Homework 07 DTMC

Math 519

William Gillespie

Water Molecule Exercise

A molecule of water is hanging around the Great Lakes region,  
consisting of the 5 lakes (M,S,H,E,O), or the air above the region  
(which we'll call "A"), from which it might fall as rain or dew back  
into a lake. Suppose it follows this transition matrix.  
I'm leaving the time step size unspecified, since I'm making up  
these numbers anyway.

M S H E O A  
M .8 0 .1 0 0 .1  
S 0 .98 .01 0 0 .01  
H 0 0 .8 .15 0 .05  
E 0 0 0 .7 .2 .1  
O 0 0 0 0 .9 .1  
A .1 .3 .3 .1 .1 .1

i) Explain what is good and what is not-so-good about this model.

**A good aspect about the model is that it disallows physically impossible transitions such as Superior to Ontario. Also, it is reasonable to think that the ‘no-change’ transitions have high probabilities (the lakes are huge, and the channels are tiny).**

**One not-so-good aspect is that it does not take seasonal effects into account. For example, transitioning from a lake to air has a higher probability in summer (when air is warm and holds more water) than in winter (when air is cold and holds less water, and lakes are covered with ice).**  
ii) Now pretend the model is perfect as I specified it,  
and find the relative amounts of water in each compartment.

**I wrote an R script that repeatedly multiplies a state vector with the transition matrix, updating the state vector each time. My code does 1,000 iterations; it converges by then (though I did not test fewer iterations). Below is a printout of the results for the initial and steady state vectors. The steady state vector has the relative amounts of water in each of the lakes (or air).**

**-Link to the code:** [**https://github.com/Wubuntu88/Math519StochasticModeling/blob/master/DTMC\_HW/ex1WaterMoleculePart\_ii.R**](https://github.com/Wubuntu88/Math519StochasticModeling/blob/master/DTMC_HW/ex1WaterMoleculePart_ii.R)

**-Link to the transition matrix (you’ll need it do run the other file):** [**https://github.com/Wubuntu88/Math519StochasticModeling/blob/master/DTMC\_HW/ex1WaterMoleculeTransitionMatrix.csv**](https://github.com/Wubuntu88/Math519StochasticModeling/blob/master/DTMC_HW/ex1WaterMoleculeTransitionMatrix.csv)

**[1] "initial state vector:"**

**Michigan Superior Huron Eire Ontario Air**

**[1,] 0.2 0.2 0.2 0.2 0.2 0.2**

**[1] "steady state vector"**

**[1] "Michigan" "Superior" "Huron" "Eire" "Ontario" "Air"**

**[1] "0.024" "0.727" "0.121" "0.077" "0.202" "0.048"**

iii) optional/project idea: make a more accurate model, and/or  
adjust the transition probabilities.

Economic Quintile Problem

Use the graph at  
http://people.emich.edu/aross15/coursepack3419/opportunity-1.png  
to create a Markov Chain model of income mobility from one generation  
to the next. The 1st quintile is the poorest, and the 5th is the richest.  
a) What is the transition matrix? Figure out where each number goes.  
There is a subtle issue here--what is it? Figure out what it is and  
fix it in a reasonable way.

**Transition matrix:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| From To | 1st Quintile | 2nd Quintile | 3rd Quintile | 4th Quintile | 5th Quintile |
| 1st Quintile | **0.42** | **0.23** | **0.19** | **0.11** | **0.06** |
| 2nd Quintile | **0.25** | **0.23** | **0.24** | **0.18** | **0.10** |
| 3rd Quintile | **0.17** | **0.24** | **0.23** | **0.17** | **0.19** |
| 4th Quintile | **0.08** | **0.15** | **0.19** | **0.32** | **0.26** |
| 5th Quintile | **0.09** | **0.15** | **0.14** | **0.23** | **0.39** |

**I believe the ‘subtle issue’ is that the proportion of people in the quintiles cannot really change. Even if people were to transition from one quintile to another, an equal number of people would be in each quintile. One person’s rise would be another’s fall, so in that sense it is a zero-sum game of economic mobility (quintile-wise). To make the economic transitions make more sense, I would change these quintiles to yearly income bins based on how much money the individuals on the borders of the quintiles make.**

b) If you are in the 5th quintile, what is the probability that  
your grandson will be in the 1st quintile?

**I performed this calculation in python and got an answer of 15.26%. I made an equation that shows the summation of the transition probabilities. These probabilities include all the ways that a 5th quintile individual can have a 1st quintile grandchild.**

**Link to the code:** [**https://github.com/Wubuntu88/Math519StochasticModeling/blob/master/DTMC\_HW/ex2b\_EconomicQuintiles.py**](https://github.com/Wubuntu88/Math519StochasticModeling/blob/master/DTMC_HW/ex2b_EconomicQuintiles.py)

c) What is the steady-state distribution? Compute it using methods  
learned in class.

**The steady state distribution ends up with 1/5 being in each state.**

**Link to the code:** [**https://github.com/Wubuntu88/Math519StochasticModeling/blob/master/DTMC\_HW/ex2c\_EconomicQuintiles.py**](https://github.com/Wubuntu88/Math519StochasticModeling/blob/master/DTMC_HW/ex2c_EconomicQuintiles.py)

d) Explain the result you got in part (c).

**This is because of the way that the transition matrix is set up. The ingoing amounts to each state add up to about 1/5 of the total, so the amounts in each state do not change.**  
e) Consider a person who is politically liberal/left-wing.  
What would they want the transition matrix in part (a) to look like?

**TODO**  
f) Consider a person who is politically conservative/right-wing.  
What would they want the transition matrix in part (a) to look like?

**TODO**

Problem 4.33 Three types of exams (also not too bad)

1. A professor continually gives exams to her students. She can give three possi- ble types of exams, and her class is graded as either having done well or badly. Let *pi* denote the probability that the class does well on a type *i* exam, and sup- pose that *p*1 = 0.3, *p*2 = 0.6, and *p*3 = 0.9. If the class does well on an exam, then the next exam is equally likely to be any of the three types. If the class does badly, then the next exam is always type 1. What proportion of exams are type *i*, *i* = 1,2,3?

**I first show the ‘broken down’ transition matrix. The 0.7, 0.4, and 0.3 in the first column represent the probability of doing poorly on test 1, 2, and 3 respectively (meaning they will have to take test 1 next time). The 0.1, 0.1, and 0.1 in the first row represents the sum the probability of doing well on test 1 (0.3). If the students do well on test 1, then there is an equally likely chance that they take test 1, 2, or 3 next time. Thus, the 0.3 is split evenly across the first row. The other rows can be filled in with the similar reasoning.**

|  |  |  |  |
| --- | --- | --- | --- |
|  | Test 1 | Test 2 | Test 3 |
| Test 1 | **0.1 + 0.7** | **0.1** | **0.1** |
| Test 2 | **0.2 + 0.4** | **0.2** | **0.2** |
| Test 3 | **0.1 + 0.3** | **0.3** | **0.3** |

**The full transition matrix is below:**

|  |  |  |  |
| --- | --- | --- | --- |
|  | Test 1 | Test 2 | Test 3 |
| Test 1 | **0.8** | **0.1** | **0.1** |
| Test 2 | **0.6** | **0.2** | **0.2** |
| Test 3 | **0.4** | **0.3** | **0.3** |

**The steady state vector for the proportion of test types taken for type 1, 2, and 3 (respectively) is:**

**steady state vector:**

**[ 0.71336, 0.14332, 0.14332]**

**Link to the code:** [**https://github.com/Wubuntu88/Math519StochasticModeling/blob/master/DTMC\_HW/ex3\_types\_of\_exams.py**](https://github.com/Wubuntu88/Math519StochasticModeling/blob/master/DTMC_HW/ex3_types_of_exams.py)

**In summary the students will be taking exam type 1 71% of the time, type 2 14% of the time, and type 3 14% of the time. The students are doomed to take the hardest exam (type 1) most of the time.**

**Tic Tac Toe**

Consider a 3-by-3 tic-tac-toe board. At each step, a marker hops from  
its current cell to an up/down/left/right adjacent cell, choosing with  
equal probability among its available options.  
It cannot stay in the same cell during a hop.   
i) Formulate a Markov chain for it.

**I created a transition probability matrix with the following numbering on the tic tac toe board. Each number represents a state.**

**Tic Tac Toe states:**

To

|  |  |  |
| --- | --- | --- |
| **1** | **2** | **3** |
| **4** | **5** | **6** |
| **7** | **8** | **9** |

**Transition Probability matrix:**

From

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| 1 | 0 | ½ | 0 | ½ | 0 | 0 | 0 | 0 | 0 |
| 2 | 1/3 | 0 | 1/3 | 0 | 1/3 | 0 | 0 | 0 | 0 |
| 3 | 0 | ½ | 0 | 0 | 0 | ½ | 0 | 0 | 0 |
| 4 | 1/3 | 0 | 0 | 0 | 1/3 | 0 | 1/3 | 0 | 1 |
| 5 | 0 | ¼ | 0 | ¼ | 0 | ¼ | 0 | ¼ | 0 |
| 6 | 0 | 0 | 1/3 | 0 | 1/3 | 0 | 0 | 0 | 1/3 |
| 7 | 0 | 0 | 0 | ½ | 0 | 0 | 0 | ½ | 0 |
| 8 | 0 | 0 | 0 | 0 | 1/3 | 0 | 1/3 | 0 | 1/3 |
| 9 | 0 | 0 | 0 | 0 | 0 | ½ | 0 | ½ | 0 |

ii) What do you think the steady-state solution will be?

**My hypothesis for estimating the steady state solution is as follows. Each corner position has two transitions. Each non-corner edge position has 3 transitions. The middle position has 4 transitions. These transitions add up to a total of 4\*2 + 4\*3 + 1\*4 = 24 possible transitions. The steady state vector would be as follows:**

**[2/24, 3/24, 2/24, 3/24, 4/24, 3/24, 2/24, 3/24, 2/24]**

**As we will see in iii, the proposed steady state vector is not right.**  
iii) Find the steady-state solution. Is it what you thought it might be? Explain.

**I calculated the steady state solution by writing a python script. I started with the steady state vector being a vector of ones with length 1. At the end iterative steps, I divided each value in the array by the sum of the array. The steady state vector is below.**

['0.093', '0.111', '0.093', '0.111', '0.185', '0.111', '0.093', '0.111', '0.093'].

**What is interesting is that the non-corner edge states are not 50% greater in value than the corner states. However, the middle state is twice as much as the corner states (as I had predicted, although the ratios are not 2/24 for the corner states and 4/24 for the middle state).**

**The code for my simulation can be found here:** [**https://github.com/Wubuntu88/Math519StochasticModeling/blob/master/DTMC\_HW/tictactoe.py**](https://github.com/Wubuntu88/Math519StochasticModeling/blob/master/DTMC_HW/tictactoe.py)  
iv) Imagine that the marker cannot hop back to the cell it just came from.  
Describe any needed changes in the Markov model, but you do not have to  
formulate the matrix.

**To make it so that the marker cannot hop back to the previous cell, we would have to expand the transition probability matrix to include the current and previous state. Note that the ‘From’ rows are expanded, not the ‘To’ columns. For cases where the previous state is n, the cell in the transition probability matrix at a given row at the nth column will be 0.**

Taxi Zones

Problem 4.52: the taxi-driver, city with 2 zones.

It's possible to do this using a 2-state or a 4-state chain;

I recommend the 2-state.

First, find the proportion of trips that start in zone A.

Then, find the expected profit per trip starting in zone A.

Then, the expected profit per trip starting in zone B.

Then, combine them for the overall expected profit per trip.

Hint: one of the intermediate values you might get in this problem

is 0.4285714

[4 for matrix, 4 for pi vector, 4 for average cost]

**The transition probability matrix for the taxi traveling through the zones is the following.**

|  |  |  |
| --- | --- | --- |
|  | To zone A | To zone B |
| From zone A | **0.6** | **0.4** |
| From zone B | **0.3** | **0.7** |

**I calculate the proportion of trips starting in A, and the proportion of trips starting in B by initializing a ‘not-yet’ steady-state vector to [1, 1]. I repeatedly multiply the ‘not-yet’ steady state vector (100 times) until it converges to its steady state value.**

**Steady state vector (pi vector):**

|  |  |
| --- | --- |
| P (start in zone A) | P (start in zone B) |
| 0.42857143 | **0.57142857** |

**To calculate the expected cost of a trip starting in zone A and a trip starting in zone B, I performed the following calculations:**

**expected\_profit\_from\_zone\_A = P ( A -> A ) \* cost( A -> A) +   
 P ( A -> B ) \* cost( A -> B)**

**expected\_profit\_from\_zone\_A = 0.6 \* 6 + 0.4 \* 12 = 8.4  
  
expected\_profit\_from\_zone\_B = P ( B -> B ) \* cost( B -> B) +   
 P ( B -> A ) \* cost( B -> A)**

**expected\_profit\_from\_zone\_B = 0.3 \* 8 + 0.7 \* 12 = 10.8**

**The results for the profit are summarized in the table below.**

|  |  |
| --- | --- |
| Expected profit from zone A | Expected profit from zone B |
| 8.4 | **10.8** |

The computations were done in the python script ‘taxi.py’.

**An Inventory Problem**

Part a) Transition Matrix

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| 0 | 5/8 | 1/8 | 1/8 | 1/8 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 4/8 | 1/8 | 1/8 | 1/8 | 1/8 | 0 | 0 | 0 | 0 | 0 | 0 |
| 2 | 3/8 | 1/8 | 1/8 | 1/8 | 1/8 | 1/8 | 0 | 0 | 0 | 0 | 0 |
| 3 | 2/8 | 1/8 | 1/8 | 1/8 | 1/8 | 1/8 | 1/8 | 0 | 0 | 0 | 0 |
| 4 | 1/8 | 1/8 | 1/8 | 1/8 | 1/8 | 1/8 | 1/8 | 1/8 | 0 | 0 | 0 |
| 5 | 0 | 1/8 | 1/8 | 1/8 | 1/8 | 1/8 | 1/8 | 1/8 | 1/8 | 0 | 0 |
| 6 | 0 | 0 | 1/8 | 1/8 | 1/8 | 1/8 | 1/8 | 1/8 | 1/8 | 1/8 | 0 |
| 7 | 0 | 0 | 0 | 1/8 | 1/8 | 1/8 | 1/8 | 1/8 | 1/8 | 1/8 | 1/8 |
| 8 | 0 | 0 | 0 | 0 | 1/8 | 1/8 | 1/8 | 1/8 | 1/8 | 1/8 | 2/8 |
| 9 | 0 | 0 | 0 | 0 | 0 | 1/8 | 1/8 | 1/8 | 1/8 | 1/8 | 3/8 |
| 10 | 0 | 0 | 0 | 0 | 0 |  | 1/8 | 1/8 | 1/8 | 1/8 | 4/8 |

Part b)

Steady state vector:

(this represents the probability of there being n items in the warehouse overnight).

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| 0.34 | **0.1** | **0.11** | **0.11** | **0.07** | **0.06** | **0.06** | **0.04** | **0.03** | **0.03** | **.05** |

The probability of having zero items in the warehouse overnight is about 34%.

Part c)

If it costs $10 per item in the warehouse overnight, what is our average inventory cost per night? Hint: it's around $28 or $29

**I calculated the average inventory cost per night by taking the steady state vector and multiplying it by an inventory cost vector. The steady state vector contains the probabilities of being in a state overnight. The inventory cost vector contains how expensive it is to have n items in the warehouse overnight. The python snippet showing the calculation is below.**

**average\_inventory\_cost = state\_vector.dot(inventory\_cost\_vector)**

**The average inventory cost per night is $28.89.**

**The code that does these calculations is in the python file ‘AnInventoryProblem.py’.**