Jesse Anderson

CSCI 446, Artificial Intelligence

Michelle Van Dyne

11-16-2016

Genetic Algorithms

Welcome to this introduction to Genetic Algorithms. The objective of this paper is to inform you, the reader, on Genetic Algorithms and give enough of a base to make one yourself. To understand and use Genetic Algorithms, we must first begin by describing what makes a Genetic Algorithm, instances in which Genetic Algorithms are typically used, giving examples in which you can review, and go over making your first basic program. (It is assumed that this document is being viewed electronically)

Overview

Genetic Algorithms are just that, Algorithms that apply the theory of Genetics. An algorithm that uses reproduction of Genes that mimic living beings is a Genetic Algorithm. As such, this topic is more about theory or actual mathematics then it is about structure. Genetic algorithms use these core concepts; Generations, Creatures, and Fitness.

Generations

Generations are the outcome of the reproduction of a creature. While reproduction may happen in many different forms, the bases are; Selection, Crossover, and Mutation. Selection being the creature with the higher fitness is chosen to live, increasing its chance to mate or reproduce in the future. Crossover being the mixing of genes between two mates.

Mutation being the random manipulation of a gene into a different gene. These reproduction methods are the core to Genetic Algorithms and are essential to be able to code one yourself.

Creatures

You can't have any work or reproduction being done if you don't have your creature or object. These creatures are made with random variables (Gene values) and are tested against the environment. Because Genetic Algorithms are often combined with other machine learning techniques, such as Neural Networks, the creature itself only needs to know how it reproduces in order to satisfy Genetic Algorithms. The success of the creature which determines its ability to reproduce or not is its fitness.

Fitness

This is the most important part in developing a Genetic Algorithm. While reproduction needs certain variables which it can pass on and use, it does not work if it is not passed on. The fitness function, in which you apply to your creature, is what determines who will reproduce and who will not. It is also key to know that in making this, you also take into consideration of how this function weights the reproduction of good genes and the killing off of not as good genes.

Applying the Algorithm

First a look at how a creature is represented. It is represented by its **Genes** which are connected to form a **chromosome**. The complete set of the genetic material is called genome. These chromosomes and its information are known as the **Genotype**, and its physical representation are known as **phenotype**. When the creature reproduces, it may reproduce

asexually or with a mate. Reproducing with a mate results in the child getting half the genes of each parent and a gene may be **mutated** in the process. This is all necessary for making a Genetic Algorithm.

In order to represent the Genes, a binary format of the problem being solved must be made. In my example to the class in which a creature contains a word and the objective is to have the correct word, a correct letter would be a 1 and an incorrect letter would be a 0. These Genes are then made randomly for the first run, try to avoid having any prebuilt solutions into the Genes as this could lead to a solution that isn't optimal for what you want but what you gave it.

It is imperative to you to understand this last part of your Algorithm as it defines the success or failure of your Algorithm and its entirety. The **fitness score** in which you apply to your creature's success must be made. Without this component, no valuable evolution may take place. The fitness score can range from being simple to complex and is entirely up to you as the coder on what it is. The only concept behind it, is that the fitness score represents the solution in which your creatures are striving to. Some can be a correct final solution, some may just be a better score, others may be more complex in that it achieves higher scores and lower penalties.

Other Information

That is it! If this paper seemed rather short or lacking, it's because most is better explained through examples. However, before we dive into examples of Genetic Algorithms there is some information to know.

History

Genetic Algorithms were developed in 1970 by John Holland at the University of Michigan, United States of America. They were discovered to be useful in avoiding peaks of local optima's in comparison to other algorithms and were used with both discrete and continuous problems. They are however computationally expensive. From a student researching on the topic long ago still hold true.

"Idea of evolutionary computing was introduced in the 1960s by I. **Rechenberg** in his work "***Evolution strategies***" (*Evolutionsstrategie* in original). His idea was then developed by other researchers. **Genetic Algorithms** (GAs) were invented by John **Holland** and developed by him and his students and colleagues. This lead to Holland's book "*Adaption in Natural and Artificial Systems*" published in 1975" (Marek Obitko, 1998)

"In 1992 John **Koza** has used genetic algorithm to evolve programs to perform certain tasks. He called his method "**genetic programming**" (GP). LISP programs were used, because programs in this language can expressed in the form of a "parse tree", which is the object the GA works on" (Marek Obitko, 1998)

It was believed that Genetic Algorithms would be a source in solving NP hard problems. That is, problems that are nondeterministic polynomials in which a guess found by it could be checked in polynomial time. If there was a machine that could guess for us, we would find it more reasonably. (paraphrased from Marek Obitko, 1998) However, it is to be noted that the problems such as the Traveling Sales Person are not yet solved.

Abstracts of Real World Research

Below are examples of real world research being done from old to new. The abstracts of the papers is enough to give an idea of the wide use of Genetic Algorithms.

This article is a tutorial on using genetic algorithms to optimize antenna and scattering patterns. Genetic algorithms are "global" numerical-optimization methods, patterned after the natural processes of genetic recombination and evolution. The algorithms encode each parameter into binary sequences, called a gene, and a set of genes is a chromosome. These chromosomes undergo natural selection, mating, and mutation, to arrive at the final optimal solution. After providing a detailed explanation of how a genetic algorithm works, and a listing of a MATLAB code, the article presents three examples. These examples demonstrate how to optimize antenna patterns and backscattering radar-cross-section patterns. Finally, additional details about algorithm design are given. (IEEE Antennas and Propagation Magazine (Volume: 37, Issue: 2, Apr 1995))

The use of genetic algorithm (GA) to simplify the structures of artificial neural network-based modulation format identification is proposed in next-generation dynamic and heterogeneous fiber-optic networks. Simulation results show that with 80 asynchronous amplitude histogram bins, by virtue of GA, the identification error rate decreases from 4.24 to 1.04 %. (Zhang, S., Peng, Y., Sui, Q. et al. Photon Netw Commun (2016))

In the classical *p*-median problem, the objective is to find a set *Y* of *p* vertices on an undirected graph $G=(V,E)$ in such a way that $Y \subseteq V$ and the sum of distances from all the vertices to their respective closest vertices in *Y* is minimized. In this paper, we have considered the weighted case where every vertex in *G* has either a positive or a negative weight under two different objective functions, viz. the sum of the minimum weighted distances and the sum of the weighted minimum distances. In this paper, we have proposed a hybrid artificial bee colony (ABC) algorithm for the positive/negative weighted *p*-median problem where each solution generated by ABC algorithm is improved by an interchange based randomized local search. In addition, an interchange based exhaustive local search is applied on some of the best solutions obtained after the execution of ABC algorithm in a bid to further improve their quality. We have compared our approach with the state-of-the-art approaches available in the literature on the standard benchmark instances. Computational results demonstrate the effectiveness of our approach. (Jayalakshmi, B. & Singh, A. OPSEARCH (2016))

Examples

There are many examples of Genetic Algorithms that getting a good list of them is not entirely possible. But this is not a problem, we will be going over these examples as they are not too complex and are visually interesting. One example is made by me in under an hour, showing how simple in design Genetic Algorithms can be.

https://www.youtube.com/watch?v=qv6UVOQ0F44 : Mari/o, a Neural Network that uses a Genetic Algorithm to play Mario games (in this case, Super Mario World). Created by Seth Bling.

https://youtu.be/GOFws_hhZs8 : Evolution Simulator, a program made by a rather young programmer of the age of 15 known as Cary k Huang, who started it before college and finished it during his first year of college.

https://www.openprocessing.org/sketch/377698 Evolution Simulator Source code.

https://youtu.be/4FvjOFwU7bE : Evolution Simulator (with hurdles), same program as before but with some obstacles in the way for the creatures. Made by Cary k Huang.

https://www.openprocessing.org/sketch/205807 :Evolution Simulator (with hurdles) Source code.

https://youtu.be/DTUlgZ2qLg8 : Evolution Simulator, Same program with a new fitness function of how high the creature can jump. Made by Cary k Huang.

https://www.youtube.com/watch?v=jAQNiL3o5lU : Evolv.io, an evolution simulator that resembles the likes of Agar.io. Made by Cary k Huang.

https://www.youtube.com/watch?v=pAqrSM3drxw : Evolv.io, continuation of the above code, showing problems that can occur.

https://drive.google.com/drive/folders/0B_6HXWjGWDmFZVlHMzlVTjJQWVE?usp=sharing : Own code: simple word guessing game in which an ai attempts to guess a word using a Genetic Algorithm.

References

http://www.ai-junkie.com/ga/intro/gat1.html

"Genetic Algorithms." *Introduction to*. Ed. Melanie Mitchell, G. Winter, and Gregory J.E. Rawlins. Navy Center for Applied Research in Artificial Intelligence Search Form Search, n.d. Web. 16 Nov. 2016. https://www.doc.ic.ac.uk/~nd/surprise_96/journal/vol1/hmw/article1.html

http://lancet.mit.edu/mbwall/presentations/IntroToGAs/

Obitko, Marek@obitko.com Marek. "GENETIC ALGORITHMS." *Main Page - Introduction to Genetic Algorithms - Tutorial with Interactive Java Applets*. N.p., n.d. Web. 16 Nov. 2016. http://www.obitko.com/tutorials/genetic-algorithms/index.php

Zhang, S., Peng, Y., Sui, Q. et al. Photon Netw Commun (2016) 32: 246. doi:10.1007/s11107-016-0606-7

Jayalakshmi, B. & Singh, A. OPSEARCH (2016). doi:10.1007/s12597-016-0271-8

Resources

http://suraj.lums.edu.pk/~cs431w02/handouts/6_2_ga.pdf

https://en.wikipedia.org/wiki/List_of_genetic_algorithm_applications