Super Mario AI

CS 512 - Artificial Intelligence Project

Rishabh Singh, Chirag Khurana, Aditya Tiwari, Abhishek Goyal
2016csb1054, 2016csb37, 2016csb1029, 2016csb1027

Abstract- In this project we created a program to navigate successfully through 2D levels based on Super Mario Bros platforming game, using various AI Algorithm considering the advantages and disadvantages.

Keywords- Super Mario, NEAT, Astar, Q-Learning

I. Introduction and motivation

With the increasing popularity of autonomous vehicle, unmanned aerial vehicle, humanoid robots and etc., new methods of controlling complex system to avoid object are points of interests among control engineers. Many classical control methods are based on accurate models or concrete knowledge to derive dynamics of systems and corresponding control laws. For complicated or unknown dynamics systems, classical control methods are not very efficient and new systematic methods are needed to solve such problems. Machine learning draws much attention in last decades and is well applied in many areas such as shopping recommendation systems, computer vision, speech recognitions and etc. Applying Machine learning to solve complicated control problems is investigated in this paper.

We suppose that games are ideal environments to carry out our tests. So we have chosen "Mario AI" to apply some concepts of machine learning algorithms. The game "Mario AI" is a 2D FPS based game in which mario agent try to clear levels as fast as possible. There are various type of obstacle like walls, enemies, pipe etc. The enemy consist of Goomba, enemy Flower, Turtle etc. The action space consist 6 keys which are the combination of the up, right, left, down, speed and Jump. So in total we can have 64 combination of actions but out of which only some moves combination are only valid.

II. Related Work

There have been a lot of work in this field of Artificial Intelligence, Some of which are listed below.

1. Gabe Grand, Kevin Loughlin. Reinforcement Learning in Super Mario Bros. 2018 (link).
2. Stanley, Kenneth O., and Risto Miikkulainen. "Efficient reinforcement learning through evolving neural network topologies." In Proceedings of the 4th Annual Conference on Genetic and Evolutionary Computation, pp. 569-577. Morgan Kaufmann Publishers Inc., 2002. (link)
3. Julian Togelius, Sergey Karakovskiy, and Robin Baumgarten. The 2009 mario ai competition. In IEEE Congress on Evolutionary Computation, pages 1–8. IEEE, 2010. (link)

III. Environment

We have basically used two mario based framework out of which we have following properties have in common.

1. Partially Observable

2. Single Agent
3. Stochastic
4. Sequential
5. Discrete.

Here are the details of each of each environment.

1. **OPEN AI GYM**

It uses FACUX emulator to run Mario game. It actually is a real time environment i.e. it gives almost 5ms to take any actions which is the main difficulty of this framework. It gives us a 16*13 state space which contain the details of the enemy, mario, objects and blank gaps.

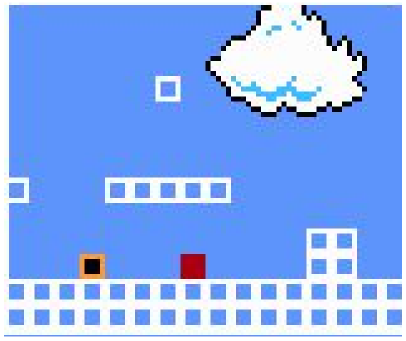
Fig1 : Showing the details of open Gym environment.

2. **MARIO AI CHAMPIONSHIP**

Mario AI championship is java framework created for Mario championship in 2009 to 2012. It is better than the above environment because it gives us a detailed information of environment and also a relaxed time frame of around 40 ms to perform any action.

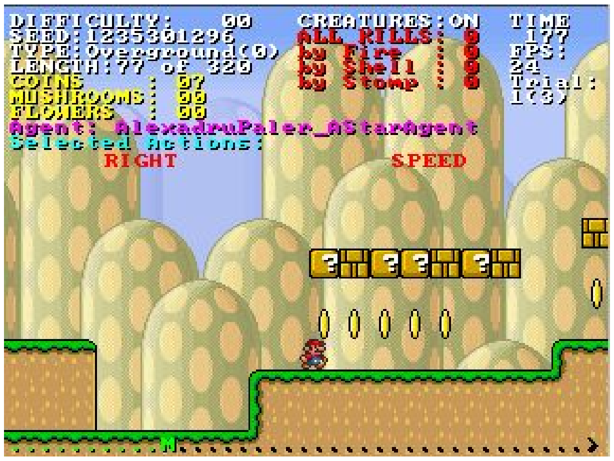

Fig2 : Mario AI Championship Environment

IV. Solution Approach

We tried different models to train mario to navigate and complete levels.

- Simpler Agents:
  - Forward Jumping Agent
  - Reflex Agent
- For Fixed Levels
  - NEAT
  - Deep RL
- Custom Randomly Generated levels
  - A* Search

IV.a. Reflex Agent

Reflex agent is very simple agent which performs actions based on some specific conditions i.e. we have some condition-action rules which help reflex agent to cross the level. We decide the best possible move for mario considering the current and some of the previous states of mario. These previous states help in predicting the speed of mario and the movement direction of enemies. For example if mario is just before a pipe and the pipe is long enough then we need to have 4 consecutive jump in four frames and then a jump and rightmove to let mario cross that pipe. Similarly if there is a gap in front of mario then it try to jump to furthest position. Basically a general rule is that mario should jump to the furthest position if that is possible and if not then it should jump to the uppermost right position.

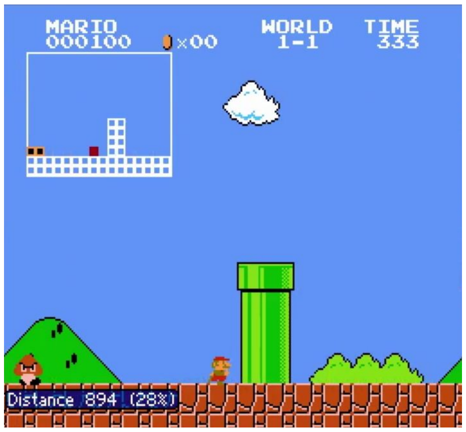

Fig 3 : Mario trying to cross the pipe

IV.b. Neuroevolution of Augmenting Topologies

NeuroEvolution of Augmenting Topologies (NEAT) is a genetic algorithm (GA) for the generation of evolving artificial neural networks (a neuroevolution technique) developed by Ken Stanley in 2002 while at The University of Texas at Austin. It alters both the weighting parameters and structures of networks, attempting to find a balance between the fitness of evolved solutions and their diversity. It is based on applying three key techniques: tracking genes with history markers to allow crossover among topologies, applying speciation (the evolution of species) to preserve innovations, and developing topologies incrementally from simple initial structures ("complexifying").

In NEAT, Single Genome consists of the neural network encoded into all the node genes and the connection genes. Mutation results in either addition of a new connection between two existing nodes, or adding a new node on an existing connection. For crossover we use following method

- Each connection gene is assigned a historical number and during crossover, same connection genes are aligned to ensure the new neural network is not mutated.
- For disjoint genes with same historical marking, the better fit parent gene is chosen for the child.

For mario we considered fitness value of each genome network to be the furthest distance rightwards that mario travels feeding all the states into that network and obtaining each action.

IV.c. Deep RL

Reinforcement Learning is about taking suitable action to maximize reward in a particular situation. It is employed by various software and machines to find the best possible behavior or path it should take in a specific situation. Reinforcement learning differs from the supervised learning in a way that in supervised learning the training data has the answer key with it so the model is trained with the correct answer itself whereas in reinforcement learning, there is no answer but the reinforcement agent decides what to do to perform the given task.

Q-values are generated by a neural network which takes state, Action and the status of mario as input and then decide the Q-value of that combination. We then used this Q-value to decide the next action of Mario and this process continues till mario is completely trained on that particular level. For reward we considered Enemies Killed, Mario status, Distance covered, Time left etc.

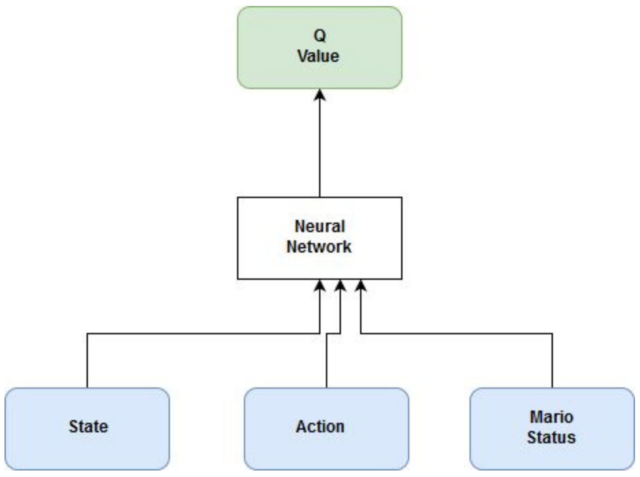

Fig 4 : Neural network to decide the Q-value of the Network.

IV.d. Weighted Astar

In computer science, **A*** (pronounced "A star") is a computer algorithm that is widely used in pathfinding and graph traversal,which is the process of finding a path between multiple points, called "nodes". Above mentioned method has a problem that they need to be trained on a level then mario is able to cross that level but A* has an advantage that is does not require any training and it is very fast and accurate. We used A* algorithm with the heuristic as the weighted average of elapsed time and Utility to complete the level from current position. We also rewarded Mario with negative penalties whenever it try to take an action which leads to death of mario. So how we applied A* without any backtracking. Actually in 40ms we evaluate every action for at least fixed frames in future which helps us in deciding which action is best for fixed or more frames and we do it again for next step. This relaxation helps mario to cross all levels without death. We overestimated our heuristic a little bit (by moving mario to the farthest position) to increase speed in expense of optimality.

The framework we were using didn't give us the ability to simulate an action to know future without actually doing it. Hence we used the AStarSimulator of Robert Baumgarten with some minor changes to incorporate our code. Since the framework was open Source, he had made Cloneable(java) environment and simulated all the physical engine of the game himself which is actually duplicating half of the framework.

Our Final heuristic is:
$w_1 \times part_1 + w_2 + part_2$
$part_1 = (w_3 \times d) - (w_4 \times c) - (w_5 \times e) - (w_6 \times p)$

w_i are weights
$Part_1$ is the Utility of function including the coins kills and etc
$Part_2$ is the time estimated to reach the end of the screen
d = damage received in doing the action
c = coins received in doing the action
e = enemies killed in doing the action
p = power ups collected in doing the action
$w_1 = 1$ (weight of Utility)
$w_2 = 2$ (weight of time,how fast does mario reaches end)
$w_3 = 100000$(weight of damage received,should be high)
$w_4 = 100$ (weight of coins)
$w_5 = 100$ (weight of enemies killed)
$w_6 = 100$ (weight of power ups)

Observations:
More is the weight , more the mario risks its life for it.
Increasing the weight of power ups has no effect as to get a power up one needs to hit the boxes and collect it which requires planning for a lot of steps in future but since there is a time cap of 40ms, the default action is returned before we find the appropriate set of actions for power ups. Default action is move right with speed.

V. Results and Discussion

We almost cross the first level in each of the above mentioned algorithm. But we need to train for around 1.5 days in order to clear mario to 1st level.

A* algorithm runs quite good and give the best result for any randomly generated levels.

VI. References

1. Gabe Grand, Kevin Loughlin. Reinforcement Learning in Super Mario Bros. 2018 (link).
2. Stanley, Kenneth O., and Risto Miikkulainen. "Efficient reinforcement learning through evolving neural network topologies." In Proceedings of the 4th Annual Conference on Genetic and Evolutionary Computation, pp. 569-577. Morgan Kaufmann Publishers Inc., 2002. (link)
3. Julian Togelius, Sergey Karakovskiy, and Robin Baumgarten. The 2009 mario ai competition. In IEEE Congress on Evolutionary Computation, pages 1–8. IEEE, 2010. (link)
4. Mario AI Championship (link)