RL_Proyect

Carlos Soto (carlos.soto362@gmail.com)

June 2022

1 Pure exchange with two commodities and two consumers or a Edgeworth box economy

This section is based on Mas-Colell et al., 1995.
Pure exchange means no production. The simplest economy with profitable exchange with no production is the one with two commodities and two consumers. No production for this economy means that, the total amount of good l that each of the consumers posses is less or equal than the initial amount of the good, with equality when there is no consumption and no waste. Denoting by $\{A, B\}$ the consumers, x_{il} the total amount of good l that consumer i posses, w_{il} the endowments, the initial amount of good l that consumer i posses and $\bar{w}_l$ the total amount of good l, no production, no consumption and no waste is represented by the equation,

$$x_{Al} + x_{Bl} = w_{Al} + w_{Bl} = \bar{w}_l. \tag{1}$$

The two tuples $\{(x_{A1}, x_{A2}), (x_{B1}, x_{B2})\}$ are called an "Allocation", all the Allocations that obey equation (1) are called feasible allocations. Given a set of prices (p_1, p_2), each consumer can exchange their initial goods with the budget constrain

$$p_1 x_{i1} + p_2 x_{i2} = p_1 w_{i1} + p_2 w_{i2}. \tag{2}$$

The consumers are going to exchange their goods in order to maximise their utility function $u_i(x_{i1}, x_{i2})$. This is a function that describes the level of happiness that the consumer fills given an allocation.

All the feasible allocations in this economy can be represented in the *Edgeworth box* figure 1, where the amount of good 1 that consumer A posses is represented in the x axis, and the amount of good 2 that consumer A posses in the y axis in the usual form, while the same applies for consumer B but with the origin in the opposite corner.

Given a utility function $u_A(x_{A1}, x_{A2})$, a set of prices (p_1, p_2) and an initial endowment (w_{A1}, w_{A2}), there exist a feasible allocation such that the consumer A maximise his utility function subject to his budget constrain. For an arbitrary set of prices, this allocation is different from the allocation that maximises $u_B(x_{B1}, x_{B2})$. A Walraisan or competitive equilibrium for the Edgeworth box

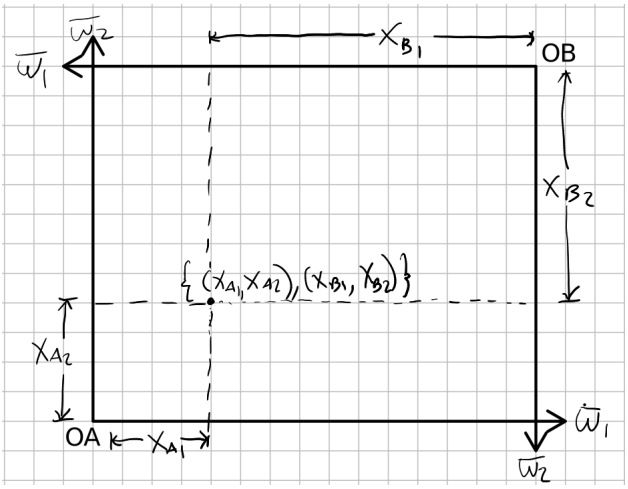

Figure 1: Representation of the Edgeworth box, where one feasible allocation has been remarked.

economy is a price vector p^* and an Allocation, such that both utility functions are maximised under their respective budget constrains.

2 Translating the Edgeworth box economy to a Reinforcement Learning problem

The Edgeworth box economy can be understood as a game, where two consumers exchange two goods, in order to maximise their utility function. Under complete knowledge of the utility functions, total amount of each good and initial endowments, it is possible to find the competitive equilibrium, so, under rationality assumptions, we could assume that the consumers would agree on the price of the goods as the price of this equilibrium, and exchange the necessary amount in order for they to maximise their utility function. However, this is far away from reality. Consumers rarely know how is the total amount of each good, the utility function of the other consumers and in most of the cases, they don't know their own utility function. This can be translated as a reinforcement learning (RL) problem with no knowledge of the world.

Elements of the RL problem

World

The world consist of the utility functions of each consumer $u_A(x_{A1}, x_{A2})$, $u_B(x_{B1}, x_{B2})$, and the set of all possible allocations and prices $\{(x_{A1}, x_{A2}), (x_{B1}, x_{B2}), (p1, p2)\}$ such that equation (1) is obeyed.

To be able to work with the methods of reinforcement learning, this world is going to be discretized in the following manner,

for $n_1, n_2 \in \mathbf{N}$, defined $\eta_1 = \bar{w}_1/n_1$, $\eta_2 = \bar{w}_2/n_2$. Then, $p_1 = 1$, $p_2 \in \{\eta_2/\eta_1, 2\eta_2/\eta_1, \ldots, \bar{w}_2/\eta_1, \eta_2/2\eta_1, \ldots, \eta_2/\bar{w}_1\}$, $x_{i1} \in \{\bar{w}_1, \bar{w}_1 - \eta_1, \ldots, 0\}$ and $x_{i2} \in \{\bar{w}_2, \bar{w}_2 - \eta_2, \ldots, 0\}$.

This discretisation transforms the Edgeworth box in a three dimensional greed world with $(n_1+1)(n_2+1)(n_1+n_2+1)$ points. The price p_1 can be set to one because what determines the competitive equilibrium is the ratio between the two prices.

In this world, there is only one point where equilibrium can be reached.

Actions

Without knowledge of their utility function, each consumer can only see the reward (the value of the utility function) after they have change the allocation. For doing so, they need to exchange with the other consumer. In order to work with this world in a similar manner than the greed world, each consumer can decide to buy or sell moving their allocation horizontally or vertically, but the movement is done only if both of the consumers agree on this decision. This agreement will be done in the following way:

If a consumer decide to sell an amount of good 2 according to his policy in the present point, he needs also to decide if he want to sell at the same, increase or decrease price, moving the state up or down in the price direction. Next, the other customer also needs to decide to buy at the new price, only if both of them agree on this, then they exchange giving the equivalent amount of good 1 in exchange of the amount of good 2.

If both customers want to buy the good 2, nothing happens. If both customers want to sell the good 2, nothing happens.

Rewards

The rewards would be the value of their utility function after the action has been taken. The utility functions don't need to be the same for both customers. They are assumed to be concave functions of the goods.

Possible simplification

In case the dynamics don't manage to reach the competitive equilibrium because the learning algorithm is not well fitted for this problem or the problem is to big, one possible simplification could be to make the prices fix. The fixation of the prices would make that the competitive equilibrium may not exist, so, to ensure the existence, the problem could be set to $u_A(x_{A1}, x_{A2}) = u_B(x_{B1}, x_{B2})$, and the initial budgets to be also equal, such that the competitive equilibrium would be $\{(1/2, 1/2), (1/2, 1/2), (1, 1)\}$ by symmetry.

3 Algorithm

Mathematical digression

Each player is going to use an actor-critic algorithm, which is based on Natural Policy Gradient using an approximation of the value function as a baseline in order to learn the best strategy. Calling $\Pi_i(a, s)$ the policy of player i, the probability that player i makes action a given that she is in the state s, then the expected reward for player i is

$$\begin{aligned} J_i &= \mathbf{E}[\sum_{t=0}^{\infty} \gamma^t R_{t+1}^{(i)}] \\ &= \sum_{s,\hat{s},a_1,a_2,t} \gamma^t P(S_t = s) P(\hat{s}|s,a) \Pi_1(a_1|s) \Pi_2(a_2|s) u_i(s, a_1, a_2, \hat{s}) \end{aligned} \tag{3}$$

where γ is a discount factor over time, $R_{t+1}^{(i)}$ is the value of the utility function of player i at time $t+1$, and P stands for probability.

The Natural Policy Gradient algorithm consist on maximising the function $J_i(\theta)$ by computing the sequence

$$\theta^{(n+1)} = \theta^{(n)} + \alpha \tilde{\nabla}_{\theta^{(n)}} J_i(\theta^{(n)}) \tag{4}$$

with $\tilde{\nabla}_\theta = F^{-1} \nabla_\theta$ the natural gradient, and F the Fisher Information Matrix with elements

$$F_{\theta_a, \theta_b} = \sum_{c,s} \Pi_i(c|s) \frac{\partial \log \Pi_i(c|s)}{\partial \theta_a} \frac{\partial \log \Pi_i(c|s)}{\partial \theta_b}. \tag{5}$$

This sequence, for suitable choice of α, any given $\theta^{(0)}$ and concavity assumptions, would converge to

$$\theta^* = \arg\max_\theta (J_i(\theta)) \tag{6}$$

It can be proven that, if the policy is parameterized by parameters θ,

$$\tilde{\nabla}_\theta J_i = \sum_{t=0}^{\infty} \gamma^t \mathbf{E}[\left(\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \right) \tilde{\nabla}_\theta \log \Pi_i(A_t|S_t)], \tag{7}$$

where $A_t \in \mathbf{A}$ the action taken on time t, and $S_t \in \mathbf{S}$ the state on time t, then, by choosing the soft-max parametrization,

$$\Pi_i(a, s) = \frac{e^{\theta_{a,s}}}{\sum_{a^*} e^{\theta_{a^*,s}}}, \tag{8}$$

$$\tilde{\nabla}_\theta J_i = \sum_{t=0}^{\infty} \gamma^t \mathbf{E}[\left(\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \right) \mathbf{I}(A_t, S_t) / \Pi_i(A_t|S_t)] \tag{9}$$

with $\mathbf{I}(x)$ a vector of zeros for every action and state that has not been taken and one for the action and state in time t.

In order to decrease the variance and make the algorithm faster at learning, the Value Function is introduced as a baseline,

$$V_i(s) = \mathbf{E}[\sum_{t=0}^{\infty} \gamma^t R_{t+1}^{(i)} | S_0 = s] \tag{10}$$

which obeys the Bellman's equation

$$V(s) = \sum_{\hat{s},a} \Pi_i^{(t)}(a|\hat{s}) P_t(\hat{s}|s,a)[u_i(s,a,\hat{s}) + \gamma V(\hat{s})]. \tag{11}$$

and the relation

$$\begin{aligned} &Q_i(s,a) = \mathbf{E}[\sum_{t=0}^{\infty} \gamma^t R_{t+1}^{(i)} | S_0 = s, A_0 = a] \Rightarrow \\ &V(s) = \sum_a \Pi_i^{(t)}(a|s) Q_i(s,a) \Rightarrow \\ &Q(a,s) = \sum_{\hat{s}} P_t(\hat{s}|s,a)[u_i(s,a,\hat{s}) + \gamma V(\hat{s})]. \end{aligned} \tag{12}$$

Using the equation (12) in equation (9) and adding the Value Function as baseline, the resulting natural gradient is

$$\tilde{\nabla}_\theta J_i = \mathbf{E}[\sum_{t=0}^{\infty} \gamma^t u_i(S_{t+1}, A_t, A_{t+1}) + \gamma V(S_{t+1}) - V(S_t)) \mathbf{I}(A_t, S_t) / \Pi_i(A_t|S_t)]. \tag{13}$$

Dropping the Expected value in the previous equation, equation (13) and the equation (4) give the ingredients for a stochastic policy gradient, which would converge for a suitable decreasing learning rate $\alpha = \alpha_t$. In order to reduce the variance and speed the algorithm, a bias is introduced by substituting the infinite sum with just one term of it. Because the Value function is also unknown, it also has to be learned. By choosing a parametric $\hat{V}(s,w)$, the intention is to minimize the distance between $V(s)$ and $\hat{V}(s,w)$, so defining the value error as

$$\bar{VE} = \sum_s \sum_{t=0}^{\infty} \gamma^t P(S_t = s)(V(s) - \hat{V}(s,w))^2, \tag{14}$$

minimizing this value error by gradient decent,

$$\nabla(\bar{VE}) = -2 \sum_s \sum_{t=0}^{\infty} \gamma^t P(S_t = s)(V(s) - \hat{V}(s,w)) \nabla \hat{V}(s,w) \tag{15}$$

which is approximated as

$$\begin{aligned}\nabla(\bar{VE})_{aprox} &= -2\sum_{s}\sum_{t=0}^{\infty}\gamma^t P(S_t=s)(u_i(S_{t+1})+\gamma\hat{V}(S_{t+1},w)-\hat{V}(S_t,w))\nabla\hat{V}(S_t,w)\\ &= -2\sum_{t=0}^{\infty}\mathbf{E}[\gamma^t(u_i(S_{t+1})+\gamma\hat{V}(S_{t+1},w)-\hat{V}(S_t,w))\nabla\hat{V}(S_t,w)]\end{aligned} \tag{16}$$

Noticing that one state can be represented by three coordinates, $S=(S_1,S_2,S_3)$, then $\hat{V}(S,w)$ is chosen to be

$$\hat{V}(S,w)=w_1S_1+w_2S_2+w_3S_3, \tag{17}$$

so, the final expression for the gradient of the value error that will be used is

$$\nabla(\bar{VE})_{aprox} = -2\sum_{t=0}^{\infty}\mathbf{E}[\gamma^t(u_i(S_{t+1})+\gamma\hat{V}(S_{t+1},w)-\hat{V}(S_t,w))S_t] \tag{18}$$

Again, dropping the expected value give reach to a Stochastic gradient decent, and using only one term of the infinite sum increase the bias but can reduce the variance and speed up the algorithm.

Finally, the algorithm: Actor-Critic

- Input $\Pi_1(a|s)$, $\Pi_2(a|s)$, $\hat{V}_1(s,w_1)$, $\hat{V}_2(s,w_2)$, α_θ and α_w.
- Initialize θ_1, θ_2, w_1 and w_2.
- Loop over episodes:
  - Initialize a state s.
  - Loop over time:
    - Pick $A=(a_1,a_2)$ according to the policy $\Pi(A|s)=\Pi_1(a_1|s)\Pi_2(a_2|s)$.
    - Observe $\hat{s},u_1(\hat{s}),u_2(\hat{s})$.
    - Compute the temporal difference error for each player $\delta_i=u_i(\hat{s})+\gamma\hat{V}_i(\hat{s},w)-\hat{V}_i(s,w)$.
    - Modify the parameters, $w_i=w_i+\alpha_w\delta_i s$, $\theta_i=\theta_i+\alpha_\theta\delta_i\mathbf{I}(a,s)/\Pi_i(a|s)$, $s=\hat{s}$.

References

Mas-Colell, A., Whinston, M. D., & Green, J. R. (1995). *Microeconomic theory.* New York, Oxford University Press.