HW6_kb42582

Khyathi Balusu(kb42582)

5/8/2020

Question 1,2

States = selfScore , oppScore , curTimeValue
#s = self score; p = opponent's score, x = current time value

Action = Roll or Hold

Bellman equation:

V(selfScore,100,curTimeValue) = 0; V(100,oppScore,curTimeValue) = 1; V(99,oppScore,curTimeValue) = 1;

V(selfScore,oppScore,curTimeValue) = max[p1(1-V(oppScore,selfScore+1,0)) + p2 V(selfScore,oppScore,curTimeValue+z) + p3 V(selfScore,oppScore,curTimeValue+3) + p4 V(selfScore,oppScore,curTimeValue+4) + p5 V(selfScore,oppScore,curTimeValue+5) + p6 V(selfScore,oppScore,curTimeValue+6) , (1- V(oppScore,selfScore+curTimeValu

```
       where Pn is probability of rolling n
```

Question 3,4

```
gameStrategy <- function(p_goal){

  goal = p_goal+1
  all_comb = expand.grid(seq(goal-2),seq(goal-1))     # Generating all combinations of each index
  all_comb$sum = all_comb[[1]] + all_comb[[2]]        # Finding max sum
  all_comb = all_comb[order(all_comb$sum, decreasing = TRUE),]

  V = array(NA, dim=c(goal+5,goal+5,goal+5))   # selfScore,oppScore,curTimeValue
  U = array(NA, dim=c(goal+5,goal+5,goal+5))   # selfScore,oppScore,curTimeValue

  # Initialize boundaries
  V[goal:(goal+5),,] = 1                 # If you have 100+ points then you win irregardless of the opp
  V[goal-1,seq(1,goal-1),] = 1           # If it is your turn and you have goal-1 points, no matter wha
  V[seq(1,goal-1),goal:(goal+5),] = 0    # If the opponent reaches the goal first then you lose.
  V[,seq(1,goal-1),goal:(goal+5)] = 1    # If you accumulate the goal amount in one round you will win

  # Fill in remaining V and U matrix
  for(r in 1:length(all_comb$sum)){
      for(curTimeValue in (goal):1){
        selfScore = all_comb[r,1]
        oppScore = all_comb[r,2]
        V[selfScore,oppScore,curTimeValue] = max( ( (1/6)*(1-V[min(oppScore,goal),selfScore+1,1]) + (1/
                          + (1/6)*V[min(selfScore,goal),oppScore,min(curTimeValue+4,goal)] + (1/6)*V[mi
                          + (1/6)*V[min(selfScore,goal),oppScore,min(curTimeValue+6,goal)] ), 1-V[min(o
        U[selfScore,oppScore,curTimeValue] = which.max( c( ( (1/6)*(1-V[min(oppScore,goal),selfScore+1,
                          + (1/6)*V[min(selfScore,goal),oppScore,min(curTimeValue+4,goal)] + (1/6)*V[mi
                          + (1/6)*V[min(selfScore,goal),oppScore,min(curTimeValue+6,goal)] ), 1-V[min(o
      }
```

```
  }
  save(list = c('V','U'),file = 'VUfile.Rdata')

}
```

Bonus

We now can't calculate the V and U matrices because we now don't know the probability of a number appearing during a dice roll.

Strategy: We played the first round with the assumption of a fair dice and used the dice rolls from the first round to estimate the dice roll distribution. We then passed the estimated probabilities into our dynamic program to re-calculate the V and U matrices for the next round.

We provided the modified server.R file that, on clicking new game, will estimate the new distribution and then call our function to re-calculate the V and U matrices. We also changed the dice to be biased, with roll probabilities of (0.2, 0.1, 0.3, 0.1, 0.2, 0.1). Be aware, that you must set the working directory to the server.R file's location and play one full game with the assumption that the dice is fair. After you finish one game, clicking "new game" will re-estimate the distribution and the next game will be run with the updated V/U matrices. Every additional game you play will improve the estimation of the distribution.