromosome string of important features selected by the genetic algorithm. The terminal results for these criteria's were 17.85% and 22.2% whereas the chromosome string revealed at the end of 168 generation (initial population size 50) run of the algorithm that all nine features contribute with equal importance . Using the mean value for comparison to the previous paper [1] that evaluated the same neural network topology, the results are similar in terms of accuracy (~79.8 %).

Keywords: Tic-Tac-Toe, Genetic Algorithm, Chromosome, Crossover, Fitness Function, Feature Selection.

1 Introduction

This paper describes the results obtained by applying a genetic algorithm to do feature selection for a feed-forward single layered neural network that is being used to recognize patterns from a dataset that represents various end games of Tic-Tac-Toe. Multiple runs of the genetic algorithm with varying parameters were applied to find the most important features required to predict an output having a low classification error.

1.1 Background

Genetic algorithms mimic Darwinian ways of natural selection to find the most optimal solutions to the problems they are fed [2].They belong to a larger set of evolutionary algorithms that define the subfield of artificial intelligence called evolutionary computation. Genetic algorithms have been known to be adaptive and efficient when employed to do feature selection as it is indicated and demonstrated by these papers and publications [3] [4] [5]. The experimenters have the option to tryout various configurations of genetic algorithm to further improve the outputs
.
Genetic algorithms applied to optimization problems output a population of candidate solutions commonly called individuals, creatures or phenotypes so as to evolve towards an improved solution. Every creature has properties (chromosomes) that can be mutated and altered, and generally the solution is a string of 0's and 1's [6] called the chromosome string.

1.2 Problem Model and Investigation Aims

The motivation to apply genetic algorithm for feature selection on a dataset that contains only 9 features and every attribute correspond to one Tic-Tac-Toe square [7] came from the curiousness to experiment whether every square of the tic-tac-toe board holds the same amount of importance or not and how accurately can the neural network recognize patterns when some of those inputs are removed.
It is to be observed that genetic algorithms applied for feature selection inherently work on a multi objective problem, one being the selection of most relevant features and the other being classification error. To keep the classification error accurate 10 K-Fold cross validation method was applied to the neural network.

2 Genetic Algorithm Design

In Genetic algorithm optimization is achieved by an iterative process of manipulating the current population (candidates) of chromosomes to generate a new one using genetic functionalities like mutation and crossover, the population of one iteration is commonly referred to as a generation. During the iterative process the fitness of every candidate in the population is evaluated and the ones with higher fitness values are selected using one of many selection functions (stochastic, remainder, uniform, roulette, etc.) to form a new generation. The algorithm comes to a halt when one of its termination conditions are met. The resulting chromosome string illustrates the optimal feature selection output as computed by the genetic algorithm.

There are five important factors in genetic algorithms that are fitness evaluation, chromosome encoding, selection criteria, and stopping criteria as acknowledged [8]. To begin with, the initial population size is set to 50 of type 'bit string' as genetic algorithm operates in a binary search space. A custom fitness function that contains the feed forward neural network employed to classify the tic-tac-toe dataset into a binary classification is constructed and it returns a classification error to evaluate the performance of its current population. Fitness scaling converts the raw fitness scores using the scaling function "rank" to a suitable value (fittest candidate has rank 1 and next fittest rank 2) for the selection function to work on. After evaluation the candidates with the highest fitness are chosen using the 'stochastic uniform' selection function to be used for mutation and crossover. To provide genetic diversity, mutation function makes small random changes to the individuals in a population which provides the genetic algorithm with a broader search space, in the experiment a 'uniform mutation' was used with a value of "0.01". Crossover, which is analogous to biological crossover and reproduction combines two parents or candidates to produce a crossover child, the "scattered" type of crossover function was used in the experiment. Fig 1 illustrates how scattered type crossover works (see Appendix A). Elite count of 0.05*Population size is used to produce elite candidates that automatically go to the next generation. The elite, reproduction from crossover and mutation populate the next generation. The algorithm terminates when one of the stopping criteria's are met. In the experiment no explicit limit was applied to the number of iterations (generations), a stall generation limit of 20 was engaged. The algorithm stops automatically when the average change in fitness function value over stall generations is smaller than function tolerance. The above criteria was reached when the experiment got to generation '168' and the algorithm stopped automatically. Table 1. Illustrates the summary of the parameters used in the genetic algorithm.

Table 1. Paramaters used by GA

GA Parameters	Value
Population Size	50
Population Type	Bit string
Fitness Function	Classification error
Crossover	Scattered
Mutation	Uniform
Mutation Rate	0.01
Selection Scheme	Stochastic Uniform
Fitness Scaling	Rank
Stall Generations	20

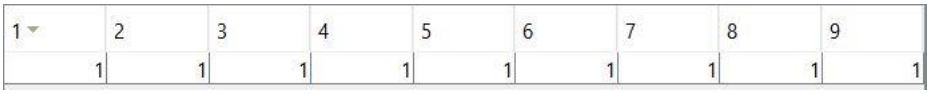
3 Results and Discussion

According to the configurations listed above in Table 1, the genetic algorithm was applied and the results obtained showed similar error rates, with varying accuracy close to the ones mentioned in the original paper [1]. According to MATLAB documentation [9], the mean fitness value and the best fitness value should be close to each other as the algorithm reaches its termination. The genetic algorithm terminated at generation 168.

The best and mean fitness value using the ANN classification error at termination were 0.1785 and 0.2227, the plotted figure (Fig 2[see Appendix A]) validates this result.

The features selected by the genetic algorithm from the Tic-Tac-Toe dataset are represented by the chromosome string output illustrated in Fig 3. Showing all nine features are important for the ANN to produce effective classification results, proving that every square on a Tic-Tac-Toe board plays an equally important part on the outcome of the game.

Fig. 3. Illustrates the Final Chromosome String Showing All Nine Features as Important



To keep the results as accurate as the ones achieved in the original paper [1] 10 K-Fold cross validation was included in the artificial neural network whose classification error was used as the fitness function evaluator.

Comparing the results with the previous paper that uses pure back propagation ANN for pattern recognition the table below shows the differences and how genetic algorithms are not an effective feature selectors for these type of datasets.

Table 2. Comparing Results

Algorithm	Accuracy	Classification Error
Back – Propagation	79.96±2%	20.4±2%
Genetic Algorithm with Back - Propagation	79.8±2.2%	20.2±2.2%

4 Conclusion and Further Work

This paper implemented a genetic algorithm to select features of a Tic-Tac-Toe dataset and used an artificial neural network to calculate the classification error as the return type of its fitness function, then used the fitness function to populate generations. It was not able to achieve its intended purpose of reducing the feature set without increasing the classification error.
However this experiments gives way to further exploratory research for future work like using genetic algorithms to optimize the topology of the artificial neural network classifying the tic-tac-toe dataset. Implementing hybrid models based on genetic algorithms and Support Vector Machines (SVM) and report result after applying it to the tic-tac-toe dataset to check for any improvements.

5 The References Section

[1] Gupta, Sukrit. "Pattern Recognition Neural Network on a Tic-Tac-Toe Game". *COMP8420* 4 pages.

[2] Mitchell, Melanie. *An Introduction to Genetic Algorithms*. Cambridge, Mass.: MIT Press, 1996. Print.

[3] Baudry Benoit. Franck Fleurey.Jean-Marc Jezequel.Yves Le Traon. Automatic Test Case Optimization: A Bacteriologic Algorithm. IEEE Software (IEEE Computer Society) 22 (2): 2005: 7682. doi:10.1109/MS.2005.30.

[4] Crosby Jack L. Computer Simulation in Genetics. London: John Wiley & Sons. ISBN 0-471-18880-8; 1973.

[5] Akbari Ziarati. A multilevel evolutionary algorithm for optimizing numerical functions" IJIEC 2; 419430; 2011

[6] Whitley, Darrell (1994). "A genetic algorithm tutorial". Statistics and Computing 4 (2): 65 - 85. doi:10.1007/BF00175354

[7] Aha, David W. *Archive.ics.uci.edu*. N.p., 2016.

[8] Babatunde Oluleye. Armstrong Leisa. Leng Jinsong Diepeveen Dean. Zernike Moments and Genetic Algorithm: Tutorial and Application. British Journal of Mathematics and Computer Science. 4(15): 2217-2236

[9] Mathworks T. Statistics Toolbox User's Guide The MathWorks, Inc. 3 Apple Hill Drive Natick, MA 01760-2098, 2013.

Appendix A

Fig. 1. Illustrates How Scattered Type Crossover Works

```
parent1 = [1 m n o p g r s]

parent2 = [1 2 3 4 5 6 7 8]

Random crossover vector =

[0 0 1 1 1 0 1 0]

Child = [1 2 n o p 6 r 8]
```

Fig. 2. GA Simulation Plot Over 168 Generations