Urkund Analysis Result

Analysed Document:	IEEE Paper refactored.docx (D67945965)
Submitted:	4/11/2020 9:02:00 AM
Submitted By:	tina.maru@sakec.ac.in
Significance:	0 %

Sources included in the report:

Instances where selected sources appear:

0

Flappy Bot using AI

Abdeali Saria Department of Computer Engineering Shah & Anchor Kutchhi Engineering College Mumbai, India abdeali.saria@sakec.ac.in Gaurav Pandav Department of Computer Engineering Shah & Anchor Kutchhi Engineering College Mumbai, India gaurav.pandav@sakec.ac.in Mihir Desai Department of Computer Engineering Shah & Anchor Kutchhi Engineering College Mumbai, India mihir.desai@sakec.ac.in Prof. Tina Maru Department of Computer Engineering Shah & Anchor Kutchhi Engineering College Mumbai, India tina.maru@sakec.ac.in

Abstract - In today’s world, an intelligent and optimal problem-solving approach is required in every field. Machines are becoming more and more efficient; many applications have been made recently to solve complex problems. In artificial intelligence, Genetic algorithm is a heuristic search technique to find the most optimized solution for a given problem based on crossover, mutation, selection and some other techniques inspired by Darwin’s theory of evolution. This report demonstrates how it approaches to an optimization problem in general. An implementation of genetic algorithm on building an AI for an arcade game. Our game has an unclear finishing point as it goes on forever and the difficulty varies overtime making genetic algorithm a good fit to train the agent.

Keywords—Genetic Algorithm, Collision Detection, Artificial Intelligence, Feed Forward Neural Network.

I. INTRODUCTION

The goal for the project is to create an AI (bot) efficient enough to master the proposed game below with genetic algorithm as the base for implementation with feed forward neural networks. Our game consists of a spherical agent trying to maneuver through the space between the obstacles in its course. These obstacles have different heights and spacing overtime, additionally the speed and number of obstacles increase progressively making it harder for the AI. The rule of the game is that our agent is always falling towards the floor and we flap (move up) to move between the obstacles. If the agent falls and touches the floor, or if the agent collides with one of the obstacles then the game ends. Every obstacle avoided increases the score which provides a metric to judge the performance of the AI.

Genetic algorithm follows the same fundamental processes that mimic the evolutionary process of nature and justify Darwin's theory of evolution also demonstrating adaptability and survival of the fittest aspect of nature. The universe is finite, its resources finite, competition for these resources leads to the fittest ones dominating others. Similarly, the fitness function of our project will determine the fitness of our agent which in turn will establish its performance factor. Genetic algorithm employs previous generation’s information to further enhance the performance of the population of the descendant generations. This evolutionary approach was most inspiring and has multiple uses in other projects as well. We looked upon previous projects based on genetic algorithm and were amused by them and hence decided to implement an AI for the game using neural networks and genetic algorithm.

Fig 1.1 Population of agents Fig 1.2 Best performing agent

II. LITERATURE SURVEY

The related work in this domain is primarily by Google Deep mind are able to successfully train agents to play the Atari 2600 games using deep reinforcement learning, surpassing human expert-level on multiple games. These works inspired this project, which is heavily modeled after their procedure. They use a deep Q-network (DQN) to assess the Q-function for Q-learning and also use trained experience replay to de-correlate learning. Their approach is naturally state-of-the-art and was the main catalyst for deep reinforcement learning, after which many papers tried to make improvements. The main quality is that they were able to train an agent despite extremely high dimensional input (pixels) and no description about essential game parameters. In fact, they are able to surpass a human expert on three out of seven Atari 2600 games. We have read papers of other similar projects based on the idea of building an AI that can master the game and found them extremely intriguing and helpful and have tried a simpler approach since we are beginners in the field of artificial intelligence and found that genetic algorithms and neural network combination is best for these sort of optimization problems.

Outline of Genetic Algorithm process -

Genetic algorithm starts off with a population of nodes (agents in our project) all containing random weights or information from the input and therefore, performing random decisions to lead towards the goal destination. Since the algorithm is evolutionary, the information about the inputs, weights and decisions of previous generations are preserved, so that the future generations perform better towards reaching the goal state, justifying evolution. The historical information of previous generations has to follow an important constraint before it is stored for future use, which is, from the entire population of agents, information of only those agents that have performed better than the others will be saved to pass onto the next generation. However, the information to be passed onto the next generation is not passed in as raw data to the future population of agents, as it will most likely in all scenarios perform exactly the same way the previous generations performed from which the data was extracted. Hence, crossover and mutation is performed to make sure the behavior of the new set of agents have some variety, and they make different or rather, better decisions than the previous population of agents. The steps in genetic algorithm are briefed below:

Fig 2.1 Process of genetic algorithm

Initialize Search Space: Creating a population of nodes inside the search space where every node of the population makes a decision to provide the solution to the problem statement is the first step. In our project, the search space would be the population of agents created that have random weights and make random decisions at the start and further refine them towards the goal state.

Selection: Selection is the process of picking the best performing or the fittest agent, so to speak from the search space by comparing the fitness values of the entire search space and save it.

Crossover: Crossover is the process of creating a new individual by using best performing individual of previous generation and similarly generating an entire population of agents. Since the new search space is derived from the previous best agents it will generally be better than the previous generations.

Mutation: Mutation is one of the imperative steps of genetic algorithm as it ensures variety in the decision making of each individual to be distinct from one another. It is performed after crossover and after every new generation of the search space is created for the next cycle. Without mutation we won’t have variety and it is important that each agent is mutated individually rather than mutating the entire population at once. It ensures that the entire new generation doesn’t make the same decisions and has diversity.

III. PROPOSED SYSTEM

Genetic algorithm has been successful in complex engineering applications that involved multiple objective, not well-defined optimization. Thus, in our project by using genetic algorithm with an underlying multi layered neural network we were able to train an agent to learn and master the game, quickly and more efficiently as compare to humans. The project also demonstrates how AI’s for other applications and games can also be created with the same idea of this project. i) Search Space - Population of the agents that will attempt on learning the game, while starting with random weights. ii) Selection - After calculating the fitness parameters of the population we select the fittest agents for further steps. iii) Crossover - Using the selected agents that performed the best out of the population, we create a new generation of agents for improving the agent’s fitness score. iv) Mutation – The new generation of agents are randomly mutated using a gaussian random function altering the weights of the neural network. These steps are repeated over generations till solution is found

Architecture: Our neural network architecture uses a feed forward neural network which can also be called a multi layered perceptron with two layers containing one hidden layer and one output layer. There are five input nodes in the neural network that are fed to the hidden layer which has eight hidden nodes which use sigmoidal activation function to process the inputs from the input layer and pass it to the output layer. The output layer has two nodes as we can narrow the outputs down to a simple classification problem of whether to move up or to not move up. It uses softmax function as the activation function which returns a probabilistic value used for decision making and performs the action based on the output values of these two nodes.

Fig 3.1 Neural network Architecture

IV. RESULT AND ANALYSIS

It can clearly be seen that there is not much concrete mathematical foundation to genetic algorithm. The algorithm itself is heavily derived from biology. The graph below visualizes the results that were obtained after a test run. It is evident that more often than not after each generation the performance of our agents improve drastically until it completely masters the game which was our final goal.

Further improvements can be made using convolutional neural networks, using deep learning or reinforcement learning techniques or improved by changing the neural network architecture and making further refined calculations to create a fitness function that would improve performance. This approach can also be used to implement AIs for more games that improve the overall experience.

Fig 4.1 Results of test run

V. CONCLUSION

We have created a working game that detects collision when the agent collides with the obstacles and masters it with every passing generation. We were excited to see that such arbitrary mutations can create AIs that can outscore any human expert. Our AI agents use a feed forward neural network architecture which can be changed for better performance. We felt great while working on this project, and learnt a lot about evolutionary approaches or algorithms. The project demonstrates an evolutionary approach to solve a non-optimized problem using AI and neural networks.

VI. REFERENCES

1. Similar project: http://cs229.stanford.edu/proj2015/362_report.pdf2

2. http://theory.stanford.edu/~amitp/GameProgramming/AITechniques.html5 3. Demonstrations of Genetic Algorithm process: https://towardsdatascience.com/genetic-algorithm-explained-step-by-step-65358abe2bf?gi=dd893c99e3e 4. An introductory guide to Neural Networks:

https://www.analyticsvidhya.com/blog/2018/10/introduction-neural-networks-deep learning/

5. A study on Genetic Algorithm and its Applications: https://www.researchgate.net/publication/309770246_A_Study_on_Genetic_Algorithm_and_its Applications

6. Genetic Algorithm theory: http://theory.stanford.edu/~amitp/GameProgramming/AITechniques.html5

7. Wiki pages of associated topics.

Hit and source - focused comparison, Side by Side:

Left side: As student entered the text in the submitted document.
Right side: As the text appears in the source.