Intro:

The reinforcement learning is one of the fundamental aspects of the artificial Intelligent in games. This project is to implementing reinforcement Learning algorithms on Minecraft and analysis the training result of each bot. The implemented control algorithms used MARLO as a framework. MARLO is a library built on top of Microsoft’s Malmo project which based on Minecraft(Microsoft, 几几年). The three control algorithms are based on Monte Carlo Tree Search, Q-learning and Deep Q-learning.

The aim of this report is to explain each of algorithms separately and analysis the summaries.

Algorithms:

1. Monte Carlo Tree Search Bot:

Monte Carlo Tree Search is a heuristic search algorithm. there are four basic steps of the Monte Carlo Tree Search: Selection, Expansion, Simulation and Backpropagation.

In selection step, the function will start from the current game state as the root and select child nodes until a leaf node L is reached.

In expansion steps, the function will create one or more nodes based on the child node and select one node from the created nodes.

In simulation step, the function will let the game randomly run from the node selected from the previous step.

In backpropagation step, the function will update information which get from the previous running result in the nodes on the path from the expansion selected node to the starting root.

The algorithm of this bot is based on the Monte Carlo Tree Search.

In this bot, the selection steps were implemented as a calculate action (calAction) function. The required information to calculate action is the location of the character (X, Y and Z value), reward that can directly obtained from the MARLO framework, and previous action that stored in the global variable. The function Calculate action taking actions from the action pool.

The Expansion steps were implemented as that the parent action perform a child action that calculate from the calculate action (calAction) function.

1. The required information is location of the character, action and reward, which is the value and reward. The reward can get directly from the MARLO framework
2. Each step
3. The function Calculate action taking actions from the action pool(3, 7, 8) and add the taken action to the child action. child action is the next step The using [UCB1] (select action Algorithms)
4. and using update value function(UVF) to calculate a value and update the value
5. The action can only go forward left and right, so the length need less than 30
6. Q-learning bot

Q-learning is a reinforcement learning algorithm. In Q-learning Q is the action-utility function which used to evaluate the pros and cons of taking an action in a particular status. The combination of status and action is limited. The Q(action-utility function) can be taken as a table that each row in the table records the status (Delta x, Delta z, Yaw), and rewards when selecting different actions (go forward, left, right).The table is initialized to zero, then each row is updated by rewards through training.

1. Deep Q-learning bot
2. Background and literature review of the Deep Q-learning

To comprehend and optimize the given deep Q-learning algorithm, the lecture slides and the online materials about the basic idea of the deep Q-learning and the difference and inner relationship between the Q-learning and DQN has been looking through.

It is obviously unrealistic to maintain a large Q table by the traditional Q-learning method due to the excessive state. However, deep Q-learning is a model-free and off-policy reinforcement learning algorithm which solves the reinforcement learning task by playing games in the emulator following an episodes-greedy policy for exploration of the search space. To approximate the value of the Q-table, neural networks train the actual training sample data.

The basic algorithm of the DQN is that

1. Initialize replay memory and action-value function and target action-value function with each random weight theta.
2. For M episodes initialize the state and preprocessed sequence

For T time steps do

* 1. Use episodes-greedy to select action from action value function
  2. Execute that action in emulator, observe the reward and the next action
  3. Set the next step state equals to this time state, this time action, reward and next time state. Set the preprocess. Then store the transition in the replay memory.
  4. Foreach memory reply, perform a gradient descent step with respect to the network parameters theta.
  5. Every setting step, setting the action-value function to the target action-value function.

Techniques Implemented

The mechanism of the given algorithm of the Deep Q-learning is that:

1. The function initializes the arguments, dictionaries, hidden layers and channels.

The algorithm initializes an environment and connect to the Minecraft client(marlo-server --port 10000).

Initializes the action-value function(q-function), replay buffer, explorer function and Stochastic Gradient Descent(SGD) optimizer function.

Check available GPU

Initialize a replay buffer and its capacity and experiment profile.

Initialize the agent.

1. Trains the agent while regularly evaluating it using the training and evaluating function from marlo framework.
2. Draw the computational graph and save it in the output directory.

However, we change some of the given part:

?????

Experimental Study:

描述为每种算法和游戏进行的实验研究。