

Homework 5, CS 177

Applications of Probability in Computer Science: Winter 2007

Due Tuesday February 20th, 12 noon (to be submitted electronically)

Suggested Reading

- Text: pages 417 to 421.
- Class notes on stationary probabilities for Markov chains.

In all problems below the term “Markov chain” should be interpreted as a discrete-time, finite-state, homogeneous Markov chain.

Your solutions should be submitted electronically to the Homework 5 dropbox for EEE. The MATLAB functions should be submitted using the function names below and the solutions to the other problems should be submitted in a single Word document or PDF file. In cases where numerical answers are required (such as vectors of probabilities) provide your answers to 4 decimal places.

Problem 1

Implement a function in MATLAB called `markov_steadystate.m` that uses the power method as described in class to compute the steady-state probabilities for a Markov chain with M states. You can assume that the matrix M that is provided to this function is irreducible and aperiodic. You should initialize the power method with the uniform vector, i.e., $x = (1/M, 1/M, \dots, 1/M)$. If the number of iterations exceeds a specified maximum number of iterations (see code below) the iterations are halted (this prevents runaway executions). Please use comments liberally in your code. The header for your function should be exactly as follows:

```
function pi = markov_steadystate(Pmatrix, initial_pi epsilon);
%
% computes the steady-state probabilities for a Markov chain
%
% INPUTS:
% Pmatrix: M x M transition probability matrix
% initial_pi: initial estimate of pi, 1 x M vector
% epsilon: convergence criterion for the power method.
% Specifically, let pi^k be the vector at iteration k, and pi^{k-1} the
% vector at iteration k-1.
% Let S = the sum of the absolute differences between the elements
% of pi_k and pi_{k-1}
% If S < epsilon, declare convergence, halt the iterations and return
% Otherwise continue to the next iteration.
% A typical value for epsilon might be 0.00001
%
% OUTPUT:
% pi: an M x 1 vector of steady-state probabilities as estimated
% using the power method.
%
% [Student name here], CS 177, Winter 2007

MAX_ITERATIONS = 10000 % the code should halt if this number of iterations
% are reached (some large number) - prevents
% runaway executions, e.g., due to bugs, etc.

% rest of the code follows below.....
```

Problem 2

Let P be the four-state Markov chain below where

$$P = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0.5 & 0 & 0 & 0.5 \\ 0 & 0.6 & 0 & 0.4 \\ 0.0 & 0.6 & 0.4 & 0 \end{pmatrix}$$

1. Use the code from the previous problem to calculate the steady state probabilities for the Markov transition matrix P above. Initialize the algorithm with $\pi^{(0)} = [0.25, 0.25, 0.25, 0.25]$.
2. Provide the $\pi^{(k)}$ vectors for $k = 1, 3, 5, 10, 20, 50, 100$ (you can temporarily modify your code to print out these numbers at these values of the iterations). $\pi^{(k)}$ is the estimate of the steady-state probability vector after the k th iteration of the power method.
3. Plot the value of $\pi_2^{(k)}$, the estimate of state 2's steady-state probability at the k th iteration, as a graph where $\pi_2^{(k)}$ is the y-axis and k is the x-axis, with $k = 1, \dots, 50$.

Problem 3

In problem 2, using 2 different initialization vectors, $\pi^{(0)} = [1, 0, 0, 0]$ and $\pi^{(0)} = [0, 0, 0, 1]$ show that it does not matter what initial vector we use, the power method still converges to the same result. For each of these 2 examples, plot the value of $\pi_2^{(k)}$, the estimate of state 2's steady-state probability at the k th iteration, as a graph where $\pi_2^{(k)}$ is the y-axis and k is the x-axis, with $k = 1, \dots, 50$. Use the same graph for both plots. Comment briefly on the results.

Problem 4

Implement a function in MATLAB called `markov_simulate.m` that simulates data from a Markov chain with M states. Please use comments liberally in your code. The header for your function should be exactly as follows:

```
function sequence = markov_simulate(Pmatrix, ivector, N, rseed);
%
% simulates a sequence of length N from a Markov chain
%
% INPUTS:
% Pmatrix: M x M transition probability matrix
% ivector: M x 1 vector of initial state probabilities
% N: length of sequence to be simulated
% rseed (optional): integer seed value to initialize the random number generator
%
% OUTPUT:
% sequence: an N x 1 vector of states simulated
%         from the Markov chain, where each state is represented by
%         a number from 1 to M., e.g., 3 3 3 1 1 2 1 3 1 2 2 2 3
%
%                                     [Student name here], CS 177, Winter 2007

% initialize the random number generator so that whenever the function
% is called with the integer value "rseed" we get the same sequence
% pseudorandom results (useful for testing our code)
if nargin > 3
    rand('state',rseed);
end

% rest of the code follows below.....
```

Problem 5

1. Use the code from Problem 4 to simulate a sequence of length 20 from the Markov chain in Problem 2, with the value of `rseed = 1234`. Compute the relative frequency (from the simulated sequence) that the system spends in each of the 4 states.
2. Now generate another simulated sequence, this time of length 100,000 (any `rseed` value is fine this time, e.g., 1234 again). Compute the relative frequency (from the simulated sequence) that the system spends in each of the 4 states.
3. Compare the answers for the steady-state probabilities you obtained in parts 1 and 2 of this problem with those computed in Problem 2 using the power method and comment on any similarities or differences.