Selfish Q Learning using Neural Nets

Aldo Curtis

March 27, 2015

Word count:2753
Jeremy Gow.
AldoCurtis@gmail.com
https://github.com/Dizzly/AISelfishQLearners

1 Abstract

The intention of this project was to test whether a group of agents using Q Learning could learn to work together given only selfish rewards. Q-Learning can associate actions with their long term rewards and so should make it possible to associate actions that benefit the team with the overall success regardless of direct reward.

Contents

2 Introduction

The concept of the challenge was to take 3 agents and give them the challenge of getting through several floors of dungeon encounters with monsters. Each agent will have different stats giving them specialities and leading to different possible adaptations to deal with the challenges. All agents will be given a maximum health, healing and damage values, with each agent specialising in one of these stats, named Tank, Hots and Deeps to represent health, healing and damage respectively. The game will be turn based, with the heros(AI agents) going first and the enemies attack second. Heros will be given a choice between a few basic action types and some more specific actions involving dealing damage, healing themselves or allies and taunting enemies to force them to attack the agent. Each hero can choose to move room as well, leading them closer to the exit.

3 Hypothesis

Q-Learning associates states with their ideal actions, and can to a certain extent associate each action with its possibility with future success. In actuality the choice of action in many games can heavily depend on the decisions of teammates involving tactics that could be seen as un-optimal for an individual. One of the most distinct examples is the "Holy Trinity" mechanics of MMO's. This is where a characters are specialised to the point where some characters only objective is to take as much damage as possible. This can be perceived as creative use of mechanics in a game to defeat difficult obstacles. The hypothesis is that this behaviour, as well as even simpler teamwork behaviours could be developed using Q-Learning without considering complicated variables such as other agents intentions. And that Q-Learning could be used to find other possible "creative" use of game mechanics to defeat obstacles that require teamwork.

4 Research

Q-Learning hopes to optimize the Q value for a state action pair $Q(s, a)$, the Q value representing how profitable that action will be given the state based on a $R(s, a)$ being the reward for that action. Q-Learning however encodes some information about future possibilities in the Q value so that $Q(s_i, a) = R(s_i, a) + MaxQ(s_t, a)$ where we find the maximum Q value given a simulation of the next state after that action. Normally a discount value is associated with the next action, so that $d{*}MaxQ(s_t, a)$ so that future actions can be considered, but weighted.

The simplest implementation of Q-Learning involves using a table to store the Q values for state-action pairs, however this explodes in complexity when any more then a few simple states are included. For more complex scenarios there are several possibilities to simplify the information for use with Q-Learning, techniques such as Fuzzy Logic could convert floating point numbers into discreet states for simpler table insertion, however this works by reducing the precision of the information given to Q-Learning. Another method is to replace the Q Learning table with a neural network, taking in all the variables and outputting the Q Values for the suitability of the action given the state. Using neural networks with Q Learning is well documented and easily extendible, it is also much easier to save to file and reproduce later.

I researched multi-agent implementations of Q-Learning to solve one of the potential problems with the project which is that each agent does not take into account other agents actions. All implementations of multi-agent Q Learning involved associating reward values with the given actions of other agents, which would contradict with the aims of this experiment. Alternative options would be to have each agent predict the actions of the others using another learning technique such as N-Grams, however implementing this would exceed the time requirements of the project.

5 Implementation

5.1 Structure

The actual "game" that the AI must learn is a simple turn based combat game. On each turn each agent will choose an action which will be performed on the state in order of choice, after all hero turns have been taken then any remaining enemies will attack their targets and that round will be over. Neither enemies nor heroes can act if they are at or below 0 health, and if an enemy finds itself targeting a hero who is considered dead they will switch targets to the next lowest health hero. One special action that the heroes can take is to move room, getting them closer to the exit while also spawning more enemies, enemies spawned in this way cannot act immediately and must way one round, this is to give the heroes time to deal with them.

The state converts to a QState which simplifies the data into a form that can be direct fed into the neural network. The neural network then replaces the Q table and outputs 10 floating point values relating to the 10 possible actions. Each action overrides a common base class called QAction, which contains the abstract function PerformOnState which is where the meat of the agent logic is held.

The neural net of a successful AI can be saved for future use or testing. Neural nets are easy to serialize and the FANN library offers a useful function to save them to file.

A *config.txt* file holds some variables that can be used to control the difficulty of the game.

5.2 Q-Learning and Neutral Net

I used extrapolative Q-Learning taking the best possible action after finding that exploration performed much more poorly even after fairly long training times the neural network found a local maxima before exploration could discover a superior strategy. For the neural network I used a library called FANN to perform on-line learning.

The Agents are given 9 basic actions to choose from each turn. These actions represent general interactions with the agents cohorts and enemies. Only one action can be chosen per turn.

1. Attack Lowest Health Enemy - ALE
2. Attack Highest Health Enemy - AHE
3. Attack Random Enemy - ARE
4. Taunt Lowest Enemy - TLE
5. Taunt Highest Enemy - THE
6. Taunt Untaunted Enemy - TUT
7. Heal Self - HSF
8. Heal Lowest Health Ally - HLA
9. Heal Most Targeted Ally - HTA
10. Move Room - MRM

The 3 attack actions arfe relatively self descriptive, from human observation ALE could be seen as the best possible choice as it has the highest chance to kill an enemy, thus removing their possibility of damaging you, but the option to do all three are included. Once an enemy has had its health reduced to 0 it will die and be removed from the game, thus if two agents select the ALE and the first agent kills its target the second ALE action will target a different enemy.

When an agent taunts an enemy it forces the enemy to attack it, this is almost always immediately punished as the agent is punished for taking threat from the enemy. TUT looks for an enemy that is not targeting you already and taunts it.

The healing abilities will not attempt to heal a target if no valid target exists, for example if all allies are at full health then HLA will not heal anybody. Agents are only rewarded for their own health going up, and HLA and HTA will not heal themselves if they are the most targeted or lowest health, this mechanic is to stop false positive associations with the abilities as a means of the agent healing itself. The variables being fed into the neural network are both simple enough to not require too many nodes on the input layer, but complex enough to accurately represent the state. It does not include variables such as individual enemy health as there could be a variable number of enemies in the scene.

1. Agent health percentage
2. Num enemies targeting agent

3. Num Enemies total
4. Enemy lowest health
5. Enemy average health
6. Number of allies
7. Proximity to exit
8. Ally lowest health
9. Ally average health

Each of the inputs are relatively self explanatory, they were chosen to represent the scene without bias towards teamwork, with values such as how many enemies are targeting allies being omitted. The reward function being based on the difference between two states is defined as such. The agent is punished for loosing health and rewarded for gaining health, increasing in intensity the lower health the agent is on. The agent is punished slightly for doing actions that would lead to more enemies or him being targeted by enemies more. However it is given a large reward for killing an enemy however no reward for reducing the total enemy health. A significant reward is giving for getting to the last state, as well as a page punishment for taking agro from enemies when below 50 % health.

6 Testing

Varying the strength and number of enemies during the dungeon will change the difficulty and so require different tactics to be developed, as such these are the variables I have chosen to change through tests. The variables that represent success will be initially if they can complete the challenge at all, after that the degrees of success can be represented by how many times the AI died finding the tactic and how quickly (in how many rounds) they can defeat the dungeon.

The stats for heros is as such.

Agent	Damage	Health	Healing power
Deeps	7	12	3
Hots	2	10	7
Tank	4	20	3

With enemy values ranging from 3 to 5 damage, health ranging from 9 to 14 to numbers ranging from 2 to 3. An extra novel tests changes the enemy to a "boss" enemy with very high health and damage, simulating the encounters that led to the holy trinity stratergy in the first place. The boss has 8 attack, 30 health and will only spawn one per room.

Each test has no limit on the number of round that it must be finished in, but the agents must complete 60 floors (or 60 move room actions) to complete the dungeon, thus spawning 60 * number of enemies per room enemies. It is not required that they kill these enemies, but they will be quickly overwhelmed if they choose to run to the exit.

7 Results

Damage/Health/Number	Average resets to completion
1/1/2	0
3/9/2	11.6
4/9/2	40.6
5/9/2	91.4
3/10/2	6.4
4/10/2	82.7
5/10/2	148.5
3/9/3	71.6
8/30/1	4306.2 (50% infinity)

To clarify the last result some sessions could not finish the boss after 10,000 attempts and so where halted.

8 Evaluation

It is visible in the results that an increase in difficulty leads to a harsher environment where it takes longer to adapt a plausible strategy to survive. Seemingly an increase in damage leads to more resets then an increase in health, this can be seen as the average attempts more then doubles for damage.

An anomaly in the results is that 3/10/2 is seemingly easier to defeat then 3/9/2. It is difficult to see why as the increase in health should not make major changes to their strategy but I can theorise that at 9 heath one attack from both the Deeps and Hots would successfully kill a single enemy, possibly causing them to reach a local maxima with one of the attack commands, and leaving the Tank to heal and move rooms, which leaves more space open for mistakes as the tanks healing is not very effective.

The boss proved to the most difficult, with 50% of the runs not being completed by 10,000 attempts. It is likely that there is a inferior strategy that reaches a local maxima but is unable to constantly beat the boss.

On the easier enemies from values ranging from 3-4 attack the most common behaviour is that both the Tank and Deeps agents do very little but attack the enemies regardless of if enemies are present or not, while Hots varies between taunting enemies, healing itself or the lowest ally and moving rooms. Hots takes a utility role while the others simply focus on killing the enemies as fast as possible. This is a viable tactic as the enemies damage at this point is not adequate to overwhelm the healers healing ability, further more with two 3 attack enemies it even leaves the healer a further action to take before having to heal which allows him to taunt enemies. It is assumed that the healer is chosen for this role since he has the lowest damage, and stands to gain the most reward from healing.

It is very common that they discover it best not to spawn more enemies when there are still enemies in the encounter, as well as healing up to full before moving room again. This is somewhat surprising as commonly Hots will heal his allies up to full during this rest period, which he is in no way rewarded for as the agents are only rewarded for their own health increasing.

During the boss however the winning strategies is very close to the Holy Trinity developed by players, the Tank chooses to taunt the boss within 2 turns of it spawning and the healer chooses to heal the most threatened target during that time. This behaviour allows them to consistently beat the boss.

It is interesting to note that while the Deeps very commonly decides to only using attacking moves it very commonly settles on ALE as its preferred method. The obvious answer is that ALE is more likely to give them the kill enemy reward however it also mimics a strategy created by players where they prefer smaller damage on single targets then more damage on every target as it will reduce the number of attackers.

Another interesting feature is that taunting targets, while directly punished by the reward system is still used by the agents, most commonly by Hots and Tank.

9 Conclusion

In conclusion it seems that even with selfish focused rewards Q-Learning can produce viable team strategies, to the point where they replicate well known and used methods developed by games designers and players. This points towards such methods being able to produce unique and novel tactics in more complex scenarios. It is fascinating that taunting was chosen by the AI, my initial expectations was that it might not be chosen at all since the reward system directly punishes the agent for having enemies targeting it. However during the boss the taunts were used effectively to absorb its large damage with the equally large Tank's health pool. The technique is not very efficient however, with what could be considered very simple mathematical puzzles being solved after hundreds and thousands of attempts. But this considered it is because these AI have been give no knowledge about what actions are good to perform, and furthermore the usefulness of their actions are dependent on the actions chosen by their allies, of which they have no way of predicting. This generative behaviour is done without communication, nor consideration of other players, simply observing behaviours. The AI works on trust, the Tank trusts that it will be healed by Hots, and because of that it decides that taunting enemies, regardless of the direct punishment it receives for doing so, is the best action. Moving forward increasing the complexity of the game may unveil either the limitations of this technique or open up further opportunities for the agents to develop creative strategies to deal with the complexity.

10 References

Anonymous, N/A, Neural Networks in SALSA. Available at: http://www.cs.indiana.edu/ gasser/Salsa/nn.html

Dini S and Serrano M, 2012. Combining Q-Learning with Artificial Neural Networks in an Adaptive Light Seeking Robot.
Available at: http://web.cs.swarthmore.edu/ meeden/cs81/s12/papers/MarkStevePaper.pdf

Mc Cullock, J, 2012, Q-Learning. Available at: http://mnemstudio.org/path-finding-q-learning-tutorial.htm