MarketsAI

A Deep Reinforcement Learning Approach to Simulate High-Dimentional and Imperefect Information Economies

Matias Covarrubias

NYU

Motivation: Deep RL as an implementation of Rational Expectations

I borrow two elements from modern AI literature:

1. Agents use Reinforcement Learning (RL) to choose actions.
   - On simple environments, agent behavior is consistent with Rational Expectation.
   - They learn by interacting with environment. No need to find explicit fixed point between expectations and policies $\rightarrow$ computationally tractable.
   - We only need to characterize their environments and simulate interactions $\rightarrow$ easy to program.
2. Agent-environment separation. Households and Firms are agents. Markets are environments. Advantages:
   - Scalability: Many techniques to increase scale of models.
   - Modularity: This allow us to construct economies combining pre-specified agent and environment modules.

Agenda

1. Centralized planing problem.
   - Can be formalized as a Markov Decision Problem $\rightarrow$ theoretical guarantees.
   - **Benchmark**. Stochastic Growth model plus investment adjustment costs.
2. Decentralized (multi-agent) problems.
   - Can be formalized as Partially Observed Markov Games $\rightarrow$ few theoretical guarantees.
   - Challenge: non-stationarity. Agents learn while others are learning (and experimenting) as well.
   - **Benchmark**. Durable good market.
3. Large-scale problems
   - Many heterogeneous agents. **Benchmark**: Krussel-Smith (1998).
   - Many markets. **Application**: Capital Goods Supply and Boom and Bust Dynamics.

The learning workflow

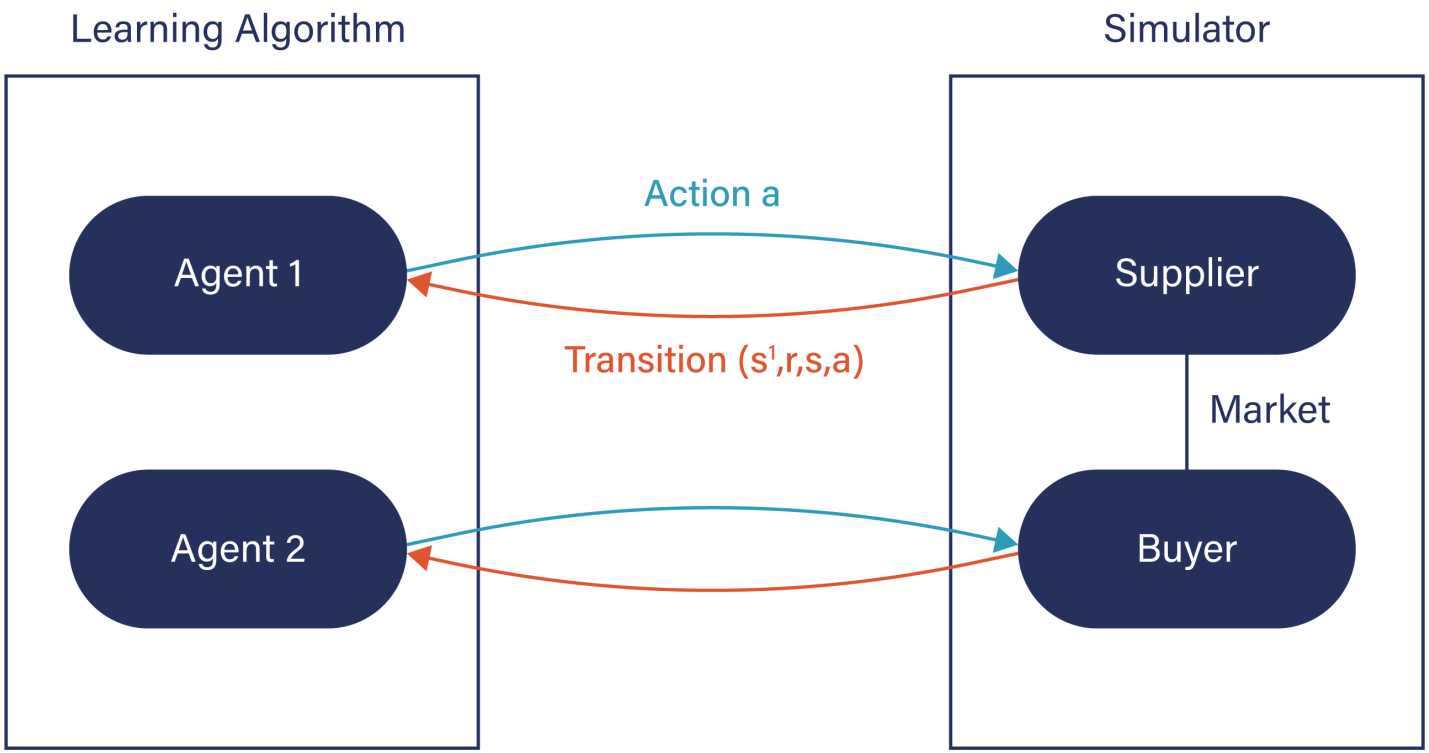


Figure: Online learning workflow

Formalizing the Simulator

We formalize an economy as a a Partially Observed Markov Game. In each period:

- n agents indexed by i choose an action $a_i \in \mathcal{A}_i$, after observing a state $s_i \in \mathcal{S}_i$.
- Let $\mathcal{S}$ be set of states, that is, the collection of all possible states S_i.
- The initial states are determined by a distribution $\rho : \mathcal{S} \mapsto [0, 1]$.
- To choose actions, each agent i uses a stochastic policy $\pi_{\theta_i} : \mathcal{S}_i \times \mathcal{A}_i \mapsto [0, 1]$.
- The combined actions produces the next state according to the state transition function $p : \mathcal{S} \times \mathcal{A}_i \times ... \times \mathcal{A}_N \mapsto \mathcal{S}$.
- Each agent i observes its reward as a function of the state and agent's action $r_i : \mathcal{S} \times \mathcal{A}_i \times ... \times \mathcal{A}_N \mapsto \mathcal{R}$, and receives a private observation $s_i' : \mathcal{S} \mapsto \mathcal{S}_i$.

Our first environment

- A representative household chooses how much to consume and how much to invest in order to maximize $\sum_{t=0}^{\infty} \beta^t u(c_t)$.

$$V_t(k_t) = \max_{s_t} U(c_t) + \mathbb{E}_t M_{t,t+1} V_{t+1}(k_{t+1}) \qquad \text{s.t.}$$
$$c_t = y_t(1 - s_t)$$
$$y_t = k_t^{\alpha}$$
$$K_{t+1} = (1 - \delta)K_t + s_t y_t$$

where:

- Utility from consumption: $U(c_t) = \frac{C^{(1-\gamma)}}{1-\gamma} + 1$.
- Production function: $f(k_t) = \theta k_t^{\alpha}$.
- Evolution of capital: $k_{t_1} = (1 - \delta)k_t + i_t$.

We add uncertainty

- A representative household chooses how much to consume and how much to invest in order to maximize $\sum_{t=0}^{\infty} \beta^t u(c_t)$.

$$V_t(k_t) = \max_{s_t} U(C_t) + \mathbb{E}_t M_{t,t+1} V_{t+1}(k_{t+1}) \qquad \text{s.t.}$$
$$c_t = y_t(1 - s_t)$$
$$y_t = \theta k_t^{\alpha}$$
$$K_{t+1} = (1-\delta)K_t + s_t y_t$$

where:

- Utility from consumption: $U(c_t) = \frac{C^{(1-\gamma)}}{1-\gamma} + 1$.
- Production function: $f(k_t) = \theta k_t^{\alpha}$.
- Evolution of capital: $k_{t_1} = (1-\delta)k_t + s_t y_t$.
- Productivity θ follows a two-state markov process with transition probability $\mathcal{P}$.

We add convex adjustment costs

- A representative household chooses how much to consume and how much to invest in order to maximize $\sum_{t=0}^{\infty} \beta^t u(c_t)$.

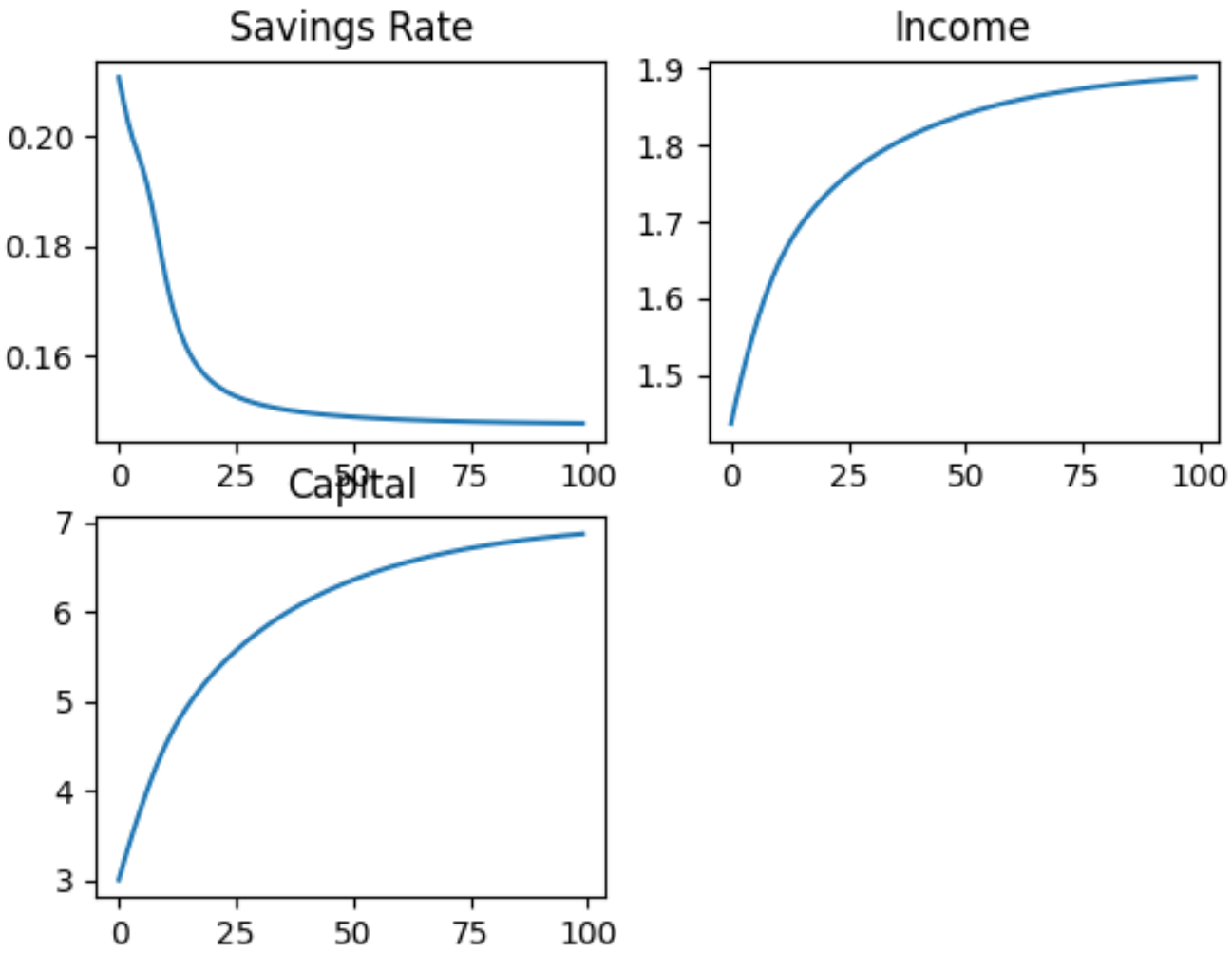
$$V_t(k_t) = \max_{s_t} U(C_t) + \mathbb{E}_t M_{t,t+1} V_{t+1}(k_{t+1}) \qquad \text{s.t.}$$

$$c_t + \phi(\frac{i}{k})^2 k = y_t(1 - s_t)$$
$$y_t = \theta k_t^{\alpha}$$
$$K_{t+1} = (1-\delta)K_t + s_t y_t$$

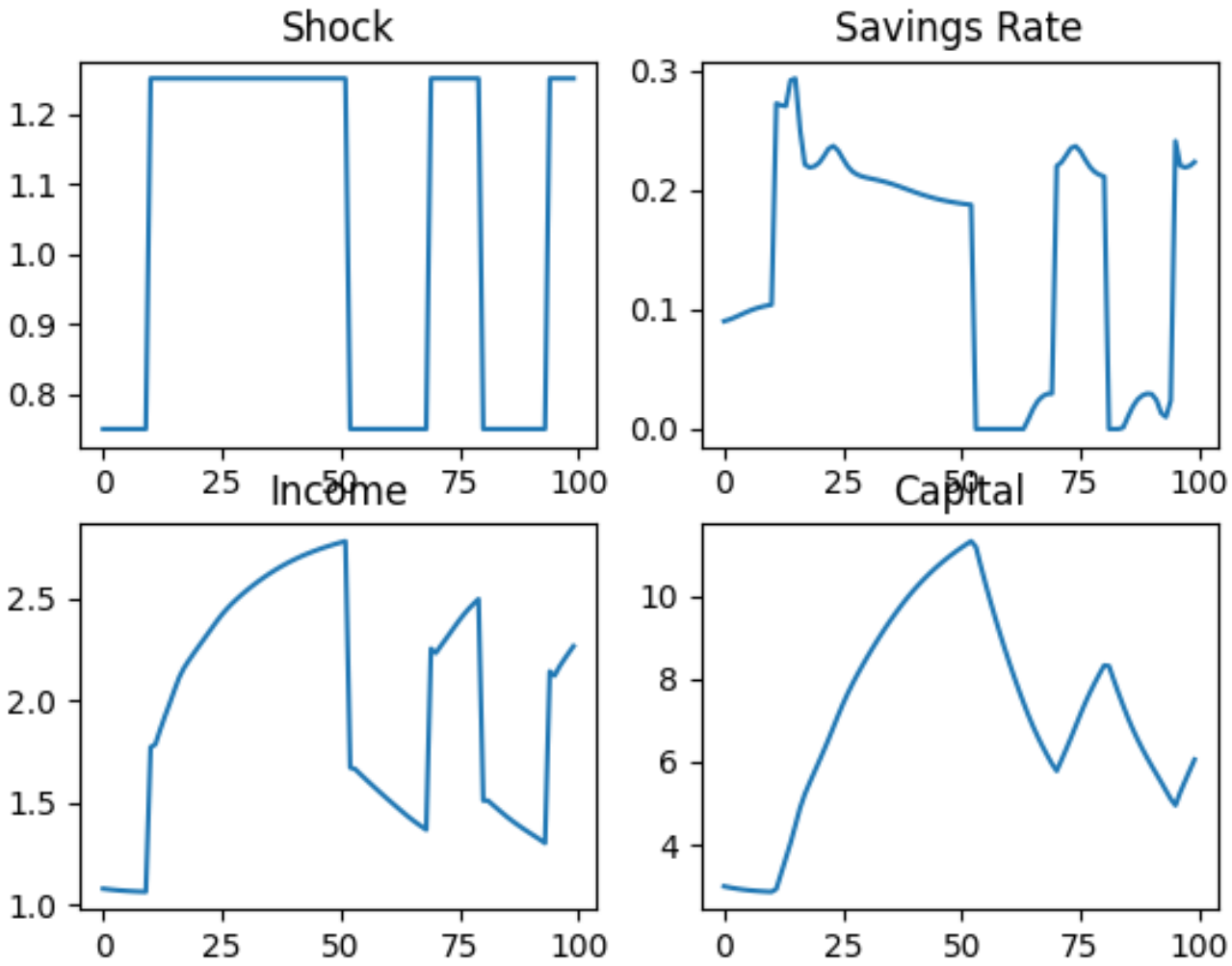
where:

- Utility from consumption: $U(c_t) = \frac{C^{(1-\gamma)}}{1-\gamma} + 1$.
- Production function: $f(k_t) = \theta k_t^{\alpha}$.
- Evolution of capital: $k_{t_1} = (1-\delta)k_t + s_t y_t$.
- Productivity θ follows a two-state markov process with transition probability $\mathcal{P}$.
- Convex adjustment cost: $C(\frac{i}{k}) = \phi(\frac{i}{k})^2 k$.

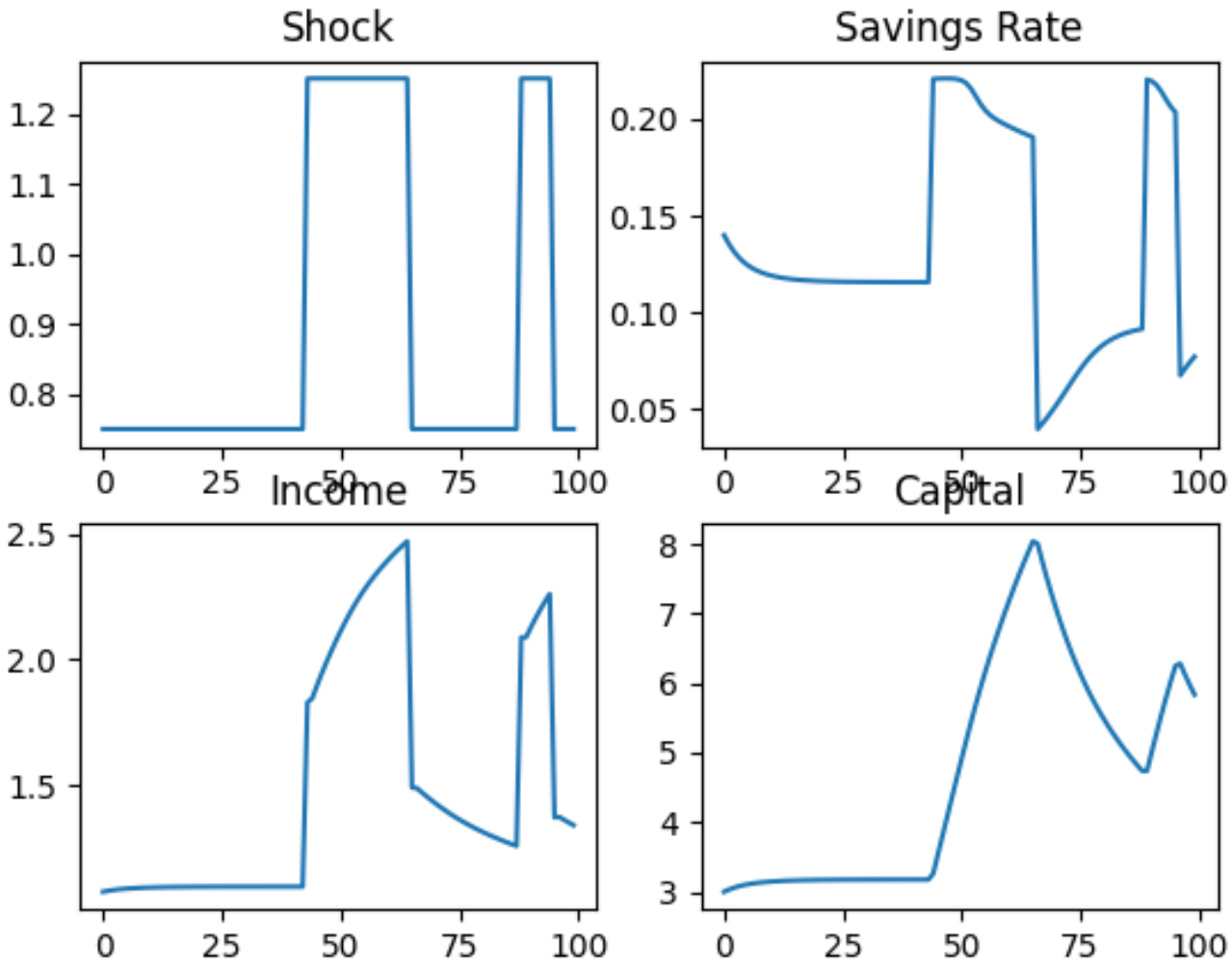
Results for deterministic case



Results for stochastic case



Results for adjustment cost case case



Endogenous Time-to-Build

- The problem is the same as before but with with an inventory process for investment and an expanded action space.
- There areN^{TTB} stages (we index them from 0 to$N^{TTB}-1$). The agent decides the scale of the new projects and the progress of the projects through the stages.
- New projects go directly to stage 0. Then, for each stage, the agent decides what percentage to move 1 stage ahead ($z_{i,i+1}$ for $i \in [0,..,N^{TTB}-1]$) and what percentage to move 2 stages ahead $z_{i,i+2}$ for $i \in [0,..,N^{TTB}-2]$).
- The rest $(1-z_{i,i+1}-z_{i,i+2})$ stays in the stage i. We have $N^{TTB}-1$ occasionally binding constraints $z_{i,i+1}+z_{i,i+2}<1$.
- The state space now includes $\{K_t, \theta_t, \{x_{i,t}\}_{i\in[0,..,N^{TTB}-1]}\}$ where $x_{i,t}$ is the stock of projects at stage i at time t.

Endogenous Time-to-Build

- The cost of progressing through stages is increasing in speed.
- Each stage represents a proportion ω_i of the total cost and $\sum_{i=0}^{N^{TTB}-1} = 1$. if you move one unit of capital that its at stage i two stages ahead, you pay $(\omega_{i+1} + \omega_{i+2})\phi$, where ϕ is the speed penalty.
- There is also a convex adjustment cost over Z_t, the total investment spending, which is the sum of the progress spending of each stage: $C(Z) = \psi(\frac{Z}{2})^2 k_t$.
- The evolution of capital is

$$k_{t+1} = (1-\delta)k_t + z_{N^{TTB}-1,N^{TTB}} + z_{N^{TTB}-2,N^{TTB}}$$

.

Market: Durable Good supplied elastically

- Agents: $n^s >= 1$ suppliers. $n^b >= 1$ buyers.
- Action space:
  - Supplier: Each supplier chooses a price $p_i' \in \mathbb{R}$ for the next period.
  - Households: each buyer chooses a quantity vector $h_i \in \mathbb{R}^{n^s}$.
- State space: The state space of all agents include:
  - The current price vector ($\mathbf{p}$), chosen the previous period.
  - The stock of good j owned by agent i: $\{H_i^j\}_{i,j}$.
  - Stochastic parameters (e.g., depreciation shock for each good).

Market: Durable Good supplied elastically

- Step `s', R = step(p',h)`:

$$R_j^s = (p_j - c_j) \sum_{j=1}^{n^b} h_i^j$$

$$R_i^b = U_i(H_i^j) + \mu_i \mathbb{1}(I_i < \sum_{j=1}^{n^s} h_i^j p_j)$$

$$H_i^{j'} = (1 - \delta_j) H_i^j + h_i^j$$

$$s' = \{p', \{H_i^{j'}\}_i\}$$

where h_i^j is the demand of agent i of product j and δ_j is the depreciation rate of good j

Decentralized markets: Durable Good supplied inelastically

- Agents: $n^s >= 1$ suppliers. $n^b >= 1$ buyers.
- Action space:
  - Suppliers: Each supplier chooses a price $p'_j \in \mathbb{R}$ and quantity $\tilde{h}^s_j$ to develop for the next period.
  - Buyers: each buyer chooses their intended quantity $\hat{h}^j_i$ give prices. This will not necessarily correspond to the obtained quantity h^j_i, since the inventory needs to be allocated.
- The state space of all agents include:
  - The current price vector ($\mathbf{p}$), chosen the previous period.
  - Each buyer i observes his stocks of good j. Each supplier j knows the stock of his good owned by each buyer i.
  - The inventory of good j available for selling: $\{h^s_j\}_j$.
  - Stochastic parameters (e.g., depreciation shock for each good).

Market: Durable Good supplied inelastically

- Step `s', R = step(p)`:

$$h_i^j = f(\{\hat{h}_i^j\}_i, h_j^s)$$

$$R_j^s = p_j \sum_{j=1}^{n^b} h_i^j - C_j(\tilde{h}_j^s)$$

$$R_i^b = U(H_i^j) + \mu_i \mathbb{1}(I_i < \sum_{j=1}^{n^s} h_i^j p_j)$$

$$H_i^{j'} = (1-\delta_j) H_i^j + h_i^j$$

$$h_j^{s\prime} = \tilde{h}_j^s + h_j^s - \sum_{j=1}^{n^b} h_i^j$$

$$s' = \{p', \{H_i^{j'}\}_i, \{h_j^{s\prime}\}_i\}$$

where h_i^j is realized quantity obtained by buyer i of product j.

Reinforcement Learning: Overview

- We start from simplest single agent algorithm. The problem is:

$$\max_{a \in A} \sum_{t=s}^{\infty} \gamma^{s-t} E_t[r_{t+s}]$$

- Two value functions:
  1. state-value function: $V(s) = \max_{a \in A}\{E[r|s,a] + \gamma E[V(s')|s,a]\}$
  2. action-value function: $Q(s,a) = E(r|s,a) + \gamma E[\max_{a' \in A} Q(s',a')|s,a]$
- Q-Learning:
  - Initialize $Q_0(s,a)$. Loop until converge:
    - agent chooses $a = \arg\max Q(s,a)$ and observes $\{r, s', s, a\}$.
    - Old Q: $Q(s,a)$.
    - Target Q: $r_t + \gamma \max_{a' \in A} Q(s',a')$.
    - New Q:

$$Q'(s,a) = (1-\alpha)Q(s,a) + \alpha[r + \gamma \max_{a' \in A} Q(s',a')]$$

Getting deeper: Deep RL

- Deep Q-learning consists in approximating the Q function with a neural net: $\hat{Q}(s,a) \equiv \mathcal{N}_\rho \sim Q(s,a)$.
- The parameters ρ are estimated in order to minimize

$$\mathcal{L}(\rho) = \mathbb{E}\left[\left(\hat{Q}(s,a) - (r_t + \gamma \max_{a' \in A} \hat{Q}(s',a'))\right)^2\right]$$

  in observed transitions $\{r, s', s, a\}$.
- A common example of a NN is a composite function made of nested linear functions:

$$\mathcal{N}_\rho(x) = \sigma_K(W_K ... \sigma_2(W_2 \sigma_1(W_1 x + b_1) + b_2) ... + b_K)$$

  where x is the input (in our case the transitions $\{r, s', s, a\}$), $W_i \in \mathbb{R}^{m_{i+1} \times m_i}$ are the weights, $b_i \in \mathbb{R}^{m_i + 1}$ are the biases, and $\sigma()$ are activation functions.

Convergence results in RL

- Tsitsiklis (1994) If state space and action space are discrete, Q-learning converges globally if:
  - All state-action pairs are visited infinitely often.
  - Conditions on learning rate α. The sum $\sum_{t=0}^{\infty} \alpha_t^2$ is finite and $\sum_{t=0}^{\infty} \alpha_t$ is infinite.
- Tsitsikli and Van Roy (1997) prove convergences in the case of linear function approximators and Maei et al (2009) prove convergence with smooth non-linear approximators (e.g., neural nets).
- For multi-agent, convergence is hard to prove. Muller et al (2019) show convergence in a large class of games to a solution concept they called α-rank.

Taxonomy of State of the Art Algorithms

There are three objects that the agents can learn:

1. q function $Q(s,a)$
   - **Q-based methods** use NNs to represent $Q(s,a)$.
   - off-policy method: the NN can learn from steps taken by any policy.
   - Model-free method.
2. policy function $\pi(s,a)$
   - **Policy-gradient methods** use NN to represent $\pi(s,a)$.
   - on-policy method. the NN only learns from steps taken from optimal policy.
   - Model-free. More stable than Q but less sample efficient.
3. Model $p(r,s'|s,a)$
   - **Model-based methods** uses different methods to represent $p(r,s'|s,a)$.
   - In recent stage of development compared to other methods.

In practice: How to create environments and agents

An environment consists of a **reset** and a **step** function.

An agent consist of a **choose_action** and a **learn** function. Loop:

- `initial_state=env.reset()`
  - The reset function gives you the initial state.
- `action = agent.choose_action(state)`
  - takes the state as input and choose the optimal action.
- `next_state, rewards = env.step(actions)`
  - Takes as input the actions of all the players and outputs the rewards for all the agents and the next state.
- `agent.learn(r,s',s,a)`
  - Takes as input the transition info $\{r, s', s, a\}$ and updates the internal estimates (e.g., policy functions).

The Framework: Agents, Markets and Economies

1. Agents. Two classification criteria:
   - Algorithm: RL Model-free , RL Model-based or FOC-based.
   - Role in markets: Household , Firm , Capital Producer , Saver, Borrower, etc.
2. Markets. I will create them in two batches:
   - Core Markets: Spot , Durable good , Labor , Credit.
   - Advanced Markets: Financial Intermediation, Stock Market, Over the counter markets.
3. Economies:
   - Fully characterized by a list of agents, a list of markets and a participation matrix that tells you which agent participates in which market.
   - Planer solution can be implemented as the cooperative multi-agent solution.

The Economy Constructor

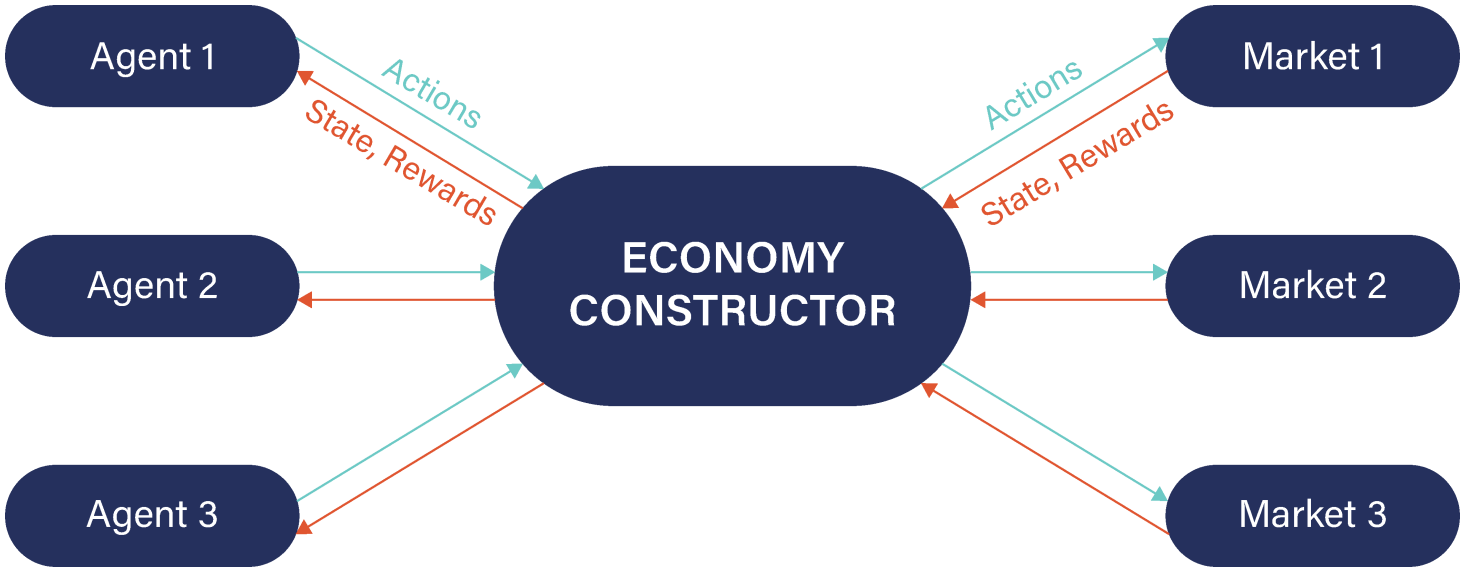


Figure: Economic Constructor

- Three types of states:
  - Market specific states (e.g. prices, depreciation).
  - Agent specific states (e.g. productivity, preferences).
  - Economic specific states (e.g. aggregate productivity).

Customization of markets

- Both discrete and continuous action spaces are supported.
- When declaring parameters, you can declare a stochastic process instead of a parameter.
  - e.g.: `substitution = iid_Normal(coeffs=[0.15,0.1])` instead of `substitution=0.15`.
  - The market automatically evaluates the stochastic function and includes i int market state.
- You can specify agents functional forms.
  - e.g.: `utility = CES(coeffs=1)`.
- If the market is created from an economy constructor, the market is automatically "Connected", which means that the imposing of budget constraints and registry of states is done at the economy level.
- If the market is "Unconnected", it automatically imposes the budget constraints and keep registry of states.

Challenges to the framework

1. F.O.C.s are useful to introduce abstractions that simplify the environment. Example: Representative Agent.
   - A solution is to introduce F.O.C. based agents, which observe the state and solves their F.O.Cs each period.
   - In the extreme, we can specify the RL problem as one of submitting full equations (e.g. a demand schedule) and the environment act as auctioneer, solving for the equilibrium.
2. Constraints do not have a natural implementation in the RL world.
   - A fast solution is to punish the agent if it violates a constraint during learning.
   - More complex solutions involve changing the action space as the environment evolves or adjusting the policy neural net so it always delivers an acceptable action.
3. Non-stationarity of environment in early stages of learning might hinder progress.
   - A solution would be to use curriculum learning and abstract markets.
   - Agents may start playing against programed counterparts (as in diff. demand example) and then they start learning against RL players.

Three possible Macro Avenues

1. Macro-financial model of Real Estate.
   - Start with Real Model, consisting of housing, CRE markets plus final (spot) market. Benchmark: Kydland and Prescott (1982).
   - Add Intermediated Credit Market. Benchmark: Kaplan, Mitmal and Violante (2020) Elenev, Landvoight and Van Nieuwerburgh (2020).
   - I can study macro effect of TTB or taxes.
2. Monetary Policy:
   - Start with basic New Keynesian model with exogenous Taylor Rule.
   - Add credit and banking. Benchmarks: Kaplan, Moll and Violante (2018).
   - I can study equilibrium selection issues and test neo-fisherian hypothesis.
3. Incomplete Information:
   - Start with basic Grossman and Stiglitz.
   - Add real sector.
   - I can study learning from prices in a dynamic setting.