Dynamic programming with Markov chain

Sequence problem

$$\max_{\{c_t\}} \sum_{t=0}^{\infty} \beta^t u(c_t)$$

subject to

$$W_0 = \sum_{t=0}^{\infty} \frac{c_t}{(1+r)^t}$$

Recursive problem

$$\max_{c} u(c) + \beta V_{t+1}[(W_t - c)(1+r)]$$

where

$$V_{t+1} = \sum_{s=t+1}^{\infty} \beta^{s-t-1} u(c_s).$$

Denote the solution to the RP problem V_t.

Key idea: V_{t+1} is also a solution to the recursive problem and so on.

This is a divide and conquer type algorithm.

Cake eating problem

$$u(c) = \ln(c)$$

and

$$r = 0.$$

Sequential Problem:

$$\max_{\{c_t\}} \sum_{t=0}^{\infty} \beta^t \ln c_t$$

s.t.

$$W_0 \geq \sum_{t=0}^{\infty} c_t.$$

FOC:

$$\beta^t \frac{1}{c_t} = \Lambda \cdot 1.$$

$$c_t = \Lambda^{-1} \beta^t$$

to pin down Λ,

$$W_0 = \Lambda^{-1} \sum_{t=0}^{\infty} \beta^t = \frac{1}{\Lambda(1-\beta)}$$

so that

$$\Lambda = \frac{1}{W_0(1-\beta)}$$

and

$$c_t = W_0(1-\beta)\beta^t$$

Recursive Problem: Find some value function $V()$ such that

$$V(W_t) = \max_c \ln c + \beta V(W_t - c) \tag{*}$$

Note:

1. eq (*) is called the **Bellman equation.**
2. it is a **functional equation**: we are looking for a $V()$ function s.t. (*) holds for every possible value of W.
3. under some conditions, $V()$ is the same as the solution to the SP.

Break until 14:29

(2) is a complication, but we use 3.5 strategies to overcome it.

Guess and verify (method of undetermined coefficients)

One guess

$$V(x) = a + b \ln x$$

Suppose this is true, my maximization problem:

$$\max_c \ln c + \beta[a + b\ln(W - c)]$$

FOC:

$$\frac{1}{c} + \beta b(-1)\frac{1}{W-c} = 0$$

or

$$c = (W - c)/(\beta b)$$

$$\frac{W-c}{c} = W/c - 1 = \beta b$$

$$c = W\frac{1}{1+\beta b}$$

Substitute this in

$$V(W_t) = \max_c \ln c + \beta V(W_t - c) \tag{*}$$

to get

$$a + b \ln W = [\ln W - \ln(1 + \beta b)] + \beta[a + b \ln(W - c)]$$

but $W - c = W(1 - 1/(1 + \beta b))$.

$$a + b \ln W = [\ln W - \ln(1 + \beta b)] + \beta[a + b \ln W + b \ln(\beta b/(1 + \beta b))] \tag{**}$$

Do a and b exist such that (**) is true for **every possible value** of W? I can ensure this if all "polynomial" terms of W are equal. For $\ln W$:

$$b = 1 + \beta b$$

this holds iff

$$b = 1/(1 - \beta).$$

For "constant terms," not depending on W,

$$a = -\ln(1 + \beta b) + \beta a + \beta b \ln(\beta b/(1 + \beta b))$$

Yes, we can solve this for a.

Value function iteration

Take the Bellman as an operator, not an equation. EQ:

$$V(W_t) = \max_c \ln c + \beta V(W_t - c) \tag{*}$$

operator:

$$Tv := (Tv)(x) = \max_c \ln c + \beta v(x - c)$$

Policy function iteration

What is a policy function?

$$c(x) := \arg\max_c u(c) + \beta V(x - c)$$

We want

$$Tc := (Tc)(x) = \arg\max_c \ln c + \beta V(x - c)$$

we need to compute V for a given $c(x)$. Given this rule, and starting from x_0, I can compute $x_1 = x_0 - c(x_0)$ and $x_2 = x_1 - c(x_1)$, etc. Plug this into utility, to get PDV:

$$V_{(c)}(x_0) = \sum_{t=0}^{\infty} \beta^t \ln c(x_t)$$

Plan:

1. Get candidate $c(x)$
2. Compute the law of motion for x: $x_0, x_1, \ldots$
3. Compute PDV to get V
4. Solve Bellman with this V to get new $c(x)$

Parametric policy function iteration (1+3)

In (1) and (4), you are constrained to use a parametric $c(x)$.

Linear policy

Suppose

$$c(x) = ax$$

with $a \in (0, 1)$.

Step 2

Then $x_1 = (1-a)x_0$, $x_t = (1-a)^t x_0$, ... and

$$c_t = c(x_t) = a(1-a)^t x_0$$

Step 3

with log utility,

$$u_t = \ln c_t = \ln a + t \ln(1-a) + \ln x_0$$

Add up PDV,

$$V(x_0) = \ldots + \ln x_0 \sum_{t=0}^{\infty} \beta^t = \ldots + \frac{\ln x_0}{1-\beta}$$

Step 4

FOC for "optimal" c, relying on the fact that

$$\frac{\partial \ldots}{\partial x_0} = 0$$

$$\frac{1}{c} + \beta \frac{\partial V}{\partial x} \frac{\partial x}{\partial c} = 0$$

$$\frac{1}{c} = \frac{\beta}{1-\beta} \frac{1}{x-c}$$

In **this case**, a did not show up, so we already have a solution in one iteration.

Break until 14:28

Dynamic programming with Markov chains

Suppose x_t follows some some Markov chain with x_0 and transition matrix $\mathbf{P}$.

General Bellman

$$V(x_t) = \max_c E[u(x_t, c) + \beta V(x_{t+1})]$$

or with deterministic utility,

$$V(x_t) = \max_c u(x_t, c) + \beta EV(x_{t+1})$$

We know how to forecast a MC, getting PMF $\pi_{t+1} = \mathbf{Pr}(x_{t+1} = k | x_t = x_t)$.

$$E[V(x_{t+1})] = \sum_{k=1}^{K} \pi_{k,t+1} v_k = \pi'_{t+1} \mathbf{v}$$

Without optimization

$$V(x) := \mathbf{v}$$

is a K by 1 vector. Bellman can be written in vector notation,

$$\mathbf{v} = \mathbf{u} + \beta \mathbf{P} \mathbf{v}.$$

For state 1,

$$v_1 = u_1 + \beta \mathbf{P}_{1.} \mathbf{v}$$

With two states,

$$v_e = w + \beta[\pi_{11} v_e + \pi_{12} v_u]$$
$$v_u = b + \beta[\pi_{21} v_e + \pi_{22} v_u]$$

We can express the value as a function of utility, transition probabilities, and the discount factor,

$$\mathbf{v} = (\mathbf{I} - \beta \mathbf{P})^{-1} \mathbf{u}.$$

With optimization

$$\mathbf{v} = \max_c \mathbf{u}(c) + \beta \mathbf{P}(c) \mathbf{v}$$

1. give me a c
2. compute value given c

$$\mathbf{v} = (\mathbf{I} - \beta \mathbf{P}_c)^{-1} \mathbf{u}_c$$

Endogenous job search

Suppose firing probability fixed δ. Job finding prb is $\Lambda(c)$ for $c > 0$.

$$\Lambda(c) = \frac{\lambda c}{1 + \lambda c}$$

Bellman for unemployed state:

$$v_u = \max_c b - c + \beta[\Lambda(c)v_e + (1 - \Lambda(c))v_u]$$

$$v_u = \max_c b - c + \beta \frac{\lambda c v_e + v_u}{1 + \lambda c}$$

FOC:

$$-1 + \beta \frac{\lambda v_e(1 + \lambda c) - (\lambda c v_e + v_u)\lambda}{(1 + \lambda c)^2} = 0$$

$$\beta[\lambda v_e(1 + \lambda c) - (\lambda c v_e + v_u)\lambda] = (1 + \lambda c)^2$$

$$\beta\lambda[v_e - v_u] = (1 + \lambda c)^2$$

$$\frac{1}{1 + \lambda c} = \sqrt{\frac{1}{\beta\lambda(v_e - v_u)}}$$

for this to be a prob, we need $\beta\lambda(v_e - v_u) > 1$.

Check second-order condition

Take the second derivative of the objective function wrt c,

$$\Lambda''(c)(v_e - v_u) < 0?$$

In optiumum, the value gap will be positive, so we just have to check the second derivative of Λ. This will be negative if and only if $\Lambda(c) > 0.5$. Otherwise, the logistic function is convex and the FOC characterizes a local *minimum*, not maximum. So we have to change our code to check if the predicted employment probability is greater than 0.5.