Marmik Pandya

Amy Hoover

CS 5100

20 April 2016

Pacman vs Ghosts Framework: Genetic Decision Tree

Problem Statement:

Creating an AI controller for the ghosts/pacman using the MS Pacman v/s Ghosts framework.

Algorithms Implemented:

Blind Search:

I implemented the following Blind Search Algorithms for Pacman for the first submission for the final project:

1. Breadth First Search
2. Depth First Search
3. Iterative Deepening Depth First Search

Implementation:

Tree.java: The tree that will be created based on the game state, for the Pacman/ Ghost to search through for the best move using either of the algorithms can be found in "MoveController" package. The formation of tree takes $O(n^2)$ complexity which is quite an overhead and needs to be worked on so that the deeper trees can be constructed faster – which will make the searching better.

Evaluation.java: This class is the core of the entire logic. This file controls the various constants for the tree generation, includes a method for evaluating every node (possible game state) and also it contains the logic for the next best move. The heuristics to determine the best moves is determined by Boolean variable COMPLETE_LEVEL. If the variable is set true, the Pacman goes for LEVEL COMPLETION by following pills rather than following the edible ghosts. If the Ghost is 20 or less than 20 units nearer to the pacman, it changes it's course of direction to move away from ghost. If the edible ghost is 100 or less units nearer to the pacman it goes nearer to the ghost to eat it (only if COMPLETE_LEVEL is set false).

Algorithms:

BFS: Uses Breadth First Search to traverse through the tree. It gets the best move by calling getBestMove() function from Evaluation class.

DFS: Uses Depth First Search to traverse through the tree. It gets the best move by calling getBestMove() function from Evaluation class.

Iterative Deepening: Uses Iterative Deepening Depth First Search to traverse through the tree. It gets the best move by calling getBestMove() function from Evaluation class.

Comparison and Limitations:

Implementation Comparison and Limitations: Currently amongst all these algorithms Iterative Deepening performs more optimally in both the scenario where the heuristics is focused on collecting points. Iterative Deepening performs well since, due to balanced depth of tree time isn't much of the overhead at the moment and also space isn't the overhead for pacman to find the best possible moves. Since, none of the algorithms have been able to complete the level of of yet no comments can be made in that scenario as of now. Moreover, this is just based on few test runs of mine and in future proper data modelling and machine learning algorithms can be applied to find the most optimal solution.

Theoretical Performance Comparison:

Evaluation	Breadth First	Depth First	Iterative Deepening
Time	B^D	B^M	B^D
Space	B^D	BM	BD
Optimal?	Yes	No	Yes
Complete?	Yes	No	Yes

Where B is Branching Factor, M is the Maximum Depth and D is the Depth of the solution.

Informed Search:

I implemented the following Informed Search Algorithms for Pacman for the first submission for the final project:

Hill Climbing: Uses greedy approach to find the best value at given point and then it goes to that node until it reaches Maxima.

Limitations: Can get struck in Local Maxima.

Adversarial Search:

I implemented the following Adversarial Search Algorithm for Pacman for the first submission for the final project:

Alpha-Beta: Uses the Minimax strategy along with pruning the unwanted branches to save traversal time. It decides the best move based on either of the two Heuristics (Complete Level or Get Maximum Points)

Evolutionary Algorithms:

I implemented the following Informed Search Algorithms for Pacman for the first submission for the final project:

Genetic Algorithm: Gets the best generation by performing crossover on two different chromosomes and gets the fitness of a particular evolution using either of the Heuristic (Maximum increase in points or Maximum Pills Swallowed)

Supervised Learning:

I implemented the following Informed Search Algorithms for Pacman for the first submission for the final project:

1. K Nearest Neighbor
2. Decision Tree

Decision Tree: Training data was already recorded and stored as a text file. Going through the data, different characteristics were used to create the decision tree: the total number of moves Pac-Man can make (left, right, up, and down), what kind of moves Pac-Man can make, and the move of the closest ghost decision was made.

K Nearest Neighbor: Using the training data, the instances where the difference between the distance between pacman and the average distance from all of the ghosts in the replay are the closest to 0 are used. The k instances are then observed to see if Pac-Man took the behavior of running away from ghosts, or going to take the nearest pill. The behavior chosen the most is then chosen as the optimal behavior for Pac-Man.

Custom Algorithm – Genetic Decision Tree

Implementation: This is a very basic decision tree, but its parameters -distance to Edible ghosts, dangerous ghosts and power pills have been evolved with a genetic algorithm.

Basically, for all members of the population, runs a heuristic (which is average of the score in 10 trials for every generation) that evaluates their fitness – Average Score in 10 trials, based on their phenotype. The evaluation of this problem's phenotype is fairly simple,and can be done in a straightforward manner. In other cases, such as agent behavior, the phenotype may need to be used in a full simulation before getting evaluated (e.g based on its performance)

The best evolution of distance to Edible ghosts, dangerous ghosts and power pills is then feed to the Decision Tree to make the decision for the best move.

Analysis and Comparison:

Breadth First			Depth First			Iterative Depth First			Hill Climber		
Trials	Score	Level	Trials	Score	Level	Trials	Score	Level	Trials	Score	Level
1	20560	2	1	32300	3	1	28270	3	1	5840	2
2	17540	2	2	5950	1	2	23330	2	2	6970	1
3	25540	2	3	9090	1	3	17050	2	3	7640	2
4	35980	4	4	47900	4	4	11060	1	4	5480	2
5	10200	1	5	19710	2	5	27970	3	5	3500	1
6	31030	3	6	32410	3	6	28860	2	6	5450	1
7	14890	2	7	19360	2	7	10160	1	7	8480	2
8	23200	2	8	11860	1	8	17570	2	8	4610	1
9	48940	5	9	12570	1	9	29870	3	9	1350	1
10	29920	3	10	22380	2	10	29700	4	10	4480	1
11	28790	2	11	24740	2	11	18000	2	11	6040	2
12	18100	2	12	27360	2	12	18950	2	12	1350	1
13	31520	3	13	37420	3	13	35300	4	13	6700	1
14	40900	4	14	60620	5	14	7610	1	14	7240	2
15	8240	1	15	7210	1	15	18080	2	15	2790	1
16	23250	2	16	9050	1	16	25520	2	16	3850	1
17	19960	2	17	45050	4	17	27080	3	17	4610	1
18	18420	2	18	38640	3	18	19950	2	18	4610	1
19	18580	2	19	20050	2	19	36380	4	19	7040	2
20	29700	4	20	25590	2	20	39300	4	20	2090	1
Average	24763	2.5	Average	25463	2.25	Average	23500.5	2.45	Average	5006	1.35

Genetic			Alpha-Beta			kNN		
Trials	Score	Level	Trials	Score	Level	Trials	Score	Level
1	280	1	1	4080	1	1	3860	1
2	400	1	2	6200	1	2	3510	1
3	290	1	3	2660	1	3	3860	1
4	400	1	4	5970	1	4	3910	1
5	200	1	5	4670	1	5	3860	1
6	200	1	6	3780	1	6	3600	1
7	200	1	7	5690	1	7	3460	1
8	200	1	8	6340	2	8	3860	1
9	290	1	9	2420	1	9	4070	1
10	280	1	10	20980	3	10	3860	1
11	580	1	11	6590	1	11	2670	1
12	280	1	12	2430	1	12	3860	1
13	1060	1	13	7550	2	13	3860	1
14	200	1	14	3290	1	14	3800	1
15	200	1	15	5940	1	15	3860	1
16	380	1	16	8190	1	16	3860	1
17	200	1	17	4670	1	17	2680	1
18	200	1	18	28110	4	18	4590	1
19	200	1	19	12760	2	19	5480	1
20	200	1	20	11260	3	20	3600	1
Average	312	1	Average	7679	1.5	Average	3805.5	1

ID3 Decision Tree				Genetic Decision Tree		
Trials	Score	Level		Trials	Score	Level
1	220	1		1	3060	1
2	220	1		2	4080	1
3	220	1		3	8230	1
4	220	1		4	5640	1
5	240	1		5	8500	1
6	220	1		6	1720	1
7	250	1		7	3950	1
8	220	1		8	3320	1
9	220	1		9	4210	1
10	240	1		10	2360	1
11	220	1		11	2290	1
12	220	1		12	1990	1
13	220	1		13	3570	1
14	220	1		14	5680	1
15	220	1		15	1690	1
16	220	1		16	3550	1
17	250	1		17	1490	1
18	260	1		18	4750	1
19	220	1		19	7020	1
20	240	1		20	510	1
Average	228	1		Average	3880.5	1

Conclusion:

Although the Custom Decision Tree was much simpler than the ID3 algorithm, using genetic algorithm to evolve the decision making parameters made the decision helped make a much better decision. Although after 20 Trials BFS, DFS and IDDFS seem to outperform Genetic Decision Tree, using much broader parameters and evolving it to more than 50 generation.