MIS 381N

Stochastic Control and Optimization: Project 6

Please email your group to Vishwakant (vishwakant@gmail.com).

In this project we will build a computer program that computes the optimal strategy to play a simple game. We will then use the computed solution in an App to allow human players to play against the computed solution.

The object of the game is to be the first player to reach 100 points. Each player's turn consists of repeatedly rolling a die. After each roll, the player is faced with two choices: roll again, or hold (decline to roll again).

- If the player rolls a 1, the player scores 1 and it becomes the opponent's turn.
- If the player rolls a number other than 1, the number is added to the player's turn total, the sum of the rolls during the turn, and the player's turn continues.
- If the player holds, the turn total is added to the player's score, and it becomes the opponent's turn.

For such a simple dice game, one might expect a simple optimal strategy, such as in Blackjack (e.g., "stand on 17" under certain circumstances, etc.). As we shall see, this simple dice game yields a much more complex strategy.

For example a couple of turns can go like this:

- Player A's turn begins. A chooses to roll and rolls a 5. Turn total is 5. Then A chooses to roll again and gets a 3. Turn total is 8. A chooses to roll again and gets a 6. With the turn total at 14 now, player A decides to hold. Now the turn total 14 is added to player A's score and the game moves to B's turn.
- Player B's turn begins. B chooses to roll and rolls a 4. Then B chooses to roll again and gets a 2. Turn total is 6. Chooses to roll again and gets a 5. Turn total is 11. Chooses to roll again and gets 1. Now that B rolled a 1, the turn total is discarded and simply a '1' is added to player B's score and the game goes to A's turn again.

Specific steps

1. First make sure you understand the coinGameRec.R file from the class. You can find it on Canvas under Dynamic programming/Code. You can use this file as a starting point.

2. For this game, list the states, actions and objective. Using these write down the Bellman Equation.

3. Compute the value function V and the decision matrices U with given goal. The value of U is to be 1 to roll and 2 to hold. Name your R file as 'gameStrategy.R'. Define the function gameStrategy(goal) in this R file. This function takes the goal and simply saves the resulting V and U into a Rdata file.

4. You will notice that it will take a sometime to calculate for the goal=100. So run it once and save the resulting values of V and U into a file 'VUFile.Rdata'. You can do this using the command

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

5. An App called 'gameApp' is provided. Before you run this app, make sure you can run the coinGameApp from canvas. Then run the 'gameApp', it currently plays using a dumb "hold after one roll strategy". You can replace this strategy putting your VUfile.Rdata into the "gameApp" directory and replacing the one in there. See how well your computed solution plays against you or your friends.

6. Upload your solution to the "shinyapps.io" website that was shown in class. You will need to first create an account with the website and link RStudio to your account. You can directly submit your app to the website using the "deployApp()" command. *https://shiny.rstudio.com/articles/shinyapps.html* provides easy, step-by-step instructions to load you app onto the website. Feel free to share the app with your friends and see if your strategy generally beats theirs.

   Check https://kumarapps.shinyapps.io/coin/ for the example shown in class.

Deliverables

1. A brief write up containing your answers to step 2 above. Please make sure to include your group number and group member names at the top of the document.

2. Your gameStrategy.R file and your VUFile.Rdata

All of the above zipped together and named **project6_gZ.zip**.

Tips and Hints

a) The objective will be to win the game and hence the V function will represent the probability of winning the game.

b) We will need three state variables, so V(i,j,k) will respresent the probability of winning when the current state is: our accumulated score is 'i', opponent's score is 'j' and 'k' is the current roll total.

c) Remember that you can use array function to generate a 3-dimension matrix. For example,

V=array(NA,c(10,10,10))

will give a 10 by 10 by 10 3-D matrix. You can access the elements of the matrix using, for example V[5,5,5]. If your state is described by 3 variables, then your V and U matrices will be 3 dimensional.

d) The value of some state variables can be zero (like when you begin the game). So when storing in matrices like V and U, be careful since vector and matrix indices are numbered from 1 and not from zero. An easy way to not get confused is always to add 1 to the index. For example in the mining example from class, we had

V(s,t)=max (p*x – x.^2/(1+s) + 0.9*V(s-x,t+1))

as our Bellman equation. Since both s and t can take the value 0, we write in R

V[s**+1**,t**+1**]= max (p*x – x^2/(1+s) + 0.9*V[s-x**+1**,t+1**+1**])

All the bolded '+1' are to stay away from trying to store in the zero index and adjust accordingly. The values corresponding to s=0 are now stored at index 1.

e) The maximum of your score (any state variable) should be goal + 5. For example, if the goal is 100, you may arrive at 99 and then roll a 6.

f) **Notice that the sum of your score and your opponent's score is increasing during the game. Use the carefully to structure your loops so that the RHS of the bellman equation is always calculated before the LHS.**

g) If you decide to hold without rolling, 1 is added to your score and it will be your opponent's turn. This is important in defining holding value when you haven't rolled a dice yet.

Extra credit (5 points)

Now for the above computation you assumed that the dice was evenly weighted. That is, the probability of any number is just a 1/6. Now what if you don't know the weightage of the dice! Can you modify the code from above to play the game when you don't know the weightage of the die used?

First describe the challenges involved and how you'd solve the problem. Then try and modify the code that you used from earlier. Feel free to modify the calling function and the line that loads up the matrices on the GUI if needed.

This problem is quite involved and there is no best or right answer! Even a well laid out description of how you'd solve this problem will get you partial extra credit. A good working code will get you full extra credit.