Repeated Matching Assignment for Econ 504
Fall 2019

[redacted]

[redacted]

[redacted]

[redacted]

1 Model

We will study an infinite horizon repeated matching game. Let there be a continuum of workers and a continuum of jobs. Let there be a unit (1) mass of both workers and jobs. Each period, workers and jobs match. After matching, worker and job states evolve and the next period begins and new matches form. Workers and jobs are forward looking. Assume every worker and job are matched; there are no unmatched workers or vacant jobs. The matching game has transferable utility, so matched agents exchange monetary transfers (wages) and the transfers enter agent payoffs additively separably.

Let the binary $x \in \{1, 2\}$ be the **type of a worker** and the scalar $y \in \{1, 2\}$ the type of a job. Both workers and jobs are forward looking, as in Rust. If a worker x matches to a job y, the transition rule for states is $P(x' \mid x, y) = \Pr(x' \mid x, y)$. Therefore, x can be called the **state of a worke**r in addition to being called the type of a worker. Likewise, $Q(y' \mid x, y) = \Pr(y' \mid x, y)$ is the transition rule for the state y of the job.

Each period, the **aggregate state** of this matching market is the vector (a_1, b_1), where a_1 is the mass of workers with $x = 1$ and b_1 is the mass of jobs with $y = 1$. The mass of workers of $x = 2$ is $a_2 = 1 - a_1$, so a_2 does not need to be tracked separately in the aggregate state. The same goes for b_2 .

The **matching** each period is given by $\mu = (\mu_{x,y})_{x,y}$, where each scalar $\mu_{x,y}$ is the mass of the matching between workers of type x and jobs of type y. The following constraints are required for the matching to be feasible:

$$\sum_y \mu_{x,u} = a_x,\ x = 1, 2$$
$$\sum_x \mu_{x,y} = b_y,\ y = 1, 2.$$

As there is a unit mass of workers and a unit mass of jobs, the sum of all the $\mu_{x,y}$ will be 1.

The **production** to a match between x and y is given by the term $\phi_{x,y}$.

To skip to the model solution, the model described here implies that the matching in a competitive equilibrium solves the social planner's problem of maximizing the present discounted value of economywide output. Written recursively, the **Bellman equation for the social planner** is

$$V(a_1, b_1) = \max_{\mu \in F(a_1,b_1)} \left\{ \sum_{x,y} \mu_{x,y}\phi_{x,y} + \delta V(a_1'(\mu), b_1'(\mu)) \right\}.,$$

where $F(a_1, b_1)$ is the set of feasible $\mu's$ given the aggregate state (a_1, b_1). See the constraints for feasibility above. Given the matching μ, individual workers and jobs have stochastic state evolutions. However, the aggregate state (a_1, b_1) is not stochastic as each agent is small relative to the market size.

Solving Bellman's equation is equivalent to solving a dynamic program with two continuous state variables (a_1, b_1) and a continuous action μ. The equilibrium matching is then $\mu(a_1, b_1)$. There is a unique equilibrium matching.

There also exists a **constant aggregate state** $(a_1^\star, b_1^\star)$, where the equilibrium matching $\mu(a_1, b_1)$ is such that $(a_1'(\mu(a_1^\star, b_1^\star)), b_1'(\mu(a_1^\star, b_1^\star))) = (a_1^\star, b_1^\star)$. In other words, once the matching market enters the constant aggregate state $(a_1^\star, b_1^\star)$, it remains there forever. A constant aggregate state Bellman equation is

$$V^\star = \max_{\mu \in F\left(a_1^\star, b_1^\star\right)} \left\{ \sum_{x,y} \mu_{x,y}\phi_{x,y} + \delta V^\star \right\},$$

where $V^\star$ is just a scalar. The goal is to find the constant aggregate state $(a_1^\star, b_1^\star)$.

2 Parameter Values

Let $\phi_{x,,y} = x \cdot y$.

Let $P(1 \mid 1, 1) = 0.6$, $P(1 \mid 2, 1) = 0.4$, $P(1 \mid 1, 2) = 0.5$ and $P(1 \mid 2, 2) = 0.35$. You can derive $P(2 \mid x, y)$ from the above.

Let $Q(1 \mid 1,1) = 0.65$, $Q(1 \mid 2,1) = 0.55$, $Q(1 \mid 1,2) = 0.45$ and $Q(1 \mid 2,2) = 0.3$.
Let $\delta = 0.95$.

3 Numerical Methods

For the full equilibrium (not the constant aggregate state), you should use a **two-dimensional** Chebyshev polynomial approximation to the value function. This is described in the lecture notes, following an algorithm in Judd's book. Use as many polynomials and nodes as needed. You can use value function approximation or linear programming to solve for the equilibrium.

You can use IPOPT for the nested step of computing the optimal matching μ for each state.

For the constant aggregate state, you might use IPOPT for both the inner loop (solving for $V^\star$) and for the outer loop solving for the constant aggregate state.

4 Questions

1. Use algebra to compute $(a_1'(\mu), b_1'(\mu))$, how next period's aggregate depends on this period's matching μ.

2. Describe a numerical algorithm to compute a constant aggregate state.

3. Compute the constant aggregate state and report the matching pattern μat the constant aggregate state. Explain in English why the numbers in the parameterizations lead to this constant aggregate state. This English portion is about thinking through the economics of the example to understand why the parameter values lead to this constant aggregate state.

4. Compute the full solution to the model by solving the social planner's Bellman equation for all states, not just the constant aggregate state.

   (a) Use different numbers of Chebyshev polynomials and Chebyshev nodes. Show that the solutions are settling down with more basis functions and nodes so that the continuation value V is likely accurate to around four decimal places. Document your experiments

   (b) Describe the solution to the social planner's continuation value function as clearly as possible. Pretend this is an academic research paper and that you want to explain quantitatively and qualitatively what the continuation value function $V(a_1, b_1)$ looks like and why. Refer back to the parameter values to explain the economics of why the continuation value looks like it does.

   (c) Do the same as part b except study the equilibrium matching probabilities $\mu(a_1, b_1)$.

5 Deliverables

The main deliverable is a well-written PDF with the answers to the questions above and instructions on how to run the code. The code should run on Julia 1.2 on a Mac (my machine) without modifying the code at all. All files should be in a zip archive and uploaded to Canvas.