Exploiting Symmetry in High-Dimensional Dynamic Programming

Mahdi Ebrahimi Kahou, Jesús Fernández-Villaverde, Jesse Perla and Arnav Sood (2021)

Boston College

April 30, 2024

Overview

- A new method for solving high-dimensional dynamic programming problems with a finite but large number of heterogeneous agents. Accurate and quick.
- Permutation invariance.
- Concentration of measure.
- Design and train deep learning architectures that exploit symmetry and concentration of measure.
- Use multi-firm investment under uncertainty model to demenstrate the performance of the method.

Motivation

- Representative agent or a continuum of agents $\Rightarrow$ a large but finite number of heterogeneous agents.
- Curse of dimensionality: cardinality of the state space is enormous + difficult to compute highly dimensional conditional expectations.
- Example: one state variable productivity high or low, 50 states in the US. Cardinality of the state space is 2^{50}. Does the order matter? Maybe only care how many states have high productivity shock. New cardinality is 51.

Permutation Invariance

- Exploit symmetry in the solution of dynamic programming problems to reduce dimensionality.
- Permutation-invariant function:

$$f(x, \pi X) = f(x, X)$$

  π is a N-dimensional permutation matrix
  Intuition: only care about the distribution of capital among agents, do not care whether agent 12 or 17 have high capital.
- How to exploit it?
  Permutation-invariant dynamic programming problem (Definition 2) $\rightarrow$ policy and value functions are permutation-invariant (Proposition 1) $\rightarrow$ representation (Proposition 2)
- Proposition 2: Representation of permutation-invariant functions

$$f(x, X) = \rho(x, \frac{1}{N} \sum_{i=1}^{N} \phi(X_i))$$

$$\rho : R^{L+1} \rightarrow R, \ \phi : R \rightarrow R^L$$

Concentration of Measure

- Can calculate high-dimensional expectations using single Monte Carlo draw.
- Proposition 3: Concentration of measure when expected gradients are bounded in N

$$P(|f(z^1) - E[f(z)]| \geq \epsilon) \leq \frac{\rho(\Sigma)}{\epsilon^2} \frac{1}{N}$$

  Provide an upper bound on the error of the evaluation of the expectation. It is usually better to approximate $E[f(z)]$ by $f(z^1)$ than by $f(\mathbf{0_N})$

- Intuition: suppose the continuation value function does not depend too much on the states of any given agent. The expected continuation utility will become essentially constant for all draws of (independent) idiosyncratic shocks.

Model

Investment under uncertainty with many firms:

$$
\begin{aligned}
v(x, X) &= \max_{u} \; p(X)x - \frac{\gamma}{2}u^2 + \beta E[v(x', X')] \\
s.t.\, x' &= (1-\delta)x + u + \sigma w + \eta\omega \\
X_i' &= (1-\delta)X_i + \hat{u}(X_i, X) + \sigma W_i + \eta\omega, \;\; for\, i \in \{2, ..., N\} \\
X_1' &= (1-\delta)X_1 + \hat{u}(X_1, X) + \sigma w + \eta\omega
\end{aligned}
$$

where $p(X)$ (the inverse demand function) is assumed to be:

$$
p(X) = \alpha_0 - \alpha_1 \frac{1}{N} \sum_{i=1}^{N} x_i^{\nu}
$$

Deep Learning

- Approximate ρ, ϕ and L using a deep learning architecture $F(\theta)$.

$$u(x, X) = \rho(x, \frac{1}{N}\sum_{i=1}^{N}\phi(X_i))$$

Networks: $\phi(Identity)$, $\phi(Moments)$ and $\phi(ReLU)$ with two layers each with 128 nodes (baseline case). 49.2K, 49.8K, and 66.8K parameters respectively, regardless of N.
ρ is approximated by networks with four layers each with 128 nodes.

- Train the network by minimizing the Euler residuals:

$$\varepsilon(x, X; \theta) \equiv \gamma u(x, X; \theta) - \beta E[P(X') + \gamma(1-\delta)u(x', X'; \theta)]$$

Deep Learning

Case I: $\nu = 1$
Analytical solution: $u(X) = H_0 + \frac{1}{N} H_1 \sum_{i=1}^{N} x_i$
$L = 1$ known but do not supply this information to the architectures.

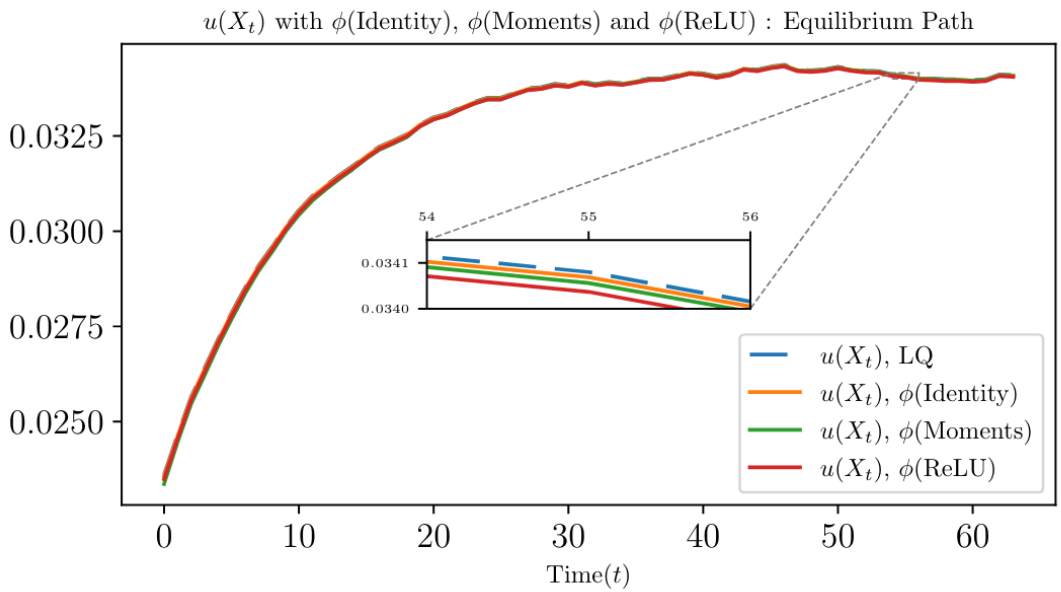

Figure 3: Comparison between the LQ-regulator solution and our three deep learning archictures for the case with $\nu = 1$ and $N = 128$.

Deep Learning

Accuracy of the approximation:

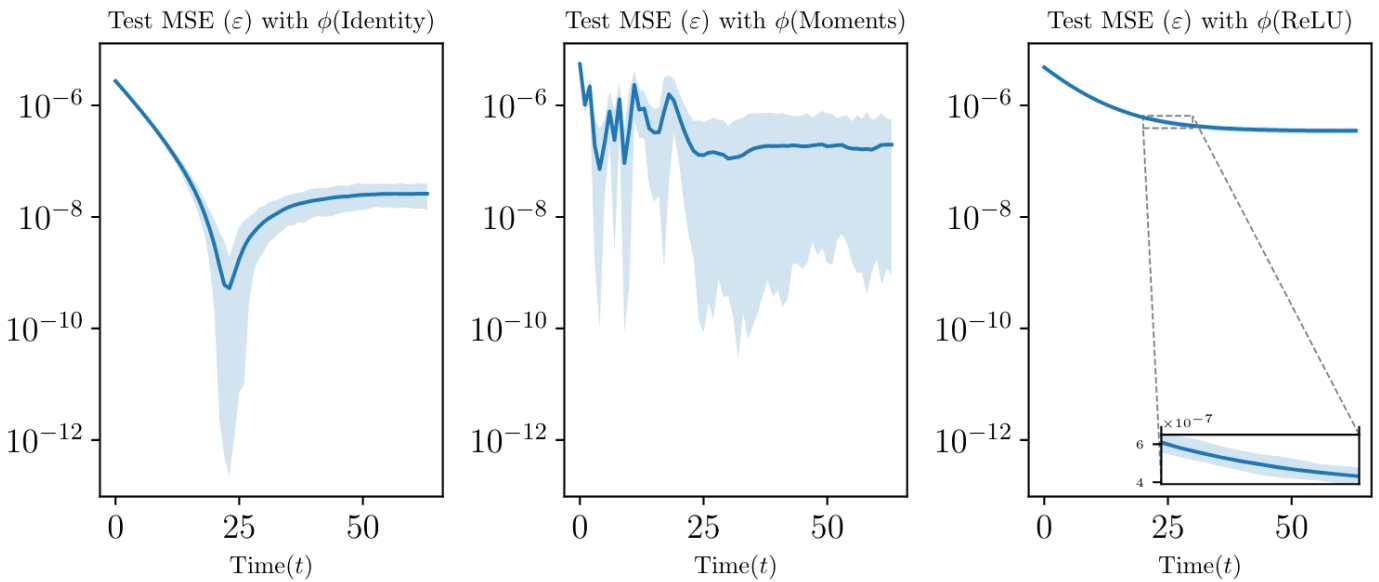

Figure 4: The Euler residuals for $\nu = 1$ and $N = 128$ for ϕ(Identity), ϕ(Moments), and ϕ(ReLU). The dark blue curve shows the average residuals along equilibrium paths for 256 different trajectories. The shaded areas depict the 2.5th and 97.5th percentiles.

The ReLU architecture is especially stable. The extra parameters of this architecture help the function to generalize very consistently.

Deep Learning

Is it practical to implement?

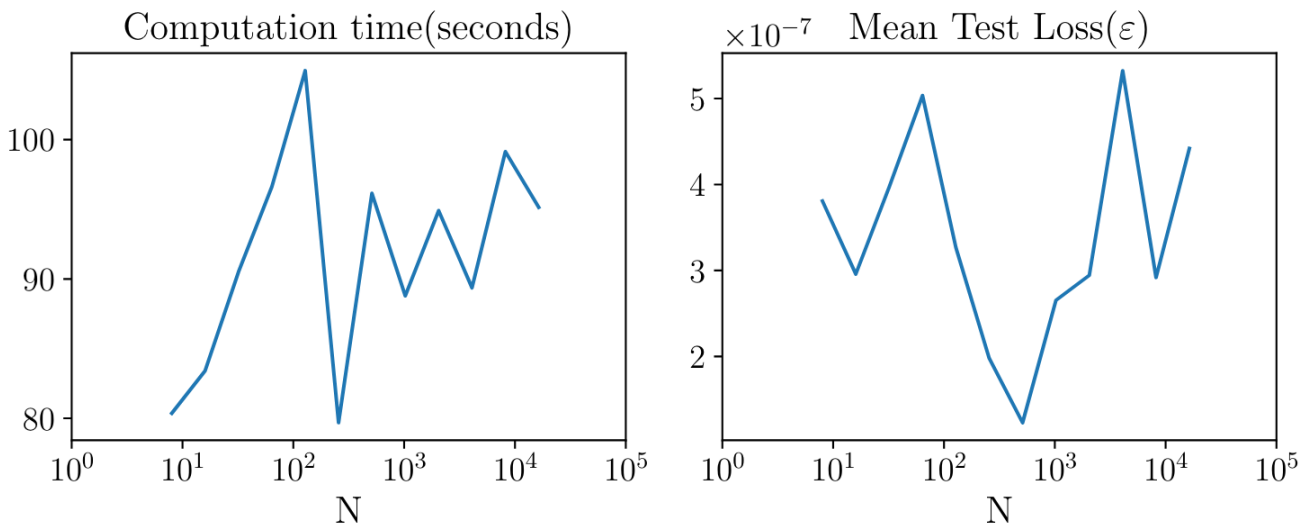

Figure 5: Performance of the ϕ(ReLU) for different N.

Computation time: of order $O(1)$ for reasonable N.

Deep Learning

Explore with different network architectures:

Table 1: Performance of different networks in solving case I: $\nu = 1$

group	description	Time (s)	Params (K)	Train MSE (ε)	Test MSE (ε)	Val MSE (ε)	Policy Error ($\lvert u - u_{\text{ref}} \rvert$)	Policy Error $\left(\frac{\lvert u-u_{\text{ref}} \rvert}{u_{\text{ref}}}\right)$
ϕ(Identity)	Baseline	42	49.4	4.1e-06	3.3e-07	3.3e-07	2.9e-05	0.10%
	Thin (64 nodes)	33	12.4	3.7e-06	2.7e-07	2.7e-07	3.4e-05	0.10%
ϕ(Moments)	Baseline	55	49.8	1.4e-06	7.6e-07	7.6e-07	2.8e-05	0.09%
	Moments (1,2)	211	49.5	2.4e-06	1.1e-06	2.3e-06	4.4e-05	0.14%
	Very Shallow(1 layer)	241	0.6	1.1e-05	8.4e-06	7.9e-06	1.1e-02	34.00%
	Thin (64 nodes)	82	12.6	1.6e-06	9.1e-07	9.2e-07	3.8e-05	0.12%
ϕ(ReLU)	Baseline	107	66.8	3.7e-06	3.3e-07	3.3e-07	2.7e-05	0.09%
	L = 2	86	66.3	1.3e-05	2.1e-07	2.2e-07	2.6e-05	0.08%
	L = 16	91	69.9	5.5e-06	1.5e-07	1.5e-07	2.1e-05	0.07%
	Shallow(ϕ : 1 layer, ρ : 2 layers)	79	17.7	2.0e-06	5.5e-07	5.5e-07	3.2e-05	0.11%
	Deep(ϕ : 4 layers, ρ : 8 layers)	242	165.1	2.1e-03	2.2e-03	2.1e-03	2.7e-03	8.50%
	Thin(ϕ, ρ : 64 nodes)	87	17.0	1.1e-05	4.5e-07	4.5e-07	3.0e-05	0.10%

- Very shallow in $\phi(Moments)$ finds a local minimum. Deep in $\phi(ReLU)$ is unsatisfactory. Explore different architectures.
- Results are better when we use a higher dimension of L.

Deep Learning

Case II: $\nu > 1$
No analytical solution.

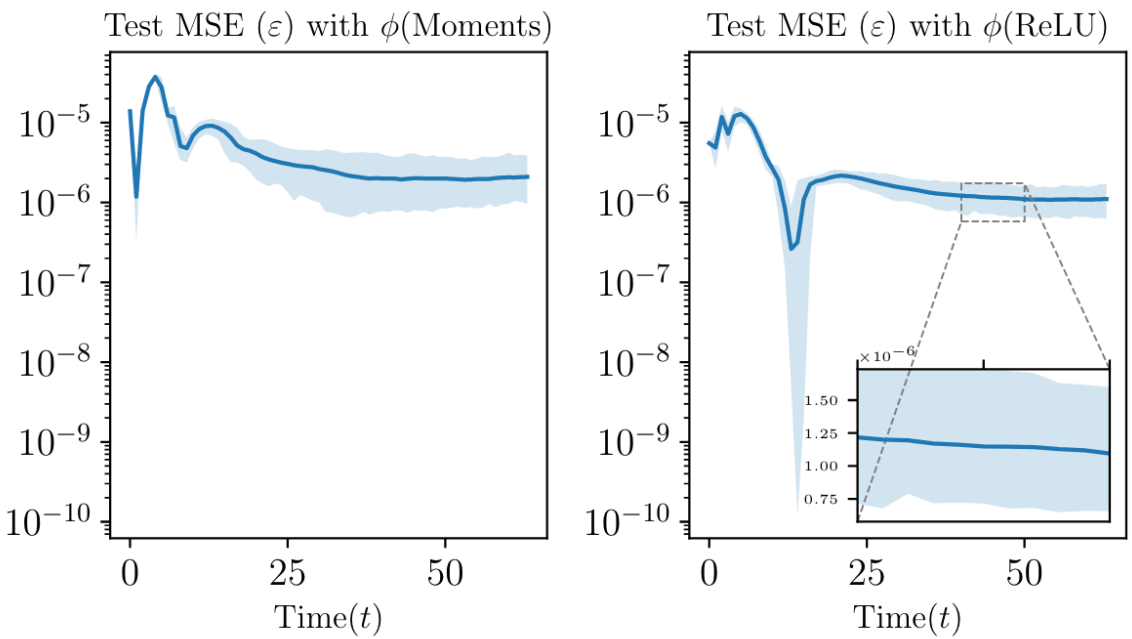

Figure 6: The Euler residuals for $\nu = 1.5$ and $N = 128$ for ϕ(Moments) and ϕ(ReLU). The dark blue curve shows the average residuals along equilibrium paths for 256 different trajectories. The shaded areas depict the 2.5th and 97.5th percentiles.

Extensions

The tools are useful for solving any high-dimensional functional equations with some degree of symmetry, especially when these equations contain high-dimensional expectations.

- Decreasing returns to scale
- Multiple productivity types
- Complex idiosyncratic states
- Global solutions with transitions and aggregate shocks

Discussion

- Solve high-dimensional dynamic programming problems in minutes.
- Double-descent: by increase the number of parameters, we can both fit the data perfectly and achieve outstanding generalization. The cure to over-fitting is to add more parameters.
- While normally we think of a high-dimensional state space as making integrals more difficult, a large number of shocks make expectations easier to calculate by concentration of measure.
- Using stochastic optimization methods with highly overparameterized models tends to robustly find solutions that fulfill a minimum norm solution in the space of approximating functions.(ADAM)