2009年度応用マクロ経済学講義ノート　マルコフチェーン

阿部修人

平成21年12月11日

概 要

1　マルコフチェーンの性質

不確実性下の経済行動を描写するとき、確率をモデルに導入することになるが、動学モデルの場合は、静学モデルと異なり、前期の様子に今期の確率変数のとる値が依存すると仮定する必要が出てくる。例えば、確率変数として、ある人間の雇用状態を考え、雇用と失業の二種類の状態をとると考えよう。最も単純な仮定としては、来期雇用されるか否かは前期の雇用状態に依存しないとするものであるが、これは極めて非現実的である。前期失業している者は今期も失業する確率は高いし、前期雇用されている者は今期も継続して雇用される確率は高いと考えるのが普通であろうし、実際に多くの実証研究で支持されている。このような前期の状態に今期の状態確率が依存する状態を描写するときに、最も単純な確率過程がマルコフチェーンである。

動学モデルの数値解析において、マルコフチェーンの果たす役割は大きい。前回の講義では不確実性のないモデルであったが、今期の資本ストックが所与の時、来季の資本ストックがどこに行くのが最適であるかをPolicy Functionで表した。これは、各行に関して、要素1が一つだけ入る特殊なマルコフチェーンとみなすことができる。経済に不確実性が導入されると、より多くのnon zero成分を含むマルコフチェーンとしてPolicy Functionが導出される。

マルコフチェーンは経済学以外にも多くの応用先のある、確率論における一大分野であり、非常に多くのトピックがある。ここでは、後の講義で議論する、Bewley-Aiyagariモデルで必要な知識に関してのみ議論する。マルコフチェーンの詳細な議論に関しては、例えば森村・木島(1991)『ファイナンスのための確率過程』や渡部隆一『マルコフチェーン』等を参照せよ。

一般に、マルコフ過程とは、二つの時間 t_1 と $t_2\,(t_1 < t_2)$ が与えられた時、$t = t_1$ における状態、例えば $C\,(t_1)$ の実現値のみによって、確率変数 $C\,(t_2)$ の分布が決まり、時刻 t_1 よりも前の歴史には無関係の場合、$C\,(t)$ をマルコ

フ過程という。未来は現在のみに依存し、過去には関係しない、ということである。現在の状態に、それまでの歴史的情報が全て含まれており、現在の状態が将来を決定するという考えは、Dynamic Programing と親和的であり、前述のように、Policy Function を経由した、状態の遷移式はマルコフ過程の一種とみなすことができるのである。

マルコフチェーンは、確率変数 $C(t)$ がとる値が整数の集合としてあらわされる時のマルコフ過程である。

現在失業しているとき、来季に就業する確率が 0.4、現在就業していて、将来失業する確率が 0.1 だとする。すると、機械的に、現在失業していて、来季も失業し続ける確率は 0.6 で、現在就業していて、来季も就業し続ける確率は 0.1 となることがわかる。

縦の列を現在の状態、横の列を来季の状態と考えると

$$\begin{array}{cc} & \text{来季} \\ \text{現在} & \begin{pmatrix} U \to U & U \to W \\ W \to U & W \to W \end{pmatrix} \end{array}$$
$$= \begin{pmatrix} 0.6 & 0.4 \\ 0.1 & 0.9 \end{pmatrix}$$

例えば、現在、失業している人が 10 人、就業している人が 20 人いるとすると、来季は

$$\begin{pmatrix} 10 & 20 \end{pmatrix} \begin{pmatrix} 0.6 & 0.4 \\ 0.1 & 0.9 \end{pmatrix} = \begin{pmatrix} 6+2 & 4+18 \end{pmatrix} = \begin{pmatrix} 8 & 22 \end{pmatrix}$$

となり、失業者は 8 人、就業者は 22 人と (確率的に) なることがわかる。さらに、次々と時間を経過させていくと

$$\begin{aligned} \begin{pmatrix} 8 & 22 \end{pmatrix} \begin{pmatrix} 0.6 & 0.4 \\ 0.1 & 0.9 \end{pmatrix} &= \begin{pmatrix} 7 & 23 \end{pmatrix} \\ \begin{pmatrix} 7 & 23 \end{pmatrix} \begin{pmatrix} 0.6 & 0.4 \\ 0.1 & 0.9 \end{pmatrix} &= \begin{pmatrix} 6.5 & 23.5 \end{pmatrix} \\ \begin{pmatrix} 6.5 & 23.5 \end{pmatrix} \begin{pmatrix} 0.6 & 0.4 \\ 0.1 & 0.9 \end{pmatrix} &= \begin{pmatrix} 6.25 & 23.75 \end{pmatrix} \\ &= \ldots \\ &= \begin{pmatrix} 6.0 & 24.0 \end{pmatrix} \end{aligned}$$

失業者が 6 人、就業者が 24 人のとき、

$$\begin{pmatrix} 6.0 & 24.0 \end{pmatrix}\begin{pmatrix} 0.6 & 0.4 \\ 0.1 & 0.9 \end{pmatrix} = \begin{pmatrix} 6.0 & 24.0 \end{pmatrix}$$

となり、来季も分布が変化しない、すなわち就業状況の分布が定常になっているのである。この、現在と来季の分布をつなげる行列は推移確率行列と呼ばれる。

一般に、マルコフチェーンは、状態が 2 の場合は下記の推移確率行列により描写される。

$$P = \begin{pmatrix} p_{11} & p_{12} \\ p_{21} & p_{22} \end{pmatrix}$$

ここで、p_{ij} は、前期状態 i にいて、今期は状態 j に移る確率である。したがって、この推移確率行列の全ての成分は非負であり、かつ、行を足すと 1 にならねばならない。

例えば、労働者は、失業している (U) か、就業 (W) しているかのどちらかの状態に常にある、と仮定しよう。

すなわち、

$$\sum_{j=1}^{n} p_{ij} = 1, 0 \leq p_{ij}. \text{ for all i.}$$

推移確率行列に関しては、下記の性質は知っておくべきである。

(1) 成分が全て 1 のベクトル f に関して、

$$Pf = f$$

(2) P, Q が確率行列であれば、PQ も確率行列である。

(3) P が確率行列のとき、1 を左右の固有値として持つ。

(4) 確率行列 P の左右の固有値の絶対値は 1 以下である。

いま、

$$P'x = x$$

を満たすベクトル x を考える。これは、固有値 1 に対応する P' の固有ベクトルである。推移確率行列の定義に従って書けば

$$x'P = x'$$

と書くことも出来る。さて、このとき、

$$x'PP = x'P = x'$$

だから、x' は P によって変化しない。推移確率行列が、今期の状態が来期にどのような状態に移るかを示していることを考えると、一期先に進んでも

分布が変わらない状態、すなわち、いつまでも分布が変化しない状態を示していることが分かる。このような x を定常分布という。定常分布は P の固有値 1 に対応する左固有ベクトルである。または、P' の固有値 1 に対応する右固有ベクトルである。

定常分布は下記の方程式の解で与えられる。

$$\begin{pmatrix} 1-P' \\ 1' \end{pmatrix} x = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$$

ただし、左辺の行列は x の次元を超えているため、最後の行を除くどこかの行を一つ除外し、正方行列にしてから計算する。

または、

$$P'x = x$$

をみたすような、転置行列 P' の固有値 1 に対応する固有ベクトルを見つければよい。

matlab では、

```
n=length(prob);
prob1=eye(n)-prob';
prob2=[prob1(2:n,:);ones(1,n)]
pai=zeros(n,1);
pai(n)=1;
sd = inv(prob2)*pai
pempl=sd(1)
```

あるいは

```
[v1,d1]=eig(prob');
[dmax,imax]=max(diag(d1));
probst1=v1(:,imax);
ss=sum(probst1);
probst1=probst1/ss;
```

推移確率行列の全ての成分が正であるとき、あるいは何回か推移を重ねるうちに全ての成分が正になるとき、定常分布はかならず存在し、かつユニークであることが知られている。上記の手法では、固有値をみつける、あるいは逆行列を計算する必要があったが、もしも定常分布がユニークであるなら、どんな初期分布から初めても、いつかは定常分布に収束するはずである。したがって、定常分布をもとめる手法として

```
n=length(prob);
```

```
value=1/n;
x=value*ones(1,n);
for i =1:100  %  100 というのは適当である。実際には収束基準を設けるべきであるが、、、100 回繰り返して収束しない場合はほとんどない。
    x=x*prob;
end
```

という方法も使うことが出来る。推移確率行列が大きく、逆行列や固有ベクトルを計算することが困難なときには便利な手法である。

1.1 AR(1) 過程のマルコフチェーンによる近似　Tauchen(1986)

$$y_t = \rho y_{t-1} + \varepsilon_t$$

という一階の自己回帰過程を考える。RBC モデルに限らず、経済学ではよく出てくる確率過程であり、y_t の分布は y_{t-1} を考慮した後は、y_{t-2} 等の過去の実現値に依存しないのでマルコフ過程でもある。この確率過程に従って生成される y の分布を考える。理想的には、y の値を連続的に扱いたいが、前回の講義ノートでみたように、状態変数の数が増加すると計算の負荷が大きくなるため、極力少数のグリッドで近似するとこを考える。このとき、離散の、有限個の値しかとらないマルコフチェーンで AR(1) を描写する手法を考えるのである。マルコフチェーンに対する近似手法には様々な手法が提案されているが、Tauchen (1986) の手法が簡単であり、それなりの高い精度で近似できることが知られている。

近似する自己回帰モデルを定常過程、すなわち、$0 < \rho < 1$ を仮定する。なお、非定常過程では、マルコフチェーンで近似することが出来ないので注意すること。

ε_t の分散が σ_ε^2 とすると、y の分布は 0 で対称となり、標準偏差は

$$\sigma = \frac{\sigma_\varepsilon}{\sqrt{1-\rho^2}}$$

である。ここで、例えば 3σ の範囲での近似を考える。n 個の点で近似するとすると、まずは、-3σ から 3σ の間で均等に n 個に分割する点をまずとる。

次に、各グリッドに前期存在すると仮定し、来期にどこのグリッドに行くか、グリッドごとの確率を AR(1) の過程から計算する。ここで、ε_t の分布の情報が必要になる。例えば、正規分布に従うとすると下記のように、Adda and Cooper の web に掲載されているプログラムとなる。

```
function [prob,eps,z]=tauchen(N,mu,ro,sig);
```

```
% Discretizes an AR(1) process into a Markov chain. Determines the optimal grid
% and transition matrix. Based on Tauchen (1991).
%
% y(t)= mu(1-ro) + ro*y(t-1) + u(t) with V(u) = sig^2
%
% syntax:
%
% [prob,eps,z]=tauchen(N,mu,ro,sig)
%
% N is the number of points on the grid.
% prob is the transition matrix
% eps contains the cut-off points from - infty to + infty
% z are the grid points, i.e. the conditional mean within [eps(i),eps(i+1)].
global mu_ ro_ sigEps_ sig_ eps_ jindx_
if N==1;prob=1; eps=mu;z=mu;
    else;
    if ro==0;
        sigEps=sig;
        eps=sigEps*norminv((0:N)/N)+mu;
        eps(1)=-20*sigEps+mu;
        eps(N+1)=20*sigEps+mu;
        aux=(eps-mu)/sigEps;
        aux1=aux(1:end-1);
        aux2=aux(2:end);
        z=N*sigEps*(normpdf(aux1)-normpdf(aux2))+mu;
        prob=ones(N,N)/N;
    else;
        sigEps=sig/sqrt(1-ro^2);
        eps=sigEps*norminv((0:N)/N)+mu;
        eps(1)=-20*sigEps+mu;
        eps(N+1)=20*sigEps+mu;
        aux=(eps-mu)/sigEps;
        aux1=aux(1:end-1);
        z=N*sigEps*(normpdf(aux1)-normpdf(aux2))+mu;
        mu_=mu;ro_=ro;sigEps_=sigEps;eps_=eps;sig_=sig;
        prob=zeros(N,N);
        for i=1:N
            for jindx_=1:N
```

```
            prob(i,jindx_)=quadl(@integ3,eps_(i),eps_(i+1),1e-6)*N;
        end
    end
  end
  z=z';
  eps=eps';
end
```