Machine, Date & Learning

Value-Iteration

Team 25 | Tribrid

- Nitin Chandak (2019101024)
- Ayush Sharma (2019101004)

Task - 1

NOTE: The trace of the state, the chosen action at that state and the value of state for all iterations of the `VI` algorithm from 0 till convergence has been logged in `outputs/part_2_trace.txt` in following format.

```
iteration=0
(W,0,0,D,0):RIGHT=[12.231]
(W,0,0,D,25):RIGHT=[6.000]
.
.
.
(C,2,3,R,100):SHOOT=[1.232]
iteration=1
(W,0,0,D,0):STAY=[8.999]
.
.
.
.(so on until the end of iterations)
```

Interpretation, Comments & Analysis

Some statistics on the trace obtained are:- * Iterations: It took 118 iterations to reach convergence. * Gamma : 0.999 * Delta : 0.001 * Step Cost : -10

Analysis from `trace`:-

- Changing values of `gamma` leads to difference in gains. Low value lead to short-term gain policy and larger value to long-term gain policy. Asymptotically, if `gamma ~ 1`, then we'll be getting policies that optimizes gain over infinite time. On the other hand, value iteration will be slower to converge.
- The tendency of IJ to `SHOOT` MM is very low, when IJ is in square `C`. Actually, there were only `4` instances where action for a given state in policy is `SHOOT`. For this , reason can be low success rate i.e. low probability to reach terminal state.
- We found that for position 'C' , IJ never took action `STAY`. This can be reasoned as to prevent the negetive reward of `-40` when MM is in `Ready` state and attacks, also less probability of damage happen due to action either `SHOOT` or `HIT`.
- We found when in position `E`, IJ always attempt to shoot the MM as there is less probability of MM transitions from `D` state `->` `R` state and opposite. i.e. `0.2` & `0.5` respectively.
- Total count of `SHOOT` action is more than `HIT` action. This can be because of higher probability of Damage using arrow than blade.
- We can notice our model i.e. `IJ` to be risk-seeking because when reaching square `E`, IJ always try to `HIT` or `SHOOT` instead of fact whether MM is in `R` state or `D` state.
- We found following count of actions for every non-terminal state classified by positions of IJ.

```
Place =  C
count of  UP  =  27
count of  Left  =  0
count of  RIGHT  =  43
count of  DOWN  =  15
count of  LEFT  =  6
count of  HIT  =  0
count of  SHOOT  =  5
count of  STAY  =  0
count of  CRAFT  =  0
count of  GATHER  =  0
Place =  E
count of  UP  =  0
count of  Left  =  0
count of  RIGHT  =  0
count of  DOWN  =  0
count of  LEFT  =  0
count of  HIT  =  39
count of  SHOOT  =  57
count of  STAY  =  0
count of  GATHER  =  0
count of  CRAFT  =  0
Place =  W
count of  UP  =  0
count of  Left  =  0
count of  RIGHT  =  59
count of  DOWN  =  0
count of  LEFT  =  0
count of  HIT  =  0
count of  SHOOT  =  24
count of  STAY  =  13
count of  GATHER  =  0
count of  CRAFT  =  0
Place =  S
count of  UP  =  54
count of  Left  =  0
count of  RIGHT  =  0
count of  DOWN  =  0
count of  LEFT  =  0
count of  HIT  =  0
count of  SHOOT  =  0
count of  STAY  =  32
count of  GATHER  =  10
count of  CRAFT  =  0
Place =  N
count of  UP  =  0
count of  Left  =  0
count of  RIGHT  =  0
count of  DOWN  =  34
count of  LEFT  =  0
count of  HIT  =  0
count of  SHOOT  =  0
count of  STAY  =  24
count of  GATHER  =  0
count of  CRAFT  =  38
```

Simulations

We have simulate the game for obtained `utility` and `policy` on two start states: * (W, 0, 0, D, 100) * (C, 2, 0, R, 100)

The order of the actions, the state transitions and comment on the results for above start states: We ran simulations `10` times for both start state and found that every time it reaches the terminal state i.e. state with last value `0` like `('S', 0, 2, 'R', 0)` and `('E', 0, 0, 'R', 0)`. One of the output is shown below for each start state. One can find all the obtained outputs on running simulation in `outputs/simulations/2_1.txt` file.

(W, 0, 0, D, 100)

```
current_state = ('W', 0, 0, 'D', 100) action_taken = RIGHT reached_state = ('C', 0, 0, 'R', 100)
current_state = ('C', 0, 0, 'R', 100) action_taken = RIGHT reached_state = ('C', 0, 0, 'D', 100)
current_state = ('C', 0, 0, 'D', 100) action_taken = RIGHT reached_state = ('E', 0, 0, 'D', 100)
current_state = ('E', 0, 0, 'D', 100) action_taken = HIT reached_state = ('E', 0, 0, 'D', 100)
current_state = ('E', 0, 0, 'D', 100) action_taken = HIT reached_state = ('E', 0, 0, 'R', 100)
current_state = ('E', 0, 0, 'R', 100) action_taken = HIT reached_state = ('E', 0, 0, 'R', 50)
current_state = ('E', 0, 0, 'R', 50) action_taken = HIT reached_state = ('E', 0, 0, 'D', 75)
current_state = ('E', 0, 0, 'D', 75) action_taken = HIT reached_state = ('E', 0, 0, 'R', 25)
current_state = ('E', 0, 0, 'R', 25) action_taken = HIT reached_state = ('E', 0, 0, 'D', 50)
current_state = ('E', 0, 0, 'D', 50) action_taken = HIT reached_state = ('E', 0, 0, 'D', 50)
current_state = ('E', 0, 0, 'D', 50) action_taken = HIT reached_state = ('E', 0, 0, 'R', 50)
current_state = ('E', 0, 0, 'R', 50) action_taken = HIT reached_state = ('E', 0, 0, 'D', 75)
current_state = ('E', 0, 0, 'D', 75) action_taken = HIT reached_state = ('E', 0, 0, 'D', 75)
current_state = ('E', 0, 0, 'D', 75) action_taken = HIT reached_state = ('E', 0, 0, 'D', 75)
current_state = ('E', 0, 0, 'D', 75) action_taken = HIT reached_state = ('E', 0, 0, 'R', 25)
current_state = ('E', 0, 0, 'R', 25) action_taken = HIT reached_state = ('E', 0, 0, 'R', 0)
```

(C, 2, 0, R, 100)

```
current_state = ('C', 2, 0, 'R', 100) action_taken = UP reached_state = ('N', 2, 0, 'R', 100)
current_state = ('N', 2, 0, 'R', 100) action_taken = CRAFT reached_state = ('N', 1, 3, 'D', 100)
current_state = ('N', 1, 3, 'D', 100) action_taken = DOWN reached_state = ('C', 1, 3, 'D', 100)
current_state = ('C', 1, 3, 'D', 100) action_taken = RIGHT reached_state = ('E', 1, 3, 'R', 100)
current_state = ('E', 1, 3, 'R', 100) action_taken = HIT reached_state = ('E', 1, 0, 'D', 100)
current_state = ('E', 1, 0, 'D', 100) action_taken = HIT reached_state = ('E', 1, 0, 'D', 100)
current_state = ('E', 1, 0, 'D', 100) action_taken = HIT reached_state = ('E', 1, 0, 'D', 100)
current_state = ('E', 1, 0, 'D', 100) action_taken = HIT reached_state = ('E', 1, 0, 'D', 100)
current_state = ('E', 1, 0, 'D', 100) action_taken = HIT reached_state = ('E', 1, 0, 'D', 50)
current_state = ('E', 1, 0, 'D', 50) action_taken = HIT reached_state = ('E', 1, 0, 'D', 50)
current_state = ('E', 1, 0, 'D', 50) action_taken = HIT reached_state = ('E', 1, 0, 'D', 0)
```

Task - 2

Indiana now on the LEFT action at East Square will go to the West Square.

Only one thing was obserably changed was that now IJ also possess policy for havng action `LEFT` for some states mainly states having MM in state `R`. Apart from this there wassn't much change by having above modification in the code. Also, the number of iterations required to converge for the algorithm is approximately same i.e. 120.

The step cost of the STAY action is now zero.

The outcome of the algorithm i.e. trace here significantly changes. * Total number of iterations required to converge decreased from `118` to `80`. * Count of action `LEFT` in position `E` increased much. * The count of action `STAY` abruptly increased in IJ's position of `W`. Because the reward is maximized by just remaining the same place and not doing anything. All other action are having `STEPCOST` negetive, except when IJ would attack and MM dies. However, the probability of that happening is so low that staying still remains beneficial.The stay action is highy preferred

Change the value of gamma to 0.25

The number of iterations for convergence of algorithm drastically decreased from `118` to `8`.