Report of implementing RL in mini-game CollectMineralShards.

Problem: the rule of the game is to simply control 2 soldiers to collect the mineral shards over the map. In each episode, there are 20 mineral shards randomly generated over the map. If all the mineral shards are collected before the deadline of the episode, new 20 mineral shards will be randomly generated over the map. The task is to collect as many mineral shards as possible within deadlines. The length of each episode is 1:30 in game time, but the real-time will depend on the speed of the agent making a decision.

Solution: e-greedy algorithm with Q-learning method is used as RL method, with the chance of 0.1 to ignore the score and pick action randomly, decay factor of 0.9 and learning\_rate of 0.01.

3 models have been developed to train the agent, one is a bit complex, with 25 possible actions and 1048576 possible states. With this model, the agent can control 2 soldiers separately and can move them to the exact position of the mineral shards.

Another model is simple, contains 5 possible actions and 256 possible states. However, the agent can only control the soldiers as a group, and can only control the movement as go up, down, left and right.

The third model is using the same actions as the simple model but the states are represented with convolutional layer.

3 reward schemes were used, which are ternary reward, blizzard scoring reward and a transferred scoring reward from blizzard scoring reward.

Each training was done in 1,000,000 steps (4000s episodes).

Expectation: With the complex model, we expect the agent will learn to manage the soldiers separately and control them to find the best routine and collect mineral shards efficiently. With the simple model, we expect the agent will simply learn to control and move the soldiers to a mineral shard by having very a few knowledge of the environment. Using convolutional layer, we expect the result could be more accurate and the agent would be trained faster.

Results: With the complex model, both Ternary reward and blizzard reward converges the score to 1 or 2 points less than the number multiplied by 20, e.g. 19, 38, 58... In addition, Ternary rewarded agent learns to control the soldiers separately to collect different minerals, where blizzard rewarded agent was using a single soldier and leave the other idle most of time. An unexpected behaviour that both agents learn is that when with only a few shards left, the agents were trying to wander between a fixed point and different visited point and leave the shards uncollected. In general, they are better than a random agent. The maximum score the agents can achieve is higher than the scripted agent but the average score is far less than the scripted agent. It took about 9 and half hours to finish the training for each agent.

With the simple model, a transferred reward was used to punish the agent if it achieves a low score. With Ternary reward and blizzard reward, the agent did not learn to have the soldier moving to shards efficiently as sometimes they still wandering around visited points. Whereas using transferred reward, the agent learns to cover the unvisited area where the places were marked as has-resources. All the average scores for 3 agents were low (appx 12) as it is a very simple model. However, unexpectedly, the maximum score for transferred rewarded agent achieved 45 and there are several scores, which means it somehow learned to collect the last shards in a round. It took about 4 hours to finish the training for each agent.

Discussion: For the complex model, major actions come from moving the soldier to a position of mineral shards, and these actions can be repeated. The result shows the convergence because when it gets closer to collect the rest few shards in a round, the fewer action can actually be right. The unexpected behaviour mentioned in the result comes from the lag between action and the state, as the actions that the agent picked usually results in a few steps of the future state, which confused the agent.

For simple model environment for an agent is really simple, the state represents only 4 areas (dividing the map to 2x2) indicating if there are resources to collect and a number to indicate the position of the soldiers in a 4x4 grid of the map. In this model, the action matches the state, there will not be a lag between action and states. It results in a serrated graph of the score because it uses the e-greedy algorithm and sometimes it will choose a random action regardless the reward; in this case, random actions barely represent correct choices. The amount of training is not enough.

Appendix:

