Assignment 3: Fog Of War

Nicolas Gundersen neg62, Maudiel Romero mar641, Ray Caringal rrc85

Abstract - Project 3 expands upon our last assignment, in which we created an Adversarial Agent on a grid game board using minimax with beta pruning. Project 3 replaces the minimax algorithm to one that relies on observations based upon an implemented Fog Of War, which restricts the players vision on the board to a limited stope around each piece.

I. Introduction

Our game takes place in a dxd game board based on user inputs of either 3,6, or 9 with Wumpuses, Mages, Heroes, and Pits as per Project 2 instructions. The fog of war will black out all of the enemy pieces so that we only see one set of pieces. Both players are unaware of the location of each other's pieces, including the pits. We instead rely on observations that we make after each player moves If there is a Wumpus in an adjacent cell, we will receive an observation as a Stench, Similarly, heat for Mages, movement for Heroes, and a breeze for pits. These observations will update the screen to let the player aware of potential adversaries near them. For such a task, the adversarial agent needs to store a probabilistic distribution that it updates after each observation, we make a Markov assumption that assumes that the state of the board at each turn depends on the state of the board in the previous turn. To start, the agent will make random moves, and each cell will hold a list of values that correspond to [P(Pxy), P(Wxy), P(Hxy), P(Mxy)]. After a player moves, observations are made and the probabilities are updated according. Then, the rest of the board is normalized and the best move is chosen.

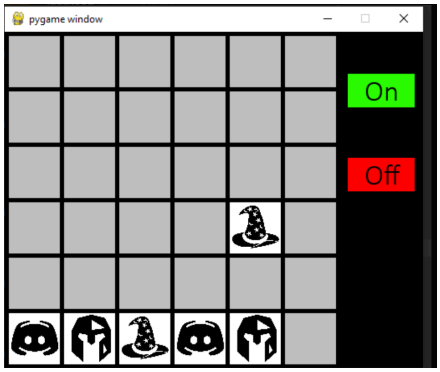
II. Probabilistic Distribution

Instead of our Adversarial Agent relying on MiniMax, our agent uses a probabilistic distribution. For each cell in our grid, we hold a list that contains the Probabilistic values of a nearby cell being a Pit, enemy wumpus, enemy mage, or enemy hero. Initially, we have our adversary select a random move, after the player moves, observations are made. In order to find the new P'(Wxy), P'(Hxy), P'(Mxy), we use the following formula:

$$P'(W_{x,y}) = (1-1/c)*P(W_{x,y}) + \Sigma_{(x',y)\in(\text{neighbors}(x,y)} P(W_{x',y'}) * P(W_{x,y}|W_{x',y'})$$

Equation 1: This is our probability function we use to create our probabilistic function.

where neighbors(x,y) are the cells within one move away and $P(W_{x,y}|W_{x',y'}) = 1/(c*\text{neighbors}(x',y'))$ We achieved this by building a grid of probabilities on start up, based on where we know our starting pieces are. From there we run P_Wumpus(X,Y), P_Mage(X,Y), P_Hero(X,Y) on every cell in the grid. These functions hold the first part of eq1, into which we then call Neighbors(X,Y). In the neighbors function we look at each adjacent cell at that location, if that location exists, we take the corresponding P(Wxy) value(or respective piece), and multiple that value by equation 2. We add those all up and thats our new P'(Wxy) value. From there, we await moves and make observations, at the start of the agents next turn, we update our probabilistic distribution to reflect those values using Baye's Theorem. Since the pits never move, the distribution of the pits will be the same before and after the opponents



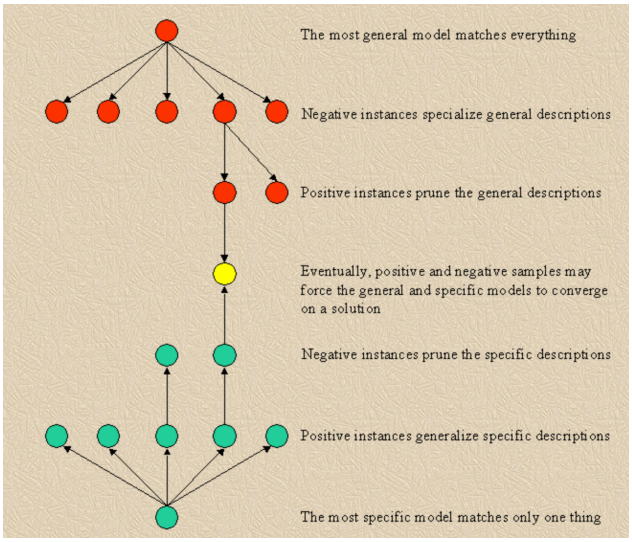
Picture 1 - our game board with fog of war

III. Selecting a good move

Now that we have an adequate probabilistic distribution, we can try to select a good move. Initially, the Adversarial Agent starts off by selecting a random move. After this, the probabilistic distribution starts getting updated and we can make better inferences on the pieces around us. In any of our adjacent cells, if we notice that there's a probability of at least 50%, we can infer that there is a piece in that cell. Our idea of a good move consists of the correct piece attacking and destroying the enemy piece, and if we can't destroy it we will try to tie, otherwise - retreat. To do this - we simply use a method goodMove() in which we take a location on the grid and our probabilistic distribution, and decide which of our probabilities is highest and most surely to give us a 'good move'. If we can be confident that the move will be good, we execute that move and then continue to update our probabilistic distribution. If we're not confident about any of our probabilities or they're inconclusive, a random move will be made.

IV. Alternative to Random

Instead of having our adversarial agent picks moves randomly, which might not be fun for a user, we could make our agent smarter by implementing a version space learning algorithm. This will allow the adversarial agent to make more sensible actions when there isn't enough criteria to make a 'good move'. What the version space learning algorithm does is represent our moves with two sets of hypotheses, one of which is a more specific set of hypotheses while the other is a more general set. This algorithm takes a set of all Hypotheses and returns a subset that are not empty or within a certain margin. Since the algorithm in incremental, backtracking is not necessary. When version space is called, taking the matrix as the example space, it sets V to the set of all hypotheses, checks to make sure V is not empty, then updates the set using Version_Space_Update - taking in V and the set of examples as as space. V will then be compared to the generalized set and specialized set and return a set of the best hypothesis. For our grid world, we can think of every possible move that an agent could pick as the generalized set of hypotheses.



Picture 2: Visualized version space learning algorithm

V. Conclusion

To sum up our project, in order for us to implement a FogOfWar to our grid game requires stripping minimax from our adversarial search. This added an entirely new set of difficulties that we had to overcome. Our new algorithm requires a probability distribution that is updated after every turn, the distribution is in the form of a 2d-array, where the bulk of our math is done with the equations in fig1 and 2, as well as bases based on observations. This gives us a good basis for what we can consider a good move. We look over all of the probabilistic values for each piece, in every cell, and from there we decide which probabilities are worth looking at. This gives us the illusion that the adversarial agent Is as informed as the player, and reacts to observations seemingly as well as human player would.