Implementation Basic Logic Gates By Multilayer Perceptron Using Genetic Algorithm

Majid Nasiri Manjili
Shahid Rajaee teacher training University, Tehran, Iran
majid.nasiri@srttu.edu

Abstract

In this report we have solved XOR classification problem by multilayer perceptron using genetic algorithm, we have aided genetic algorithm to reach optimum weights of multilayer perceptron. Also we have used this algorithm for implementing other common gates like AND, OR, NAND, NAND, NOR XOR and XNOR. This report is based on Machine Learning course.

***Keywords:** MLP, Genetic Algorithm, XOR problem*

1 Introduction

In computer science and operations research, a genetic algorithm (GA) is a meta-heuristic inspired by the process of natural selection that belongs to the larger class of evolutionary algorithms (EA). Genetic algorithms are commonly used to generate high-quality solutions to optimization and search problems by relying on bio-inspired operators such as mutation, crossover and selection [1].

Historically in the 1960s Hans-Joachim Bremermann published a series of papers that also adopted a population of solution to optimization problems, undergoing recombination, mutation, and selection. Bremermann's research also included the elements of modern genetic algorithms.

In this report have solved XOR classification problem by multilayer perceptron using genetic algorithm, we have aided genetic algorithm to reach optimum weights of multilayer perceptron. Also we have used this algorithm for implementing other common gates like

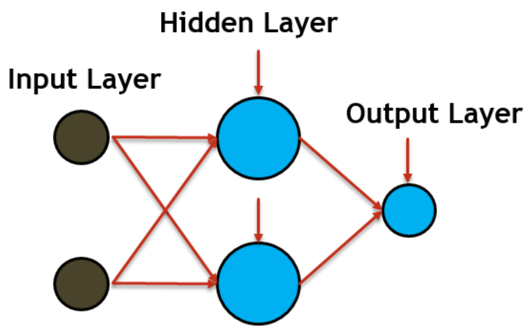

Figure 1: Multilayer Perceptron Model for XOR problem

AND, OR, NAND, NAND, NOR XOR and XNOR. This report is based on Machine Learning course. The MLP is shown in figure 1.

2 Implementation

As shown in figure 1 our model have 9 parameters (weights) to initialize, in fact for solving this problem using GA we have 9 gene in each chromosome of our population. Often, the initial population (first generation) is generated randomly, allowing the entire range of possible solutions. In consequent steps we update our generation by operation such as mutation, crossover and randomly.

In our algorithm to form next population we gave more chances to those parents that have had better performance in current generation. This process performed by roulette selection [2]. We have used mutation and crossover algorithm to generate next generation from roulette output population. Mutation operation is assigning random value to a random gene of a random chromosome of roulette output population. And also we have used crossover algorithm relation (1) in [3] to generate chromosomes for next generation.

$x_{child} = random\_in\_range(x_{parent\ 1}, x_{parent\ 2})$ (1)

For non-decreasing performance of algorithm we copy best chromosome of current generation to next generation. And also to guarantee convergence of algorithm, we generate a portion of next generation's chromosomes randomly.

3 Results

We have applied our GA model for obtaining our MLP optimum weights to solve logic gates especially XOR, for visualization accuracy of each generations we have drawn accuracies of ssuccessive generations in figure 2-7 for different logic gates.

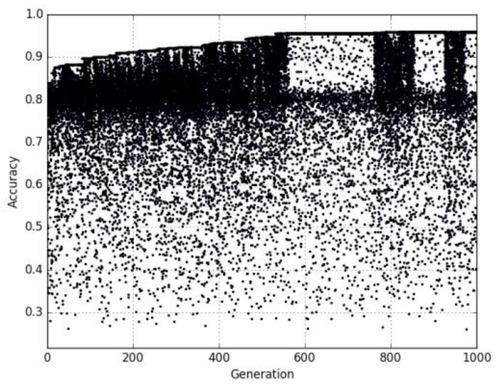

Figure 2: Generation accuracy for AND logic gate.

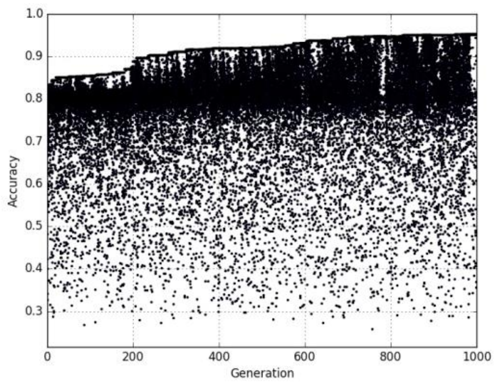

Figure 3: Generation accuracy for NAND logic gate.

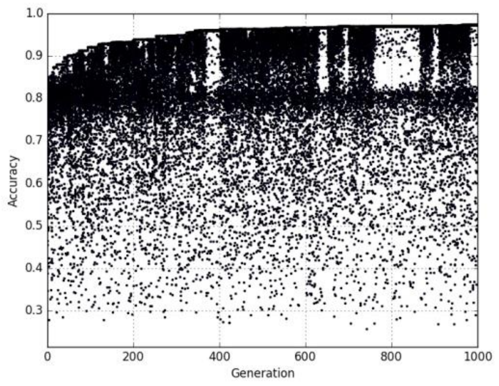

Figure 4: Generation accuracy for NOR logic gate.

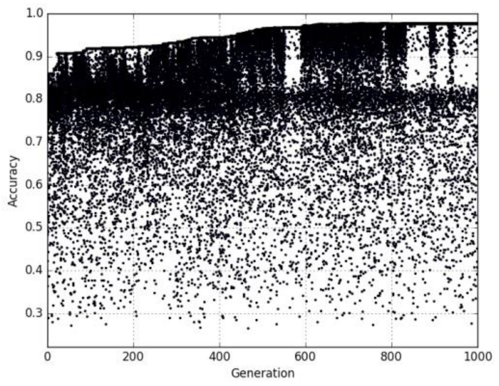

Figure 5: Generation accuracy for OR logic gate.

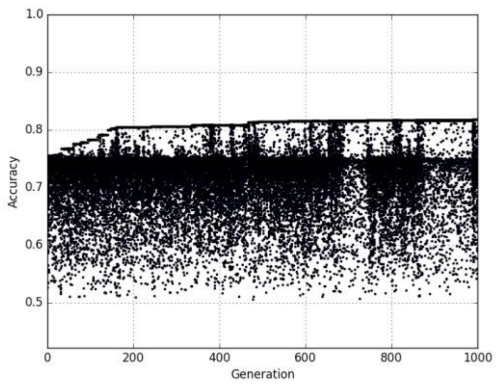

Figure 6: Generation accuracy for XNOR logic gate.

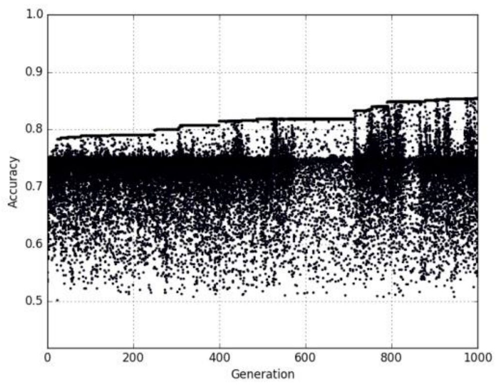

Figure 7: Generation accuracy for XOR logic gate.

4 Conclusion

As shown in figures it’s obvious that by more generation result will converge to more accuracy in all of logic gates. It refers to this point that GA is a universal approximator, and this ability is as result of randomly selection a portion of each generation, that guarantee convergence of error to minimum.

References

[1]https://en.wikipedia.org/wiki/Genetic_algorithm

[2] Houck, Christopher R., Jeff Joines, and Michael G. Kay. "A genetic algorithm for function optimization: a Matlab implementation." *NCSU-IE TR* 95.09 (1995).

[3] Golub, Marin. "An implementation of binary and floating point chromosome representation in genetic algorithm." *Proceedings of the 18th International Conference on Information Technology Interfaces*. 1996.