Bob Lin

Math381

Homework5

The problem is to roll a dice(fair and six-sided), and we let score S = 0 be the initial score of our rolls. We will add or subtract number to this score until we get 20 or more. Here are the rules:

1. When the number we roll **can** divide the current score, we update score by **adding** that number to our current score.

2. When the number we roll **cannot** divide the current score, we update score by **subtracting** that number to our current score.

3. If the score becomes negative, we set it to zero.

In order to investigate the game above, I use 21x21 transition matrix, and first row and column is initial state, while the last row and column is absorbing state. By using the matrix(with row represents the current score from 0 to 20 and column represents the next step’s score from 0 to 20, and the last column represents all scores equal or greater than 20), I can see the probability that we get a score ‘b’ from a score ‘a’. Here is the code to generate the matrix in Matlab:

A = zeros(21);

for row = 1:21

for col = (row + 1):(row + 6)

if row == 21

A(row,21) = 1;

elseif mod(row - 1, col - row) == 0

if col < 21

A(row,col) = 1/6;

else

A(row,21) = A(row,21) + 1/6;

end

else

sub = 2\*row - col;

if sub > 0

A(row,sub) = 1/6;

else

A(row,1) = A(row,1) + 1/6;

end

end

end

end

With this matrix, I will further explore several properties about the game:

**First:**

I want to see the expected number of rolls for me to make the score at least 20.

The matrix transition A can be written in canonical form, which is:

A = )

Where Q is the matrix without the last row and last column of matrix A, J is the identity matrix with 1 in diagonal and 0 in elsewhere, and O is a matrix with all zeros.

By using the Q above, a theorem shows that gives important information of:

*The sum of the i-th row of N gives the mean number of steps until absorbtion when the chain is started in state i*.

So, The sum of first row of N is the expected number of steps until we are in absorbing state 20, then by using code:

Q = zeros(20)

for row = 1:20

for col = 1:20

Q(row,col) = A(row,col)

end

end

N = inv(eye(20) - Q);

sum = 0;

for col = 1:20

sum = sum + N(1,col)

end

we get that the sum equals to , so we need about 115.35 rolls on average in order to get a score equal or greater than 20

**Second:**

I want to see the probability for me to get a score at least 20 under different times limits:

Here, by definition, is the probability that we can get from state i to state j in k rolls. Thus, by setting i to 1, j to 21, we can get the probability to the goal state in k rolls, so by simply setting k to 50, 100, and 200,

This can be computed by following code:

|  |  |  |  |
| --- | --- | --- | --- |
| Times  - |  |  |  |
| Probability | 20 |  |  |

**Third:**

I want to see how many rolls will guarantee that there is at least 90 for me we get a score at least 20.