closed and transient sets in arbitrary Markov chain probability transition matrix

Nasser M. Abbasi

March 9, 2008 page compiled on July 1, 2015 at 2:00pm

The problem: Given an arbitrary matrix which represents the probability of transition from one state to another state in one step for a Markov chain, determine all closed sets of states and the transient state.

This small note describes an algorithm that I developed which solves this problem. The algorithm is recursive in nature.

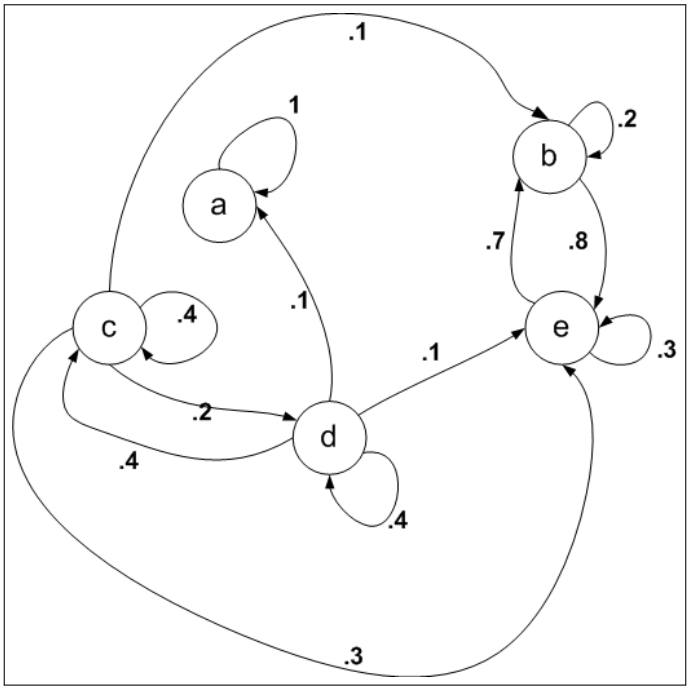
A Matlab implementation is given below with an example run on 3 different matrices and how to call the matlab function.

1 Introduction

This algorithm describes how to find all closed sets and the transient set (if any) given as input the p matrix which contains the initial one step finite chain Markov probability transition from one state to another. The algorithm works directly on the matrix p itself and does not require prior knowledge of which states are recurrent.

An implementation of the algorithm is provided (in Matlab) with a number of examples.

As an example of an input, let us consider the following simple Markov chain state diagram and its corresponding p matrix and then we show the output generated from this algorithm. In this diagram, the numbers on the arrows between the states (the circles) is the probability of going from the source state to the end state.



The matrix p for the above diagram is the following

	a	b	c	d	e
a	1	0	0	0	0
b	0	.2	0	0	.8
c	0	.1	.4	.2	.3
d	.1	0	.4	.4	.1
e	0	.7	0	0	.3

The output of the algorithm below will be the following
closed sets: $\{a\},\{b,e\}$, transient set: $\{c,d\}$
Once the closed sets and the transient set are known, generating the canonical form is a simple matter of formatting the p matrix in the following form

	a	b	e	d	c
a	1	0	0	0	0
b	0	.2	.8	0	0
e	0	.7	.3	0	0
d	.1	0	.1	.4	.4
c	0	.1	.3	.2	.4

Q matrix

Now the algorithm is described.

2 Algorithm description

The basic idea of the algorithm is the following: For each state, we generate the path of states that can be travelled from that state. Next, we collect the states which has the same states on its path. Each set of states which has the same states on their path are one closed set. The remaining states, if any, would all have different set of states on their path. These states make up the set of transient states.

The algorithm will terminate since the length of the longest path that could possibly be travelled from any state is finite (which is the number of states in the chain).

Since there are n states, and for each state we need to examine $n-1$ other states at most, the algorithm is of order $O\left(n^2\right)$.

Here is a description of the algorithm
Input: A square n by n probability transition matrix (i.e. each row in the matrix sums to 1)
output: all sets which are closed and transient set (if it exists)

1. For each state i in the matrix p generate the list of states which state i can reach in one step. These are the states j in which $p_{ij} \neq 0$. Call the list of states i as LHS, and call the list of states j as the RHS. Hence the LHS will be a list of length n and the RHS will be a list of lists where each list is at most of length n and at least is of length 1.

2. For each state i in the LHS set, scan its RHS list of states j, and do the following for each state j found

   (a) If $j = i$ then do nothing.

   (b) if $j \neq i$ then replace j by $\{j \cup RHS \text{ states of } j\}$

   (c) At the end, remove all duplicate states in the new RHS list.

3. Compare the updated RHS list from step (2) with the RHS at the start of step (2). If these 2 sets are the same then go to step 4 else go back to step 2.

4. Now the process is complete. We now analyze the final RHS list and generate from it the closed sets as follows. Find all states in the LHS set which has the same list of states on its RHS. If the union of the LHS states equals the states in the RHS, then the states in the LHS thus found make one closed set otherwise the states in the LHS make the transient set.

To illustrate how this algorithm works, we apply it to the example Markov chain shown above. In step 1, we generate the LHS and the RHS lists from the input p matrix.

$$\overbrace{\begin{Bmatrix} a \\ b \\ c \\ d \\ e \end{Bmatrix}}^{LHS} = \overbrace{\begin{Bmatrix} a & & & & \\ b & e & & & \\ b & c & d & e & \\ a & c & d & e & \\ b & e & & & \end{Bmatrix}}^{RHS} \tag{1}$$

The above is send to step 2. In step 2 we start by scanning the RHS list of each state listed in the LHS set. For state a we see that there is only state a in its RHS. and from the rule in 2.a we see that we do nothing and go to the next state in its RHS set. But this is the only state in this set. Hence we are done with state a and we go to state b in the LHS set.

We start again by looking at the set of states of the RHS of state b, which are $\{b, e\}$. we start with state b which is the same as the state in the LHS, so we skip this per rule 2.a, and go to the next state which is e. According to 2.b, we replace this by $e \cup RHS$ states of e, where RHS of state e is seen to be $\{b, e\}$ which is the last RHS set in the last entry of the LHS set as shown in (1). Hence now the RHS of state b becomes $\{b, e, \{b, e\}\}$. Since there are no more states in the RHS of b we have completed this state. Now we remove duplicated entries in the above list to obtain $\{b, e\}$. We now go to state c in the LHS set. and do the same. We do this until we processed all states in the LHS. At the end we have the following LHS and RHS generated

$$\overbrace{\begin{Bmatrix} a \\ b \\ c \\ d \\ e \end{Bmatrix}}^{LHS} = \overbrace{\begin{Bmatrix} a & & & & & \\ b & e & & & & \\ b & e & c & d & a & \\ a & c & b & d & e & \\ b & e & & & & \end{Bmatrix}}^{RHS} \tag{2}$$

We now compare RHS sets in (1) with those in (2). Since they are not the same, we repeat step 2 again, but use the above RHS now as an input to step 2.

Again, we start by scanning the RHS list of each state in the LHS. We repeat the same thing as before, lets say we reached state c in the LHS set now. We see that its RHS is $\{b, e, c, d, a\}$, we start with state b and replace that with $b \cup RHS$ states of b which is $\{b, \{b, e\}\}$. We go to state e, and replace that with $e \cup RHS$ states of e which is $\{e, \{b, e\}\}$. Next we do state c and replace that with $c \cup RHS$ states of c which is $\{c, \{b, e, c, d, a\}\}$, next we do state d, and replace that with $d \cup RHS$ states of d which is $\{d, \{a, c, b, d, e\}\}$, and finally we do state a and replace that with $a \cup RHS$ states of a which is $\{a, \{a\}\}$, now that we completed the RHS processing of state c we remove all duplicated states in its RHS and we obtain $\{b, e, c, d, a\}$. Next we go to state d in the LHS set and process that similarly. We then do state e in the LHS set. Now we have completed step 2 again and the final result is the following

$$\overbrace{\begin{Bmatrix} a \\ b \\ c \\ d \\ e \end{Bmatrix}}^{LHS} = \overbrace{\begin{Bmatrix} a & & & & \\ b & e & & & \\ b & e & c & d & a \\ a & c & b & d & e \\ b & e & & & \end{Bmatrix}}^{RHS} \tag{3}$$

We see now that the RHS in (3) is the same as the RHS in (2) hence we stop and go to step 4 in the algorithm.

Now we collect all states from the LHS which has the same list in its RHS. We see that state a is its own class. And since the LHS of this class which is state a is the same as the state in the RHS which is a, then this class is closed set. Next we see that states $\{b, e\}$ has the same states on their RHS which is $\{b, e\}$ and since the LHS states is the same as the RHS states, then this is a close set class that contains states $\{b, e\}$, next we see that state c and state d also has the same RHS set which is $\{b, e, c, d, a\}$ but now since $\{c, d\} \neq \{b, e, c, d, a\}$, then the states $\{c, d\}$ make a class which is the transient states. Hence $\{c, d\}$ are transient and all other states are recurrent.

The following is a test run of the implementation showing in each case the matrix p and the output of the algorithm.

```
EDU>> clear all
EDU>> close all
EDU>> nmaTestMarkov()
**************************

    1.0000         0         0         0         0
         0    0.2000    0.8000         0         0
         0    0.7000    0.3000         0         0
    0.1000         0    0.1000    0.4000    0.4000
         0    0.1000    0.3000    0.2000    0.4000

found the following closed sets
{1}
{2,3}
found the following transient set
{4,5}


**************************
```

```
    0.1000    0.3000         0         0    0.6000
         0    0.2000    0.7000         0    0.1000
         0    0.7000    0.3000         0         0
    0.1000         0    0.1000    0.4000    0.4000
         0    0.1000    0.3000    0.2000    0.4000

found the following closed sets
{1,2,3,4,5}
No transient set found


**************************

    0.5000    0.5000         0         0         0         0
         0         0    1.0000         0         0         0
    0.3333         0         0    0.3333    0.3333         0
         0         0         0    0.5000    0.5000         0
         0         0         0         0         0    1.0000
         0         0         0         0    1.0000         0

found the following closed sets
{5,6}
found the following transient set
{1,2,3,4}
**************************

    1.0000         0         0         0         0         0
         0    1.0000         0         0         0         0
         0         0    1.0000         0         0         0
         0         0         0         0    0.5000    0.5000
    0.2500    0.2500         0         0         0    0.5000
    0.2500         0    0.2500         0    0.5000         0

found the following closed sets
{1}
{2}
{3}
found the following transient set
{4,5,6}
**************************

    1.0000         0         0         0         0         0         0         0         0
         0    1.0000         0         0         0         0         0         0         0
    0.2222    0.1111         0    0.0833    0.1111    0.1389    0.1389    0.1111    0.0833
    0.1389    0.1667         0    0.7500         0         0         0         0         0
    0.1111    0.1667         0         0    0.7222         0         0         0         0
    0.1389    0.1667         0         0         0    0.6944         0         0         0
    0.1389    0.1667         0         0         0         0    0.7222         0         0
```

```
    0.1111    0.1667         0         0         0         0         0    0.7222         0
    0.0833    0.1667         0         0         0         0         0         0    0.7500

found the following closed sets
{1}
{2}
found the following transient set
{3,4,5,6,7,8,9}
EDU>>
```

3 Source code

1. nmaTestMarkov.m.txt
2. nmaChainMarkov.m.txt

4 example output

```
clear all
close all
nmaTestMarkov()

***************************

    1.0000         0         0         0         0
         0    0.2000    0.8000         0         0
         0    0.7000    0.3000         0         0
    0.1000         0    0.1000    0.4000    0.4000
         0    0.1000    0.3000    0.2000    0.4000

found the following closed sets
{1}
{2,3}
found the following transient set
{4,5}

***************************
    0.1000    0.3000         0         0    0.6000
    0    0.2000    0.7000         0    0.1000
         0    0.7000    0.3000         0         0
    0.1000         0    0.1000    0.4000    0.4000
         0    0.1000    0.3000    0.2000    0.4000

found the following closed sets
{1,2,3,4,5}
No transient set found

***************************
    0.5000    0.5000         0         0         0         0
```

```
         0         0    1.0000         0         0         0
    0.3333         0         0    0.3333    0.3333         0
         0         0         0    0.5000    0.5000         0
         0         0         0         0         0    1.0000
         0         0         0         0    1.0000         0

found the following closed sets
{5,6}
found the following transient set
{1,2,3,4}

**************************
    1.0000         0         0         0         0         0
         0    1.0000         0         0         0         0
         0         0    1.0000         0         0         0
         0         0         0         0    0.5000    0.5000
    0.2500    0.2500         0         0         0    0.5000
    0.2500         0    0.2500         0    0.5000         0

found the following closed sets
{1}
{2}
{3}

found the following transient set
{4,5,6}
**************************
    1.0000         0      0         0         0         0         0         0         0
         0    1.0000      0         0         0         0         0         0         0
    0.2222    0.1111      0    0.0833    0.1111    0.1389    0.1389    0.1111    0.0833
    0.1389    0.1667      0    0.7500         0         0         0         0         0
    0.1111    0.1667      0         0    0.7222         0         0         0         0
    0.1389    0.1667      0         0         0    0.6944         0         0         0
    0.1389    0.1667      0         0         0         0    0.7222         0         0
    0.1111    0.1667      0         0         0         0         0    0.7222         0
    0.0833    0.1667      0         0         0         0         0         0    0.7500

found the following closed sets
{1}
{2}
found the following transient set
{3,4,5,6,7,8,9}
```