Exploiting Symmetry in High-Dimensional Dynamic Programming

Mahdi Ebrahimi Kahou[1] Jesús Fernández-Villaverde[2] Jesse Perla[2] Arnav Sood[3]

March 25, 2024

[1] Bowdoin College

[3] University of Pennsylvania

[3] University of British Columbia, Vancouver School of Economics

[4] Carnegie Mellon University

Motivation

- Most dynamic models in macro (and other fields) deal with either:
  - Representative agent or few agents.
  - A continuum of agents.
- However, many models of interest in macro (IO and trade) deal with **finite** (but large) number of agents and idiosyncratic/aggregate uncertainty:
  - Industry dynamics with many firms, agents and industries, even models with networks.
- These models are becoming increasingly popular, **but**:
  - They pose computational challenges as we add more agents.
  - **No (non-heuristic)** algorithm exists providing **global** solutions in the presence of aggregate uncertainty.

Challenges: the curse of dimensionality in equilibrium models

Three components to the curse of dimensionality with many agents (Bellman, 1958, p. IX)

1. The cardinality of the state space is enormous.
    - With 266 state variables, with 2 values per state (zero and one), we have more arrangements (2^{266}) than the estimated number of protons in the universe.
2. With idiosyncratic and aggregate shocks we need to calculate high-dimensional conditional expectations.
3. Finding equilibrium paths to the steady-state (ergodic distributions) are extremely hard in high-dimensions.

Contribution I

Inspired by economic theory, providing novel method for **globally** solving high-dimensional heterogeneous agent models with **aggregate shocks** which relies on:

- A **symmetry** present in many heterogeneous agent models, i.e., **exchangeability** of agents.
  - Example: In general equilibrium models the **Walrasian** auctioneer removes indices.
  - The solution (function) must be faithful to this symmetry caused by this exchangeability of agents.

Contribution II

- **Concentration of measures**, something that resembles the law of large numbers to deal with conditional expectations (very fast).
  - More agents makes it easier to forecast the evolution of distributions.
  - Conditional on the aggregate shock, we can use something similar law of large numbers to calculate expectations.
- We show how to implement the symmetry when using **deep neural networks**.

With these we **globally** solve a model with **10,000** agents which was **not possible** before.

Literature Review

- Deep learning as a functional approximation: Maliar et al. (2019), Fernández-Villaverde et al. (2022), Duarte (2018), Azinovic et al. (2022), Han et al. (2021) (a mean-field approach).
- Symmetry in statistics and machine learning: Bloem-Reddy and Teh (2020), Zaheer et al. (2017), and Yarotsky (2018).
- Symmetry in computer science (MDP/RL): Ravindran and Barto (2001) and Narayanamurthy and Ravindran (2008), van der Pol et al. (2020).
- Symmetry in micro and games: Jovanovic and Rosenthal (1988), Hartford et al. (2016)

Background: Deep learning for functional equations

Equilibrium conditions as functional equations

Most theoretical models in economics with equilibrium conditions can be written as functional equations:

- Take some function(s) $\psi \in \Psi$ where $\psi : \mathcal{X} \rightarrow \mathcal{Y}$ (e.g. asset price, investment choice, best-response).
- Domain $\mathcal{X}$ could be state (e.g. dividends, capital, opponents state) or time if sequential.
- The "model" is $\ell : \Psi \times \mathcal{X} \rightarrow \mathcal{R}$ (e.g., Euler and Bellman residuals, equilibrium FOCs).
- The solution is the root of the model (residuals operator), i.e., $\mathbf{0} \in \mathcal{R}$, at each $x \in \mathcal{X}$.

Then a **solution** is a $\psi^* \in \Psi$ where $\ell(\psi^*, x) = 0$ for all $x \in \mathcal{X}$. How do we find an approximate solution?

Classical solution method for functional equations

Quick **review** of collocation-like methods:

1. **Pick** finite set of D points $\hat{\mathcal{X}} \subset \mathcal{X}$ (e.g., a grid).
2. **Choose** approximation $\hat{\psi}(\cdot;\theta) \in \mathcal{H}(\Theta)$ with coefficients $\Theta \subseteq \mathbb{R}^M$ (e.g., Chebyshev polynomials).
3. **Fit** with nonlinear least-squares

$$\min_{\theta \in \Theta} \sum_{x \in \hat{\mathcal{X}}} \ell(\hat{\psi}(\cdot;\theta), x)^2$$

   If $\theta \in \Theta$ is such that $\ell(\hat{\psi}(\cdot;\theta), x) = 0$ for all $x \in \hat{\mathcal{X}}$ we say $\hat{\psi}(\cdot;\theta)$ **interpolates** $\hat{\mathcal{X}}$.

4. The goal is to have good **generalization**:
   - The approximate function is close to the solution outside of $\hat{\mathcal{X}}$.

Deep Neural Networks

Deep learning is **highly-overparameterized** $\mathcal{H}(\Theta)$ ($M \gg D$) designed for good generalization.

- Example: one layer neural network, $\hat{\psi} : \mathbb{R}^Q \to \mathbb{R}$:

$$\hat{\psi}(x;\theta) = W_2 \cdot \sigma\left(W_1 \cdot x + b_1\right) + b_2$$

- $W_1 \in \mathbb{R}^{P \times Q}$, $b_1 \in \mathbb{R}^{P \times 1}$, $W_2 \in \mathbb{R}^{1 \times P}$, and $b_2 \in \mathbb{R}$.
- $\sigma(\cdot)$ is a nonlinear function applied element-wise (e.g., $\max\{\cdot, 0\}$).
- $\Theta \equiv \{b_1, W_1, b_2, W_2\}$ are the coefficients, in this example $M = PQ + P + P + 1$.
- Making it "deeper" by adding another "layer": $\hat{\psi}(x;\theta) \equiv W_3 \cdot \sigma(W_2 \cdot \sigma(W_1 \cdot x + b_1) + b_2) + b_3$.
- Very flexible to design $\mathcal{H}(\Theta)$ using economic insights (e.g., encode symmetry).
- Composing $\mathcal{H}(\Theta)$ from multiple function (e.g., deeper) tend to **generalize better** better in practice.

Over-parameterization and convergence

If the number of coefficients is much larger than the number of grid points $M \gg D$, there are many different sets of coefficients that achieve interpolation.

- What is going on?
  - Deep neural networks and their optimizers have an inherent **implicit bias** toward a **unique** class of interpolating solutions.
  - Figuring out this property is a very active field in computer science and optimization theory.
  - Converges to **"simple"** (flat) interpolating functions.
    - They have a built-in **Occam's razor**.
- Modern ML, uses high-dimensional **non-convex** optimizations. Does the initialization of the coefficients matter?

Application

How do we pick our application to show how all this works?

- In terms of application, there are two routes:

  1. Introducing a sophisticated application where the method "shines".
  2. Or, applying it to a well-known example.

- If I tell you about a sophisticated application, how do we know our "solution" method works?

- So we study a well-known example (with a twist).

- Study the more sophisticated applications in future projects.

Our application

A variation of the Lucas and Prescott (1971) model of investment under uncertainty with N firms.

Why?

1. Ljungqvist and Sargent (2018), pp. 226-228, use it to introduce recursive competitive equilibria.
2. Simple model that fits in one slide.
3. Under one parameterization, the model has a known Linear-Quadratic (LQ) solution, which gives us an exact benchmark.
4. By changing one parameter, the model is nonlinear, with no known solution. Our method handles the nonlinear case as easily as the LQ case with high accuracy.

Investment under uncertainty

- Industry consisting of $N > 1$ firms, each producing the same good.
- Firm of interest produces output x (x units of capital).
- Thus, the vector $X \equiv [X_1, \dots X_N]^\top$ is the production (or capital) of the whole industry.
- The inverse demand function for the industry is, for some $\nu \geq 1$ (this is our twist):

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

- The firm does not consider the impact of its individual decisions on $p(X)$.
- Due to adjustment frictions, investing u has a cost $\frac{\gamma}{2} u^2$.
- Law of motion for capital $x' = (1 - \delta)x + u + \sigma w + \eta \omega$ where $w \sim \mathcal{N}(0, 1)$ an i.i.d. idiosyncratic shock, and $\omega \sim \mathcal{N}(0, 1)$ an i.i.d. aggregate shock, common to all firms.
- The firm chooses u to maximize $\mathbb{E}\left[\sum_{t=0}^{\infty} \beta^t \left(p(X)x - \frac{\gamma}{2} u^2\right)\right]$.

Recursive problem

The recursive problem of the firm taking the exogenous policy $\hat{u}(\cdot, X)$ for all other firms as given is:

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

Take FOCs and equation using standard steps to write equilibrium as the LOM and Euler equation

$$\boxed{\gamma u(X) = \beta \mathbb{E}\left[p(X') + \gamma(1-\delta)u(X')\right]}$$

Curse of dimensionality: a closer look

$$\boxed{\gamma u(X) = \beta \mathbb{E}\left[p(X') + \gamma(1-\delta)u(X')\right]}$$

Let's say there are 1000 **agents**:

1. The domain of the unknown functions of interest $u(\cdot)$ is 1000-**dimensional**.
2. The expectation operator $\mathbb{E}\left[\cdot\right]$ is a 1001-**dimensional** integral.
    - 1000 idiosyncratic shocks + 1 aggregate shock.
3. What about the stationarity of the solution and the **transversality condition**?
    - I will come back to this.

General class of problems: A “big X, little x” dynamic programming

$$
\boxed{\begin{aligned}
v(x, X) &= \max_u \left\{ r\left(x, u, X\right) + \beta \mathbb{E}\left[v(x', X')\right] \right\} \\
\text{s.t. } x' &= g\left(x, u\right) + \sigma w + \eta \omega \\
X' &= G\left(X\right) + \Omega W + \eta \omega \mathbf{1}_N
\end{aligned}}
$$

1. x is the individual state of the agent.

2. X is a vector stacking the individual states of all of the N agents in the economy.

3. u is the control variable.

4. w is random innovation to the individual state, stacked in $W \sim \mathcal{N}(\mathbf{0}_N, \mathbf{I}_N)$ and where, w.l.o.g., $w = W_1$.

5. $\omega \sim \mathcal{N}(0, 1)$ is a random aggregate innovation to all the individual states.

Permutation Groups

- A permutation matrix is a square matrix with a single 1 in each row and column and zeros everywhere else.

- Let $\mathcal{S}_N$ be the set of all $n!$ permutation matrices of size $N \times N$. For example:

$$S_2 = \left\{ \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \right\}$$

- Multiplying vector $v \in \mathbb{R}^N$ by $\pi \in S_N$ reorders elements of v

Permutation-invariant dynamic programming

Definition

A ‘big X, little x’ dynamic programming problem is a **permutation-invariant dynamic programming problem** if, for all $(x, X) \in \mathbb{R}^{N+1}$ and all permutations $\pi \in \mathcal{S}_N$

1. The reward function r is **permutation invariant**:

$$r(x, u, \pi X) = r(x, u, X)$$

2. The deterministic component of the law of motion for X is **permutation equivariant**:

$$G(\pi X) = \pi G(X)$$

3. The covariance matrix of the idiosyncratic shocks satisfies

$$\pi\Omega = \Omega\pi$$

Permutation invariance of the optimal solution

Proposition

The optimal solution of a permutation-invariant dynamic programming problem is permutation invariant. That is, for all $\pi \in \mathcal{S}_N$:

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

and:

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

Can $u(x, X)$ permutation invariance guide $\mathcal{H}(\Theta)$ choice?

Representation of permutation-invariant functions

Proposition

(based on Wagstaff et al., 2019) Let $f : \mathbb{R}^{N+1} \to \mathbb{R}$ *be a continuous permutation-invariant function under* $\mathcal{S}_N$*, i.e., for all* $(x, X) \in \mathbb{R}^{N+1}$ *and all* $\pi \in \mathcal{S}_N$*:*

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

Then, there exist a latent dimension $L \leq N$ *and continuous functions* $\rho : \mathbb{R}^{L+1} \to \mathbb{R}$ *and* $\phi : \mathbb{R} \to \mathbb{R}^L$ *such that:*

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

Representation of permutation-invariant functions: Discussion and intuition

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

- This proposition should remind you of Krusell-Smith (1998), $L = 1$, $\phi(X_i) = X_i$.
- Key benefit for approximation is the **representation** (ρ, ϕ), **not explicit** dimensionality reduction.
- Fitting a ρ and ϕ rather than f directly leads to **far better generalization** on $\mathcal{X}$. Why?:
  - Imposing structure on $\mathcal{H}(\Theta)$, functions that know a lot about the economic problem.
- In practice: $L \ll N$ generalizes very well. ⏩ Regression Example

High-dimensional expectation

Euler's equation:

$$\beta \mathbb{E}\left[p(X') + \gamma(1-\delta)u(X')\right]$$

- Linear example: $p(X') = 1 - \frac{1}{N}\sum_{i=1}^{N} X_i'$.
- Conditioned on the aggregate shock ω, **law of large numbers**:

$$\mathbb{E}\left[p(X') \mid \omega\right] \approx p(X')|\omega, \quad \text{for large } N.$$

- Can we say the same thing about $u(X')$?

$$\mathbb{E}\left[u(X') \mid \omega\right] \approx u(X')|\omega, \quad \text{for large } N?$$

- **Yes**, but $u(\cdot)$ has to satisfy some properties.

Expected gradient bounded in N

Definition (Expected gradient bounded in N)

Let $f : \mathbb{R}^N \to \mathbb{R}$ be a bounded function in N and $z \sim \mathcal{N}(\mathbf{0}_N, \mathbf{I}_N)$ be a normalized Gaussian random vector. The function f has its expected gradient bounded in N if there exists a C such that:

$$\mathbb{E}\left[\|\nabla f(z)\|^2\right] \leq \frac{C}{N},$$

where C does not depend on N.

$$\mathbb{E}_W\left[\|\nabla u(X')\|^2\right] \leq \frac{C}{N}$$

- The policy to be well-behaved (non-explosive gradients).
- Other agent's influence vanishes.

Main result II: Concentration of measure

Proposition

Suppose $z \sim \mathcal{N}(\mathbf{0}_N, \Sigma)$, where the spectral radius of Σ, denoted by $\rho(\Sigma)$, is independent of N, z^1 a draw from z, and $f : \mathbb{R}^N \to \mathbb{R}$ is a function with expected gradient bounded in N. Then:

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

- As Ledoux (2001) puts it: "A random variable that depends in a Lipschitz way on many independent variables (but not too much on any of them) is essentially constant."
- With concentration of measure, dimensionality is not a curse; it is a blessing.

Implication: We can calculate $\mathbb{E}_W[u(X')|\omega]$ with a *single draw* of idiosyncratic shocks W:

- $\mathbb{E}_W[u(X')|\omega] \approx u(x', X')|\omega$.
- Reducing an $N+1$-dimensional conditional expectation to a 1-D one (with good approximation).

Summarizing results

- The structure symmetry imposes on the functions leads to better **generalization**
    - Functions extrapolate better outside of the grid points $\hat{\mathcal{X}}$.
- Concentration of measures provides a fast method for calculating the conditional expectations.
    - Calculate with **one draw** of the idiosyncratic shocks (conditional on the aggregate shock).
- **No non-heuristic** algorithm exists to solve this problem.

Solving the Model

Design of $\mathcal{H}(\Theta)$: Deep learning architectures

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

Three cases for ϕ:

1. Identity function: One moment → ϕ(Identity).
2. Up to degree four polynomials: 4 moments → ϕ(Moments).
3. A deep neural network for ϕ, with $L = 4 \rightarrow \phi$(ReLU).

If polynomials for ϕ: A finite set of moments à la Krusell-Smith.

- In all cases, ρ is a highly over-parameterized neural network with four layers.
- The baseline ϕ(Identity), ϕ(Moments), and ϕ(ReLU) have $49.4K$, $49.8K$, and $66.8K$ coefficients.

Solution method follows “interpolation” methods

1. **Pick**: $\hat{\mathcal{X}}$ as simulated trajectories from X_0:
   - Only need 100 to 1000 points regardless of dimensionality of the state space N.
   - Because we use economic insight, i.e., symmetry which gives us good generalization.
2. **Choose**: Design the $\mathcal{H}(\Theta)$ with ρ and ϕ as discussed:
   - ϕ(Identity), ϕ(Moments), and ϕ(ReLU).

Utilizing concentration of measures:

- **One** draw $\hat{W} = \{\hat{W}_1, \ldots, \hat{W}_N\}$ of the idiosyncratic shocks. For a given $u(\cdot;\theta)$, and aggregate shock ω calculate:

$$X_i' = (1-\delta)X_i + u(X) + \sigma \hat{W}_i + \eta\omega, \quad \text{for } i \in \{1, ..., N\}.$$

Solution method follows "interpolation" methods

- Approximate the Euler residuals

$$\varepsilon\left(X; u(\cdot;\theta)\right) \equiv \gamma u(X;\theta) - \beta\mathbb{E}\left[P(X') + \gamma(1-\delta)u(X';\theta)\right]$$

  using concentration of measures (one draw of W in X'). ⏩ error analysis in N

3. **Fit**: The residuals $\varepsilon\left(X; u(\cdot;\theta)\right)$, that is the "model" i.e., ℓ.

$$\min_{\theta\in\Theta} \sum_{X\in\hat{\mathcal{X}}} \varepsilon\left(X; \hat{u}(\cdot;\theta)\right)^2$$

4. **How to Verify/Test**: Given the approximate solution simulate new paths from X_0 and check the Euler residuals (ε).

Study two cases: linear ($\nu = 1$) and nonlinear ($\nu > 1$) demand functions

Case 1: Linear to verify algorithms and methods

- With $\nu = 1$, we have a linear demand function: $p(X) = 1 - \frac{1}{N}\sum_{i=1}^{N} X_i$.
- It generates a Linear-Quadratic (LQ) dynamic programming problem (only the mean of X_i matters).
- We can find the exact $u(x, X)$, LQ has algebraic solutions.
- The LQ solution gives us a benchmark against which we can compare our deep learning solution.
- The neural network figures out very quickly that the solution is $u(x, X) = H_0 + \frac{1}{N} H_1 \sum_{i=1}^{N} X_i$ and finds a high-dimensional approximation which matches that for the training grid.

Equilibrium Paths: Linear case

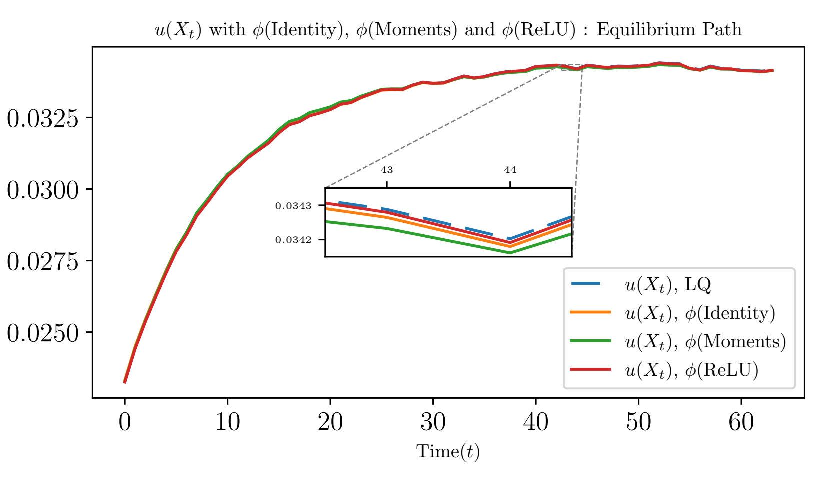


Figure 1: Comparison between baseline approximate solutions and the LQ solution for the case with $\nu = 1$ and $N = 128$.

Accuracy of Solutions: Linear case

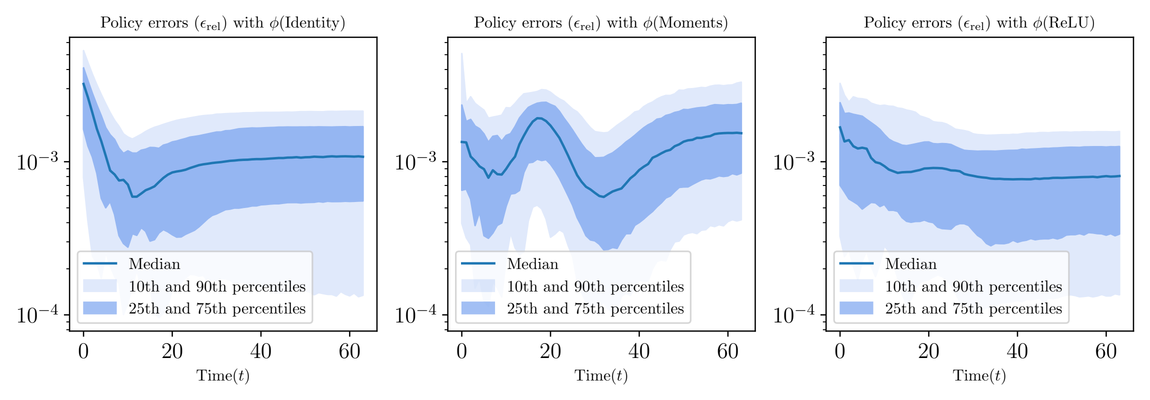


Figure 2: The relative errors for $\nu = 1$ and $N = 128$ for ϕ(Identity), ϕ(Moments), and ϕ(ReLU). The dark blue curve shows the average residuals along equilibrium paths for 256 different trajectories.

Computation time: Linear case

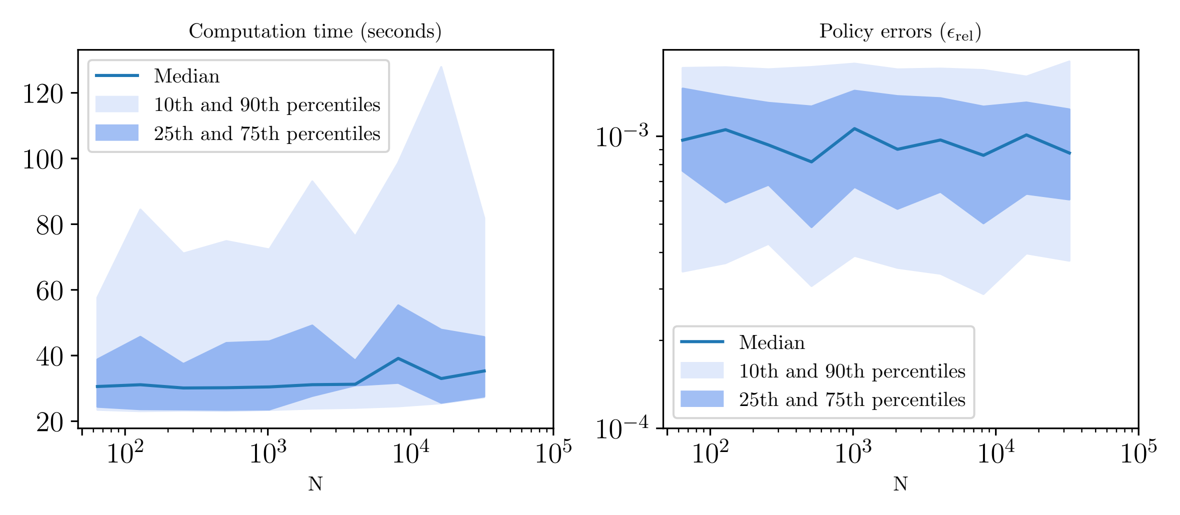


Figure 3: Performance of the ϕ(ReLU) for different N (median value of 21 trials).

Case 2: Nonlinear case with no "closed-form" solution

- With $\nu > 1$, we have a nonlinear demand function: $p(X) = 1 - \frac{1}{N}\sum_{i=1}^{N} X_i^{\nu}$.
- Notice how, now, the whole distribution of X_i matters.
- But we can still find the solution to this nonlinear case using exactly the same functional approximation and algorithm as before.
- We do not need change anything in the code except the value of ν.
- Since the LQ solution no longer holds, we do not have an exact solution to use as a benchmark, but can check residuals.
- Same model and method. Computation time by N nearly the same to linear case
  - Link to the code

Euler residuals: Nonlinear case

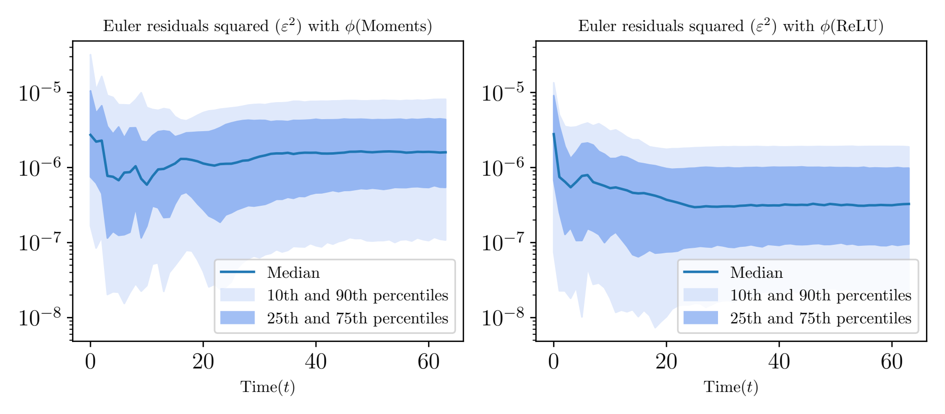


Figure 4: The Euler residuals for $\nu = 1.5$ and $N = 128$ for ϕ(Moments) and ϕ(ReLU). The dark blue curve shows the average residuals along equilibrium paths for 256 different trajectories. The shaded areas depict the 2.5th and 97.5th percentiles.

Equilibrium paths: Nonlinear case

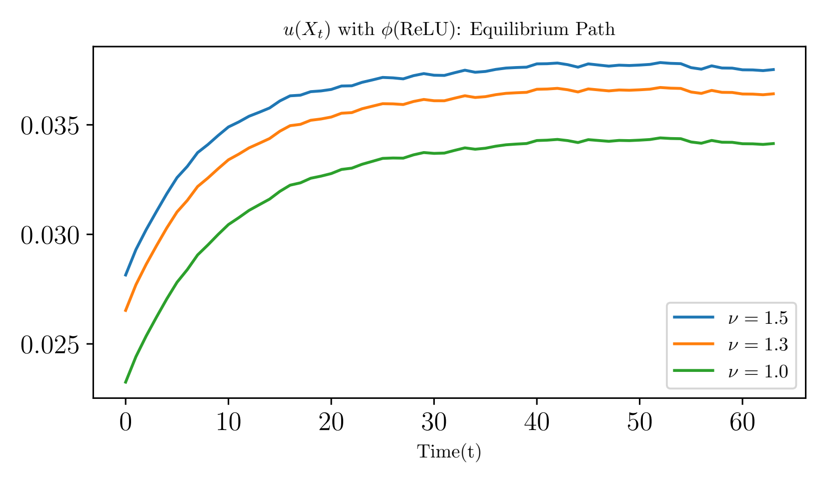


Figure 5: The optimal policy u along the equilibrium paths for $\nu = [1.0, 1.05, 1.1, 1.5]$ and $N = 128$. Each path shows the optimal policy for a single trajectory.

Some challenging question: Generalization puzzle

Question I: Generalization

- From statistical learning and numerical analysis we know:
  - More coefficients in the family of parametric functions $\mathcal{H}(\Theta)$ leads to over-fitting and poor generalization (bias-variance trade-off).
  - We have $70K$ parameters, and $< 1K$ grid points.
  - The results indicate the opposite: More coefficients → better generalization.

How come we achieve great generalization?

Some challenging questions: Multiplicity and transversality puzzle

Question II: Multiplicity and transversality

$$\gamma u(X) = \beta \mathbb{E}\left[p(X') + \gamma(1-\delta)u(X')\right]$$
$$X_i' = (1-\delta)X_i + u(X) + \sigma W_i + \eta\omega, \quad \text{for } i \in \{1, ..., N\}$$

with linear prices. Guess and verify with $u(X) \equiv H_0 + \frac{1}{N}H_1\sum_{i=1}^{N} X_i$

- The Euler equation is quadratic → **two** solutions: $(H_0^-, H_1^-), (H_0^+, H_1^+)$:
  - $H_1^- < 0 \rightarrow$ **stationary** solution, $H_1^+ > 0 \rightarrow$ **non-stationary** solution.
  - We have **no explicit** device in our algorithm to weed out the second solution.

How come we never observe the non-stationary solution in the results?

Understanding the **implicit bias** of deep neural networks answers both questions. Ebrahimi Kahou et al. (2022,2024) address these two challenging questions.

Implicit bias, Generalization, and Stationarity.

Representation with linear prices

Recall the representation,

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

For $\nu = 1$ we can show that the following **exact solution** holds with our representation

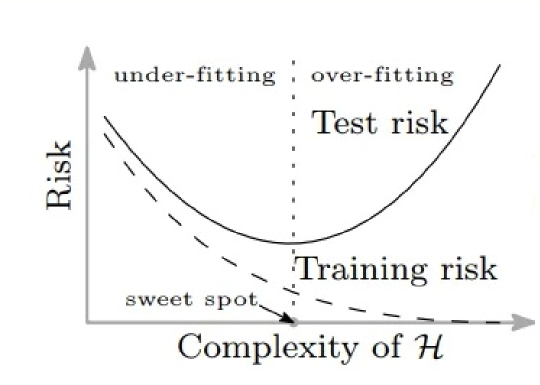
- $\phi(X_n) = X_n$ identity, $L = 1$.
- $\rho(x, y) = \theta_1 + \theta_2 y$.
- Doesn't matter how to generate X since only need 2 points.
- Let's do it with 3 points.

Extreme example of generalizability of neural networks

- **Extreme** case : Forget we know any closed form, and see if over-fitting hurts us.
- Fit **three** grid points in $\mathbb{R}^{512}$ (an economy with $N = 512$ agents).
- Flexible functional form with 17.7 K coefficients.
- Now, evaluate for a whole bunch of reasonable trajectories from the initial condition and check the policy error:
  - 5×10^{-5} MSE of Euler, approximately **0.06%** relative error of $u(X)$.

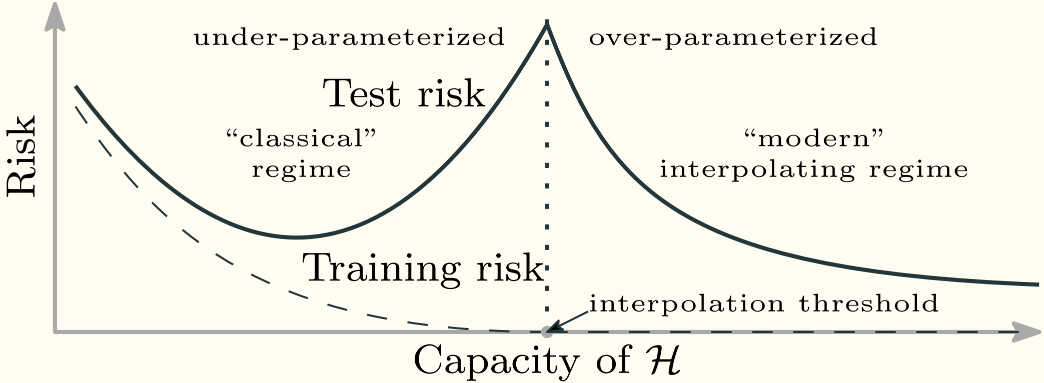
This is related to a literature called **double descent**.

Classical bias-variance trade-off



- Complexity of $\mathcal{H}$: think of
  - Degree of Chebyshev polynomials, polynomials, ...
  - Coefficients of the neural networks (weights and biases)

The cure to over-fitting is to add more parameters



Belkin et al., 2019: Traditional statistics/bias-variance trade-off stop around the interpolation threshold.

Deep learning optimizes in a space of functions

Remember

$$\min_{\theta\in\Theta}\sum_{x\in\hat{\mathcal{X}}}\ell(\hat{\psi}(\cdot;\theta),x)^2$$

- Deep learning: number of coefficients is much larger than the number of grid points.
- Since $M \gg D$, it is possible for $\hat{\psi}$ to interpolate and the objective value will be ≈ 0.
- Since $M \gg D$ there are many solutions (e.g., θ_1 and θ_2),
  - Agree on the grid points: $\hat{\psi}(x;\theta_1) \approx \hat{\psi}(x;\theta_2)$ for $x \in \hat{\mathcal{X}}$.
- Since individual θ are irrelevant it is helpful to think of optimization directly within $\mathcal{H}$

$$\boxed{\min_{\hat{\psi}\in\mathcal{H}}\sum_{x\in\hat{\mathcal{X}}}\ell(\hat{\psi},x)^2}$$

But which $\hat{\psi}$?

Deep learning and interpolation

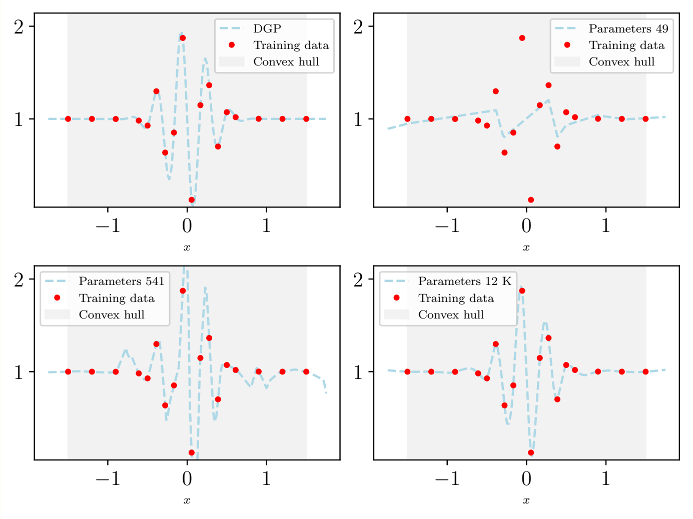
- For M large enough, optimizers **tend to** converge to **unique "simple"** $\hat{\psi}$ (w.r.t to some norm $\| \cdot \|_S$). Unique both in $\hat{\mathcal{X}}$ and $\mathcal{X}$. There is a **bias** toward a specific class of solutions.

- **How to interpret:** interpolating solutions for some functional norm $\| \cdot \|_S$

$$\boxed{\begin{aligned} &\min_{\hat{\psi} \in \mathcal{H}} ||\hat{\psi}||_S \\ &\text{s.t. } \ell(\hat{\psi}, x) = 0, \quad \text{for } x \in \hat{\mathcal{X}} \end{aligned}}$$

  - Comp Sci literature refers to this as the **inductive bias** or **implicit bias**: optimization process is biased toward particular $\hat{\psi}$.
  - Small values of $\| \cdot \|_S$ corresponds to **flat** solutions with **small gradients** (w.r.t. input).

Flat and smooth interpolation: Illustration



Implicit bias: More details

Let ψ_1 and ψ_2 be two differentiable function from a compact space $\mathcal{X}$ in $\mathbb{R}$ to $\mathbb{R}$ such that

$$\int_{\mathcal{X}} \left| \frac{d\psi_1}{dx} \right|^2 dx > \int_{\mathcal{X}} \left| \frac{d\psi_2}{dx} \right|^2 dx$$

then

$$\|\psi_1\|_S > \|\psi_2\|_S.$$

Recently shown the optimizers (first order e.g. SGD) regularize Sobolev semi-norms: Ma, Ying (2021).

Answering the challenging questions

- Answering **generalization puzzle**: Flat interpolation leads to good generalization:
  - If the true underlying functions is flat between (and outside) the points.
  - The cure to over-fitting is to add more parameters.
- Answering **multiplicity puzzle**: In the linear set-up, the explosive solution has larger derivatives (less flat) than the non-explosive one i.e, $|H_1^+| > |H_1^-|$:
  - The deep-learning based solution **automatically** satisfies stationarity.
- Ebrahimi Kahou et al. (2022, 2024) explore this for many more dynamic models in macroeconomics (e.g., neoclassical growth and asset pricing) we show:
  - We can have short- and medium-run accurate solutions without being worried about the long-run behavior.
  - We dont need to calculate the steady-state (ergodic distribution).

Conclusions

Extensions

1. Decreasing returns to scale: the policy becomes a function of x.
2. Multiple productivity types (e.g., two different groups).
3. Complex idiosyncratic states (e.g., a agent is described with more than one variable).

Summarizing our contribution

- **Method** for solving **high-dimensional** dynamic programming problems and competitive equilibria with idiosyncratic and aggregate shocks relying
  - Symmetry.
  - Concentration of measures: Dimensionality is a **blessing** not a curse.
- Using **economic theory** (i.e., exchangeability) and **deep learning** for function approximation with a huge # of parameters ($\gg$ grid points)
  - Achieve great generalization: key to alleviate the curse of dimensionality.
- Implementation
  - Can deal with 10000+ agents.
  - Can deal with 10000+ dimensional expectations with one Monte-carlo draw.

Appendix

Example: Symmetry in a regression problem

Data generating process:

$$y = f(X) = \left(\frac{1}{N} \sum_{i=1}^{N} x_i^{\zeta} \right)^{\frac{1}{\zeta}}$$

- Pick $a \sim \mathcal{U}(a_{\text{min}}, a_{\text{max}})$
- Draw $X = (x_1, \cdots, x_N) \sim \mathcal{N}(a, \sigma^2) \,:\, M$ data points
- $\zeta = 1.5$, $a_{\text{min}} = 1$, $a_{\text{max}} = 2$, $\sigma = 0.2$, $M = 10$
- Generate a set of new data and look at the mean of the relative errors

$$|\frac{\hat{f}(X) - f(X)}{f(X)}|$$

Results: Symmetry in a regression problem

Table 1: Results for the regression problem

$\hat{L}$	1	2	3	4	5	20	NONE
N							
2	1.85%	1.19%	1.91%	1.27%	1.7%	1.75%	1.36%
4	1.04%	1.7%	1.76%	0.76%	0.97%	0.83%	1.65%
8	0.57%	0.57%	0.71%	0.64%	0.6%	0.56%	3.68%
16	0.5%	0.49%	0.51%	0.45%	0.42%	0.48%	4.91%
32	0.32%	0.35%	0.32%	0.28%	0.37%	0.51%	5.69%
64	0.32%	0.27%	0.27%	0.33%	0.28%	0.29%	5.77%
128	0.32%	0.22%	0.24%	0.28%	0.23%	0.22%	5.77%

▸▸ back

Definition (Bounded functions in N)

Let:

$$\mathcal{L}(M) \equiv \{y \in \mathbb{R}^N : |y_i| \leq M \ \forall i = 1, \dots, N\}$$

be an N-dimensional hypercube in $\mathbb{R}^N$. A function $f : \mathbb{R}^N \to \mathbb{R}$ is bounded in N if for every M there exists K_M such that

$$\sup_{y \in \mathcal{L}(M)} |f(y)| < K_M,$$

where K_M is a constant that does not depend on N, but may depend on M.

- Example $f(y) = \frac{1}{N} \sum_{i=1}^{N} y_i \to \sup_{y \in \mathcal{L}(M)} |f(y)| < M$.
- To avoid $f(y) = \sum_{i=1}^{N} y_i \to \sup_{y \in \mathcal{L}(M)} |f(y)| < NM$.

⏩ back

Concentration of measure is the bless of dimensionality

In the linear case we know the closed form solution for u

$$\boxed{\begin{aligned} \hat{\varepsilon}(X; u) - 0 &\sim \mathcal{N}\left(0, \frac{\sigma_{\varepsilon}^{2}}{N}\right) \\ u(\hat{X}') - \mathbb{E}\left[u(X') \mid \omega\right] &\sim \mathcal{N}\left(0, \frac{\sigma_{u}^{2}}{N}\right) \end{aligned}}$$

- Conditional expectation becomes constant as N gets large.
  - One **single Monte-carlo draw** of the idiosyncratic shocks is enough.

▸▸ back

Analytic euler error due to the concentration of measure

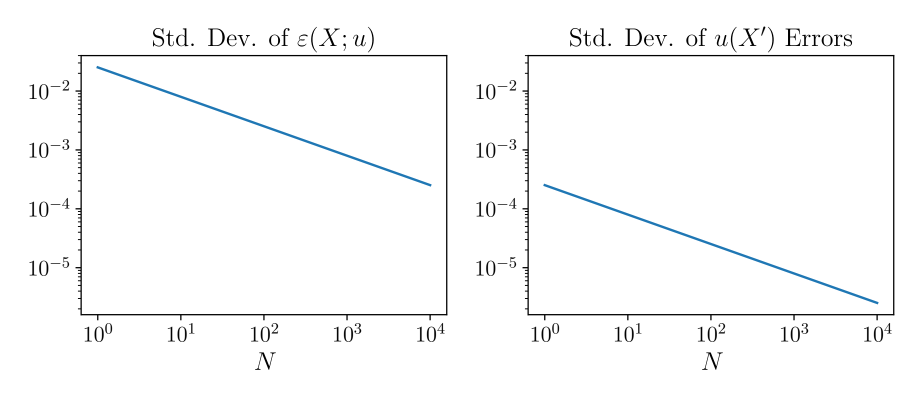


Table 2: Performance of Different Networks in Solving the Linear Model

		Success	Parameters	Time	Train MSE	Val MSE	Test MSE	Policy Error
		(%)	(Thousands, K)	(s)	(ε)	(ε)	(ε)	(ϵ_{rel})
Group	Description							
Identity	Baseline	48%	49.4	28	6.7e-07	4.9e-07	5.0e-07	0.10%
Moments	Baseline: Moments (1,2,3,4)	59%	50.3	36	9.0e-07	8.7e-07	1.2e-06	0.13%
	Moments (1,2)	54%	50.0	33	1.0e-06	8.7e-07	1.0e-06	0.12%
	Very Shallow (1 layer)	0%	0.8	-	-	-	-	-
Deep Sets	Baseline: L$=$ 4	97%	199.6	17	1.3e-06	3.6e-07	3.6e-07	0.09%
	L $=$ 2	93%	201.1	17	1.3e-06	4.0e-07	4.3e-07	0.09%
	L $=$ 16	93%	204.2	14	1.5e-06	3.5e-07	3.5e-07	0.10%
	Deep (ϕ : 2 layers, ρ : 4 layers)	100%	215.9	25	2.0e-06	3.8e-07	3.7e-07	0.10%
	Shallow (ϕ : 1 layer, ρ : 2 layers)	1%	68.0	16	1.6e-07	3.3e-07	3.5e-07	0.10%

Table 3: Nonlinear Model Performance

		Time	Params	Train MSE	Test MSE	Val MSE
		(s)	(K)	(ε)	(ε)	(ε)
group	description					
ϕ(Moments)	Baseline	26	49.8	6.0e-06	5.0e-06	3.8e-06
	Moments (1,2)	27	49.5	8.0e-06	5.1e-06	3.6e-06
	Very Shallow (1 layer)	252	**0.6**	8.3e-06	**1.4e+00**	5.0e-06
	Thin (32 nodes)	66	3.2	1.1e-05	9.7e-06	4.4e-06
ϕ(ReLU)	Baseline	60	67.1	1.4e-05	4.7e-06	3.3e-06
	L = 8	73	68.1	1.1e-05	4.9e-06	2.0e-06
	L = 16	72	70.2	1.5e-05	5.4e-06	1.7e-06
	Very Shallow (ϕ, ρ : 1 layer)	136	**1.4**	8.9e-06	**4.8e+06**	4.9e-06
	Shallow (ϕ, ρ: 2 layers)	47	34.3	1.0e-05	9.2e-06	2.8e-06
	Thin (ϕ, ρ :32 nodes)	52	4.5	1.3e-05	6.0e-06	2.7e-06

Parameters

- $\gamma = 90$, $\beta = 0.95$, $\sigma = 0.005$, $\eta = 0.001$.