Wolf Sheep Game

1 Part 1

Using MCTS
The main part of this part is MCTS:
In general the MCTS has 4 main steps:
Selection
Find the leaf node from the root node
Expand
If the leaf node is not terminal node, add one or more child node
Simulation
This process is to estimate the value of the child node have just added from the expand process.
To have some approximations, place some game and compute the win loose and other statistic values
Back Propagation
The node have some estimations after simulation process, now the information is back propagate to parent and the node above.

We can write simple MCTS like this

Main loop:

Play the game
Init game
Loop until the end game:
    Base on which player turn the algorithm choose the best action using MCTS
    Take the action(move)
    If the game end
        Decide who is winner
    From new node make a full game play and get result

MCTS:

Root node as current game state
Find leaf node
If leaf not is not a terminal node:
Expend new node
Now all child nodes of the root node are candidates for select as the best move base on number of UBC
c.wins/c.visits + UCTK * sqrt(2*log(self.visits)/c.visits

This UBC function is the method for solving the exploitation and exploration problem in MCTS

Self play game

```
you want to play auto? (y,n)y

    0   |  _     S     _     S     _     S     _     S
    1   |  _     _     _     _     _     _     _     _
    2   |  _     _     _     _     _     _     _     _
    3   |  _     _     _     _     _     _     _     _
    4   |  _     _     _     _     _     _     _     _
    5   |  _     _     _     _     _     _     _     _
    6   |  _     _     _     _     _     _     _     _
    7   |  _     _     _     _     W     _     _     _
------------------------------
```

Internal state:

```
Best Move: index:0 _x:1 _y:0 x:0 y:1 v:2


    0   |  _     _     _     _     _     _     _     S
    1   |  S     _     _     _     S     _     _     _
    2   |  _     _     _     S     _     _     _     _
    3   |  _     _     _     _     _     _     _     _
    4   |  _     _     _     _     _     _     _     _
    5   |  _     _     _     _     W     _     _     _
    6   |  _     _     _     _     _     _     _     _
    7   |  _     _     _     _     _     _     _     _
------------------------------
```

If set wolf smarter than sheep (wolf: 10000 steps, sheep: 100 steps) after 13 steps the wolf win the game

Player and computer:

```
Best Move: index:-1 _x:4 _y:7 x:5 y:6 v:1


    0   |  _     S     _     S     _     S     _     S
    1   |  _     _     _     _     _     _     _     _
    2   |  _     _     _     _     _     _     _     _
    3   |  _     _     _     _     _     _     _     _
    4   |  _     _     _     _     _     _     _     _
    5   |  _     _     _     _     _     _     _     _
    6   |  _     _     _     _     _     W     _     _
    7   |  _     _     _     _     _     _     _     _
------------------------------
TURN 2
1 : 0
3 : 0
5 : 0
7 : 0
Your move: 1,0,2,1
Best Move: index:0 _x:1 _y:0 x:2 y:1 v:2


    0   |  _     _     _     S     _     S     _     S
    1   |  _     _     S     _     _     _     _     _
    2   |  _     _     _     _     _     _     _     _
    3   |  _     _     _     _     _     _     _     _
    4   |  _     _     _     _     _     _     _     _
    5   |  _     _     _     _     _     _     _     _
    6   |  _     _     _     _     _     W     _     _
    7   |  _     _     _     _     _     _     _     _
```

The input move is old position of sheep and new position of sheep (target position)
For example, the image above is to move the first sheep from position (1,0) to (2,1)
So player input: 1,0,2,1

2 Part 2

This part is a the simple version of neutral network for estimate the value function. In other game like AlphaZero there are a neutral network that predict the value function (how good player in a state) and estimate the next move from the state that play are current at.
The prediction is use by MCTS method to calculate the value and policy. If we see the terminal state, we don't need the network to predict any more, just use that value because it is true value

This part use value function approximation by linear combination of features
Assume we have feature x = [x1,x2,x3,x4]
The weight will be W =[w1,w2,w3,w4]
The V wis calculated:
V = transpose(W).X
Feature selection:
For Wolf
+The first feature is how long the wolf to goal
wolf_to_goal = abs(max_y_sheep - wolf.pos_y): find the distance: how many cells that the wolf have to go to win the game (above all the sheep)
the first feature is 8-wolf_to_goal the smaller the better
there are another 4 features are the distance from the
for s in sheeps:
    state.append(min(abs(s.pos_x - wolf.pos_x),abs(s.pos_y - wolf.pos_y)))
For Sheep we choose the opposite

state.append(wolf_to_goal)
    for s in sheeps:
        state.append(8-min(abs(s.pos_x - wolf.pos_x),abs(s.pos_y - wolf.pos_y)))

The weights are the linear combination of the features
np.dot(np.array(self.theta) , np.array(state))

with theta is defined:
theta = [1.0,0.2,0.2,0.2,0.2]
the reason to choose the first weight is the biggest because the distance of wolf to goal is the most important

The second part of part 2 is to estimate the value V by modifying the parameters W

Auto play for Value function estimation and TD are the same as part 1
The expected result is near the part 1 but it can not be as part 1 because it need more step for wolf to win the game (20-30 steps). The reason is that the feature selection need to be improve as well as the hyper parameter must be optimized.

3 Part 3

This part is for to play the game with all the algorithm develop so far
I define 3 algorithms
player_pair = [('mcts','mcts_v'),('mcts','mcts_td'),('mcts_td','mcts_v')]

'mcts' : MCTS algorithm
‘'mcts_v’ estimated valie function as linear of selected feature
‘'mcts_td’ use TD for improving the value function estimation
The config above there are 3 match each rounds, the number of rounds is set : the bigger the better

4 Conclustion:

I learnt a lot from working on this project. The MCTS is a powerful algorithm for mastering game. This method have an advantage of not searching all the nodes of tree, it focus on the better promising node that seem good for furture.
When the number of simulation big enough the estimation become more stable and make the value function more accuracy. It mean that the value function give estimate for a node (state of game), at the beginning it is

bad estimation, when the MCTS make more simulation the real outcome is backpropagate though the tree and make the value function change according to the error with the real estimation. The result is that the weights is changes using gradient decent.
The advantage is that to test these agents that take a lot of time for waiting the test to be completed. We can’t set small simulation because it will not good for estimation the outcome so the value is not stable when set small simulation. The game seem bias on wolf because the wolf is easy to win then sheep win. To make sheep win we have to set very high simulation for sheep and low down the simulation for wolf