Operator Learning in Macroeconomics

Yaolang Zhong

University of Warwick

Overview

- This paper proposed a new numerical framework to solve a prevalent class of structural models: the heterogeneous agent models with aggregate shocks
- In this context, the cross-sectional distribution of all individual states, which is an infinite-dimensional object, becomes part of the agents' state variable in the recursive form of their utility-maximization problem
- This leads to a severe computational challenge for researchers adopting structural models for their problems of interest

Overview

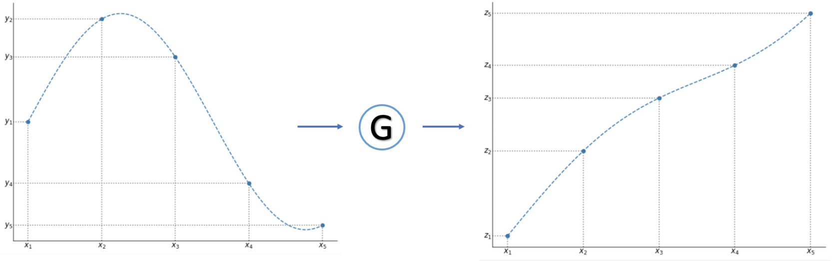
- Advantages:
  - Computationally Efficient: In experiments on a Bewley-Huggett-Aiyagari type model (Den Haan, Judd and Juillard, 2008), my proposed approach outperforms the state-of-the-art alternative in the literature (Maliar, Maliar and Winant, 2021) ▸ Figure
  - Generally Applicable: The application of my approach can be generalized to other research fields such as labor, IO, and network (e.g., Khan and Thomas (2008))
- The key insight is threefold:
  - Reformulation of the problem of solving the model into learning an operator
  - Parameterization of the operator by the neural operator (Li et al., 2020)
  - Implementation by a specific training scheme (policy function iteration)

Setup

- My approach focuses on the case where:
  - There is a continuum of agents $i = 1, ..., N(N \to \infty)$, truncate to e.g. $N = 1,000$
  - The information set of an agent includes not only her individual state vector $s_i \in \mathcal{S}$, but also the states of all other agents $s^N = (s_1, ..., s_N) \in \mathcal{S}^N$ (eqivalently a distribution)
  - All agents share the same policy function $\mathbf{g} : \mathcal{S} \times \mathcal{S}^N \to \mathcal{S}$ such that $s_i' = \mathbf{g}(s_i, s^N)$
- The goal is to solve for the optimal policy function $\mathbf{g}^*$
- Considers an intuitive case: $s_i = (k_i), k_i \in [k_{\text{min}}, k_{\text{max}}]$, the agent's capital holding
- Krusell-Smith Framework:
  - $\mathbf{g}^*(s_i, m), m$ is a set of moments of s^N: lose information, inner-outer loop algorithm
- Neural Network Framework:
  - function approximator: $\mathbf{g}_{\text{NN}}(s_i, s^N) \approx \mathbf{g}^*(s_i, s^N)$: dimensionality N determines the parameterization ▸ Figure

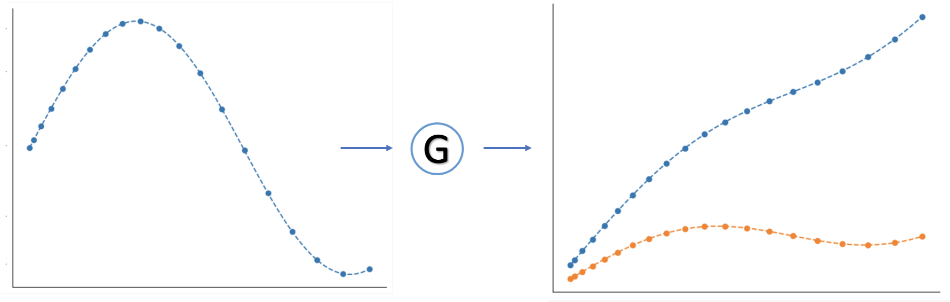
Introduction: Operator

- An operator is a mapping between function spaces
- Examples: $\mathbf{h}(x) = \mathbf{G}(\mathbf{f})(x) = \frac{\mathrm{d}\mathbf{f}}{\mathrm{d}x}(x)$ and $\mathbf{h}(x) = \mathbf{G}(\mathbf{f})(x) = \int \mathbf{f}(x)\mathrm{d}x$
- $\mathbf{f} : (\{x_1, ..., x_J\}, \{y_1, ..., y_J\})$ and $\mathbf{h} : (\{x_1, ..., x_J\}, \{z_1, ..., z_J\})$



Introduction: Operator (cont.)

- Increasing J-grid gives higher approximation accuracy
- The J-grid is not necessarily uniform
- We can have $\mathbf{G}(\mathbf{f}) = (\mathbf{h_1}, \mathbf{h_2})$



Introduction: The Bewley-Huggett-Aiyagari Model

- Lowercase letters for individual variables, uppercase letters for aggregate variables and bold letters for operations.
- A continuum of infinitely lived and ex-ante identical agents, each period:
  - the time endowment $\bar{l}$
  - earn the after-tax wage $(1-\tau_t)\bar{l}W_t$ if employed $(\epsilon = 1)$
  - earn the unemployment benefit μW_t if unemployed $(\epsilon = 0)$
  - W_t is the per unit of time wage rate, τ_t is the tax rate, and μ is a model parameter denoting the fraction of wage

Introduction: The Bewley-Huggett-Aiyagari Model (cont.)

- Market is incomplete: non-zero capital holding $k_t \geqslant 0$
- The net rate of return for capital: $R_t - \delta$, R_t is market-determining interest rate and δ is the fixed depreciation rate
- Agents' maximization problem

$$\mathbb{E}\sum_{t=0}^{\infty}\beta^t \frac{(c_t)^{1-\gamma}-1}{1-\gamma}$$

subject to

$$c_t + k_{t+1} = R_t k_t + \left[(1-\tau_t)\,\bar{l}\epsilon_t + \mu\,(1-\epsilon_t)\right] W_t + (1-\delta)k_t$$

Introduction: The Bewley-Huggett-Aiyagari Model (cont.)

- Firms: Cobb-Douglas production function $Y_t = Z_t K_t^{\alpha} \left(\bar{l} L_t\right)^{1-\alpha}$
- K_t is the per capita capital, L_t is the employment rate, and $\alpha \in [0, 1]$ is the capital sharing. Z_t is a binary aggregate productivity shock: $Z_t \in \{Z_b, Z_g\}$
- Government: keep budget balanced by redistributing all taxation
- Firms' first-order optimality + Government's budget constraint:

$$R_t = \alpha Z_t \left(\frac{K_t}{\bar{l} L_t}\right)^{\alpha-1}, \quad W_t = (1-\alpha) Z_t \left(\frac{K_t}{\bar{l} L_t}\right)^{\alpha}, \quad \tau_t = \frac{\mu(1-L_t)}{\bar{l} L_t} \qquad (1)$$

- Shocks: Z_t is first-order Markovian, ϵ_t is first-order Markovian conditional on the transition of Z_t, and confront to the law of the large number
- $(\epsilon_t, Z_t) \sim \mathbf{\Pi}$: the element $\pi_{\epsilon\epsilon' Z Z'}$ denotes $\mathrm{P}((\epsilon, Z) \to (\epsilon', Z'))$

Introduction: The Bewley-Huggett-Aiyagari Model (cont.)

- Denote $\mathbf{\Gamma}$ the distribution of agents over capital holdings
- Denote the law of motion of $\mathbf{\Gamma}$ by $\mathbf{H} : \mathbf{\Gamma}' = \mathbf{H}(\mathbf{\Gamma}, Z, Z')$
- The agents' problem can be therefore express recursively as

$$\mathbf{V}(k, \epsilon; Z, \mathbf{\Gamma}) = \max_{c,k'} \{\mathbf{U}(c) + \beta \mathbb{E}[\mathbf{V}(k', \epsilon'; Z, \mathbf{\Gamma}') \mid \epsilon, Z]\} \tag{2}$$

subject to

$$\begin{aligned} c + k' &= Rk + [(1-\tau)\bar{l}\epsilon + \mu(1-\epsilon)]W + (1-\delta)k, && (3) \\ \epsilon', Z' &\sim \mathbf{\Pi}(\epsilon, Z), && (4) \\ \mathbf{\Gamma}' &= \mathbf{H}(\mathbf{\Gamma}, Z, Z'), && (5) \\ k' &\geqslant 0 && (6) \end{aligned}$$

- Denote the solution to (2) subject to (3), (4), (5), (6) $\mathbf{V}^*(\cdot)$ and corresponding policy function $\mathbf{g}^*(\cdot)$

Contribution to the literature

Table 1: Comparison of Three Numerical Frameworks for the Desirable Properties

Property	Framework		
	KS[1]	NN[2]	Operator[3]
Full Information of Distribution	×	✓	✓
Discretization-Invariance	✓	×	✓
Permutation-Invariance	✓	×	✓
Sharing-Aggregation	✓	×	✓

[1] Krusell-Smith

[2] Deep Learning with feed-forward neural network

[3] Deep Learning with neural operator (This Paper)

Permutation-Invariance

- Let us start from $k_i' = \mathbf{g}(k_i, k^N)$
- I propose using the empirical cumulative distribution function (ECDF) $\mathbf{\Gamma} \in \mathcal{T} : [k_{\text{min}}, k_{\text{max}}] \rightarrow [0, 1]$ to characterize $k^N = (k_1, ..., k_N)$
- $\mathbf{\Gamma}$ is represented by the interpolation of the tuple $\{(\tilde{k}_1, ..., \tilde{k}_N), (\frac{1}{N}, ..., \frac{N}{N})\}$, where $(\tilde{k}_1, ..., \tilde{k}_N)$ is in ascending order
- Permutation-Invariance:
  - In principle, $k_i' = \mathbf{g}(k_i, k^N) = \mathbf{g}(k_i, \hat{k}^N)$, where $\hat{k}^N$ is any permutation of k^N
  - But in practice, a large amount of simulated data and computational time is required to learn this property
- The form $k_i = \mathbf{g}(k_i, \mathbf{\Gamma})$ implicitly fulfills this property

Sharing-Aggregation

- I propose recasting the policy into an operator form:

$$k_i' = \mathbf{g}(k_i, \mathbf{\Gamma}) = \mathbf{G}(\mathbf{\Gamma})(k_i)$$

- $\mathbf{G} : \mathcal{T} \to \mathcal{H}$ such that $\mathbf{h} = \mathbf{G}(\mathbf{\Gamma}) : [k_{\text{min}}, k_{\text{max}}] \to [k_{\text{min}}, k_{\text{max}}]$ is the "conditional policy function"
- Process the aggregation part $\mathbf{\Gamma}$ and individual part k_i separately and sequentially
- Sharing-Aggregation:
  - $k_i' = \mathbf{g}(k_i, \mathbf{\Gamma})$: repeat the processing of $\mathbf{\Gamma}$ for $i = 1, ..., N$, the cost is $\mathcal{O}(N^2)$ ▸ Figure
  - $k_i' = \mathbf{G}(\mathbf{\Gamma})(k_i)$: process $\mathbf{\Gamma}$ once-for-all to obtain $\mathbf{h} = \mathbf{G}(\mathbf{\Gamma})$, then $k_i' = \mathbf{h}(k_i)$ for $i = 1, ..., N$, the cost is $\mathcal{O}(N)$ ▸ Figure

Discretization-Invariance

- I propose parameterizing the operator $\mathbf{G}$ by the neural operator θ
- In the case of $\mathbf{g}_{\mathsf{NN}}(k_i, k^N)$, the neural network approximates the operation in vector space through a series of matrix multiplications
- In the case of $\mathbf{G}_\theta(\mathbf{\Gamma})$, the neural operator approximates the operation in function space through a series of convolutions (*Universal Approximation Theorem for Operator*)
- Fourier Neural Operator: transforms the input function into the Fourier domain, imposes a series of matrix multiplications, and then returns the output to the spatial domain (*The Convolution Theorem*) ▸ Figure
- Discretization-Invariance: Parameterization in the Fourier domain is independent of the discretization of the input and output function in spatial domain (choice of N)

Summary

- Roadmap of the transformation:

$$\mathsf{k}_i = \mathbf{g}(k_i, k^N) \rightarrow \mathbf{g}(k_i, \mathbf{\Gamma}) \rightarrow \mathbf{G}(\mathbf{\Gamma})(k_i) \rightarrow \mathbf{G}_\theta(\mathbf{\Gamma})(k_i)$$

That's all, thank you!

Figure

Return

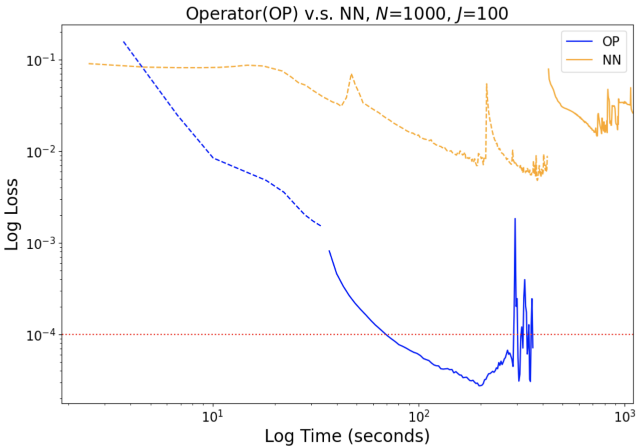


Figure 1: Training Loss vs. Time (seconds). My approach (blue) reached the 1% error (square root of loss) in around 5 mins and the alterative approach (yellow) took more than 20 mins

Figure

Figure

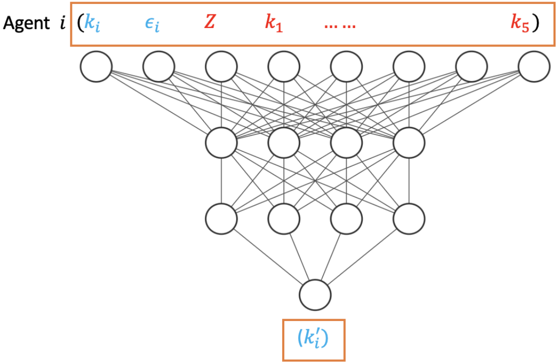


Figure 2: An example Neural Network in the case of $N = 5$ agents

Figure

Return

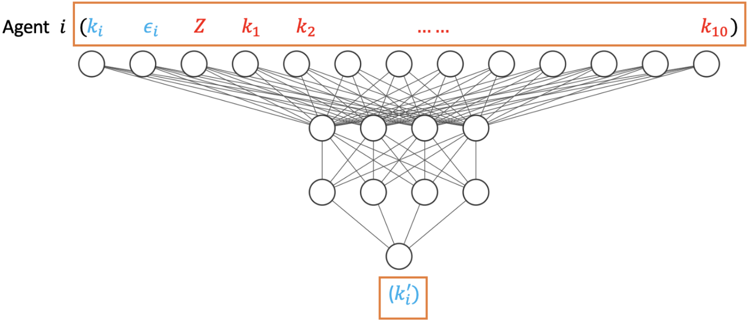


Figure 3: An example Neural Network in the case of $N = 10$ agents

Figure

▸ Return

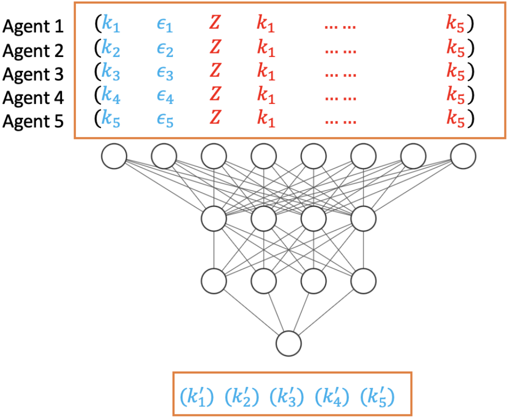


Figure 4: Processing the transition in the case of policy function

Figure

Return

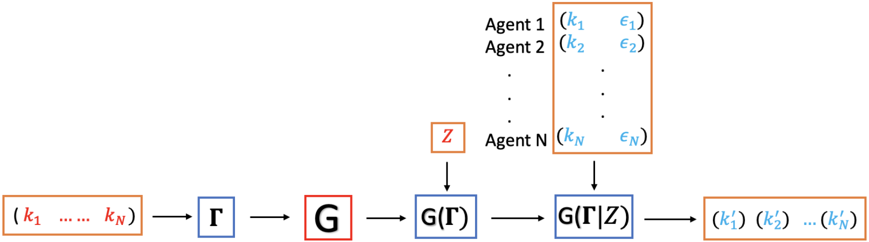


Figure 5: Processing the transition in the case of policy operator

Figure

Layer

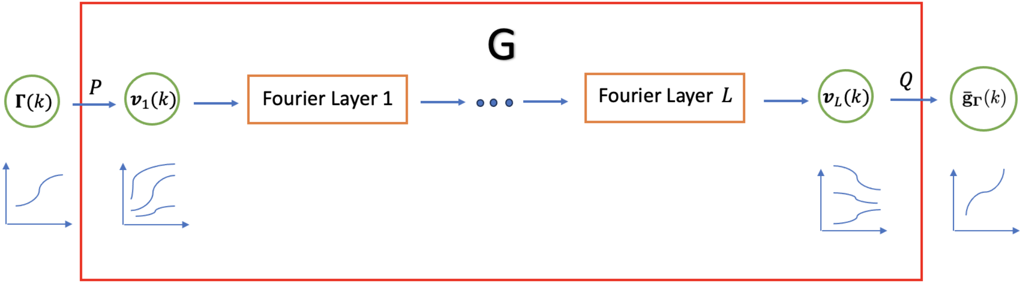


Figure 6: Fourier Neural Operator Architecture

Figure

Return

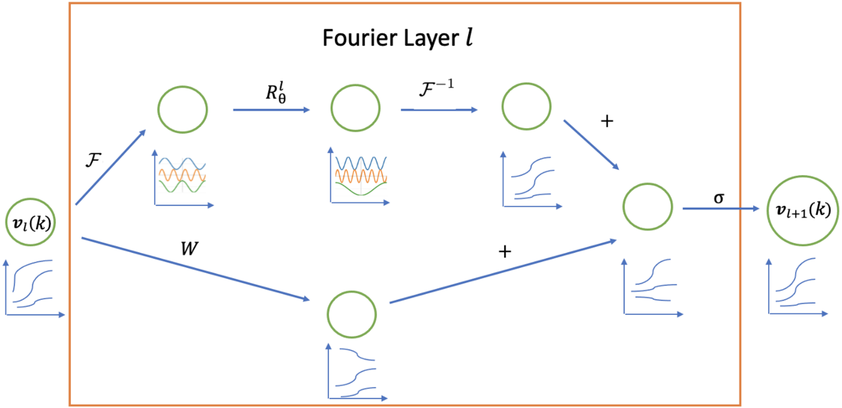


Figure 7: A Typical Fourier Layer l

Haan, Wouter J Den, Kenneth L Judd, and Michel Juillard, "Computational suite of models with heterogeneous agents: model specifications," *Journal of Economic Dynamics and Control, this issue*, 2008.

Khan, Aubhik and Julia K Thomas, "Idiosyncratic shocks and the role of nonconvexities in plant and aggregate investment dynamics," *Econometrica*, 2008, *76* (2), 395–436.

Li, Zongyi, Nikola Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar, "Fourier neural operator for parametric partial differential equations," *arXiv preprint arXiv:2010.08895*, 2020.

Maliar, Lilia, Serguei Maliar, and Pablo Winant, "Deep learning for solving dynamic economic models.," *Journal of Monetary Economics*, 2021, *122*, 76–101.