Continuation Value is All You Need

Simple Deep Learning for HA Models with Aggregate Shocks

Jeffrey Sun, University of Toronto

Computing in Economics and Finance, June 20, 2024

Overview

1. Describe model class and algorithm
2. Contrast with literature
3. Discuss performance, flexibility, and accuracy
4. Krusell-Smith benchmark

Algorithm

Model Class

- Discrete time. Het. atomistic agents. Beginning of period aggregate state:

$$\Gamma \equiv (\underbrace{\Lambda}_{\text{Household state distribution}}, \underbrace{S}_{\text{Other aggregate state variables}}).$$

- Timing within period:
    1. Beginning of period: Household with state $x \in X$ has value $V^{\text{start}}(x; \Gamma)$
    2. End of (household) period: Households have value $V^{\text{end}}(x; \Gamma)$ and distribution Λ^{end}
    3. Aggregate shock: $\Gamma' = \Omega\big(\Gamma, \Lambda^{\text{end}}, \varepsilon\big)$, with $\varepsilon \sim \text{Cat}(\{p_i\})$
- Key assumption: $V^{\text{start}}|_\Gamma : X \to \mathbb{R}$ is a function of $V^{\text{end}}|_\Gamma : X \to \mathbb{R}$ and prices[1]

[1] "prices" $\equiv$ some set of equilibrium values

Data Representation

$$\text{Fix some } \Gamma \equiv (\underbrace{\Lambda}_{\text{Household state distribution}}, \underbrace{S}_{\text{Other aggregate state variables}})$$

- Represent $V^{\text{start}}|_\Gamma$, $\Lambda^{\text{start}}|_\Gamma$ by data arrays $A_\Gamma^{V\text{start}}$, $A_\Gamma^{\Lambda\text{start}}$. Similarly, $A_\Gamma^{V\text{end}}$, $A_\Gamma^{\Lambda\text{end}}$.
- Define the **Intra-Period Problem** function mapping V backward and Λ forward:

$$\text{IPP} : \left(A_\Gamma^{V\text{end}}, A_\Gamma^{\Lambda\text{start}}, S_\Gamma\right) \mapsto \left(A_\Gamma^{V\text{start}}, A_\Gamma^{\Lambda\text{end}}\right)$$

- IPP can typically be implemented *conventionally* (no neural net)
- To truly solve a model, the Hard Part is knowing $A_\Gamma^{V\text{end}}$
- Strategy: Train a neural net $\mathcal{N}(\Gamma;\theta)$ to approximate $A_\Gamma^{V\text{end}}$
  - $\mathcal{N}$ uses "generalized moments" of Han et al. (2024)

Algorithm for Continuation Value Neural Net (One-Period Lookahead)

1. Guess neural net parameters θ_0
2. For each epoch $i \in \{1, \ldots, I\}$, simulate the model given $\mathcal{N}(\,\cdot\,; \theta_i)$, then update $\mathcal{N}$:
    - 2.1 Initialize state $\Gamma_{i0} \equiv \left(A_{i0}^{\Lambda\text{start}}, S_{i0}\right)$
    - 2.2 For each period $t \in \{1, \ldots, T\}$:
        - 2.2.1 Approximate end-of-period value array $A_{it}^{V\text{end}} \leftarrow \mathcal{N}(\Gamma_{it}\ ;\ \theta_i)$
        - 2.2.2 Compute $\left(A_{it}^{V\text{start}}, A_{it}^{\Lambda\text{end}}\right) \leftarrow \text{IPP}\left(A_{it}^{V\text{end}}, A_{it}^{\Lambda\text{start}}, S_{it}\right)$
        - 2.2.3 Draw $\varepsilon_{it} \sim \text{Cat}(\{p_i\})$
        - 2.2.4 Iterate state $\Gamma_{i,t+1} \leftarrow \Omega\left(A_{it}^{\Lambda\text{end}}, \Gamma_{it}, \varepsilon_{it}\right)$
    - 2.3 Update $\theta_i \rightarrow \theta_{i+1}$ with cost function, for sample periods $\mathcal{T}_i \subseteq \{1, \ldots, T\}$:

$$\frac{1}{|\mathcal{T}_i|} \sum_{t \in \mathcal{T}_i} \left| A_{it}^{V\text{end}} - \frac{1}{K} \sum_{k=1}^{K} p_k \widehat{A_{i,t+1}^{V\text{start}}}(\Gamma_{it}, \theta_i \mid \varepsilon_{it} = k) \right|^2$$

Details

Literature

Key choices (within HA model solutions w/ aggregate shocks)

- Time:
  - **Discrete**
  - Continuous: Gu et al. (2024), Fernández-Villaverde et al. (2023), etc.
- Solution scope
  - **Global**: Most DL based methods
  - Local: Most projection/perturbation methods: Bhandari et al. (2023), Bilal (2023), Auclert et al. (2021), Winberry (2018), etc.
- Policy function
  - **Conventional**: Krusell and Smith (1998), Hull (2015), etc.
  - Deep Learning: Han et al. (2024), Azinovic et al. (2022), Maliar et al. (2021), etc.
- Household simulation (all compatible with continuation value strategy):
  - Discrete State: Gu et al. (2024) do both, Kaplan et al. (2020)
  - Finite Agent: Krusell and Smith (1998), Han et al. (2024), etc.
  - Personal preference: **Gridded CDF**: fast, deterministic, less biased than point-mass

Classification

- Krusell-Smith/Hull with NN V and generalized moments
- Q-learning with equilibrium
- Approximate Dynamic Programming with Post-Decision States (ADP-POST) (Powell, 2007) with deep learning and equilibrium

Discussion

Flexibility

Key advantage: No need to train a policy function approximator, often the hardest part

- Only need to implement

$$\text{IPP}: \left(A_{\Gamma}^{V\text{end}}, A_{\Gamma}^{\Lambda\text{start}}, S_{\Gamma}\right) \to \left(A_{\Gamma}^{V,\text{start}}, A_{\Gamma}^{\Lambda\text{end}}\right)$$

- Can typically be done conventionally. Immediately correct as a function of $A_{\Gamma}^{\Lambda,\text{end}}$

I provide code to modularly implement IPP for household problems featuring:

- Consumption-saving decisions
- Idiosyncratic income shocks
- Binding borrowing constraints
- Multiple assets
- Multiple locations and frictional migration
- Real estate markets
- All of the above simultaneously

Performance

$$\text{IPP} : \left(A_\Gamma^{V,\text{end}}, A_\Gamma^{\Lambda,\text{start}}, S_\Gamma\right) \to \left(A_\Gamma^{V,\text{start}}, A_\Gamma^{\Lambda,\text{end}}\right)$$

- IPP function is inner loop of algorithm: costly need for many simulations
- IPP represents the bottleneck, but also the target for optimization
- The available code for building IPP functions is highly optimized and reusable
- Ideally, one person can contribute a new IPP module (or "stage"), many can use
- Cannot handle high-dimensional *individual* state

Accuracy

Suppose IPP is implemented:

$$\text{IPP} : \left(A_\Gamma^{V,\text{end}}, A_\Gamma^{\Lambda,\text{start}}, S_\Gamma\right) \mapsto \left(A_\Gamma^{V,\text{start}}, A_\Gamma^{\Lambda,\text{end}}\right).$$

- Immediately correct as a function of $A_\Gamma^{\Lambda,\text{end}}$. Only V needs to be trained
- Formally, IPP represents the solution to a one-period model where $A_\Gamma^{\Lambda,\text{end}}$ represents terminal payoffs
- If prices cannot be solved analytically, two options:
    1. Solve by inner loop around each IPP (slow but accurate)
    2. Introduce new price approximator neural net à la Azinovic et al. (2022)

Benchmark: Krusell-Smith Model

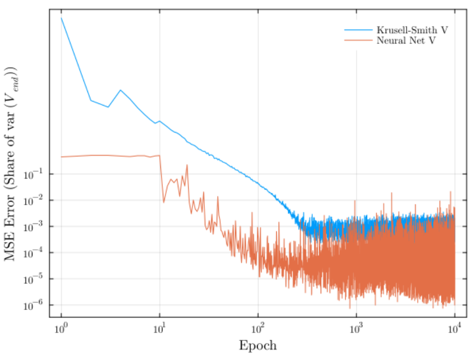
Model Setup

- Standard Krusell-Smith Model Details
- 65 wealth gridpoints, 3 income gridpoints = 195 idiosyncratic gridpoints
- Identical simulate-update_V loop, except V can either be parameterized as:
  1. Neural network Details
  2. Interpolation over k-l-K-A grid Details
- Error function:

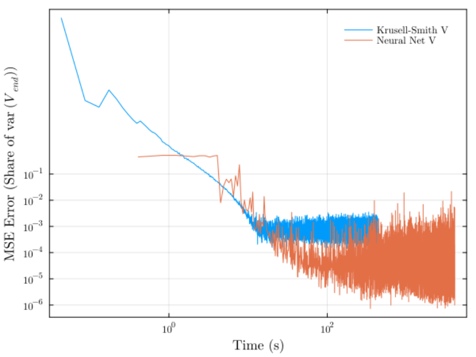
$$\frac{1}{|\mathcal{T}_i|}\sum_{t\in\mathcal{T}_i}\left|A_{it}^{V\text{end}} - \frac{1}{K}\sum_{k=1}^{K} p_k \widehat{A_{i,t+1}^{V\text{start}}}(\Gamma_{it}, \theta_i \mid \varepsilon_{it} = k)\right|^2$$

- Report: V error as share of var(V) across population
- Hardware: One laptop CPU thread (i9-13900H)

Learning Curve Comparison



V error as share of V variance by training epoch



V error as share of V variance by training time

- Both stop improving within about 300 epochs (122s for NN, 13s for KS)
- After Epoch 300, mean $\log_{10}$ error is -4.431 for NN, -3.152 for KS

Conclusion

Conclusion

- Describe a solution method for HA models with agg. shocks that overcomes curse of dimensionality (of aggregate state)
- Method uses neural nets only where needed – solution otherwise conventional
- Much complexity is offloaded to **Intra-Period Problem** (IPP) function
- In other work, provide tools to implement IPP flexibly, easily, performantly

Discussion

- Key advantages:
  - Complex household problems supported
  - No need to train policy network
- Disadvantages:
  - Individual state must be low-dimensional (≤ 6 or so)
  - Prices require inner loop around IPP or price neural net à la Azinovic et al. (2022)
- Future work:
  - Train to tighter tolerance
  - Assess other error metrics, e.g. Euler equation error
  - Compare economics of solutions

Appendix

Algorithm Details

- I use a gridded CDF representation of Λ, but using a finite number of agents is also possible. However, they have to interpolate over continuation V
- Training data for each simulated period is $A_{it}^{V\text{end}}$ together with

$$\mathbb{E}\left[A_{i,t+1}^{V,\text{start}} \mid \Gamma_{it}\right] = \frac{1}{K}\sum_{k=1}^{K} \text{IPP}_V\left(\mathcal{N}(\Omega(\Gamma_{it},k)\ ;\ \theta_i), A_{it}^{\Lambda\text{end}}, \Omega(\Gamma_{it},k)\right)$$

- For large models, if memory is constrained, you can update θ_i as you go, accumulating gradients but not storing the entire simulation

Back

Neural Network Implementation

Neural net $\mathcal{N}(A_{\Gamma}^{\Lambda\text{start}} ; \theta)$ has following components:

1. Generalized-moment of Han et al. (2023):

$$\text{GM}_{\Gamma} = \sum_{j \in J} \left(A_{\Gamma}^{\Lambda\text{start}}\right)_j \mathcal{N}_{\text{GM}}(X_j)$$

2. One layer $(1 \Rightarrow 10)$ neural net on aggregate productivity A
3. Dense feedforward neural net on input: $(X_j, \text{GM}_{\Gamma}, \mathcal{N}_A(A))$
    - Three hidden layers with 8, 8, and 5 neurons
    - Elu activation

Back

Krusell-Smith Model Details

- 65 wealth gridpoints
- 3 income gridpoints
- $\beta = 0.98$
- Income process by Tauchen discretization with persistence 0.95 and std 0.1
- Log-linear wealth grid from 1k to 10m
- Income states: 15.4k, 40.3k, 105.4k
- Risk aversion: 0.9
- Capital share: 0.36
- Depreciation rate: 0.025

Back

Krusell-Smith Method Details

- 5 aggregate capital gridpoints: 100k, 150k, 200k, 250k, 300k
- Linear extrapolation outside aggregate capital grid
- 2 aggregate productivity states: 0.5 and 1.0
- Unlike Krusell and Smith (1998), use gridded CDF population distribution representation for cleaner comparison

Back