The AI economist: Improving Equality and Productivity with AI-Driven Tax Policies

Stephan Zheng, Alexander Trott, Sunil Srinivasa, Nikhil Naik, Melvin Gruesbeck, David C. Parkes, and Richard Socher, 2020, mimeo.

Presenter: Yoji Tomita

RL-GT ゼミ June 17, 2021

Table of Contents

1. Introduction

・イントロダクション

2. Economic Simulations: Learning in Gather-and-Build Games

- Economic environment について.
- まずは税の無い設定 (”free-market”) で説明する.

2.1 Notation and Preliminaries

- Partial-observable multi-agent Markov Games(MGs): $(S, A, r, \mathscr{T}, \gamma, o, \mathscr{I})$
  - S : 状態空間 (state space)
  - A : 行動空間 (action space)
  - $r_{i,t}$: 報酬関数 (reward function)
  - $\mathscr{T}$: 遷移関数 (transition function) $s_{t+1} \sim \mathscr{T}(\cdot \mid s_t, \boldsymbol{a}_t)$
  - γ : 割引因子 (discount factor)
  - $o_{i,t}$: 観測 (observation)
  - time step $t = 0, 1, \ldots, H$.

- Agents' policy : $\pi_i(\cdot \mid o_{i,t}, h_{i,t}; \theta_i)$
  - $h_{i,t}$: hidden state（自分の私的情報と, 過去の history）
  - θ_i : policy の parameter
  - エージェント i は次の最大化問題を得く policy を求める:

$$\max_{\theta_i} \mathbb{E}_{a_i \sim \pi_i, \boldsymbol{a}_{-i} \sim \boldsymbol{\pi}_{-i}, s' \sim \mathscr{T}} \left[\sum_t \gamma^t r_{i,t} \right]. \tag{1}$$

- データ効率性のため, すべてのエージェントは training の間パラメータ θ を共有する.
- 彼らの行動 $\pi_i(a_i \mid o_i, h_i; \theta)$ は, agent-specific observations o_i と hidden-state h_i によって異なる.

t	time
i, j, k	agent indices
θ, ϕ	model weights
s	state
o	observation
a	action
r	reward
π	policy
γ	discount factor
$\mathcal{T}$	state-transition, world dynamics
h	hidden state

x	endowment
x^c	coin
x^s	stone
x^w	wood
z	income
l	labor
u	utility
T	tax
τ	tax-rate
π_p	planner policy
`swf`	social welfare
ω	social welfare weight
g	social marginal welfare weight
`gini`	Gini index
`eq`	Equality index

Table 1: Notation. Subscripts are indices. Superscripts are labels.

Gather-and-Build game

- 2 次元の grid (25×25) からなる世界が舞台.
- エージェントはフィールドを歩き回り, 資源 (石と木) を集め, それらを 1 つずつ使って家を建て, また資源を coin を介してトレードする.
- 資源は空タイルに確率的に産み出される.
- エージェントは家を建てると coin が得られるが, 得られる coin は agent の skill ごとに異なる.

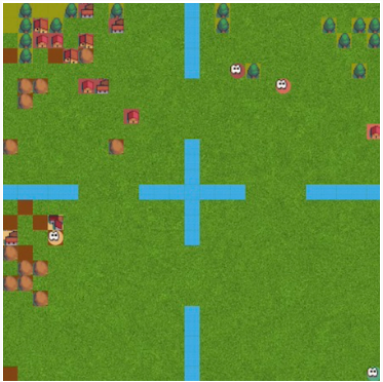
Labor and Skill.

- Agent の action(moving, gathering, trading, building) にはそれぞれ labor cost が設定されている.
- 各 time に agent がどれか 1 つ行動をとると, その labor cost がかかる.

- building skill (1 以上 3 以下) が各 agent に設定されていて, 家を建てると agent は $10\times$ skill 分の coin を得る.
- collection skill (1 以上 2 以下) もあり, 資源を拾うとこの skill 分の資源を得る (skill 1.2 の場合, 確定で 1 つ資源を得て, さらに確率 0.2 でもう 1 つ資源を得る)

Environment Scenario.

- field は水により 4 つの区域に別れている（水部分は通れない）
- 資源は空間的に集まって発生する.
- 4 agents
- buildng skills は 1.13, 1.33, 1.65, 22.2（Pareto 分布 w/ exponent $a = 4$, scale $m = 1$ の quartiles を元に設定）
- 1 episode は $H = 1000$ time steps からなる.



- Agent の utility function:

$$u_i(x_{i,t}, l_{i,t}) = \text{crra}\left(x^c_{i,t}\right) - l_{i,t}, \quad \text{where } \text{crra}(z) = \frac{z^{1-\eta} - 1}{1-\eta}, \ \eta > 0. \tag{2}$$

  - $x_{i,t} = (x^w_{i,t}, x^s_{i,t}, x^c_{i,t})$: i の保有する木・石・コイン.
  - $l_{i,t}$: 蓄積労働量.
  - η: エージェントの utility function の non-linearity をコントロールするパラメータ.
- Rational economic agent は以下の最大化を行う.

$$\forall i : \max_{\pi_i} \mathbb{E}_{a_i \sim \pi_i,\ \boldsymbol{a}_{-i} \sim \boldsymbol{\pi}_{-i}, s' \sim \mathscr{T}} \left[u_i(x_{i,0}, l_{i,0}) + \sum_{t=1}^{H} \gamma^t \underbrace{\left(u_i(x_{i,t}, l_{i,t}) - u_i(x_{i,t-1}, l_{i,t-1})\right)}_{=r_{i,t}} \right]. \tag{3}$$

Deep RL agents

- deep neural network を用いる agent policy を modelling する:

$$a_{i,t} \sim \pi(o_{i,t}^{\text{world}}, o_{i,t}^{\text{agent}}, o_{i,t}^{\text{market}}, o_{i,t}^{\text{tax}}, h_{i,t-1}; \theta)$$

  - $o_{i,t}^{\text{world}}$: 近くの状況に関する観測.
  - $o_{i,t}^{\text{agent}}$: public な agent の状況 (資源・コイン保有) と, private agent states(skill 値と labor performed)
  - $o_{i,t}^{\text{market}}$: transfer market の状況（bid, ask offer）
  - $o_{i,t}^{\text{tax}}$: tax rates

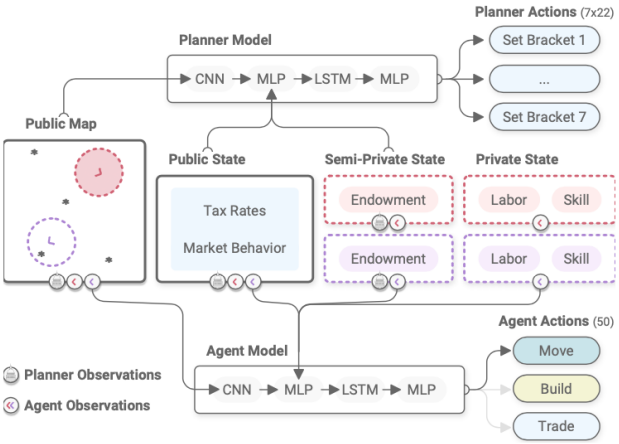
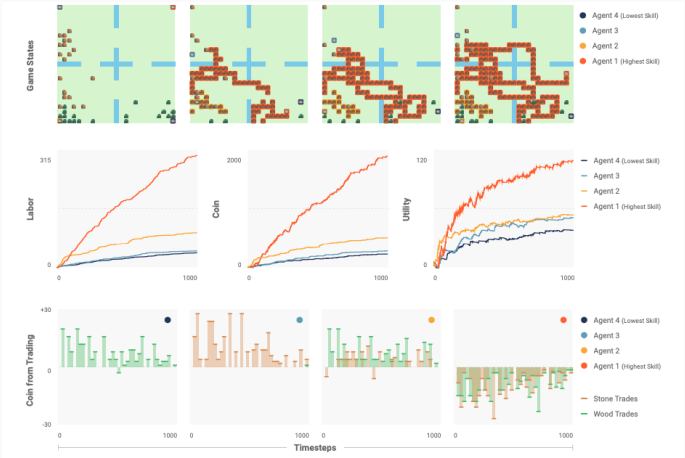


Figure 3: Schematic overview of the general network architecture used in our work. Spatial observations are processed by a stack of two convolutional layers (CNN) and flattened into a fixed-length feature vector. This feature vector is concatenated with the remaining observation inputs and the result is processed by a stack of two fully connected layers (MLP). The output is then used to update the hidden state of an LSTM and action logits are computed via a linear projection of the updated hidden state. Finally, the network computes a softmax probability layer for each action head. For the agent policy, there is a single action space and action head. For the tax policy, there is a separate action space and action head for each tax rate the tax policy controls (described below).

Emergent Behavior of AI Agents



- 左図は no-tax 下で train 後の AI agents の 1 episode の行動の例.
- low-skill agents（紺, 水）は資源を集めて market で売ることに徹している.
- high-skill agent（オレンジ）は market で資源を買って家を建てている.
- 黄色は最初は家を建ててるが, のちに資源を得る方にスイッチしている.

3. Machine Learning for Optimal Tax Policies

- 課税と再分配を行う social planner を導入する.
- Social planner は, 生産性と平等性の trade-off に直面している.
  - 無課税（free-market）では生産性は最大化されるが, 不平等.
  - 課税・再分配を行うと平等性が増すが, 生産性が落ちる.

- ここでは, free-market, US-federal, Saez framework, AI economist による social planner を試す.

Income Taxes.

- Tax period は M steps 続く（実験では $M = H/10$ とし, 1 episode に 10 tax periods があるものとする）
- ピリオド p の税は, time step t から $t + M$ までの収入 z_i^p に課される.

- Tax period の初めに, social planner は tax schedule $T(z)$ を決めて公表する.
  - 各 agent i は, 収入 z_i^p に応じて $T(z)$ を支払う.
  - 集められた税は, 全 agent に平等に分配される.
  - よって, 分配後の agent i の収入は,

$$\tilde{z}_i^p = z_i^p - T(z_i^p) + \frac{1}{N} \sum_{j=1}^{N} T\left(z_j^p\right). \tag{5}$$

Bracketed Tax Schedules.

- Scheme 間の比較を可能にするため, tax schedule は次のように ”bracketed” されたもののみを考える.
- Cut-off income levels $\{m_b\}_{b=0}^{B}$ s.t. $0 = m_0 \leq m_1 \leq \cdots \leq m_{B-1} \leq m_B = +\infty$ が先に与えられている.
- Social planner は, 各 bracket b に含まれる収入に対して適用される marginal tax rate $\tau_b \in [0, 1]$ を選ぶことで, tax schedule $T(\cdot)$ を決定する.

$$T(z) = \sum_{b=0}^{B-1} \tau_b \cdot ((m_{b+1} - m_b) \cdot 1[z > m_{b+1}] + (z - m_b) \cdot 1[m_b < z \leq m_{b+1}]) .$$

Social Welfare Functions

- Social planner の目的関数である social welfare function は, 生産性と平等性の trade-off を組み込めるように次のように決める.
- エージェントのコイン保有 $\boldsymbol{x}^c = (x_1^c, \ldots, x_N^c)$ に対し, equality を次で定義:

$$\mathbf{eq}(\boldsymbol{x}^c) = 1 - \mathbf{gini}(\boldsymbol{x}^c) \cdot \frac{N}{N-1}, \quad 0 \leq \mathbf{eq}(\boldsymbol{x}^c) \leq 1. \tag{7}$$

where

$$\mathbf{gini}(\boldsymbol{x}^c) = \frac{\sum_{i=1}^N \sum_{j=1}^N |x_i - c - x_j^c|}{2N \sum_{i=1}^N x_i^c}, \quad 0 \leq \mathbf{gini}(\boldsymbol{x}^c) \leq \frac{N-1}{N} \tag{8}$$

  - $\mathbf{eq}$ は, 1 で完全に平等 (全員同じ収入), 0 で完全に不平等 (1 人が全コインを独占).

- 生産性は,

$$\mathbf{prod}(\boldsymbol{x}^c) = \sum_{i=1}^{N} x_i^c. \tag{9}$$

- この **eq** と **prod** を social welfare function とする.[1]

$$\mathbf{swf}_t(\boldsymbol{x}_t^c) = \mathbf{eq}_t(\boldsymbol{x}_t^c) \cdot \mathbf{prod}_t(\boldsymbol{x}_t^c). \tag{10}$$

[1] Social welfare function として, weight $\omega_i \geq 0$ を用いて

$$\mathbf{swf}_t(\boldsymbol{x}_t^c, \boldsymbol{l}_t) = \sum_{i=1}^{N} \omega_i \cdot u_i \left(x_{i,t}^c, l_{i,t}\right). \tag{11}$$

を用いることも可能.

The Planner’s Problem.

.