1 Notes on Markov Chains

A Markov chain is a finite-state process that can be characterized completely by two objects – the finite set of states $Z = \{z_1, ..., z_n\}$ and the transition matrix $\Pi = [\pi_{ij}]_{i,j=1,n}$ that describes the probability of leaving state i and entering state j. Π must have rows that sum to 1 and no element can be negative:

$$\begin{aligned} \pi_{ij} &\geq 0 \\ \sum\nolimits_j \pi_{ij} &= 1 \ \ \forall i. \end{aligned}$$

That is, with probability 1 you must move from state i to some state j and transition probabilities are never negative. Π gives the probability of transitioning between states in t and $t+1$; correspondingly, Π^n gives the probability of transitioning between states in t and $t+n$.

For dynamic programming purposes, we will need the transition probability matrix to be time-invariant. We also need the **invariant distribution**, which is the "long-run" probability of being in a given state, to be unique. Markov chains that have two properties possess unique invariant distributions.

Definition 1 *State i **communicates** with state j if $\pi_{ij}^n > 0$ and $\pi_{ji}^n > 0$ for some $n \geq 1$. A Markov chain is said to be **irreducible** if every pair (i, j) communicate.*

An irreducible Markov chain has the property that it is possible to move from any state to any other state in a countable number of periods. Note that we do not require that such a movement is possible in one step, so $\pi_{ij} = 0$ is permitted.

Definition 2 *The **period** of state i is given by*

$$k = \gcd \{n : \pi_{ii}^n > 0\}$$

where gcd *is "greatest common divisor". If $k = 1$ for all i the Markov chain is **aperiodic**.*

Essentially, a state has a period k if any return to it occurs only in multiples of k steps.

For a Markov chain that is both irreducible and aperiodic, Π possesses only one eigenvalue of modulus 1 (that is has one such eigenvalue is a consequence of the Perron-Frobenius theorem). It also does not possess any **absorbing states** (a state i^* such that $\pi_{i^*i^*} = 1$) or **transient states** (a state i^* such that $\pi_{i^*j} = 0 \ \forall j \neq i^*$). If these two conditions are satisfied, then there exists a set of probabilities $[\pi_i^*]_{i=1,n}$ that define the unconditional probability of being in state i. This vector solves the eigenvector equation

$$\Pi\pi^* = \pi^*,$$

so that the invariant distribution is the eigenvector associated with the single unitary eigenvalue after the eigenvectors are orthonormalized; given that there is only one such eigenvector, the invariant distribution is unique. An alternative method is to compute

$$\Pi^* = \lim_{t \to \infty} \Pi^t$$

which produces a matrix with identical columns, each of which is the invariant distribution. Under the assumptions of irreducibility and aperiodicity, this limit exists and (of course) is unique. We can interpret the invariant distribution in two ways: (i) π_i^* is the unconditional probability that the chain is currently in state i; (ii) π_i^* is the probability that the chain will be in state i in t steps as t diverges to ∞. For the invariant distribution, it follows that

$$\pi_j^* = \sum\nolimits_i \pi_{ij} \pi_i^*$$

for each j.

Taking conditional expectations of Markov chains just requires finite summation:

$$E\left[z' | z = z_i\right] = \sum\nolimits_j \pi_{ij} z_j.$$

Other moments are straightforward to compute as well, such as the conditional variance

$$\text{var}\left(z' | z = z_i\right) = \sum\nolimits_j \pi_{ij} z_j^2 - \left(\sum\nolimits_j \pi_{ij} z_j\right)^2.$$

The long-run mean and variance can be obtained using the invariant distribution:

$$\begin{aligned} E\left[z\right] &= \sum\nolimits_i \pi_i^* z_i \\ \text{var}\left(z\right) &= \sum\nolimits_i \pi_i^* z_i^2 - \left(\sum\nolimits_i \pi_i^* z_i\right)^2. \end{aligned}$$

We can also calculate statistics like the "return time," the expected number of periods before the chain revisits state i:

$$\begin{aligned} m_i &= E\left[\inf\left\{n > 1 : z^n = z_i | z = z_i\right\}\right] \\ &= \frac{1}{\pi_i^*}. \end{aligned}$$

Another statistic of interest is the "mobility", which provides a measure of how quickly you move through the states of the chain. The typical measure is the ratio of the second largest non-unit eigenvalue to the largest; let the eigenvalues be ordered $1 > |\lambda_1| > |\lambda_2| ...$, so that the measure of mobility is

$$\frac{\lambda_2}{\lambda_1}.$$

The larger this number, the more mobile the chain. Alternatively, we can look at the "average mixing time", which is

$$\min_t \left\{\pi_{ij}^t - \pi_j^*\right\} \leq \frac{1}{4} \ \forall i, j;$$

that is, how many periods ahead do we need to look before we are sufficiently close to the stationary distribution for state j no matter which state i we start in.

1.1 Rouwenhorst's Method of Approximation

Suppose that

$$z' = \rho z + \sigma \epsilon'$$

with $\epsilon' \sim N(0,1)$. Define

$$\sigma_z^2 = \frac{\sigma^2}{1-\rho^2}$$

as the unconditional variance of z.

In keeping with finite-state dynamic programming, it is often convenient to "discretize" this process as a finite-state Markov chain. The current "state of the art" is Rouwenhorst's method, which recently was extended by Kopekcy and Suen. Let

$$p = q = \frac{1+\rho}{2};$$

$p = q$ is a consequence of the AR process being symmetric. The N state transition matrix can be constructed using the recursion

1. For $N = 2$, set
$$\Pi_2 = \begin{bmatrix} p & 1-p \\ 1-q & q \end{bmatrix}.$$

2. For $N \geq 3$, construct
$$p \begin{bmatrix} \Pi_{N-1} & \mathbf{0}_N \\ \mathbf{0}_N^T & 0 \end{bmatrix} + (1-p) \begin{bmatrix} \mathbf{0}_N & \Pi_{N-1} \\ 0 & \mathbf{0}_N^T \end{bmatrix} + (1-q) \begin{bmatrix} \mathbf{0}_N^T & 0 \\ \Pi_{N-1} & \mathbf{0}_N \end{bmatrix} + q \begin{bmatrix} 0 & \mathbf{0}_N^T \\ \mathbf{0}_N & \Pi_{N-1} \end{bmatrix}.$$

3. Divide all rows except the top and bottom by 2 to obtain Π_N.

The realizations $\{z_1, ..., z_N\}$ are set to be evenly spaced between $z_1 = -\psi$ and $z_N = \psi$ with $\psi = \sigma_z \sqrt{N-1}$.

This method was extended by Galindev and Lkhagvasuren to approximate VARs:

$$\begin{bmatrix} z_1' \\ z_2' \end{bmatrix} = \begin{bmatrix} \rho_{11} & \rho_{12} \\ \rho_{21} & \rho_{22} \end{bmatrix} \begin{bmatrix} z_1 \\ z_2 \end{bmatrix} + \begin{bmatrix} \sigma_{11} & \sigma_{12} \\ \sigma_{12} & \sigma_{22} \end{bmatrix} \begin{bmatrix} \epsilon_1' \\ \epsilon_2' \end{bmatrix}.$$

The basic idea is to orthogonalize the two rows (say, by using a spectral decomposition

$$\begin{bmatrix} \rho_{11} & \rho_{12} \\ \rho_{21} & \rho_{22} \end{bmatrix} = PDP^{-1}$$

where D is the diagonal matrix of eigenvalues and P is the corresponding matrix of eigenvectors), using Rouwenhorst's method on each row separately, then undoing the orthogonalization. In effect, we take a process with N states per

row and convert it to a joint process with N^2 states. Since their code handles the one-dimensional case we will use it.

There are other methods in use, in particular the Tauchen and Tauchen-Hussey methods, but these seem to be inferior to Rouwenhorst's method with respect to matching conditional and unconditional first and second moments. How to approximate nonlinear and nonnormal processes is an open area of research.

1.2 Numerical Integration

However, for some applications Markov chain approximations are not desireable; as an example, Gaussian processes with hidden states imply the optimal learning mechanism is the Kalman filter, which means we only need track conditional means (and sometimes variances), but a Markov chain with N states means we must track $N-1$ probabilities. The idea will be to approximate the expectation of some function using a finite sum constructed from a particular class of orthogonal polynomials – this procedure is called quadrature.

For Gaussian random variables these are the Hermite polynomials:

Table 1
Orthogonal Families

Family	$w(x)$	$[a,b]$	Definition
Hermite	e^{-x^2}	$(-\infty,\infty)$	$H_n(x)=(-1)^n e^{x^2}\frac{d^n}{dx^n}\left(e^{-x^2}\right)$

(1)

The recursion is

$$\begin{aligned} H_{n+1}(x) &= 2xH_n(x)-2nH_{n-1}(x) \\ H_0(x) &= 1 \\ H_1(x) &= 2x \\ H_2(x) &= 4x^2-2. \end{aligned}$$

Notice that the weighting function e^{-x^2} is proportional to the density of a Gaussian random variable with mean 0 and variance $\frac{1}{2}$.

The quadrature rule is given by

$$\int_{-\infty}^{\infty} f(x)\,e^{-x^2}dx=\sum_{i=1}^{n}\omega_i f(x_i)+\frac{n!\sqrt{\pi}}{2^n}\frac{f^{(2n)}(\xi)}{(2n)!};$$

the weights and nodes are given in the following table (the nodes are $\pm x_i$ with

the same weights on both), and the last term is ignored.

Table 1
Gauss-Hermite Quadrature

n	x_i	ω_i	n	x_i	ω_i
2	0.7071067811	0.8862269254	7	2.651961356	0.000971781245
				1.673551628	0.05451558281
				0.8162878828	0.4256072526
3	1.224744871	0.2954089751		0.0000000000	0.8102646175
	0.000000000	1.1816359	10	3.436159118	0.0000076404
4	1.650680123	0.08131283544		2.532731674	0.001343645746
	0.5246476232	0.8049140900		1.756683649	0.03387439445
5	2.020182870	0.01995324205		1.036610829	0.2401386110
	0.9585724646	0.3936193231		0.3429013272	0.6108626337
	0.0000000000	0.9453087204			

Code exists to generate the weights and nodes for other values of n (note that you can do quadrature with $n = 1$, which has node $x_1 = 0$ and weight $\omega_1 = 1$, which amounts to replacing the expectation with the value at the mean). One caveat: be sure to normalize the weights so that they add exactly to one by dividing by their sum.

To apply Gauss-Hermite quadrature to expectations of functions of nonstandard normal random variables, we simply use the linear change of variables

$$x = \frac{y - \mu}{\sigma\sqrt{2}}$$

and compute

$$\begin{aligned} E\left[f\left(Y\right)\right] &= \int_{-\infty}^{\infty} f\left(y\right) e^{-(y-\mu)^2/\left(2\sigma^2\right)} dy \\ &\approx \sum_{i=1}^{n} \omega_i f\left(\sigma\sqrt{2}x_i + \mu\right). \end{aligned}$$

Formulas exist for truncated normals.

As an example, let's take our AR(1) process:

$$\begin{aligned} z' &= \rho z + \sigma\epsilon' \\ \epsilon' &\sim N\left(0,1\right) \end{aligned}$$

and consider the conditional expectation

$$E\left[f\left(z'\right)|z\right];$$

this expression would be the one from a dynamic programming problem. We

therefore have

$$\begin{aligned} E\left[f\left(z'\right)|z\right] &= \int_{-\infty}^{\infty} f\left(z'\right) \exp\left(-\frac{1}{2\sigma^2}\left(z'-\rho z\right)^2\right) dz' \\ &\approx \sum_{i=1}^{n} \omega_i f\left(\sigma\sqrt{2}x_i + \rho z\right). \end{aligned}$$

In a dynamic programming context, given a set of nodes for z we would need to evaluate the value function at z' points that are clearly not going to be in the grid, so it is common to use linear interpolation (because extrapolation will be needed). In my experience this approach creates some problems, so you need to watch it carefully.

Other sets of polynomials exist for uniform (Legendre) and exponential (Laguerre) random variables. For other distributions, you will need to try various things, including using the Legendre polynomials to integrate the product of the function and the density or to convert the problem to a Gaussian by multplying and dividing by the Gaussian kernel. That is,

$$E\left[f\left(Y\right)\right] = \int_{-\infty}^{\infty} f\left(y\right) g\left(y\right) dy$$

for some density $g\left(y\right)$ can be rewritten as

$$\begin{aligned} E\left[f\left(X\right)\right] &= \int_{-\infty}^{\infty} f\left(x\right) \frac{g\left(x\right)}{\exp\left(-x^2\right)} \exp\left(-x^2\right) dx \\ &\approx \sum_{i=1}^{n} \omega_i f\left(x_i\right) \frac{g\left(x_i\right)}{\exp\left(-x_i\right)^2}. \end{aligned}$$

This transformation amounts to a form of "importance sampling" and can be dangerous if the densities get too small at the nodes.

A last resort for numerical integration is Monte Carlo or random sampling. Matlab provides routines to draw (pseudo-)random numbers from a wide array of distributions. Under very mild regularity conditions, we know that

$$\lim_{T\to\infty}\left\{\frac{1}{T}\sum_{t=1}^{T} f\left(x_t\right)\right\} = E\left[f\left(X\right)\right],$$

so if you want the expectation all you need is a large number of sampled points. This approach is very flexible, as it can handle unusual distributions and unusual supports (you simply drop any draws that fall outside the region of interest), but tends to converge very slowly (meaning you need large T). Acceleration methods exist, in particular antithetic variation, which can be illustrated easily. Suppose we are drawing a one-dimensional random number from a symmetric distribution. Antithetic variation says that if we draw x_t we compute both $f\left(x_t\right)$ and $f\left(-x_t\right)$, which guarantees that the sample is symmetric and covers the entire space more efficiently; the number of effective points is more than

twice the number of sampled points. In general you flip your sample around the median of the distribution, which can be difficult to do for some distributions, so antithetic variation is not necessarily easy to implement.