Design of intelligent non-player character in games

Submitted May 2017, in partial fulfilment of
the conditions of the award of the degree undergraduate.

Like Chen
16518695

School of Computer Science
University of Nottingham

I hereby declare that this dissertation is all my own work, except as indicated in the text:

Signature __Like Chen____

Date 2019/01/08

Content

Introduction

Ever since video games have become an important part of teenagers' life, the variety of games are developing rapidly at the same time. RPG (Role-Playing Game), is one of the most important and most frequently used game types. In a RPG game, players control a character in the game world; players can explore the world, fight against each others... NPC (non-player character), is one of the most significant parts of RPG, NPC can interact with player such as talking to player, fight against player, they can be player's assistant etc. Moreover, NPC may interact with environment, sit down under the tree rest, hold weapon and be alert, or just walk around the street. NPC interact with each other, they may clash with each other, then start to fight with weapons. Maybe two NPC has different camp, and fights once they meet each other. This is, all about building an immersion game world, to let the player enjoy the game play.

When I was playing NPC game such as The Witcher 3, Legend of Zelda: Breath of the Wild, The Last of Us etc. I found that most of the enemy NPC in these games have a lot in common. They patrol some area, they have an 'alert system' (when they first sight player, they do not direct chasing player, they turn into an alert mode, often have a question mark on their head, when alert value full, then they turn in to chase mode, begin chasing player, and when player get out of their sight, they will investigate area that the player finally exists), they interact with game world (pick up weapon when they are chasing player, sitting by the fire, chopping tree with axe etc.), they interact with each other (soldier attack monster, monsters dancing with each other by the fire etc.). In each game developing, developer use much time on design NPC behaviors and NPC decisions. But there is a difficulty in the game: if NPC is too 'smart', the game will be too hard to play, it may frustrate player's confidence, but if NPC is too 'stupid', the game will be too simple for player to feel the achievement. In a word, NPC's intelligent level is hard to control by the developer.

Figure 1: NPC alert value UI

Figure 2: NPC changed mode

Here, the aim of the project is to develop an NPC brain that can be controllable more concisely, while it's inevitable to face and solve the problem that NPCs need to have multiple behaviors in common.

In this project, NPC does not have complex behaviors like this, it may just chase enemy when needed, since this strategy cover most of NPC behaviors. For demo building, I need to build proper game scene, otherwise, I cannot start training. Moreover, I will explain how do I using machine learning tools to solve this problem, since there is no very much document of it, this dissertation may help the beginner to start their research.

Related Terminology and Notions

Unity 3D is One of the most popular game develop editor in the world, used by thousands of developer and adapt by multiple platform(Android, iOS, Windows, MacOS). It is closed source but free for noncommercial usage.

Reinforcement Learning(RL) has been used in machine learning which aims to optimize the prospective rewards of software agent's motivation base on different environments. While comparing agent's normal action to it's optimal performance, the difference gives us the notion of regret. Thus it will analyze the long term reward of its actions even though the immediate reward will probably not positive. Thus, RL is very useful while comparing a long-term vs. short-term reward trade-off. This area has been analyzed in disciplines such as game theory, operation research, simulation-based optimization… [1]

Unity Machine Learning Agent(mlagents) is an open-source plugin that can set up environments for simulation to train the intelligent agent, such that agent can be trained by machine learning methods by Python API, such as reinforcement learning. It also contains state-of-the-art algorithms allow game developers to train for 3D, VR games. The trained agents have multiple uses, here we especially talking about controlling NPC behavior in a variety of settings such as multi-agent and adversarial. [2]

Proximal Policy Optimization(PPO) is a RL algorithm release by OpenAI, it is optimizing RL training, it can make the deviation between current and last policy smaller to minimize the cost function, which is using by mlagents for train the model.

Solving Methods

Summary of problem:

We should build a NPC brain (controller), that can adapt to different environment, which only for the NPC which behavior is human-like. Game developer can simply implement the brain to their NPC, the NPC would have basic behaviour that common one have.

Brute Force:

Code controller script, contain a long if-chain, and it can let NPC act like a human. In fact, nowadays, most of the NPC in game is control by human made script, it is strong and robust for a specific environment, but player is easy to find a pattern of NPC behaviour, also when environment change, the script should be rewrite in some situation since a script cannot adapt itself to different environment. The most important part is the script is specific for single game, any other game cannot use the script. Obviously, a human made script is not the solution for our problem, we need a better solution.

Implement reinforcement learning:

What we should do is to make a NPC agent that can learn by itself, it should adapt to new scene by further learning in new environment by itself. Reinforcement learning is the one that often use in game AI building, since computer game scene can have enough data for the training and it can control the whole environment, hence, when people doing research on Reinforcement learning, their often build game scene for it. So what I think is that I should use Reinforcement learning for my project. There is many Reinforcement Learning algorithm such as Q-Learning, Sasha which people often use to train models.

I can choose the algorithm that fit the project and try to implement it, this implementation of reinforcement learning algorithms should be good enough to train a agent that can adapt to environment update.

However, the problem is we should make a model which can adapt to different games, a model for single game may not able to use in other game

since the data of it is totally different. We should find a more general way to solve the problem.

Using ml-agents

When I do document retrieve, I found that Unity was released a project call Unity Machine Learning Agent(ml-agent) in 2017, this new project connects Unity game scene with TensorFlow, and using PPO which is the advanced Reinforcement learning algorithm release by OpenAI for training.

Moreover, ml-agents is running on the most popular game editor Unity 3D, it using TensorFlow(which is one of the most popular machine learning tool) to train and produce the model, the model can be use in any unity project by the implementation of ml-agents to project which would not change a lot to game scenes.

Project Specification

The project is aim to build an adaptable NPC agent model using ml-agents, this report is more refined than the project proposal.

In the early thinking of this project, I decide to build two demo using same game scene, but the player controllers is implement by script and machine learning model, then find out the difference between this two methods.

Develop Environment Requirement:

Unity 3D 2015 or higher version
ML-Agents beta v0.5 or higher version
TensorFlowShape Unity package

Non-functional Requirement:

Adaptability for different games.

Design

Build up simple game scene for training. Game scene should have different area, such as point ground, alert ground and outside ground, and there is a treasure in the center of the point ground, when thieves run in to point ground, protector should protect the treasure by thieves, and while the improvement of agent model, I can adding extra component into scene. Agent suppose to protect point ground from thieves' damage.

Game should be simple and easy to build, it is nice for train a model since the model is train from nothing.

Train a model that can protect some area or object, automatically choose the patrol route, and it should automatically chase and kill the enemy to protect the area or object.

Compare the trained model with pre-made script, to see the performance, then change the environment and compare again.

Implementation

This section contain how I build up a new game scene and how do I apply mlagents to do the training. And I will explain how to use mlagents also.

First I should introduce how mlagents work.

When we need to use mlagent for model training, we should build our game scenes in correct way:

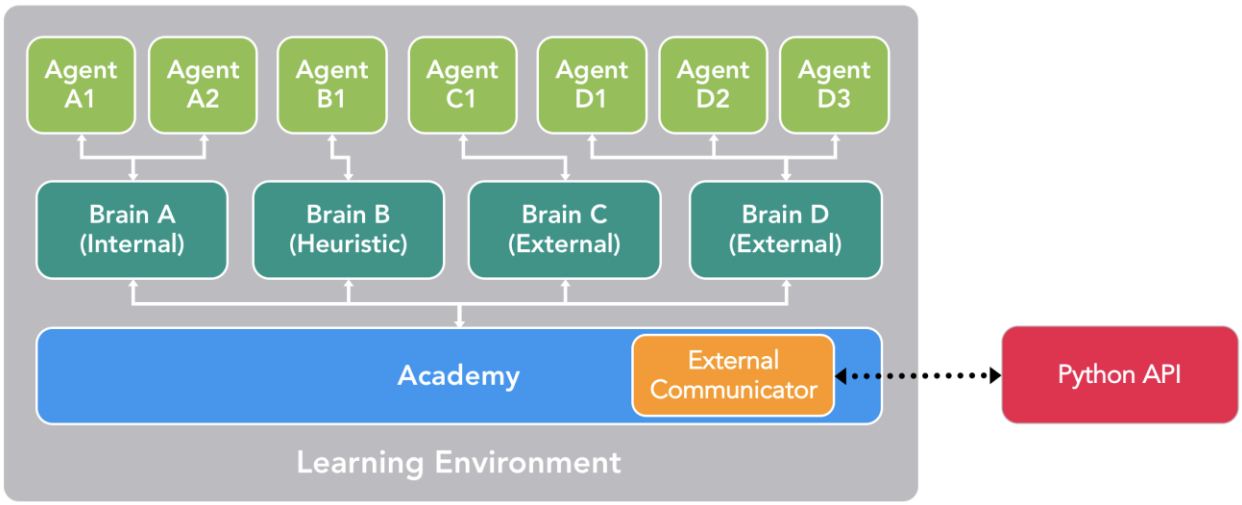
Add **academy**, **brain**, and **agent** to scene, all inherit mlagent basic script, then we can edit our own train script.

Academy: Control the training scene, this game scene is simple but we still need a basic academy set to default for training.

Brain: Make decision for the agent.

Agent: Send information it get to brain and apply the decision brain make, add reward to agent when needed.

In most situation, including this one, we can just edit our agent script, if there is not some special operation we need to apply to our scene. The main idea of agent is that add input data to observer and apply action to agent. The brain would process the input data by TensorFlow and use PPO as reinforcement learning algorithm then send decision as actions back to agent. Mlagents allow multiple brain and multiple agent. We can see clearly in the diagram of Unity document.



[4]

The Internal, Heuristic and External is three mode of brain, allow brain use trained model, human made script and training program to make decision.

I will explain more about how to write a script and build game scene for training in next part.

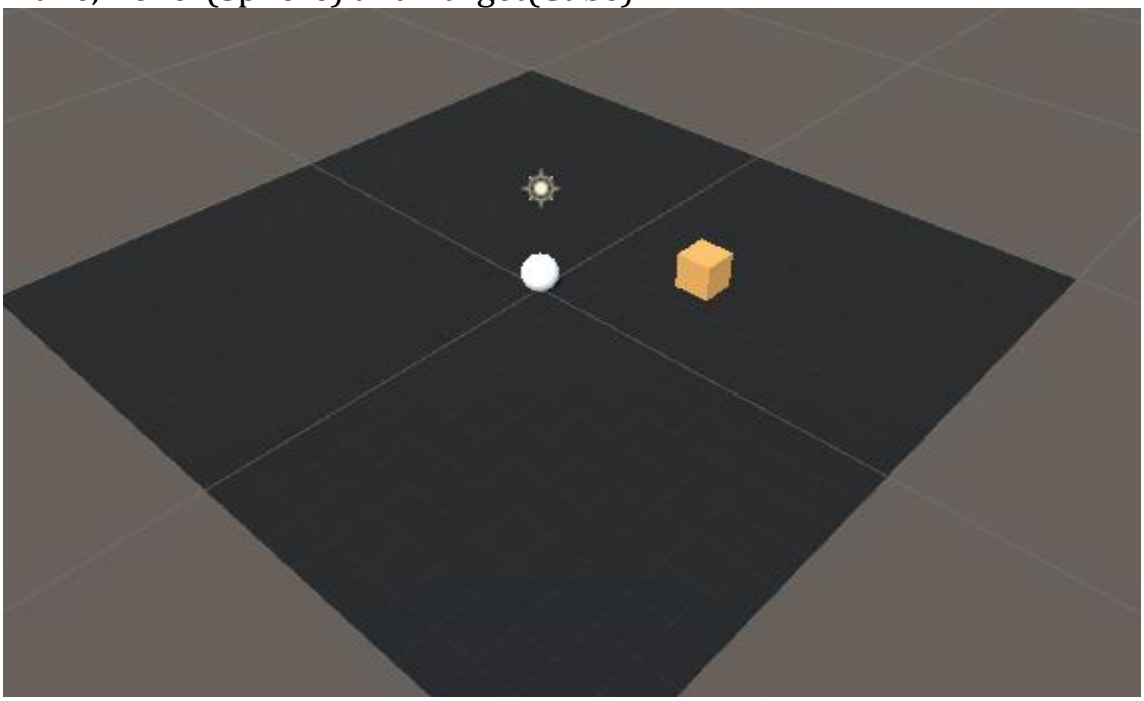
First training scene:

The training the project need a game environment, after discussion with my supervisor, I decide to build a protector and thieves game, game scene have three area identify by color, which is point ground, alert ground and outside ground, and there is a treasure in the center of the point ground, when thieves run in to point ground, protector should protect the treasure by thieves, and while the improvement of agent model, I can adding extra component into scene.

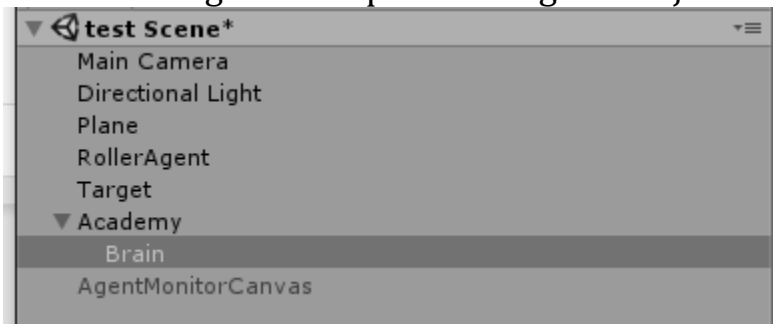
While I trying to build up my own scene, I found that ml-agent already have some example scene for beginner, I build up my own basic game scene for testing.

The test game scene is a simple game: Agent adding force to white sphere, trying to reach the goal cube and try not to fall from the black plane. This game is build all by myself with mlagents' document.

The scene is simple, only contain a few object:
Plane, Roller(Sphere) and Target(Cube).



And link mlagents' script into the game object Brain, Academy and Agent.



Training Setting:

To train a model, brain need input data for decision making, and inside agent script we have a recall function for apply new action to out agent.

Observation:

Use function AddVector inside override function CollectObservations to add input to brain.

```
List<float> observation = new List<float> ();
public override void CollectObservations () {
    // calculate relative position
    Vector3 relativePosition = Target.position - this.transform.position;

    // add relative position
    AddVectorObs (relativePosition.x / 5);
    AddVectorObs (relativePosition.z / 5);

    // Distance to edge
    AddVectorObs ((this.transform.position.x + 5) / 5);
    AddVectorObs ((this.transform.position.x - 5) / 5);
    AddVectorObs ((this.transform.position.z + 5) / 5);
    AddVectorObs ((this.transform.position.z - 5) / 5);

    // Agent Speed
    AddVectorObs (rBody.velocity.x / 5);
    AddVectorObs (rBody.velocity.z / 5);
}
```

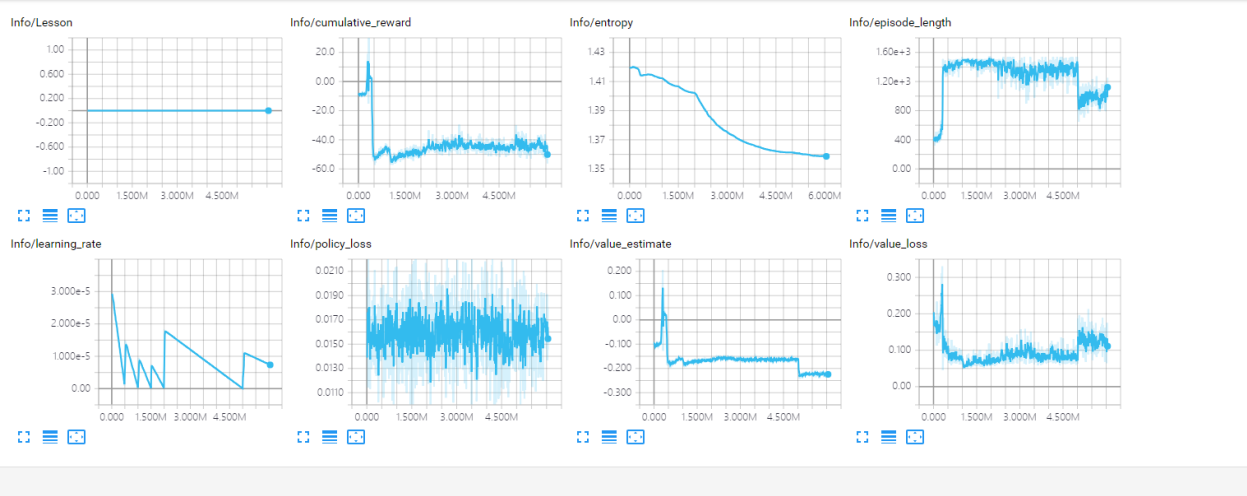
Relative position of agent and goal.
Agent's speed.
Distance to edge of plane.

Reward Setting:

In reinforcement learning, we should tell agent which result is good and which result is bad, then it will adjust itself for optimal decision.

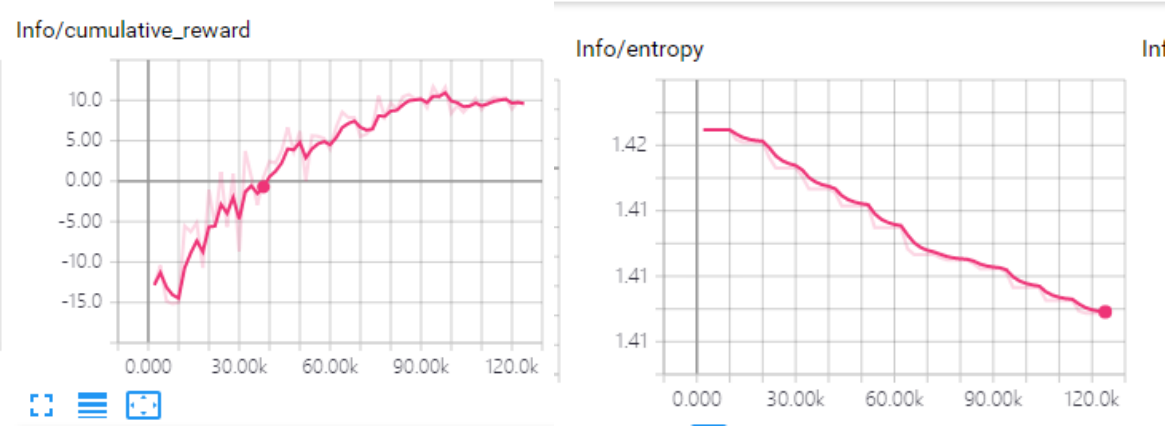
When agent fall from plane add negative reward.
Reach goal cube add positive reward.
Move closer to goal add positive reward.
Give a time punishment to let the training process not dilatory.

While and after training, we can use TensorBoard for produce diagram of training process.



We should more focus on cumulative reward and entropy, while training step increasing, if the cumulative reward is increasing and entropy is decreasing, means that the training is good or we should adjust the hyperparameter or agent script.

Let look back to the test scene training. The result of training was good:

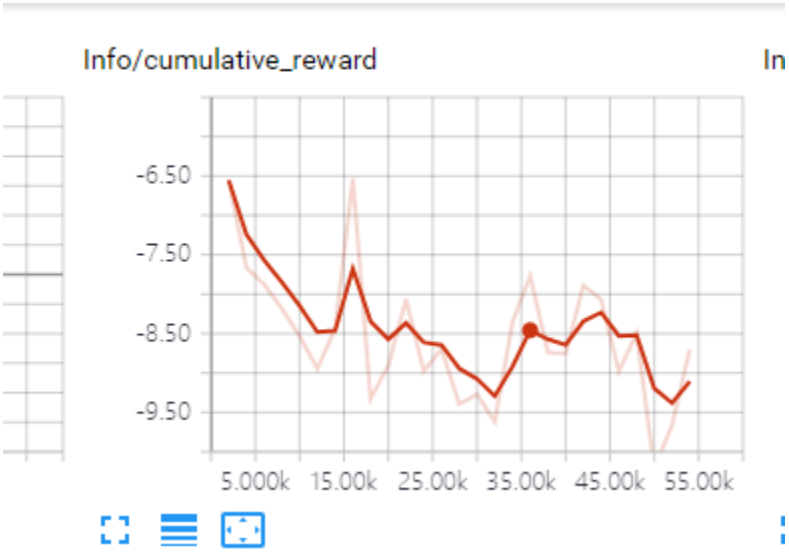


We can see that cumulative reward is keep increasing and entropy is decreasing by when training step increases, it reach 10 and not keep increasing since 90k steps, it reach the optimal position.

This is only 5 minute training on my computer with CPU Intel Core i7-4710HQ and Graphic card NVIDIA GeForce 980M. The training was efficient.

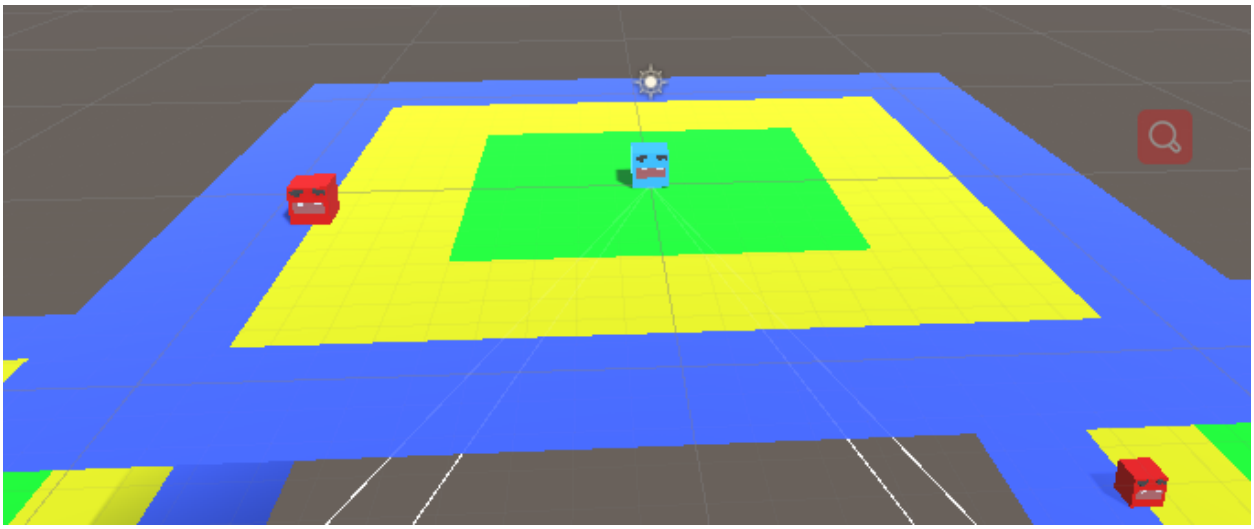
After the testing I found that the mlagents is actually work and work well for train model with Unity game scene. I can said that this tool is useful for my project.

Then I trying to build up my own scene. I try with 10 randomly moving thieves and protector just need to catch them, without any treasure or obstacles in scenes, because I think the game I thought about is too complex for training the first model which is a child without any knowledge before. But the task is still too difficult for agent, even I reduced the complexion of the scene.

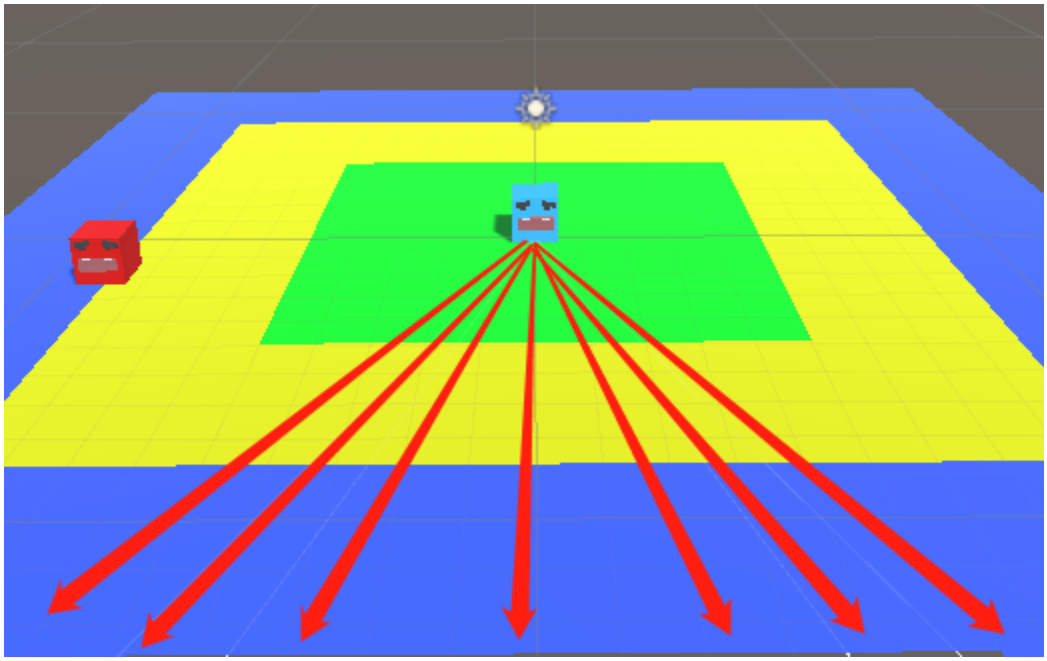


We can see one of the cumulative reward diagram of my test training, the reward was unstable even I trained 50k step for it. It must be something wrong of my work.

So that I read document again, and I found that agent task should begin with a very simple task like 1 + 1 = 2, then I can go 2x + y = 3, so I rebuild the whole scene with simple cubes and one theft (cannot move) and protector only. I think I should start with a really simple problem.



For the training part of the demo scene, I use the idea from mlagents' example(banana collector). Using a perceptron to scan object in front of the agent. Shoot a collider to different direction, return the first game object that



hit then use one-hot data structure to store the data and add to observer. Which is the data input to agent's brain. At this point game scene only have two detectable object: Enemy and wall(the edge of plane has 4 wall).

```
string[] detectableObjects = { "Enemy" , "wall"};
```

Sample one-hot data would be:

1	0	0	1	1

Means that this object is an enemy the relative position of it and agent would be (1,1).

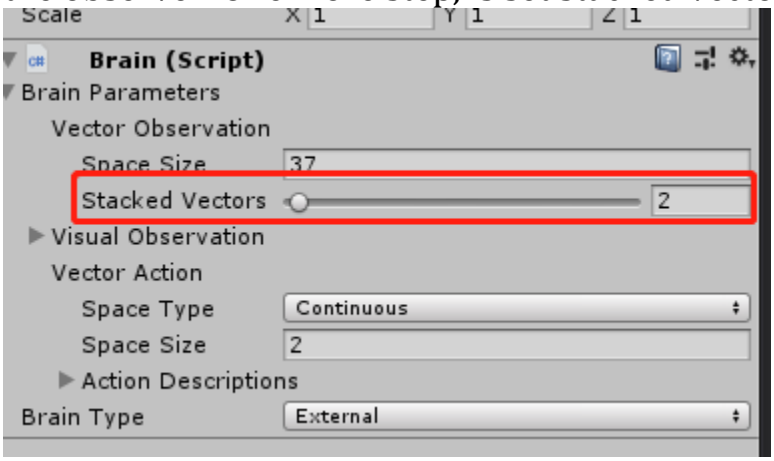
0	0	1	0	0

Means percepter detect nothing.

There would be 7 of these data input into brain since I set up 7 angles for detect.

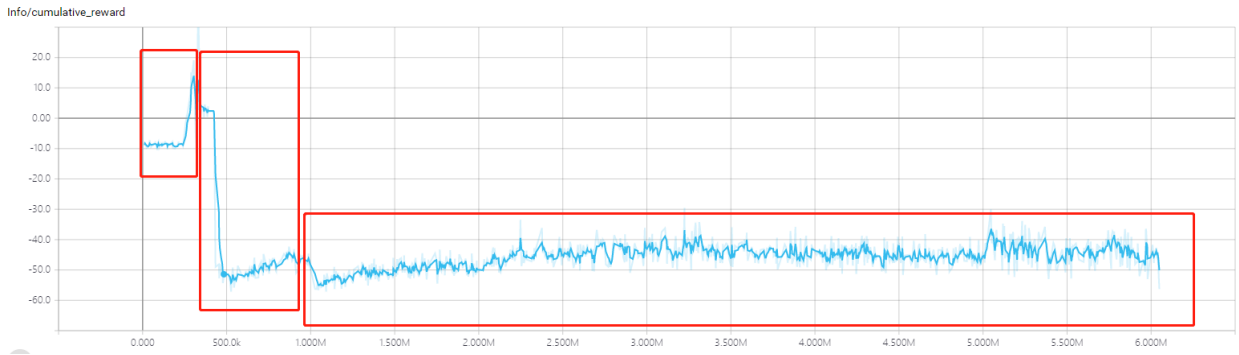
```
float[] rayAngles = { 20f, 90f, 160f, 45f, 135f, 70f, 110f};
```

Moreover, to let agent knows that where it is face to, I add the facing vector to the observer. One more step, is set stacked vector parameter in brain to 2:



This letting brain know that how the data changing, in other word, it knows enemy speed by input of last position and current position.

This two observer settings took me more than 20 adjustments, I use this one since the result of it is work (by observing game scenes, agent can find enemy and would not hit the wall through some training), and for the improvement of model, I try to change the game scene by letting enemy moving, add more enemies to scene, it keep training and keep work well in continue update of scene.



This is the cumulative reward diagram of the model which I train more than 6 million steps and keep trying to adjust the scene.
In the first part of diagram:

Scene setting:
Enemies cannot move when catched it randomly respawn on plane.
30 second per episode.

Reward setting:
Negative reward for hit the wall, adding episode time punishment to it, otherwise agent will keep hit the wall, to let episode end early and the average reward would not be low.
Time punishment reward.
Positive reward for catch enemy.
Positive reward for move closer to enemy.

Result:
The reward is increasing, means that the agent start learning to catch the enemy and not touch the wall.
when I saw that improvement I immediately change scene and let enemy start randomly moving in scene.

Second part of diagram:

Scene setting:
Enemy can move randomly on floor.
30 second per episode.

Reward setting:

Remove some reward since the model do not need that much ‘teaching’ now. so the cumulative reward should be low since the closer reward is removed.
Time punishment reward.
Positive reward for catch enemy.

Result:
The reward is getting significant lower(because the removement of closer reward) then still increasing, the agent is keep learning to catch the moving enemy.

Third part of diagram:

Scene setting:
Enemy can move randomly on floor.
One more enemy in scene.
30 second per episode.(for observing the average reward per episode which show in command line)

Reward setting:
Same as before.

Result:

The cumulative reward become unstable, I think the model need more training, so I train it 5 million steps, but when I observed the game scene the only change is that agent moving become less random, it get to it goal more ‘confident’. Hence, I think it reach the optimal point for current scene, since after this much training the cumulative reward is not increased.

Progress:

Know the installation of mlagents:
- Clone project from mlagents github
- Install mlagents-learn command line tools
- Add TensorFlowSharp to the project
- Change setting to enable internal mode of brain(use trained model)

Using Unity 3D editor to building proper training scene:

- Knowing how mlagents work, why we need academy, brain and agent script, and how is mlagents train models.
- Building game scene to fit mlagents training environment.

Know how to training using mlagents:

- Adjust training config for more training step, adjust gamma value to control the entropy change etc.
- Know what information should input to brain and which action I should use(brain can produce continuous value or discrete value for different using), and how to adjust awards gain in agent script.
- Use TensorBoard for observing the training result, and change scene to improve model in time.
- Know that for a fresh model, we should give more information to it by add reward for different situation, since the model need more 'teaching' in the beginning.

Demo building:

- First model can catch two enemies by itself.

Reference

[1] https://en.wikipedia.org/wiki/Reinforcement_learning#Research

[2] https://github.com/Unity-Technologies/ml-agents

[3] https://blog.openai.com/openai-baselines-ppo/#ppo

[4]https://github.com/Unity-Technologies/ml-agents/blob/master/docs/ML-Agents-Overview.md

Sample Chapter

Introduction

Ever since video games have become an important part of teenagers' life, the variety of games are developing rapidly at the same time. RPG (Role-Playing Game), is one of the most important and most frequently used game types. In a RPG game, players control a character in the game world; players can explore the world, fight against each others... NPC (non-player character), is one of the most significant parts of RPG, NPC can interact with player such as talking to player, fight against player, they can be player's assistant etc. Moreover, NPC may interact with environment, sit down under the tree rest, hold weapon and be alert, or just walk around the street. NPC interact with each other, they may clash with each other, then start to fight with weapons. Maybe two NPC has different camp, and fights once they meet each other. This is, all about building an immersion game world, to let the player enjoy the game play.

When I was playing NPC game such as The Witcher 3, Legend of Zelda: Breath of the Wild, The Last of Us etc. I found that most of the enemy NPC in these games have a lot in common. They patrol some area, they have an 'alert system' (when they first sight player, they do not direct chasing player, they turn into an alert mode, often have a question mark on their head, when alert value full, then they turn in to chase mode, begin chasing player, and when player get out of their sight, they will investigate area that the player finally exists), they interact with game world (pick up weapon when they are chasing player, sitting by the fire, chopping tree with axe etc.), they interact with each other (soldier attack monster, monsters dancing with each other by the fire etc.). In each game developing, developer use much time on design NPC behaviors and NPC decisions. But there is a difficulty in the game: if NPC is too 'smart', the game will be too hard to play, it may frustrate player's confidence, but if NPC is too 'stupid', the game will be too simple for player to feel the achievement. In a word, NPC's intelligent level is hard to control by the developer.

Figure 1: NPC alert value UI

Figure 2: NPC changed mode

Here, the aim of the project is to develop an NPC brain that can be controllable more concisely, while it's inevitable to face and solve the problem that NPCs need to have multiple behaviors in common.

In this project, NPC does not have complex behaviors like this, it may just chase enemy when needed, since this strategy cover most of NPC behaviors. For demo building, I need to build proper game scene, otherwise, I cannot start training. Moreover, I will explain how do I using machine learning tools to solve this problem, since there is no very much document of it, this dissertation may help the beginner to start their research.