Exploiting Symmetry in High-Dimensional Dynamic Programming

Mahdi Ebrahimi Kahou* Jesús Fernández-Villaverde† Jesse Perla* Arnav Sood§

December 7, 2021

Abstract

We propose a new method for solving high-dimensional dynamic programming problems and recursive competitive equilibria with a large (but finite) number of heterogeneous agents using deep learning. We avoid the curse of dimensionality thanks to three complementary techniques: (1) exploiting symmetry in the approximate law of motion and the value function; (2) constructing a concentration of measure to calculate high-dimensional expectations using a single Monte Carlo draw from the distribution of idiosyncratic shocks; and (3) designing and training deep learning architectures that exploit symmetry and concentration of measure. As an application, we find a global solution of a multi-firm version of the classic Lucas and Prescott (1971) model of investment under uncertainty. First, we compare the solution against a linear-quadratic Gaussian version for validation and benchmarking. Next, we solve the nonlinear version where no accurate or closed-form solution exists. Finally, we describe how our approach applies to a large class of models in economics.

Keywords: Dynamic programming; deep learning; breaking the curse of dimensionality.
JEL codes: C45, C60, C63.

*University of British Columbia, Vancouver School of Economics; †University of Pennsylvania; and §Carnegie Mellon University.

‡ We would like to thank Fernando Álvarez, Benjamin Bloem-Reddy, Samira Ebrahimi Kahou, Kevin Leyton-Brown, Vincent Michalski, Aviv Nevo, Yaniv Plan, John Rust, Kevin Song, and participants at many seminars for useful comments. Excellent research assistance was provided by Amedeus Dsouza, Alim Faraji, and Wenxin Ma. A `Github` repo with code for this paper is at https://github.com/HighDimensionalEconLab/symmetry_dynamic_programming.

1 Introduction

We propose a new method for solving high-dimensional dynamic programming problems and recursive competitive equilibria with a large (but finite) number of heterogeneous agents. The key to our approach is to exploit symmetry in dynamic programming and concentration of measure to build deep neural networks that are remarkably efficient, generalize well across the entire state space, and are easy to train.

Traditionally, most models in macroeconomics (and other fields) have dealt with either a representative agent (the canonical real business cycle and New Keynesian models) or a continuum of agents (the Aiyagari and Krusell-Smith models). Some models take an intermediate position with two (or a few) agents, for example, models of financial frictions (with a borrower and a lender) or models of international business cycles (with a home country and a foreign country).

More recently, models with many finite agents are becoming very popular, in large part due to the explosion of available microdata. Think about models of industry dynamics with many firms, models with many locations, or models with many households. In fact, one can argue that we only use models with a continuum of agents in macroeconomics or international trade as a convenient abstraction: the U.S. economy had around 128.45 million households in 2020, not an infinite number of them.

Unfortunately, dealing with multi-agent models is challenging, as we quickly bump into the curse of dimensionality (Bellman, 1958, p. ix). As soon as we have more than a few agents, it becomes nearly impossible to solve these models or take them to the data. As noted by Powell (2007), there are two sources for the curse of dimensionality in dynamic programming. First, the cardinality of the state space is enormous. Second, it is difficult to compute highly dimensional conditional expectations.

To illustrate these two sources, consider the classic multi-region business cycles model of Backus et al. (1992). In this model, each location (i.e., a country or a region) is characterized by seven state variables. Let us think about a version of this model applied to study the 50 U.S. states. The representative household in California (or any other state) must keep track of its own seven states *and* the seven state variables in each of the other 49 locations of the model, for a total of 350 state variables. Using just a few points in the grid of each state variable (if we rely on value function iteration as our solution method) or employing a few Chebyshev polynomials in each dimension of the problem (if we are using a projection method) will require an amount of computer memory beyond feasibility. Even the most aggressive sparse grid methods à la Brumm and Scheidegger (2017) cannot handle 350 state variables.

However, not only must the representative household in California keep track of 350 state variables, but also must it compute the conditional expectation of its continuation value function over these 350 states. This computation requires either large sums (if the states are finite) or complex highly dimensional integrals (if the states are continuous). Both tasks become prohibitively

expensive as soon as the number of state variables grows above single digits.

One could try to tackle this problem using perturbation methods (as Backus et al., 1992, do), but that approach will fail as soon as we incorporate interesting features into the model, such as occasionally binding constraints, or if we are interested in non-ergodic properties of the model (e.g., models without a steady state or with large transitional dynamics).

This simple example might seem discouraging: we cannot even compute a simple economy! But it also suggests intriguing possibilities. For example, does the representative household in California need to keep track of the states of every single other region in the model? Just for a moment, let us imagine that we have a version of the model with only one state variable: a productivity shock, which can be high or low. Following the idea of "exchangeability" of Pakes and McGuire (2001), under an appropriate symmetry assumption, we only need to keep track of how many regions have a high productivity shock in this period. The indices (e.g., whether California or Texas is the region with high productivity today) are irrelevant. The cardinality of the new state space becomes much smaller: instead of being 2^{50} (all the combinations of high/low productivity across the 50 regions), the cardinality is 51 (do we have zero, one, ..., or 50 regions with high productivity?).

But, much more importantly, any function representing the solution to the problem above belongs to the family of symmetric functions, i.e., the solution is a function on a set rather than on a vector space (Macdonald, 1998). To formalize this idea, we will introduce the notion of permutation-invariant dynamic programming and describe how we can use powerful theorems to represent functions invariant to permutations. This theorem can lead to an exponential reduction in the dimensionality of the state space, and it sets up a suitable geometry for deep learning approximations.

This result is very general: it holds in each case where we can find symmetry in the problem. Since we can include heterogeneity among the agents (e.g., the representative household in California is more risk-averse than the representative household in Vermont) in the definition of the states, there will be plenty of cases where symmetry holds.[1] In fact, most heterogeneous agent models in economics have a latent but obvious symmetry imposed by the presence of a Walrasian auctioneer. In general equilibrium, the auctioneer collects all the excess supplies of each agent, aggregates them, and sells them. This aggregation removes the indices of agents in the economy and the solution of the model is invariant under all the permutations of other agents' states.[2] For

[1] Although there are also cases where symmetry does not hold because we cannot easily track heterogeneity as an extra state variable, for example, if the representative household in California has a CRRA utility function and the representative household in Vermont has a CARA utility function. Keeping a whole utility function as a state variable is cumbersome.

[2] The symmetry imposed by the auctioneer is often used implicitly in models with heterogeneous agents. In many papers, the problem is manually transformed so that agents only need to forecast the auctioneer's prices, thereby simplifying the state space, even though the forecast of the prices themselves becomes difficult with aggregate shocks. This transformation is at the heart of Krusell and Smith (1998) and the recent literature exploiting functional derivatives with perturbations such as Bilal (2021).

symmetry to hold, we only require that the agents behave the same when they happen to have the same state values. In other words, if Texas has the same capital, population, and other state variables as California, it would behave as California does.

Also, our functional representation theorem sheds light on why the method in Krusell and Smith (1998) works so well so often. In many models, the moments of the distribution of agents perform the dimensionality reduction required by our theorem either exactly or nearly exactly. In other words, the spirit of our method is to take the "big K, little k" approach and generalize it to a "many big K, one little k" where each "k" can accurately forecast functions of the evolution of the other "K" that are invariant to relabeling and permutation. Our approach has two clear advantages. First, we can build from theorems that tell us which representation functions we need, instead of relying on the intuition of the researcher. Second, and we will argue momentarily, our representation theorem is easily implementable with deep neural networks.[3]

But our argument goes even further. Suppose we have many agents and there is an underlying symmetry assumption. Will we not have something that resembles a law of large numbers when computing the conditional expectation of the continuation value function? Intuitively, suppose the continuation value function does not depend too much on the states of any given agent. In that case, the expected continuation utility will become essentially constant for all draws of (independent) idiosyncratic shocks. Hence, we can calculate the expectation in the continuation utility with a single Monte Carlo draw from the distribution of idiosyncratic shocks (or with a few draws in case we do not have too many agents).

We prove this possibility by relying on recent developments in high-dimensional probability theory and concentration of measure (Barvinok, 2005, and Vershynin, 2018). Loosely speaking, concentration of measure states that a "well-behaved" (Lipschitz continuous) function that depends on lots of independent random variables is essentially constant (when conditioned on variables with a disproportionate impact on the function.). Applying this idea to the continuation value function: if we have many agents and their idiosyncratic shocks are independent, each draw from the distribution of idiosyncratic shocks will deliver, approximately, the same result.

In fact, in models with a continuum of heterogeneous agents, economists already use an extreme form of this result: usually, we assume that a law of large numbers holds in the economy regarding the idiosyncratic shocks. Similarly, the use of random simulations to calculate high-dimensional integrals has a long history in many applied fields, such as in Rust (1997) and Keane and Wolpin (1994). Our paper explains why this feature is natural and pervasive, and how symmetry can simplify the related function approximation, which the literature in past decades

[3]By emphasizing the connection to Krusell and Smith (1998), our method also offers a path forward for models with heterogeneous agents: instead of relying on a continuum approximation, we can have economies with tens of thousands of agents and track the distribution of agents with the same granularity as in the microdata. Alternatively, we can group our agents into bins and use the resulting histogram as a fantastic approximation to a continuum distribution.

could not easily implement as it took place before the recent deep learning revolution.[4]

The beauty of this result is that, instead of having to integrate over the idiosyncratic shocks of each agent and the aggregate shocks to solve the dynamic programming problem of agent i, we only need to integrate over the idiosyncratic shocks of agent i and the aggregate shocks, a much more manageable task. In other words: 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.

With all these results, we define a general class of recursive competitive equilibrium models with heterogeneous agents and establish that the value function and optimal policy of each agent are invariant under permutation of other agents in the economy. Thus, we take advantage of the representation theorems for permutation-invariant functions and concentration of measure to design a deep neural network architecture that solves, in an incredibly efficient way, a permutation-invariant dynamic programming problem. In particular, we choose training points for the deep neural network inspired by the symmetry of the problem. Our strategy is justified by the observation that neural networks are universal approximators, i.e., they can get ϵ-close to any continuous function (Cybenko, 1989, and Hornik, 1991). Also, neural networks are flexible in their design, and we have efficient software (e.g., Pytorch and TensorFlow) and hardware (e.g., GPUs) platforms for training them. Recall, nonetheless, that the neural networks do not cause a reduction of dimensionality: symmetry does (although we use the idea of symmetry in the design of the architecture of the network).

We illustrate our arguments with a multi-firm variation of the classic Lucas and Prescott (1971) model of investment under uncertainty. We pick this model for two reasons. First, this model generates one of the simplest nontrivial recursive competitive equilibria. For instance, Ljungqvist and Sargent (2018, ch. 7) use this model to introduce recursive competitive equilibria to students. Second, by changing just one parameter in the inverse demand function, the model can either have a known and *exact* linear-quadratic (LQ) solution or be fully nonlinear. This makes the Lucas and Prescott (1971) model the perfect testbed for our method: it is simple and interesting, yet it nests cases with known and unknown solutions.

We solve first the case when the inverse demand function is linear both with the LQ approach and our deep neural network. By benchmarking the solution from our deep neural network against the exact LQ solution, we demonstrate that our deep neural network solution is highly accurate even when we have hundreds of firms across a large section of the state space far outside of the trajectories used for training. Then, we solve the case when the inverse demand function

[4] As we will explain in Section 3, drawing from the distribution of idiosyncratic shocks, even just once, is the right thing to do. Instead, setting the shocks to zero (as a naive intuition would suggest if the idiosyncratic shocks are Gaussian) would miss the typical set of the distribution, which in high dimensions does not include the mode of the distribution (i.e., zero). Our argument of drawing once from the distribution of idiosyncratic shocks, thanks to concentration of measure, is different from Judd et al. (2017), who propose precomputing the integrals required in Bellman or Euler equations.

is nonlinear. Our method handles the nonlinear case as easily as the LQ case. In fact, we only need to change one parameter value in the deep learning code that solves the model: everything else is exactly as it was for the linear case. While, in this nonlinear case, we do not have an exact solution to benchmark our deep neural network, we still get a highly accurate solution according to the Euler residuals.

Furthermore, we can solve the global solution when the inverse demand function is linear by only fitting it to two data points, regardless of the number of firms. Given that the closed-form model with linearity only requires two parameters, this should not be a surprise. The interesting point is that we can do so even in cases where the guessed solution is heavily over-parameterized (e.g., if we pretend not to know the exact functional form of the solution and, instead, we try to fit a nonlinear solution with $17.7K$ parameters). An even more exciting result is that this solution is found without directly applying a transversality condition or even using a stationary solution as a special case of a boundary. Despite possibly tens of thousands of parameters, our approach finds the "right" equilibrium, which generalizes well outside of the small number of points used in the approximation.[5]

This example also highlights the importance of deep learning for our approach. In the literature on randomized algorithms in IO and labor (e.g., Rust, 1997, and Keane and Wolpin, 1994) among many others), functions tend to be approximated in relatively low dimensions. The need for this goes beyond computational tractability: it becomes essential for good generalization and extrapolation performance without overfitting. For many of the same reasons, Kalouptsidi (2017) uses LASSO to select the coefficients of approximating functions, tinkering with the classic bias-variance tradeoff to achieve less overfitting and better generalization. Counterintuitively, by radically increasing the number of parameters, we both perfectly fit the data and achieve outstanding generalization due to the "double-descent" phenomena.[6]

To frame our paper within the literature, notice that symmetry has been applied in economics before. For instance, Hartford et al. (2016) use permutation-equivariant symmetry in order to reduce dimensionality to study the best responses in a normal form game. Mertens and Judd (2018) exploit the symmetry of the DSGE models with incomplete markets and a finite but arbitrarily large number of heterogeneous agents to propose a perturbation approach that overcomes the curse of dimensionality. Symmetry is also a key component of anonymous games (Jovanovic and Rosenthal, 1988) and in models of Markov perfect equilibria (Pakes and McGuire, 2001), where the idea of symmetry is known as the "exchangeability assumption" (Aguirregabiria and Nevo, 2013). With respect to this literature, we show that symmetry leads to powerful

[5]The mathematical theory of highly over-parameterized models (when the number of parameters $\gg$ the data points) shows that this class of models, when fitted with stochastic optimization algorithms, generalize by converging to a minimum-norm solution within the space of approximating functions (see Belkin et al., 2019, and Advani et al., 2020). This minimum-norm solution coincides with the equilibrium we are interested in.

[6]Code to implement our methods can be found at https://github.com/HighDimensionalEconLab/symmetry_dynamic_programming.

functional representation theorems and efficient deep neural networks. Notice, as well, that symmetry is implicitly used when economists solve models with a mean-field approximation whenever a continuum of agents is assumed.

There is a literature on Markov decision processes (MDPs) that uses symmetry to reduce dimensionality, but it has mainly focused on discovering symmetries in MDPs rather than exploiting a priori knowledge of symmetry, as we do; see Ravindran and Barto (2001) and Narayanamurthy and Ravindran (2008).

Using neural networks as a family of approximating functions to solve a stochastic dynamic problem in economics can be traced back to Duffy and McNelis (2001). Due to recent substantial hardware and software advances in training deep neural networks, this method has become popular in solving dynamic models in economics. See, among others, Maliar et al. (2019), Fernández-Villaverde et al. (2019), Duarte (2018), and Azinovic et al. (2019).

Finally, recent developments in the machine learning literature have exploited symmetry to significantly alleviate the curse of dimensionality in regression and classification problems. For instance, see Bloem-Reddy and Teh (2020), Zaheer et al. (2017), and Yarotsky (2018). Our choice of functional representation is also closely linked to the literature of symmetric functions; see Prasad (2018) and Zabrocki (2015).

The rest of the paper is organized as follows. Section 2 formally introduces the ideas of permutation-invariant dynamic programming and concentration of measure. Section 3 presents our application. Section 4 introduces our deep learning implementation to solve both the case where the inverse demand function is linear and when it is not. Section 5 further discusses the role of concentration of measure. Section 6 explores the full generality of these techniques. Section Section 7 concludes.

2 Permutation-Invariant Dynamic Programming and Concentration of Measure

This section introduces permutation-invariant dynamic programming and concentration of measure. Permutation invariance will give us a structure in the solution of dynamic programming problems that we can exploit, even when the number of states is large. Concentration of measure will let us deal with the complex conditional expectations that appear in highly-dimensional dynamic programs. In Section 3, we will present a multi-firm model of investment under uncertainty where we can apply these two ideas.

But, before introducing these ideas, we briefly discuss the concept of a permutation matrix and the associated symmetric group of permutation matrices under matrix multiplication.

2.1 The symmetric group of permutation matrices

We denote a column vector of length N of 1s or 0s as $\mathbf{1}_N$ and $\mathbf{0}_N$ respectively. Similarly, we write matrices of size $M \times N$ of 0s or 1s as $\mathbf{0}_{MN}$ and $\mathbf{1}_{MN}$ respectively, and an identity matrix of size N as $\mathbf{I}_N$. Notice that $\mathbf{1}_{NN} = \mathbf{1}_N \mathbf{1}_N^\top$, a result that will be helpful momentarily.

A permutation matrix is a square matrix with a single 1 in each row and column and zeros everywhere else. These matrices are called "permutation" because, when they premultiply (postmultiply) a conformable matrix A, they permute the rows (columns) of A.

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

$$\mathcal{S}_2 = \left\{ \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \right\}.$$

The first element of $\mathcal{S}_2$ is the identity matrix, which leaves rows and columns of a multiplied matrix A unchanged, while the second element swaps the rows or columns of A. The set $\mathcal{S}_N$ forms a group under matrix multiplication that is isomorphic to the symmetric group on N letters (i.e., the indeces of agents in the model). See Appendix A for further notation.

2.2 Permutation-invariant dynamic programming

We want to study a class of dynamic programming problems where the aggregate state vector X can be permuted. Using recursive notation, we start by defining a general dynamic programming problem for environments with many agents.

Definition 1 (A "big X, little x" dynamic programming problem)**.** *Consider the dynamic programming problem:*

$$v(x, X) = \max_u \left\{ r\big(x, u, X\big) + \beta \mathbb{E}\left[v(x', X') \right] \right\} \tag{1}$$

$$s.t.\ x' = g(x, u) + \sigma w + \eta\omega \tag{2}$$

$$X' = G(X) + \Omega W + \eta\omega \mathbf{1}_N. \tag{3}$$

Here, x is the individual state of the agent, X is a vector stacking the individual states of all of the N agents in the economy, u is the control, w is random innovation to the individual state of the agent, which gets stacked in $W \sim \mathcal{N}(\mathbf{0}_N, \mathbf{I}_N)$ where, without loss of generality $w = W_1$, and $\omega \sim \mathcal{N}(0, 1)$ is a random aggregate innovation to all the individual states. Also, $v : \mathbb{R}^{N+1} \to \mathbb{R}$ is the value function, $r : \mathbb{R}^{N+2} \to \mathbb{R}$ is the return function, $g : \mathbb{R}^2 \to \mathbb{R}$ is the deterministic component of the law of motion for x, $G : \mathbb{R}^N \to \mathbb{R}^N$ is the deterministic component of the law of motion for X, and $\mathbb{E}\left[\cdot\right]$ is the conditional expectations operator.

Extending the previous notation to the case where every agent has several state or control

variables is straightforward but tedious. To simplify derivations, from now on, we focus on the case where $r(\cdot)$, $g(\cdot)$, and $G(\cdot)$ are differentiable.

Definition 2 (Permutation-invariant dynamic programming)**.** *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 $\pi \in \mathcal{S}_N$, the reward function r is permutation invariant:*

$$r(x, u, \pi X) = r(x, u, X), \tag{4}$$

the deterministic component of the law of motion for X is permutation equivariant:

$$G(\pi X) = \pi G(X), \tag{5}$$

and the covariance matrix of the idiosyncratic shocks satisfies

$$\pi \Omega = \Omega \pi. \tag{6}$$

The previous definition tells us that we are interested in dynamic programming problems where we care, for instance, about the distribution of capital among the agents in the economy, but not whether the agent with high capital is agent 12 or agent 27. If agent 12 swaps her capital with agent 27 (and other relevant states such as her preference shock or income level), the equilibrium dynamics of the economy will remain unchanged.

The next proposition shows a basic property of permutation-invariant dynamic programming.

Proposition 1 (Permutation invariance of the optimal solution)**.** *The optimal solution of a permutation-invariant dynamic programming problem is permutation invariant. That is, for all $\pi \in \mathcal{S}_N$:*

$$u(x, \pi X) = u(x, X) \tag{7}$$

and

$$v(x, \pi X) = v(x, X). \tag{8}$$

Proof. Appendix C.1 □

Building on our previous example, Proposition 1 tells us that, if the equilibrium dynamics of the economy do not change if agent 12 swaps her capital with agent 27, the policy and value functions of agent 1 will not change either. This straightforward property allows us to write the solution of the problem using a remarkable result regarding the representation of permutation-invariant functions.

Proposition 2 (Representation of permutation-invariant functions)**.** *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 $L \leq N$ and continuous functions $\rho : \mathbb{R}^{L+1} \to \mathbb{R}$ and $\phi : \mathbb{R} \to \mathbb{R}^L$ such that:

$$f(x, X) = \rho\left(x, \frac{1}{N}\sum_{i=1}^{N} \phi(X_i)\right), \tag{9}$$

where X_i is the ith element of X.

Proof. The proof can be found in Wagstaff et al. (2019). □

This result is powerful: if we can find functions ρ and ϕ with a small L, we can represent highly dimensional functions –such as the policy and value functions of models with high N– very efficiently. For instance, consider the case of one extreme permutation-invariant function: the mean function $f(X) = \frac{1}{N}\sum_{i=1}^{N} X_i$ (for clarity, in this paragraph and the next only, we will drop the nuisance x variable, since symmetry does not apply to it). Hence, we have the trivial representation $\phi(X_i) = X_i$ and $\rho(y) = y$, where only $L = 1$ is required.

In comparison, other extreme permutation-invariant functions are harder to represent with Proposition 2. Consider the max function: $f(X) = \max\{X_1, \dots X_N\}$. While there are several ways to represent the max function, all of them require $L = N$.[7]

But the max example should not demoralize us. Proposition 2 is an instance of a general class of functional representation theorems. With some minor changes in the proof of Proposition 2, we can reformulate equation (9) as $f(x, X) = \rho\left(x, \max\{\phi(X_1), \dots, \phi(X_N)\}\right)$. In that case, the representation of the max function becomes trivial, with $\rho(y) = y$, $\phi(X_i) = X_i$, and $L = 1$ (of course, this representation would have difficulty representing the mean function, which would now require $L = N$). Many other variations of equation (9) are possible.

In other words, if some care is put into how we formulate the representation of the dynamic programming problem, we will have that $L \ll N$ for many cases of interest even if Proposition 2 only ensures $L \leq N$. We can use our understanding of the economics of the problem to choose a convenient permutation-invariant reduction in our representation and implement such an approximation in code using what machine learning researchers call a "pooling layer."

Implementing the functional representation In practice, we will not know ρ, ϕ, or L for most problems. But we will be able to train neural networks to approximate ρ, ϕ, and L. Here

[7] For instance, let $\phi(X_i)$ calculate the first N powers of X_i to ensure that the equations are not collinear. Then, $\rho(y)$ solves the system of N equations to reverse-engineer the original $X \in \mathbb{R}^N$ from $y \in \mathbb{R}^N$ before applying the max itself.

we highlight two points. First, even when L is high, an $\widehat{L}$ (our guess for L in the architecture of a network) such that $\widehat{L} \ll L$ can be good enough in terms of numerical accuracy. Let us return to our example of the max function. Despite the theoretical requirement that $L = N$, we can choose $\widehat{L} = 2$. Then, for a fixed α, let $\phi(X) = [e^{\alpha C}, xe^{\alpha X}]$ and $\rho([u, v]) = v/u$. As $\alpha \to \infty$, this approximation will converge to $\max\{X_1, \ldots, X_N\}$ independently of N. Depending on the dispersion of X, this approximation may or may not be practical. Training a high-dimensional neural network for ϕ and ρ with $\widehat{L} = 2$ may find similar representations.

Second, neural networks are very good at fitting cases within a high-dimensional parameter space even when a low-dimensional manifold is sufficient. Thus, if we err by specifying an $L \ll \widehat{L}$ (but still $\widehat{L} \ll N$), the performance of our neural networks is unlikely to deteriorate. To show this, in Table 2, we will solve a model where, thanks to economic theory, we know $L = 1$ would be sufficient. Nonetheless, the performance and accuracy of our approximation are nearly identical if we deliberately ignore that knowledge and set $\widehat{L}$ to 16 and beyond.[8]

Functional representations and the Krusell-Smith method We can link Proposition 2 with Krusell and Smith (1998): $\widehat{L}$ corresponds to the dimensions that an agent uses to predict the law of motion of the distribution of agents in the economy. For example, if only the first two moments are required to keep track of the distribution, an approximation with $\widehat{L} = 2$ here would be sufficient. Nonetheless, researchers have discovered that, even when the distribution is infinite-dimensional, a low $\widehat{L}$ is often sufficient to accurately approximate a distribution.

2.3 Concentration of measure

We complete this section by introducing the idea of concentration of measure. First, we need to have an appropriate notion of boundedness and how it applies to gradients of functions of Gaussian random variables.

Definition 3 (Bounded functions in N)**.** *Let:*

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

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

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

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

[8] The optimization methods used in deep learning –such as reverse-mode auto-differentiation for gradients– can effectively deal with high dimensions if the approximation has been transformed to have an easier geometry, as we do with our representation theorem. It is much easier for these optimization algorithms to ignore moments than to ignore grid points if we were implementing a projection.

A simple example of a symmetric function fulfilling Definition 3 is the mean, $f(X) = \frac{1}{N}\sum_{i=1}^{N} X_i$ since $\sup_{y\in\mathcal{H}(M)} |f(y)| < M$ for all N. The main purpose of this definition is to reject functions that explode in the number of states, such as the $f(X) = \sum_{i=1}^{N} X_i$ since $\sup_{y\in\mathcal{H}(M)} |f(y)| = NM$.

Definition 4 (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.

Loosely speaking, Definition 4 tells us that a function has an expected gradient bounded in N when its value does not change too fast when its Gaussian random arguments vary by a small amount.

The next proposition delivers the intuitive result that, for a large N, a function $f(\cdot)$ with an expected gradient bounded in N is essentially a constant. That is, the measure of its values is "concentrating."

Proposition 3 (Concentration of measure when expected gradients are bounded in N)**.** *Suppose the random variable $z \sim \mathcal{N}(\mathbf{0}_N, \Sigma)$, has a spectral radius, $\rho(\Sigma)$, independent of N. If $f : \mathbb{R}^N \to \mathbb{R}$ is a function with expected gradient bounded in N and z^1 is a single random draw from z, then:*

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

Proof. Appendix C.2 □

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." But, if this is the case, then we can evaluate $\mathbb{E}\left[f(z)\right]$ with a single Monte Carlo draw z^1 from $\mathcal{N}(\mathbf{0}_N, \Sigma)$: $\mathbb{E}\left[f(z)\right] \approx f(z^1)$.

It is usually better to approximate $\mathbb{E}\left[f(z)\right]$ by $f(z^1)$ than by $f(\mathbf{0}_N)$, that is, $f(\cdot)$ evaluated at the mean of the distribution. An N-dimensional Gaussian random variable concentrates on a hypersphere of radius $\sqrt{N}$, rather than at the origin. This hypersphere is the typical set relevant to compute $\mathbb{E}\left[f(z)\right]$ when $f(\cdot)$ is not linear. Also, the distance between the typical set and the origin increases with N grows.[9] Even when N is not sufficiently large for a single Monte Carlo draw to be accurate enough, concentration of measure tells us we can still get high numerical accuracy when we evaluate an expectation by drawing a small number of $\{z^1, \ldots z^M\}$: $\mathbb{E}\left[f(z)\right] \approx \frac{1}{M}\sum_{i=1}^{M} f(z^i)$.

[9] See Vershynin (2018) and Fernández-Villaverde and Guerrón-Quintana (2021) for details.

There is an important difference between the functional representation in Proposition 2, an implicit dimensionality reduction result, and our concentration measures, which provide an upper bound on the error of the evaluation of the expectation. Also, the results of Proposition 3 rely on ensuring that Definition 4 holds, and does not require any symmetry. However, we conjecture that permutation invariance and Definition 3 are sufficient for Proposition 3 to hold (i.e., Definition 4 is implied). Loosely, functions that are invariant to permutations either have a concentration of measure or are unbounded in N. In particular, functions like $f(X) = \sum_{i=1}^{N} X_i$ do not fulfill Definition 3, but are of limited use since expectations diverge as N increases.[10]

This helps explain part of the Krusell and Smith (1998) intuition where "agents" do not need to calculate any integrals, since they can simply use the approximate law of motion of the simulation itself (after conditioning on an aggregate shock, as we do here) to form expectations of the distributional variables. Similarly, continuous-time mean-field games (Kaplan et al., 2018) implicitly use the exchangeability of all agents with identical states to define permutation-invariant functions, and this leads to a concentration of measure for all payoff relevant values.

3 An Application: A Model of Investment

To illustrate how permutation-invariant dynamic programming and concentration of measure operate in a popular economic application, we propose an extension of the classic model of investment under uncertainty by Lucas and Prescott (1971). We will consider an industry with a large number of firms and follow the recursive formulation in Prescott and Mehra (1980). In contrast to Lucas and Prescott (1971), we will not require that all firms have symmetric states, just that they have symmetric policy functions.

More concretely, we study an industry consisting of $N \geq 1$ price-taking firms, each producing a single good using capital under constant returns to scale. By picking appropriate units, we can consider that a firm i produces output x with x units of capital. We can stack the production (or capital) of the whole industry in the vector $X \equiv [x_1, \dots x_N]^\top$.

The capital of the firm depreciates at a rate $\delta \in [0, 1]$, is increased by investment u, and is shifted by an i.i.d. idiosyncratic shock $w \sim \mathcal{N}(0, 1)$ and a single i.i.d. aggregate shock, $\omega \sim \mathcal{N}(0, 1)$, common to all firms (adding persistence for the shocks is trivial but complicates notations). Thus, the law of motion for capital is:

$$x' = (1 - \delta)x + u + \sigma w + \eta\omega.$$

[10] To understand the intuition, consider that the key criterion for a concentration of measure is that a single random variable cannot affect the result a disproportionate amount relative to the others. Permutation invariance implies that individual variables affect the function in a symmetric way, and the same applies to their gradients (i.e., if $f(X)$ is permutation invariant, then $\nabla f(X)$ is permutation equivariant). If the function does not explode as N increases (i.e., Definition 3), then symmetry ensures that all of the elements of $\nabla f(z)$ are of the same scale relative to N, ensuring that Definition 4 is fulfilled.

Although this law of motion allows for capital to be negative, we will consider only the cases where $x > 0$ (for instance, because investment is sufficiently large on average in relation to the standard deviation of the idiosyncratic and aggregate shocks and, therefore, $x < 0$ only occurs with a trivially small probability). Due to adjustment frictions, investing u has a cost, in valuation terms, $c(u) = \frac{\gamma}{2}u^2$.

We assume that the inverse demand function for the industry is, for some $\nu \geq 1$:

$$p(X) = \alpha_0 - \alpha_1 \frac{1}{N} \sum_{i=1}^{N} x_i^{\nu}. \tag{10}$$

When $\nu = 1$, this inverse demand function means that the model can be represented as a linear-quadratic (LQ) dynamic programming problem as described in Ljungqvist and Sargent (2018, Section 7.2), except with $N \geq 1$ firms. When $\nu > 1$, we will have a fully nonlinear model.

Since we will be looking at competitive equilibria where a firm is a price taker, the firm does not consider the impact of its individual decisions on $p(X)$. Also, all firms use their cash flow to finance their investment, u. Hence, the period profits of firm x are:

$$\text{profits} = p(X)x - \frac{\gamma}{2}u^2.$$

The dynamic problem of the firm is to choose u to maximize the expected present discounted value of the firm at discount rate β. The corresponding Bellman equation of the firm takes the evolution of X as given and determined by the equilibrium policy of all firms, including itself.

First, we write such a Bellman equation considering X as a vector, as it is the most common practice in economics.

Bellman equation with vector X The firm of interest with capital x takes a symmetric policy $\hat{u}(\cdot, X)$ as given for all other firms in X. Without loss of generality, we assume that the firm is the first in the vector, i.e., $x = X_1$. With this assumption, we can write the recursive problem of the firm taking the exogenous policy $\hat{u}(\cdot, X)$ as:

$$v(x, X) = \max_{u} \left\{ p(X)x - \frac{\gamma}{2}u^2 + \beta \mathbb{E}\left[v(x', X')\right] \right\} \tag{11}$$

$$\text{s.t. } x' = (1-\delta)x + u + \sigma w + \eta\omega \tag{12}$$

$$X_i' = (1-\delta)X_i + \hat{u}(X_i, X) + \sigma W_i + \eta\omega, \quad \text{for } i \in \{2, ..., N\} \tag{13}$$

$$X_1' = (1-\delta)X_1 + \hat{u}(X_1, X) + \sigma w + \eta\omega. \tag{14}$$

Four points deserve further explanation. First, the expectation in equation (11) is taken over the i.i.d. idiosyncratic shocks $\{w, W_2, \ldots W_N\}$ and the aggregate shock ω. Second, equation (12) is the law of motion of capital of the firm given its control u. Third, equation (13) is the

forecast the firm makes over the evolution of the distribution of capital of all firms given the exogenous policy. Fourth, equation (14) takes care of the contribution of the firm's own policy to the aggregate state X. When $\sigma > 0$, we can let the firm consider that $w = W_1$. When N is sufficiently large, this becomes irrelevant for the optimal u and we could remove the special case of that correlated shock, combining (14) into (13). When $N = 1$, this formulation nests the textbook "big-K, little-k" example derived in Ljungqvist and Sargent (2018), Section 7.2.

Next, we show that it is easy to rewrite the previous problem considering X as a set. While both formulations look similar cosmetically, the conditions for treating the aggregate distribution as a set rather than a vector are essential to our method.

Bellman equation with set X To use our results, we need to show that the investment model described in (11) to (14) satisfies all the conditions of a permutation-invariant dynamic programming problem in Definition 2.

First, to show that the payoff is permutation invariant according to equation (4), it is sufficient for the $p(X)$ function in equation (10) to be permutation invariant. This is easily shown since the function is an affine transformation of a power mean.

Second, the law of motion must be permutation equivariant according to equation (5). Appendix D.1 shows this formally, but the intuition is that, conditional on a permutation-invariant $\hat{u}(\cdot, X)$, we can apply a π to reorder any of the indices in equations (13) and (14), i.e., $X_i' = (1-\delta)X_i + \hat{u}(X_i, X)$ to $X_j' = (1-\delta)X_j + \hat{u}(X_j, \pi X)$ since $\hat{u}(X_j, \pi X) = \hat{u}(X_j, X)$. The permutation equivariance is a direct consequence of symmetric equilibrium.

Finally, to show that the covariance matrix fulfills condition (6) from equations (13) and (14), notice that the idiosyncratic shocks of other agents are embodied in the identity matrix times σ. Therefore, for all permutations $\pi \in \mathcal{S}_N$, $\pi\sigma\mathbf{I}_N = \sigma\mathbf{I}_N$.

Hence, by Proposition 1, the solution to the investment model (u, v) is permutation invariant and can be treated as a function on sets rather than a vector space (i.e., the ordering of the inputs does not matter). If we create a tuple of the states and the corresponding shocks as (X, W), the set equivalent of equations (13) and (14) becomes:

$$\begin{aligned} X' = &\{(1-\delta)\hat{x} + \hat{u}(\hat{x}, X) + \sigma\hat{w} + \eta\omega \text{ for all } (\hat{x}, \hat{w}) \in (X, W) \setminus (x, w)\} \\ &\cup \{(1-\delta)x + \hat{u}(x, X) + \sigma w + \eta\omega\}\,. \end{aligned} \tag{15}$$

As before, the special case of the x in expression (15) is to ensure the firm does not take its choice into account when forecasting X', but can still allow for the correlation of shocks when calculating expectations. While this is important for correctness with very small N –and in particular nesting the $N = 1$ case– it has almost no quantitative impact as N increases. For that reason, we will drop the special case in further calculations in this section, effectively decoupling x from the set X. Since the forecast X' is only used for expectations, this is equivalent to assuming

that the firm does not take into account that its contribution to X is correlated with its own shock when calculating expectations of X'.

Using this simplification, we can rewrite equations (11) to (14) as:

$$v(x, X) = \max_u \left\{ p(X)x - \frac{\gamma}{2}u^2 + \beta \mathbb{E}\left[v(x', X')\right] \right\} \tag{16}$$

$$\text{s.t. } x' = (1-\delta)x + u + \sigma w + \eta\omega \tag{17}$$

$$X' = \left\{ (1-\delta)\hat{x} + \hat{u}(\hat{x}, X) + \sigma\hat{w} + \eta\omega \text{ for all } (\hat{x}, \hat{w}) \in (X, W) \right\}. \tag{18}$$

The expectation in (16) is taken over $w \in \mathbb{R}$, $\omega \in \mathbb{R}$, and $W \in \mathbb{R}^N$, all independent normal shocks. This reformulation allows us to solve for the equilibrium with functions on sets rather than vectors and to use the representation in Proposition 2.

Since the problem fulfills the requirements of Definition 2 for any permutation-invariant $\hat{u}(\cdot, \cdot)$, Proposition 1 tells us that we have a permutation-invariant optimal solution. Since in the symmetric equilibrium every agent solves the same policy, the $\hat{u}(x, X)$ is also permutation invariant, which we needed to ensure that the law of motion, (5), is equivariant. Thus, we are ready to define a symmetric competitive equilibrium.

Definition 5. *An equilibrium is a $v(x, X)$ and $\hat{u}(x, X)$ such that:*

- *Given $\hat{u}(x, X)$, $v(x, X)$ is the value function solving (16) for each agent and $u(x, X)$ is the optimal policy function.*
- *The optimal policy is symmetric, i.e., $u(x, X) = \hat{u}(x, X)$.*

In this paper, we focus on problems for which the optimal solution is unique and can be found via an Euler equation, though our techniques are not specific to that class of models.

Proposition 4. *For a permutation-invariant $p(X)$, the Euler equation takes the form:*

$$\gamma u(x, X) = \beta \mathbb{E}\left[P(X') + \gamma(1-\delta)u(x', X')\right] \tag{19}$$

$$X' = \left\{ (1-\delta)\hat{x} + u(\hat{x}, X) + \sigma\hat{w} + \eta\omega, \text{ for } (\hat{x}, \hat{w}) \in (X, W) \right\}. \tag{20}$$

Proof. See Appendix C.3. □

The economic interpretation of the Euler equation (19) is standard. The firm weighs the marginal cost of an additional unit of investment, $\gamma u(X)$, against the benefit of additional profits from that marginal increase in output at a price $p(X')$ plus the value of investment after depreciation. In this application, the policy becomes independent of x. That result is, however, incidental. If we had some curvature on x in the revenue function (as we will have in an extension in Section 6.1), the policy would not be independent of x.

Dimensionality of the problem While seemingly simple, the dimensionality of the Euler equation (19) is nontrivial. If $N = 1,000$, then $u(x, X)$ is a policy function on a 1,001 dimensional state-space. Furthermore, even given a $u(x, X)$ function, the expectation is taken over 1,001 dimensions. It is not feasible to solve this problem with standard methods.

But hope is not lost. First, think about the expectation in the simple case of $\eta = 0$ but $\sigma > 0$: while considering 1,000 individual forecasts of x' from (20) is computationally intractable, a firm may have an accurate forecast of functions of the distribution of x'. In fact, the firm does not care about the individual x' unless its $u(\cdot, X)$ depends a great deal on a single $\hat{x}$.

In the case where firms do not care about the individual x', a forecast of the distribution is good enough. Furthermore, when $\eta = 0$ and N is large, the distribution evolves almost deterministically. Similarly, even when there are aggregate shocks, an agent may have an accurate forecast of functions of the distribution *conditional* on the realization of an aggregate shock. The extreme case of this result appears in models à la Aiyagari-Krusell-Smith. With a continuum of agents, we only care about the distribution, not the position of each agent on it.

Next, consider the structure of $u(\cdot, X)$. Since the price function was invariant to permutations, we know that the policy function will be invariant as well. Hence, even if the firm cannot accurately predict an individual $\hat{x} \in X$ without large amounts of computations, it only needs to predict the general distribution of X', rather than each element in it.

If $u(\cdot)$ and $p(\cdot)$ are functions of a large number of random $\hat{x}' \in X'$, but do not depend too much on any individual one of them (as defined by the notion of bounded gradients, Definition 4), then the expectation conditional on the aggregate shock may be close to a deterministic function. Then, $p(X')$ and $u(\cdot, X')$ are fully predictable. Thus, for a large enough N, we can exploit Proposition 3 and use a single Monte Carlo draw of individual shocks.

In the case of a linear $p(\cdot)$, we could rely on certainty equivalence and normalize $\hat{w} \in W$ to have zero mean. However, when $p(\cdot)$ is not linear, we draw a random W. As emphasized in Subsection 2.3, high-dimensional expectations of a function $f(\cdot)$ are not well approximated by the function evaluated at its mode, but are well approximated with random draws from the distribution itself.

4 A Deep Learning Implementation

Our goal is to find a policy function, $u(X)$, that satisfies Proposition 4 (we will drop the dependence of u on x when we want to highlight that we are searching for the policy function of all firms). While we are interested in a global solution, we will not require that solutions be accurate for all $X \in \mathcal{X}$. We just want the $u(\cdot)$ to generalize well on paths $\left\{X_t^1, X_t^2, \ldots X_t^J\right\}_{t=0}^T$ that are likely were we to start from some distributional initial conditions $\left\{X_0^1, X_0^2, \ldots X_0^J\right\}$ of interest. Fortunately, we will see that the solutions we find will, quite often, be accurate in a

much larger set of points.

Our strategy is to exploit the representation in Proposition 2:

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

since it encodes the symmetry built into our model. We approximate ρ, ϕ, and L using a deep learning architecture $\mathcal{F}(\theta)$, where θ is the vector of weights in the network. As a result, we get a parameterized policy function $u(x, X; \theta)$ that is as close as possible to $u(x, X)$ under some appropriate metric. This strategy is much more efficient than implementing a blind neural network and resembles similar applications in other environments.[11]

4.1 Architecture of the network

We specify three different network architectures to deal with ϕ. First, we approximate ϕ as a function of a first moment of X à la Krusell-Smith, but allowing this moment to enter in a fully nonlinear way as in Fernández-Villaverde et al. (2019). We call this approximation ϕ(Identity). Second, we approximate ϕ as in the first architecture, but using four moments. We call this approximation ϕ(Moments). Third, we approximate ϕ with a flexible ReLU network, with two layers each with 128 nodes.[12] We call this approximation ϕ(ReLU). Across all of their layers, the baseline ϕ(Identity), ϕ(Moments), and ϕ(ReLU) have 49.4K, 49.8K, and 66.8K parameters respectively, regardless of N.

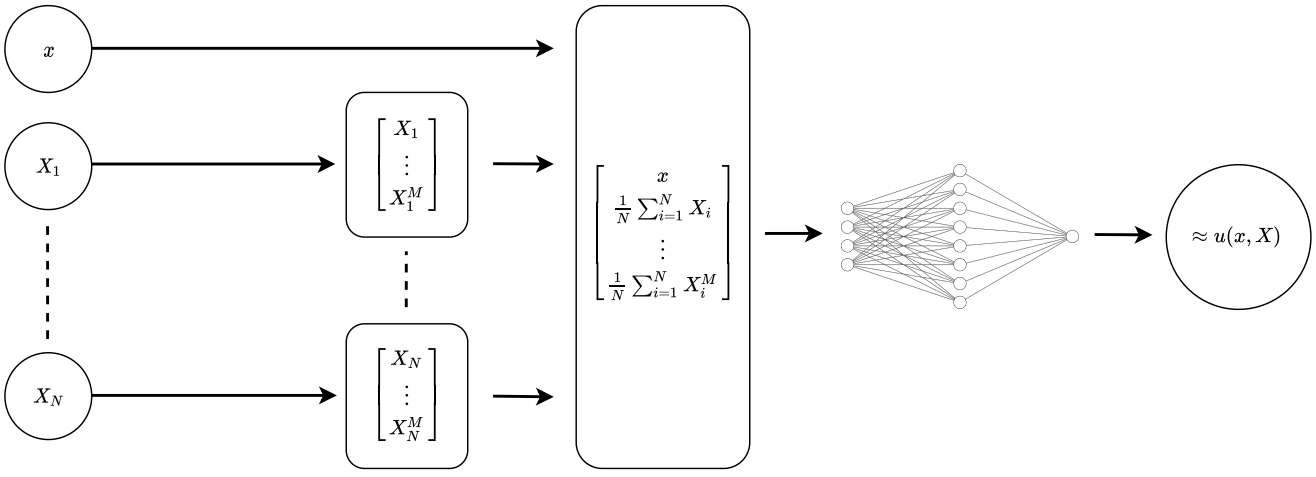

Figure 1: Moments architecture; M is the number of moments.

[11] For instance, Hartford et al. (2016) use permutation-equivariant symmetry to reduce dimensionality in the approximation of best responses in a normal form game.

[12] A rectified linear unit (ReLU) is an activation function $f(x; \theta_1, \theta_2) = \theta_1 \max(0, \theta_2 x,)$, where the max is the point-wise maximum, and θ_1 and θ_2 are weights found in the training procedure.

In these three architectures, ρ is represented by a neural network with four layers, each with 128 nodes.[13] Figures 1 and 2 illustrate these three architectures.

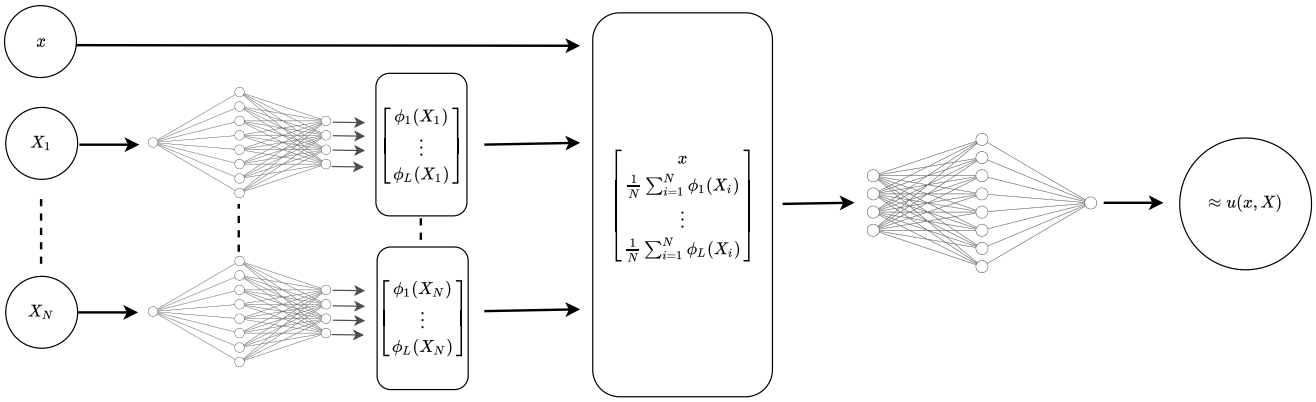

Figure 2: ReLU architecture

4.2 Training the network

We will train the network by minimizing the Euler residuals defined when we plug $u(x, X; \theta)$ into equation (19) instead of $u(x, X)$, and using equation (20) for the law of motion of X':

$$\varepsilon(x, X; \theta) \equiv \gamma u(x, X; \theta) - \beta \mathbb{E}\left[P(X') + \gamma(1-\delta)u(x', X'; \theta)\right]. \tag{21}$$

More concretely, we evaluate $\varepsilon(x, X; \theta)$ using simulations from a single initial condition and only 16 trajectories from $t = 0$ to 63 (including both aggregate and idiosyncratic shocks). We call the evaluated Euler error, in each simulation m and period t, $\varepsilon_{m,t}(x, X; \theta)$. We use the concentration of measure described in Appendix B to calculate the integral over the W shocks in (19). To do so, though, we must condition on the aggregate ω shock to ensure that Definition 4 holds.

Finally, we minimize the sum of square Euler errors:

$$\min_{\theta} \frac{1}{M} \sum_{m=1}^{M} \sum_{t=0}^{T} \left(\varepsilon_{m,t}(x, X; \theta)\right)^2 \tag{22}$$

using ADAM, a stochastic gradient descent with adaptive moment estimation, although any other suitable optimization algorithm could be applicable.

The pseudocode below summarizes our steps.

[13] Experience suggests it is better to be deep than wide for the same number of parameters, though the computer science theory explaining this result is in its infancy.

Algorithm Network training

1: Given by network architecture $\mathcal{F}(\theta)$ for $u(x, X)$, define the Euler residuals:

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

2: Pick $\{X^m(0), ..., X^m(T)\}$ for $m = 1, .., M$ trajectories given some initial point of interest.

3: Evaluate $\varepsilon_{m,t}(x, X; \theta)$ for some or all the points above.

4: Solve:

$$\min_{\theta} \frac{1}{M} \sum_{m=1}^{M} \sum_{t=0}^{T} \left(\varepsilon_{m,t}(x, X; \theta)\right)^2$$

We highlight three additional points. First, we found in preliminary testing that it is not important for the performance of the training that the transition be very close to any stationary solution (i.e., this does not require a hidden transversality condition or some form of backward induction from a steady state) or that a small or large number of trajectories are used. In fact, in Section 4.5, we solve a version with only a few of those data points.

Second, there are different possibilities regarding the choice of points to evaluate the Euler errors. The points can be some collocation nodes, simulated points à la Rust, or many others. Our main divergence from collocation-style approaches is that, when we minimize the Euler errors, we use relatively few points in $\mathcal{X}$ and ignore stationarity in both the simulation and the solution. Our method succeeds because, as we will see later, the approximations it finds for $u(\cdot)$ i) generalize and extrapolate well outside of the evaluated points; and (ii) provide stable solutions that converge to the ergodic sets and do not violate transversality. Neither of those conditions would hold, in general, for classical collocation-style solutions.

Third, a surprising benefit of a high-dimensional approximation is the "double-descent" phenomenon in machine learning (see Belkin et al., 2019, and Advani et al., 2020). In classical statistics, the bias-variance trade-off tells us that the generalization error of a function approximation becomes large when the number of free parameters approaches the number of data points (at the limit, when the number of free parameters is equal to the number of data points, we are just interpolating). For that reason, one typically wants to choose a smaller number of free parameters than the number of data points. Furthermore, we know that when we are exactly interpolating, the approximation can be especially poor exactly at the edges (e.g., Runge's phenomenon), which also makes extrapolation fail in general. The double-descent phenomenon, which is shown with many machine learning algorithms, is that if one *further* increases the number of parameters far above this interpolation threshold, then the generalization error begins to decrease again. Often, the cure to over-fitting is to add more parameters. We explore this topic in Section 4.5, where we consider the lack of treatment of transversality conditions and stationarity in more detail.

4.3 Case I: $\nu = 1$

We implement our deep learning architectures by fixing most of the model parameters at conventional levels: $\sigma = 0.005, \eta = 0.001, \alpha_0 = 1, \alpha_1 = 1, \beta = 0.95$, and $\gamma = 90$. To match a common feature of the data, we make the standard deviation of the idiosyncratic shock much larger than the standard deviation of the aggregate shock. Also, in this way, concentration of measure phases in more slowly, better showing its power to approximate expectations even in challenging parameterizations.

In this subsection, we set $\nu = 1$, the last parameter of the model. We pick this parameterization because, with it, our investment model becomes an LQ dynamic programming problem that we can solve using classical optimal control as described in Anderson et al. (1996). This will give us an exact solution (up to machine precision) with the form:

$$u(X) = H_0 + \frac{1}{N} H_1 \sum_{i=1}^{N} x_i,$$

where H_0 and H_1 are scalars. In this case, regardless of the number of firms N, the problem is two dimensional in the space of parameters. See Appendix D.4 for the functional representation of $v(x, X)$. We can use this exact solution to benchmark our deep learning approximations.

Notice how in terms of our Proposition 2,

$$u(X) = H_0 + \frac{1}{N} H_1 \sum_{i=1}^{N} x_i = \rho\left(\frac{1}{N}\sum_{i=1}^{N} \phi(X_i)\right),$$

where $\phi : \mathbb{R} \to \mathbb{R}^1$ (i.e., $L = 1$) is the identity function $\phi(X) = \Big[X\Big]$ and $\rho : \mathbb{R}^1 \to \mathbb{R}$ such that $\rho(y) = H_0 + H_1 y$ for some undetermined H_0 and H_1. However, we will *not* supply this information to our deep learning architectures. We want to gauge whether they can "learn" this form all by themselves.

Figure 3 plots investment decision $u(X_t)$ along the equilibrium path of an economy with 128 firms and aggregate shocks over a simulation of 63 periods for our four solutions. The blue dashed trajectory provides the LQ solution, the orange trajectory depicts the solution obtained by $\phi\big(\text{Identity}\big)$, the green trajectory depicts the solution obtained by $\phi\big(\text{Moments}\big)$, and the red trajectory depicts the solution obtained by $\phi(\text{ReLU})$. All four paths are almost identical. This shows that our deep learning solutions are incredibly accurate even when, as in this figure, the initial conditions are such that $u(X_0)$ (slightly below 0.024) is far from its mean value in the ergodic distribution of the economy (around 0.034). The neural networks "learn" very quickly that the solution is $u(x, X) = H_0 + \frac{1}{N} H_1 \sum_{i=1}^{N} x_i$ even if they never get such information.

The accuracy of the approximation is confirmed by Figure 4, which shows the mean squared error of the Euler residuals at each time step on the equilibrium path (averaged over 256 different

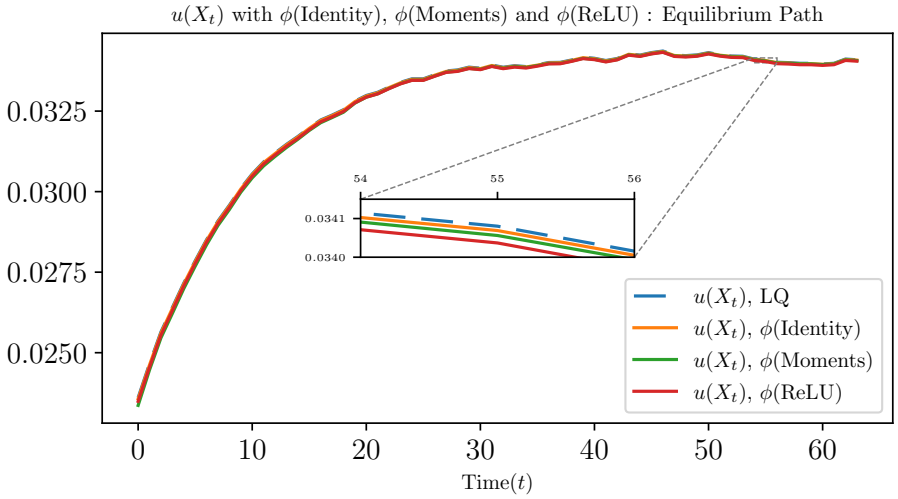

Figure 3: Comparison between the LQ-regulator solution and our three deep learning archictures for the case with $\nu = 1$ and $N = 128$.

simulated trajectories) and the corresponding 95th percentile bandwidth of those errors. The left panel shows the results for ϕ(Identity), the center panel shows the results for ϕ(Moments), and the right panel shows the results for ϕ(ReLU). While all architectures are highly accurate, the ReLU architecture is especially stable. This result can be interpreted as the function generalizing very consistently because of —rather than in spite of— the extra parameters of this architecture.

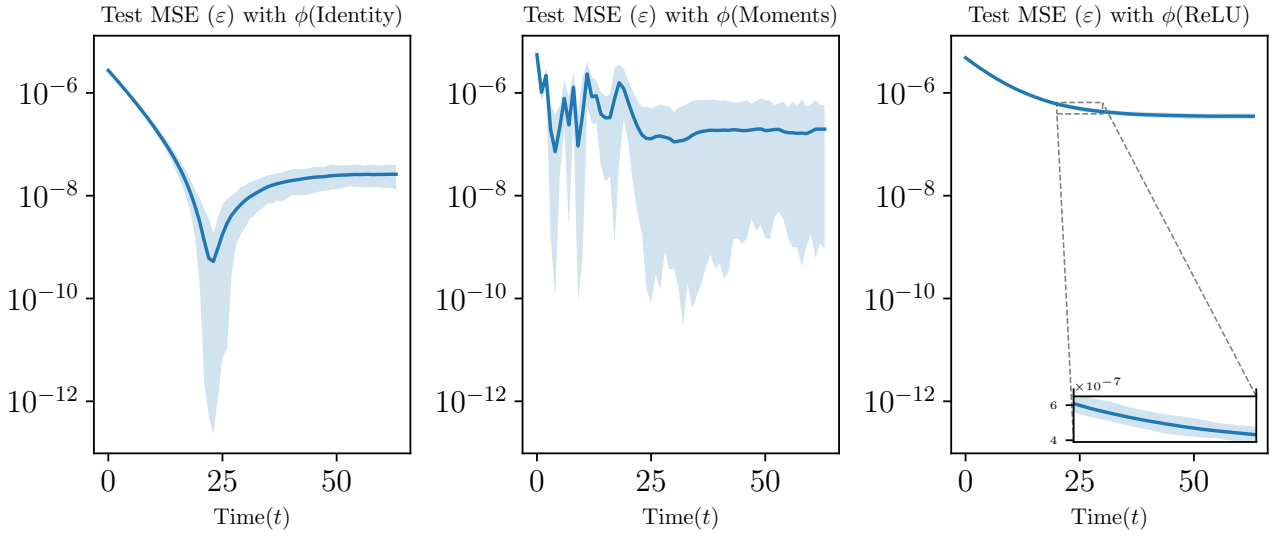

Figure 4: The Euler residuals for $\nu = 1$ and $N = 128$ for ϕ(Identity), ϕ(Moments), and ϕ(ReLU). The dark blue curve shows the average residuals along equilibrium paths for 256 different trajectories. The shaded areas depict the 2.5th and 97.5th percentiles.

While these figures show that the solution is very accurate, we need to gauge whether the performance is practical, and whether the curse of dimensionality has been sidestepped. Figure 5

shows the performance for ϕ(ReLU), the more densely parameterized architecture, as a function of N. The left panel shows the computation time in seconds.[14] The right panel shows the MSE of Euler residuals (i.e., ε) on the test data. The reported numbers are the median values of 21 trials with different seeds. The algorithm finds the global solution of the model with high accuracy in just around a minute. Furthermore, this computation is not a systematic function of N, but of order $\mathcal{O}(1)$ for reasonable N. The variation in solution time is largely due to the use of random initial conditions in a high-dimensional space (the 66.8K parameters).

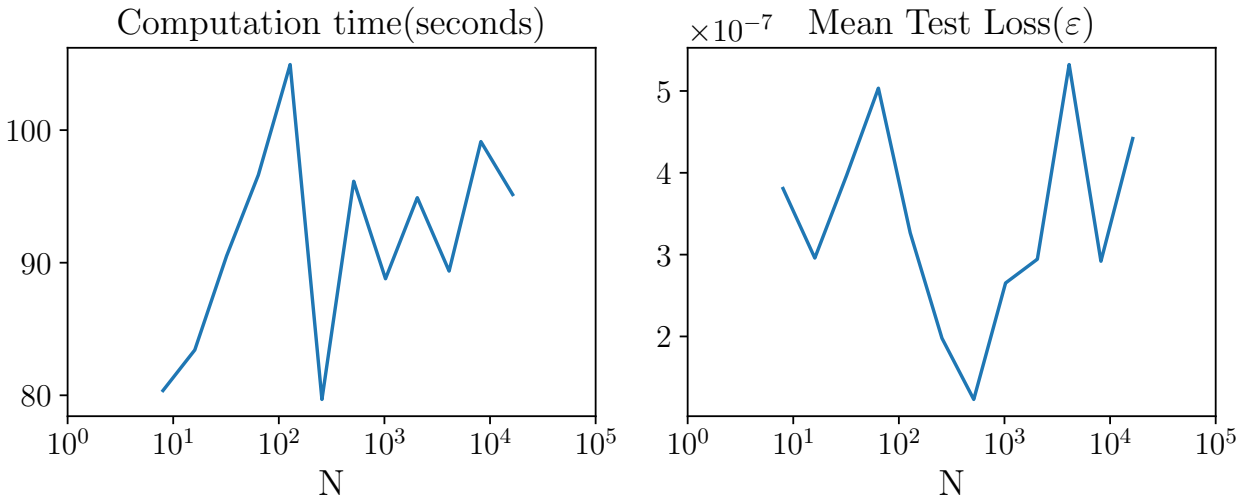

Figure 5: Performance of the ϕ(ReLU) for different N.

Finally, Table 1 shows the results for different variations in the approximations, ϕ(Identity), ϕ(ReLU), and ϕ(Moments). The columns are as follows: Time is measured in seconds. Params represents the number of parameters (θ) in functional approximation, in thousands. Train MSE reports the mean squared error of the Euler residuals for the simulated trajectories used for training. Test MSE documents the mean squared error of the Euler residuals on simulated data after the training is complete, and which internally uses the approximation itself during the simulation. Val MSE displays the mean squared error on the validation data, that is, trajectories used to check the quality of the generalization during the training process, but where the data are not used to fit the model directly. Policy error ($\left|u - u_{\text{ref}}\right|$) is the mean absolute value of the approximate solution and the accurate solution (u_{ref}) on the test data. Policy error $\left(\frac{|u-u_{\text{ref}}|}{u_{\text{ref}}}\right)$ is the mean absolute value of the relative error of the approximate solution and the accurate solution (u_{ref}) on the test data. The results in Table 1 are the median among 21 trials with different seeds. See Table 2 in Appendix E.1 for additional experiments.

[14]The time is calculated from the beginning of the fitting process until the mean squared Euler error reaches 1×10^{-6}, which is chosen to be on the same order of magnitude as the error coming from the concentration of measure approximation of the integral in Figure 9 when N =128.

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%

The first lesson from Table 1 is that the results are mainly robust to variations in the networks' architecture. However, for small networks, as in the case of the *Very Shallow* example in ϕ(Moments), sometimes the optimization procedure finds a local minimum that is not close to the global minimum solution. Not coincidentally, this is the model with the fewest number of parameters, which should give us pause since it is still orders of magnitude above the dimensionality of 2 parameters required for the analytical solution. Also, the case ϕ(ReLU), with Deep (ϕ : 4 layers, ρ : 8 layers) is unsatisfactory. This result highlights the importance of exploring different networks' architectures, something easily done with modern machine learning libraries.

The second lesson from Table 1 is that the results are, in general, good when we use a larger number of moments (equivalently, a higher dimension of L) than when we use only the first moment ($L = 1$, which we know from the analytical solution is a sufficient statistic for payoff-relevant values). Compare, for example, the ϕ(Identity) case with ϕ(ReLU($L = 16$)). The latter solves the model with many more parameters, but the neural network finds the lower-dimensional manifold in about the same amount of time.

4.4 Case II: $\nu > 1$

Now, we move to the case where $\nu > 1$, i.e., the inverse demand function is nonlinear, while all the other parameter values are the same as before. Nonetheless, we can find the solution to this nonlinear case using exactly the same algorithm, code, and functional approximations as in Section 4.3: we just change, in our code, the value of ν and drop the identity architecture since, by construction, it cannot encode the nonlinearity in the problem. As before, we fit the model and check its generalization with a set of simulated trajectories.

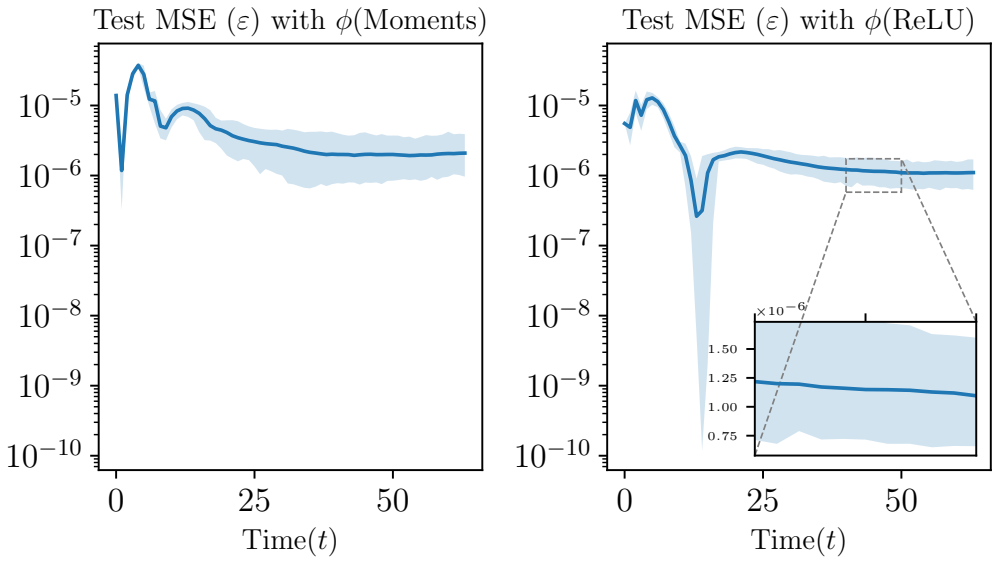

Figure 6: The Euler residuals for $\nu = 1.5$ and $N = 128$ for ϕ(Moments) and ϕ(ReLU). The dark blue curve shows the average residuals along equilibrium paths for 256 different trajectories. The shaded areas depict the 2.5th and 97.5th percentiles.

However, in this situation, the LQ solution no longer holds, and we do not have an exact solution against which to benchmark our deep learning solution. Instead, we need to jump directly to Figure 6 to show the average Euler residuals over 256 simulated trajectories, with the corresponding 95th percentile bandwidth. The economy uses $\nu = 1.5$ and $N = 128$. The left panel shows the results for ϕ(Moments) and the right panel the results for ϕ(ReLU). As in Figure 4, the mean squared error is fairly low and tight. Table 3 in Appendix E.2 provides additional information on this experiment. The main conclusion is that we are indeed finding a highly accurate solution, as in case I, because we are using the same algorithm.

We can also investigate the qualitative behavior of the approximate optimal solution, $u(\cdot)$ as ν approaches one, which coincides with case I where an accurate solution exists. Figure 7 depicts the optimal policy computed with ϕ(ReLU) and $N = 128$ along equilibrium paths with aggregate shocks and initial conditions far away from the mean of the ergodic distribution for $\nu = [1.0, 1.05, 1.1, 1.5]$. Figure 7 shows how the optimal policy for a trajectory smoothly approaches the known analytical solution as ν approaches one, showing the strength of our deep learning approximation.

4.5 Generalization: Where is the transversality condition?

Intriguingly, to get our approximation, we neither applied a transversality condition nor solved a stationary problem. While we fitted the data to a T where the equilibrium was getting close to some ergodic distribution, we did not use that behavior as a special case. However, in general,

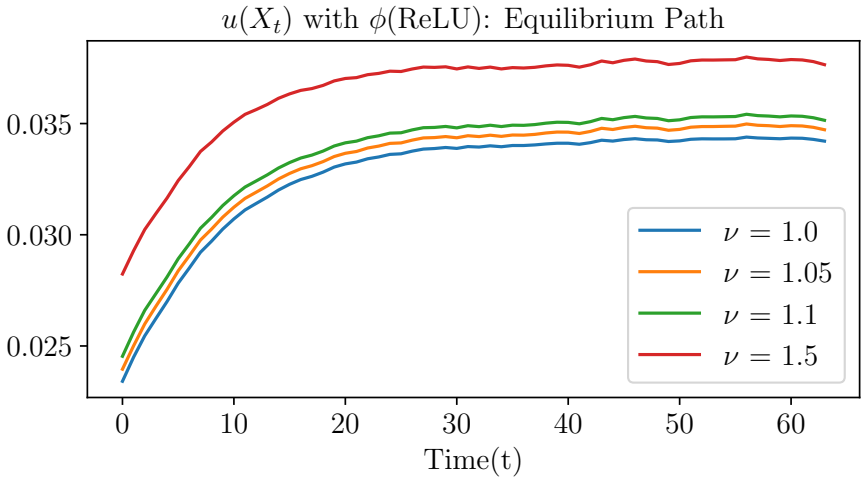

Figure 7: The optimal policy u along the equilibrium paths for $\nu = [1.0, 1.05, 1.1, 1.5]$ and $N = 128$. Each path shows the optimal policy for a single trajectory.

we can have many candidate solutions that satisfy the first-order conditions of a dynamic model if we do not constrain the long-run behavior of these solutions in one form or another. In fact, that is what we did to get the LQ solution: the classic control solutions implicitly applied a transversality condition. And, yet, our neural networks consistently picked the solution that generalizes best.[15] Why would we so carelessly disregard what has been thought of as an essential boundary condition for solving dynamic models in economics?

This question can be recast in terms of generalization and regularization. First: Why would our algorithm choose the "right" solution to the functional equation in the absence of explicit handling of the long-run behavior? Second, if the algorithm chooses a particular solution (rather than a random one), is it the one we want (as measured by its ability to generalize)?

In terms of answering the first question, one might suspect that this is because we are simulating with a great deal of data, but this is not the case. To show this, we can conduct an even more extreme example by fitting a nonlinear network with a parameter vector in $\mathbb{R}^{17.7K}$ using only 3 random data points in $\mathbb{R}^{512}$ for the linear demand function.[16] Recall that, in Subsection 4.3, we showed that the exact solution only has two parameters (i.e., H_0 and H_1). Therefore, our neural network is heavily over-parameterized ($2 \ll 17700$).

The results are shown in Figure 8. In this experiment, we fit a model with both ρ and ϕ represented by a neural network with two and one hidden layers respectively, each with 128 nodes and $\nu = 1$. The output dimension of ϕ is $L = 4$. Each step is one gradient updating in stochastic gradient descent optimization. We draw 3 random data points within $\mathbb{R}^{512}$ from

[15] With shorter simulations, the generalization error for very large $t > T$ worsens because there is an insufficient numbers of data points in a domain of interest; it is not a consequence of whether the system is ergodic.

[16] In this case, the solution is consistent with a transversality condition. In other words, imposing a stationary solution boundary condition only needs 2 points to interpolate.

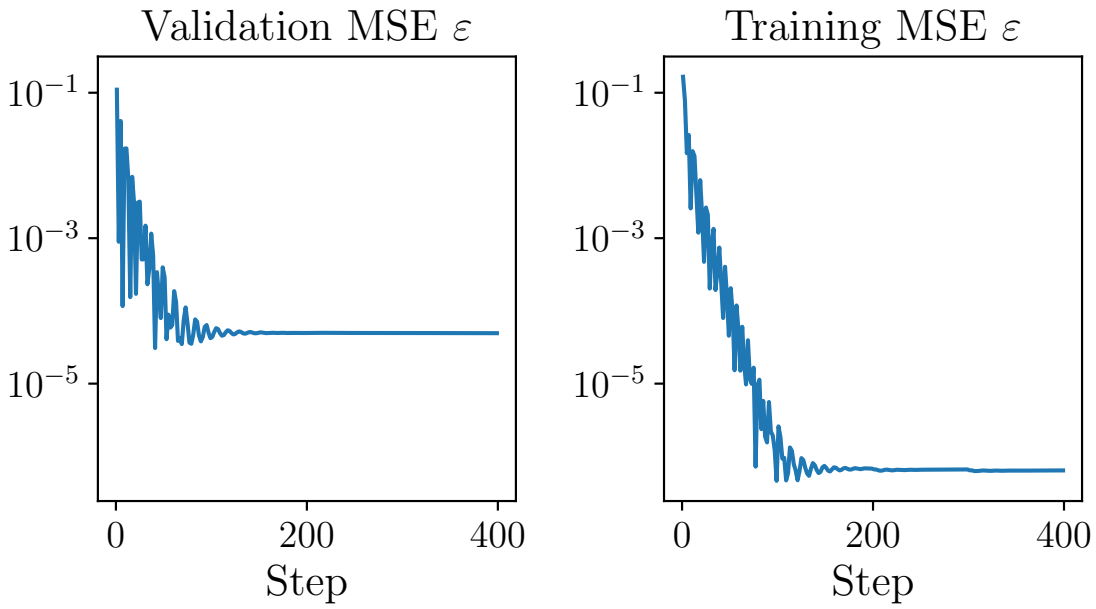

Figure 8: Training and validation loss of the Euler residuals ε. The training procedure utilizes 3 data points in the state space for an economy with $\nu = 1$ and $N = 512$.

simulated trajectories. The model fits using random "batches" of data of size 2, where the randomization in the stochastic gradient descent is an essential part of regularizing the solutions. The right panel of Figure 8 shows the convergence of the MSE of the training data, which the optimizer is minimizing. The left panel of Figure 8 shows the MSE of the Euler errors on "validation" data, i.e., random data used to test the generalization error during training, but which do not enter the optimization objective directly. The training error drops to within the numerical tolerance, as expected. But rather than over-fitting, the validation error also drops to less than 5×10^{-5}, which corresponds to a relative policy error of 0.06%, on the global solution "out of sample" with aggregate and idiosyncratic shocks, despite a large number of parameters and minimal data.

While the theoretical understanding of this over-fitting phenomenon is still an active area of research within computer science, it is robustly found in many functional approximation problems. In application after application, using stochastic optimization methods with highly over-parameterized models tends to robustly find solutions that fulfill a minimum norm solution in the space of approximating functions. Recent work on this issue includes Arora et al. (2018), Allen-Zhu et al. (2019), and Nakkiran et al. (2019).

For our specific problem, the minimum norm solution seems to coincide with the most generalizable solution and it seems to automatically fulfill the boundary-value conditions (e.g., a transversality condition or a boundary at a terminal period) that we often need to apply directly when approximating with lower-dimensional models.

5 What about Concentration of Measure?

Section 4 made it very clear how symmetry allowed us to come up with a particularly efficient deep learning architecture. It was harder, however, to gauge the role of concentration of measure.

To investigate this point, we return to the case where $\nu = 1$ and propose a modified LQ regulator where we use Monte Carlo to evaluate the required expectations instead of relying on certainty equivalence. By comparing the exact solution that we get from the classic LQ regulator with the approximated solution from our modified LQ regulator, we can assess the importance of concentration of measure and explicitly demonstrate permutation invariance. Finally, our modified LQ regulator is of interest in itself, as it can be helpful for non-Gaussian LQ problems where certainty equivalence does not hold.

5.1 The linear-quadratic regulator formulation

We start by working with a vector (notice that we are not applying Proposition 4 here, now it is just to simplify notation). As before, without loss of generality, assume that the x in (17) is the first element of X. Denote the state of all firms and a constant term as $\vec{x} \equiv \begin{bmatrix} 1 & x & X^\top \end{bmatrix}^\top \equiv \begin{bmatrix} 1 & x & x & X_2 & \dots & X_N \end{bmatrix}^\top \in \mathbb{R}^{N+2}$ and the vector of all Gaussian shocks as $\vec{w} \equiv \begin{bmatrix} \omega & W^\top \end{bmatrix}^\top = \begin{bmatrix} \omega & w & W_{\text{-}1}^\top \end{bmatrix}^\top = \begin{bmatrix} \omega & w & W_2 & \dots & W_N \end{bmatrix}^\top \sim \mathcal{N}(\mathbf{0}_{N+1}, \mathbf{I}_{N+1})$. The set of idiosyncratic shocks of all of the other firms is denoted $W_{\text{-}1}$ for convenience.

Next, define $\pi \in \mathcal{S}$ to be a permutation represented by an $(N+2) \times (N+2)$ permutation matrix where the lower-right $N \times N$ block is in $\mathcal{S}_N$, i.e., holding the first and second entries fixed. That is, we are looking at permutations of the form

$$\pi = \left[\begin{array}{cc:c} 1 & 0 & \mathbf{0}_N^\top \\ 0 & 1 & \mathbf{0}_N^\top \\ \hdashline \mathbf{0}_N & \mathbf{0}_N & \pi_N \end{array}\right], \tag{23}$$

where $\pi_N \in \mathcal{S}_N$ as defined in Section 2.1.

If we define the matrices:

$$A \equiv \left[\begin{array}{cc:c} 1 & 0 & \mathbf{0}_N^\top \\ 0 & 1-\delta & \mathbf{0}_N^\top \\ \hdashline H_0\mathbf{1}_N & \mathbf{0}_N & (1-\delta)\mathbf{I}_N + \frac{H_1}{N}\mathbf{1}_N\mathbf{1}_N^\top \end{array}\right] \tag{24}$$

$$B \equiv \left[\begin{array}{cc:c} 0 & 1 & \mathbf{0}_N^\top \end{array}\right]^\top \tag{25}$$

$$C \equiv \left[\begin{array}{c:cc} 0 & \mathbf{0}_N^\top \\ \eta & \sigma & \mathbf{0}_{N-1}^\top \\ \hdashline \eta\mathbf{1}_N & \sigma\mathbf{I}_N \end{array}\right] \tag{26}$$

$$R \equiv \left[\begin{array}{cc:c} 0 & \frac{\alpha_0}{2} & \mathbf{0}_N^\top \\ \frac{\alpha_0}{2} & 0 & -\frac{\alpha_1}{2N}\mathbf{1}_N^\top \\ \hdashline \mathbf{0}_N & -\frac{\alpha_1}{2N}\mathbf{1}_N & \mathbf{0}_N\mathbf{0}_N^\top \end{array}\right] \tag{27}$$

$$Q \equiv \left[\frac{\gamma}{2}\right], \tag{28}$$

we can write the problem as an LQ optimal regulator as derived in Appendix D.1.

5.2 The solution

We solve our LQ formulation using the following proposition.

Proposition 5 (Monte Carlo LQ formulation)**.** *Taking H_0 and H_1 as given, each firm solves:*

$$v(\vec{x}) = \max_u \left\{ -\vec{x}^\top R\vec{x} - u^\top Q u + \beta \mathbb{E}\left[v(\vec{x})\right] \right\} \tag{29}$$

$$s.t. \vec{x}' = A\vec{x} + Bu + C\vec{w} \tag{30}$$

With solutions characterized by a P, d, and F:

$$v(\vec{x}) = -\vec{x}^\top P\vec{x} - d \tag{31}$$

$$u(\vec{x}) = -F\vec{x}. \tag{32}$$

A symmetric recursive competitive equilibrium is defined as H_0 and H_1 such that:

$$F = -\left[H_0 \quad 0 \quad \frac{H_1}{N} \quad \frac{H_1}{N} \quad \dots \quad \frac{H_1}{N}\right] = -\left[\begin{array}{cc:c} H_0 & 0 & \frac{H_1}{N}\mathbf{1}_N^\top \end{array}\right]. \tag{33}$$

Furthermore, both the control $u(\cdot)$ and value function $v(\cdot)$:

1. *are invariant to permutations $\pi \in \mathcal{S}$ as defined by (23);*

2. *have expected gradients bounded in N, according to Definition 4:*

$$\nabla_{W_{\text{-}1}} u(\vec{x}\,') = \frac{\sigma H_1}{N} \mathbf{1}_{N-1}; \tag{34}$$

3. *have a concentration of measure according to Proposition 3 when calculating with a Monte Carlo draw, $\vec{x}\,'^1$, rather than certainty equivalence, which leads to —for some C_v independent of N:*

$$\mathbb{P}\left(\left|u(\vec{x}\,'^1) - \mathbb{E}\left[u(\vec{x}\,') \mid w, \omega, \vec{x}\right]\right| \geq \epsilon\right) \leq \frac{\sigma^2 H_1^2}{\epsilon^2} \frac{1}{N-1} \tag{35}$$

$$\mathbb{P}\left(\left|v(\vec{x}\,'^1) - \mathbb{E}\left[v(\vec{x}\,') \mid w, \omega, \vec{x}\right]\right| \geq \epsilon\right) \leq \frac{\sigma^2 C_v^2}{\epsilon^2} \frac{1}{N-1}; \tag{36}$$

Also, when we use a Monte Carlo draw instead of certainty equivalence, the Euler residual $\varepsilon(X;u)$ is a Gaussian random variable:

$$\varepsilon(X;u) \sim \mathcal{N}\Big(0, \frac{N-1}{N^2} \beta^2 \sigma^2 \left[\alpha_1 - \gamma(1-\delta) H_1\right]^2\Big). \tag{37}$$

Similarly, the error in forecasting the policy in the next period is Gaussian random variable as well:

$$u(\vec{x}\,'^1) - \mathbb{E}\left[u(\vec{x}\,') \mid w, \omega, \vec{x}\right] \sim \mathcal{N}\Big(0, \frac{N-1}{N^2} H_1^2 \sigma^2\Big). \tag{38}$$

Proof. See Appendix D.1 for the proof using classic LQ-Gaussian control, Appendix D.2 for permutation invariance, and Appendix D.3 for concentration of measure

As a demonstration of some of the symmetry properties, we will sketch out the proof for $u(\cdot)$ in Proposition 5, where we take the functional form of F as given.

1. Since F as defined in (33) is identical for all but the first two elements, it aligns with the block structure of the permutations. Hence, for any of the π in (23), $\pi F = F$, which ensures that $u(\cdot)$ is symmetric in all changes to the X vector.

2. The gradient of an affine function is constant. Hence, for any $\vec{w}$, the policy next period is $u(A\vec{x} + Bu + C\vec{w})$. Given the block structure of F and C, we differentiate with respect to $W_{\text{-}1}$ in the $\vec{w}$ to find:

$$\nabla_{W_{\text{-}1}} u(A\vec{x} + Bu + C\vec{w}) = \frac{\sigma H_1}{N} \mathbf{1}_{N-1}. \tag{39}$$

Therefore, the square of the norm of the gradient of the policy can be bounded as:

$$\mathbb{E}\left[\|\nabla_{W_{\text{-1}}} u(\vec{x}\,')\|^2\right] = \left(\sum_{i=2}^{N} \frac{\sigma H_1}{N}\right)^2 \leq \frac{\sigma^2 H_1^2}{N-1}. \tag{40}$$

3. Since $W_{\text{-1}} \sim \mathcal{N}(\mathbf{0}_{N-1}, \mathbf{I}_{N-1})$, the spectral radius of $\mathbf{I}_{N-1}$ is 1 and the concentration of measure follows from (34) and Proposition 3.

□

In terms of solving for an equilibrium in Proposition 5, we guess an H_0 and H_1, solve the LQ problem as a matrix Ricatti equation, and then find the fixed point of H_0 and H_1. In comparison with classic LQ control, we do not use certainty equivalence. The variance of the shock only enters into the d of the solution (31). The d can be decomposed into a component related to the ω aggregate shock and another for the W associated with the finite number of firms. As N goes to infinity, the component of d associated with the idiosyncratic shocks goes to 0. That is, conditioning on the aggregate shock, the firm can get closer and closer to the deterministic forecast of the distribution of X'.

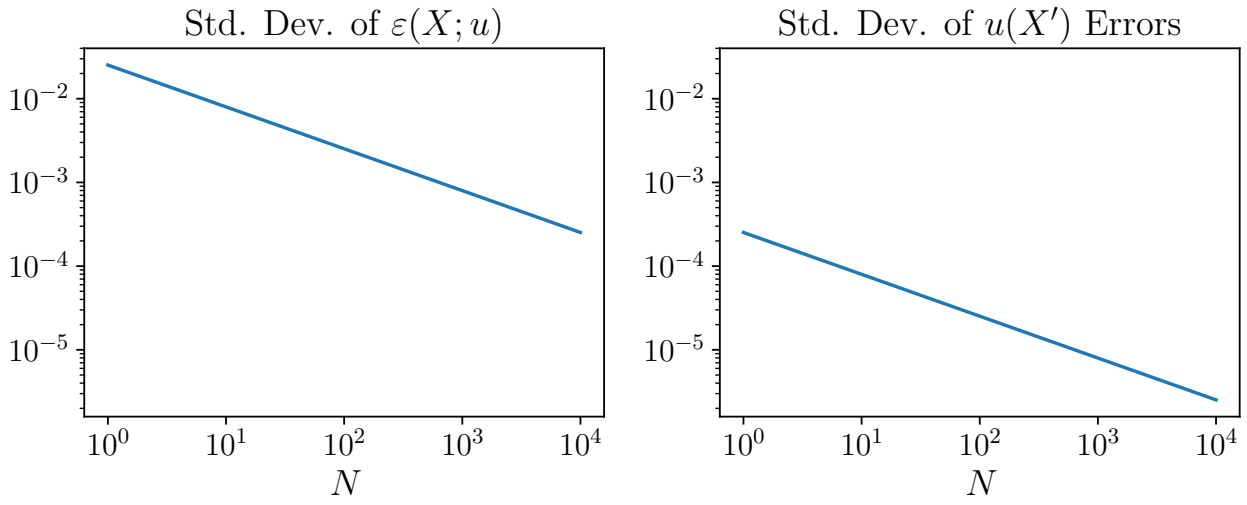

Figure 9: The concentration of Euler residuals $\varepsilon(X;u)$ and the optimal policy $u(X')$.

To illustrate how quickly and strongly concentration of measure kicks in, we use the same parameter values as in Section 4.3. Also, we use a single Monte Carlo draw for the expectation, which is an extreme case but one that can serve as a guide for the quality of the approximation in the many cases of interest where certainty equivalence would not hold.

The left panel of Figure 9 plots the standard deviation of the Euler residuals $\varepsilon(X;u)$ induced by the concentration of measure approximation as we increase the number of firms in the industry

and we solve the model with modified LQ formulation (this standard deviation comes directly from equation 37). By the time we get to 10,000 firms, the standard deviation of the Euler errors is 2.4×10^{-4}. In the right panel of Figure 9, we draw the standard deviation of the optimal policy $u(X')$ (equal to $H_1\sigma\sqrt{\frac{N-1}{N}}/\sqrt{N}$) for the linear aggregate prices given the same Monte Carlo approximation and find that the standard deviation of the policy error is 2.5×10^{-6} for $N = 10,000$ and 2.2×10^{-5} for our baseline case in many of our previous experiments of $N = 128$.

While the LQ algorithm we just outlined works in delivering accurate solutions, as N becomes large, it is too slow, as it requires computations of order $\mathcal{O}(N^3)$. In comparison, we argued before that our neural networks only require computations of order $\mathcal{O}(1)$ for a reasonable N.

6 Extensions

Did we choose a setup that was particularly amenable to our methods? In this section, we consider how these general techniques can be used for a much larger class of models. Most broadly, our tools are useful for solving any high-dimensional functional equations with some degree of symmetry—and are especially useful when these equations contain high-dimensional expectations. The functional equation could come from static problems, Bellman equations, or Euler equations, and can be solved by various approximation techniques.[17]

6.1 Decreasing returns to scale

The constant returns to scale in production and the linear marginal cost of investment in equation (16) led to the value function in Proposition 4 being independent from x. As the primary interest was in analyzing the high-dimensional state, this was a convenient result.

But what if the value function (11) has decreasing returns to scale and, thus, the policy becomes a function of x? Imagine, for example:

$$\text{profits} = p(X)\frac{x^{1-\mu}}{1-\mu} - \frac{\gamma}{2}u^2.$$

This formulation leads to an Euler equation:

$$\gamma u(x, X) = \beta \mathbb{E}\left[P(X')x^{-\mu} + \gamma(1-\delta)u(x', X')\right].$$

Given this equation, the solution techniques are identical except that $\mathbb{E}[\cdot]$ now depends strongly on the idiosyncratic shock associated with state x. Extending the previous approach,

[17] For example, we could use reinforcement learning (also known as approximate dynamic programming or neuro-dynamic programming). See van der Pol et al. (2020) for an example applying symmetry to reinforcement learning.

this expectation can be approximated by a two-dimensional Gaussian quadrature in w and ω, where the expectation with respect to idiosyncratic shocks W is still preformed with a Monte Carlo draw. The only other modification required is to ensure we are using a representation in Proposition 2 where $\rho(x, y)$ and $\phi(x)$ where $y \in \mathbb{R}^L$.

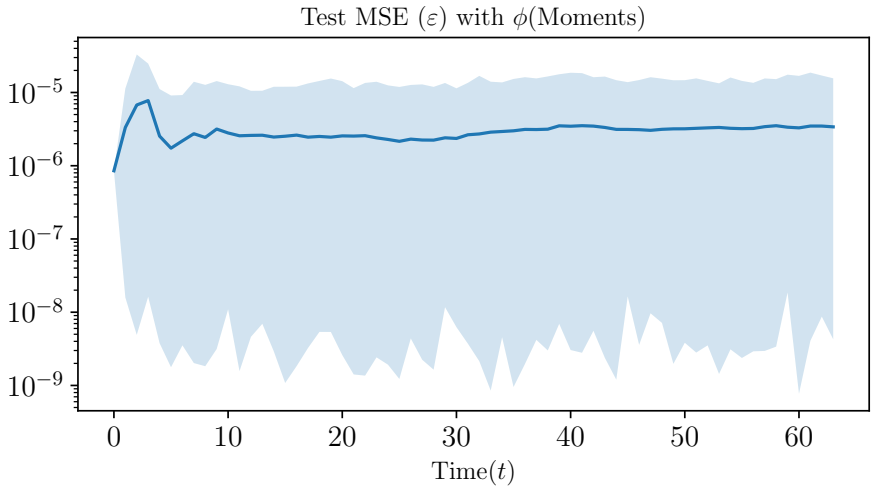

Figure 10: The Euler residuals for $\nu = 1$, $\mu = 0.05$, and $N = 128$, over 256 different trajectories. The shaded areas depict the 2.5th and 97.5th percentiles. In this case, the baseline ϕ(Moments) is used to approximate the optimal policy.

Figure 10 depicts the average Euler residuals along the equilibrium path (averaged over 256 trajectories) and the corresponding 95th percentile bandwidth for an economy with $\nu = 1$, $\mu = 0.05$, and $N = 128$. In this experiment baseline ϕ(Moments) is used to approximate the optimal policy. The generalization properties are excellent, and the mean squared Euler error is on the order of 1×10^{-6} for most of the trajectories and periods.

6.2 Multiple productivity types

We consider now the case when firms have profits$_j = \zeta_j p(X) \frac{x^{1-\mu}}{1-\mu} - \frac{\gamma}{2} u^2$ for some finite set of productivities, e.g., ζ_1, ζ_2. Then, rather than having permutation invariance based on the entire X, we could group them such that the policy becomes $u(x, X^1, X^2)$, where X_1 contains all firms with ζ_1, etc. That is, the elements of X could be symmetric in blocks, and this could be implemented in the network with a representation such as

$$u(x, X^1, X^2) \approx \rho\left(x, \frac{1}{N_1}\sum_{i=1}^{N_1} \phi_1(X_i^1), \frac{1}{N_2}\sum_{i=1}^{N_2} \phi_2(X_i^2)\right).$$

If there were reasons in the problem structure to think that the ϕ_1 and ϕ_2 could be proportional to each other, then the representation might even be

$$u(x, X^1, X^2) \approx \rho\left(x, \frac{1}{N_1}\sum_{i=1}^{N_1}\phi(X_i^1), \frac{1}{N_2}\sum_{i=1}^{N_2}\phi(X_i^2)\right).$$

6.3 Complex idiosyncratic states

Next, we study the case where firms also have a stochastic productivity y, such that the firm is identified by the tuple $z \equiv (x, y)$ and the aggregate state is a collection of tuples $Z \equiv \{(X_i, Y_i)\}_{i=1}^N$. We can apply our techniques by considering that it is the symmetry with respect to the z tuple of states, rather than permutation within that tuple, that is required. We can then implement this symmetry in an approximation along the lines of

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

for some $\phi : \mathbb{R}^2 \to \mathbb{R}^L$ and (x, y) the idiosyncratic state of interest.[18] Hence our techniques are easily applied to models with exchangeable "agents" containing a finite number of states.

Our approach can even be extended to cases where the collection of underlying states is itself exchangeable. For example, consider if, in addition to its x, each firm has up to M customers of varying sizes, y^j, where those customers are otherwise exchangeable in the firm's payoffs.[19] Then we can denote the customers themselves as a set, and use $z \equiv (x, \{y^j\}_{j=1}^M)$ as an individual firm's state. If we let $Y_i \equiv \left\{y_i^j\right\}_{j=1}^M$, the aggregate distribution is $Z \equiv \{X_i, Y_i\}$.

A neural network imposing this symmetry might be

$$u(x, Y, Z) \approx \rho_1\left(x, \frac{1}{M}\sum_{j=1}^{M}\phi_1(y^j), \frac{1}{N}\sum_{i=1}^{N}\rho_2\left(X_i, \frac{1}{M}\sum_{j=1}^{M}\phi_2(Y_i^j)\right)\right),$$

where the training process would fit parameters for ρ_1, ϕ_1, ρ_2, and ϕ_2, and the ρ_2 function itself could have a latent dimension greater than one. This last example demonstrates why deep learning, in the sense of many layers of highly parameterized functions, is so important for dealing with symmetry. Without modern (and easily available) machine learning libraries, it would be impractical to fit this nested approximation.

Given the flexibility of working with states that are themselves complicated structures, a natural application of these methods is to solve dynamic models with networks—expanding the computational feasibility of solving extensions of models surveyed in Carvalho and Tahbaz-Salehi

[18] In a model with a continuum of agents, this would simply lead to a two-dimensional distributional state, and the modeler would typically write payoffs as expectations using that distribution.

[19] See Hartford et al. (2018) for methods to represent permutation equivariance with interactions between sets.

(2019). In particular, it holds the promise of making multi-period dynamic programming in models such as Oberfield (2018) possible by enabling network connections as a persistent state variable and enabling firms to form expectations of the network's evolution.

6.4 Global solutions with transitions and aggregate shocks

In many applications, researchers use a continuum of agents. Unfortunately, as soon as we add several aggregate shocks to them, these models are difficult to tackle. Since the distributional state becomes a stochastic partial differential equation (or its discrete-time equivalent), agents need to solve optimal control problems with an infinite-dimensional state, which in turn depends on their decisions. The forward equation for the evolution is difficult enough to simulate, even as an initial value problem, but becomes intractable when it is part of an infinite-dimensional optimal control problem. The continuum approximation is then the source of the numerical intractability rather than the solution to it, hinting that leaving things discrete might help.

As a path forward, our method offers the possibility of simulating economies with tens of thousands of agents even when the model is far away from its ergodic mean or when the economy is hit by a large shock. Then, we can either have a fine granular grouping of agents that matches the microdata (i.e., the same number of agents or groups in the model as in the microdata) or approximate a continuum of agents distribution with histograms as in Mertens and Judd (2018). In comparison, in perturbation solutions such as Kaplan et al. (2018) in continuous time and Den Haan (2010) in discrete time, one requires appropriate stationarity properties for the perturbation to work accurately.

This insight might be particularly fruitful when solving growth models with heterogeneous agents, which often have steady states and balanced growth paths that are notoriously sensitive to parameters, multiplicity of equilibria, and hysteresis (i.e., they may not even be ergodic). By using a finite number of agents—along the lines of the models discussed in Buera and Lucas (2018)—models with endogeneity of both innovation and diffusion as in Benhabib et al. (2021) without ergodicity could be solved from arbitrary initial conditions.

7 Conclusion

In this paper, we have shown how to exploit symmetry in dynamic programming problems and the solution of recursive competitive equilibria with a large (but finite) number of heterogeneous agents using deep learning. Symmetry, together with concentration of measure, allowed us to break the curse of dimensionality. The method is easy to implement using standard, open-source software libraries and fast to run on a standard laptop. Our results open the door to many applications of interest, including macro, finance, international finance, industrial organization, international trade, spatial economies, and other fields.

References

Advani, M. S., A. M. Saxe, and H. Sompolinsky (2020): "High-dimensional dynamics of generalization error in neural networks," *Neural Networks*, 132, 428–446.

Aguirregabiria, V. and A. Nevo (2013): "Recent developments in empirical IO: Dynamic demand and dynamic games," in *Advances in Economics and Econometrics: Tenth World Congress*, ed. by D. Acemoglu, M. Arellano, and E. Dekel, Cambridge University Press, vol. 3, 53?122.

Allen-Zhu, Z., Y. Li, and Z. Song (2019): "A convergence theory for deep learning via over-parameterization," in *International Conference on Machine Learning*, PMLR, 242–252.

Anderson, E. W., E. R. McGrattan, L. P. Hansen, and T. J. Sargent (1996): "Mechanics of forming and estimating dynamic linear economies," *Handbook of Computational Economics*, 1, 171–252.

Arora, S., R. Ge, B. Neyshabur, and Y. Zhang (2018): "Stronger generalization bounds for deep nets via a compression approach," in *Proceedings of the 35th International Conference on Machine Learning*, ed. by J. Dy and A. Krause, vol. 80, 254–263.

Azinovic, M., L. Gaegauf, and S. Scheidegger (2019): "Deep equilibrium nets," *Available at SSRN 3393482*.

Backus, D. K., P. J. Kehoe, and F. E. Kydland (1992): "International Real Business Cycles," *Journal of Political Economy*, 100, 745–775.

Barvinok, A. (2005): "Math 710: Measure concentration," *Lecture notes*.

Belkin, M., D. Hsu, S. Ma, and S. Mandal (2019): "Reconciling modern machine-learning practice and the classical bias–variance trade-off," *Proceedings of the National Academy of Sciences*, 116, 15849–15854.

Bellman, R. (1958): *Dynamic Programming*, Princeton University Press.

Benhabib, J., J. Perla, and C. Tonetti (2021): "Reconciling models of diffusion and innovation: A theory of the productivity distribution and technology frontier," *Econometrica (Forthcoming)*.

Bilal, A. (2021): "Solving Heterogeneous Agent Models with the Master Equation," Tech. rep., University of Chicago.

Bloem-Reddy, B. and Y. W. Teh (2020): "Probabilistic Symmetries and Invariant Neural Networks." *Journal of Machine Learning Research*, 21, 90–1.

Boucheron, S., G. Lugosi, and P. Massart (2013): *Concentration Inequalities: A Nonasymptotic Theory of Independence*, Oxford University Press.

Brumm, J. and S. Scheidegger (2017): "Using Adaptive Sparse Grids to Solve High-Dimensional Dynamic Models," *Econometrica*, 85, 1575–1612.

Buera, F. J. and R. E. Lucas (2018): "Idea flows and economic growth," *Annual Review of Economics*, 10, 315–345.

Carvalho, V. M. and A. Tahbaz-Salehi (2019): "Production networks: A primer," *Annual Review of Economics*, 11, 635–663.

Cybenko, G. (1989): "Approximation by superpositions of a sigmoidal function," *Mathematics of Control, Signals and Systems*, 2, 303–314.

Den Haan, W. J. (2010): "Comparison of solutions to the incomplete markets model with aggregate uncertainty," *Journal of Economic Dynamics and Control*, 34, 4–27.

Duarte, V. (2018): "Machine learning for continuous-time finance," Tech. rep., Gies College of Business.

Duffy, J. and P. D. McNelis (2001): "Approximating and simulating the stochastic growth model: Parameterized expectations, neural networks, and the genetic algorithm," *Journal of Economic Dynamics and Control*, 25, 1273–1303.

Fernández-Villaverde, J. and P. A. Guerrón-Quintana (2021): "Estimating DSGE Models: Recent Advances and Future Challenges," *Annual Review of Economics*, 13, 229–252.

Fernández-Villaverde, J., S. Hurtado, and G. Nuño (2019): "Financial frictions and the wealth distribution," Working Paper 26302, National Bureau of Economic Research.

Hartford, J., D. R. Graham, K. Leyton-Brown, and S. Ravanbakhsh (2018): "Deep Models of Interactions Across Sets," in *Proceedings of the 35th International Conference on Machine Learning*, ed. by J. Dy and A. Krause, PMLR, vol. 80 of *Proceedings of Machine Learning Research*, 1909–1918.

Hartford, J. S., J. R. Wright, and K. Leyton-Brown (2016): "Deep learning for predicting human strategic behavior," in *Advances in Neural Information Processing Systems*, 2424–2432.

Hornik, K. (1991): "Approximation capabilities of multilayer feedforward networks," *Neural Networks*, 4, 251–257.

Jovanovic, B. and R. W. Rosenthal (1988): "Anonymous sequential games," *Journal of Mathematical Economics*, 17, 77–87.

Judd, K. L., L. Maliar, S. Maliar, and I. Tsener (2017): "How to solve dynamic stochastic models computing expectations just once," *Quantitative Economics*, 8, 851–893.

Kalouptsidi, M. (2017): "Detection and Impact of Industrial Subsidies: The Case of Chinese Shipbuilding," *Review of Economic Studies*, 85, 1111–1158.

Kaplan, G., B. Moll, and G. L. Violante (2018): "Monetary policy according to HANK," *American Economic Review*, 108, 697–743.

Keane, M. P. and K. I. Wolpin (1994): "The Solution and Estimation of Discrete Choice Dynamic Programming Models by Simulation and Interpolation: Monte Carlo Evidence," *Review of Economics and Statistics*, 76, 648–672.

Krusell, P. and A. A. Smith, Jr (1998): "Income and wealth heterogeneity in the macroeconomy," *Journal of Political Economy*, 106, 867–896.

Ledoux, M. (2001): *The Concentration of Measure Phenomenon*, 89, American Mathematical Society.

Ljungqvist, L. and T. J. Sargent (2018): *Recursive Macroeconomic Theory*, MIT Press, 4 ed.

Lucas, R. E. and E. C. Prescott (1971): "Investment under uncertainty," *Econometrica*, 659–681.

Macdonald, I. G. (1998): *Symmetric Functions and Hall Polynomials*, Oxford University Press.

Maliar, L., S. Maliar, and P. Winant (2019): "Will artificial intelligence replace computational economists any time soon?" Tech. Rep. 14024, C.E.P.R. Discussion Papers.

Mertens, T. M. and K. L. Judd (2018): "Solving an incomplete markets model with a large cross-section of agents," *Journal of Economic Dynamics and Control*, 91, 349–368.

Nakkiran, P., G. Kaplun, Y. Bansal, T. Yang, B. Barak, and I. Sutskever (2019): "Deep double descent: Where bigger models and more data hurt," *arXiv preprint arXiv:1912.02292*.

Narayanamurthy, S. M. and B. Ravindran (2008): "On the hardness of finding symmetries in Markov decision processes," in *Proceedings of the 25th International Conference on Machine Learning*, 688–695.

Oberfield, E. (2018): "A theory of input-output architecture," *Econometrica*, 86, 559–589.

Pakes, A. and P. McGuire (2001): "Stochastic Algorithms, Symmetric Markov Perfect Equilibrium, and the 'Curse' of Dimensionality," *Econometrica*, 69, 1261–1281.

Powell, W. B. (2007): *Approximate Dynamic Programming: Solving the Curses of Dimensionality*, vol. 703, John Wiley & Sons.

Prasad, A. (2018): "An introduction to Schur polynomials," *arXiv preprint arXiv:1802.06073*.

Prescott, E. C. and R. Mehra (1980): "Recursive competitive equilibrium: The case of homogeneous households," *Econometrica*, 48, 1365–1379.

Ravindran, B. and A. G. Barto (2001): *Symmetries and Model Minimization in Markov Decision Processes*, University of Massachusetts.

Rust, J. (1997): "Using Randomization to Break the Curse of Dimensionality," *Econometrica*, 65, 487–516.

van der Pol, E., D. E. Worrall, H. van Hoof, F. A. Oliehoek, and M. Welling (2020): "MDP homomorphic networks: Group symmetries in reinforcement learning," *arXiv preprint arXiv:2006.16908*.

Vershynin, R. (2018): *High-Dimensional Probability: An Introduction with Applications in Data Science*, vol. 47, Cambridge university press.

Wagstaff, E., F. B. Fuchs, M. Engelcke, I. Posner, and M. Osborne (2019): "On the limitations of representing functions on sets," *arXiv preprint arXiv:1901.09006*.

Yarotsky, D. (2018): "Universal approximations of invariant maps by neural networks," *arXiv preprint arXiv:1804.10306*.

Zabrocki, M. (2015): "Introduction to symmetric functions," *unpublished work http://garsia. math. yorku. ca/ghana03/mainfile. pdf*.

Zaheer, M., S. Kottur, S. Ravanbakhsh, B. Poczos, R. R. Salakhutdinov, and A. J. Smola (2017): "Deep sets," in *Advances in Neural Information Processing Systems*, 3391–3401.

Appendix A Mathematics Definitions and Background

Definition 6 (The permutation group and the permutation operator)**.** *Define:*

- *A permutation group is a group G whose elements are permutations of a given set $\mathcal{I}$ and whose group operation is the composition of permutations in G.*
- *The symmetric group of a set $\mathcal{I}$ is the group of all permutations of set $\mathcal{I}$.*

Definition 7 (Permutation invariance)**.** *A function $f : \mathbb{R}^N \to \mathbb{R}$ is permutation invariant if, for all permutations $\pi \in \mathcal{S}_N$:*

$$f(\pi X) = f(X).$$

Definition 8 (Permutation equivariance)**.** *A function $f : \mathbb{R}^N \to \mathbb{R}^N$ is permutation equivariant if, for all permutations $\pi \in \mathcal{S}_N$:*

$$f(\pi X) = \pi f(X).$$

Proposition 6. *Let $\mathcal{X}$ be a compact subset of $\mathbb{R}^N$ and $f : \mathcal{X} \subseteq \mathbb{R}^N \to \mathbb{R}$ be a continuous permutation invariant function under $\mathcal{S}_N$, i.e., for all $X \in \mathbb{R}^N$ and all $\pi \in \mathcal{S}_N$:*

$$f(\pi X) = f(X).$$

Then, there exist $L \leq N$, and continuous functions $\varrho : \mathbb{R}^L \to \mathbb{R}$ and $\varphi : \mathbb{R} \to \mathbb{R}^L$, such that

$$f(X) = \varrho\left(\frac{1}{N}\sum_{i=1}^{N}\varphi(X_i)\right),$$

where X_i is the ith element of X.

Proof. See Wagstaff et al. (2019) and Zaheer et al. (2017). □

Appendix B Concentration of Measure

Proposition 7 (Gaussian Poincaré inequality)**.** *Suppose $f : \mathbb{R}^N \to \mathbb{R}$ is a continuously differentiable function and $z \sim \mathcal{N}(\mathbf{0}_N, \mathbf{I}_N)$ is a standard Gaussian random vector. Then:*

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

Moreover, if $z \sim \mathcal{N}(\mathbf{0}_N, \Sigma)$ is a Gaussian random vector with mean zero and covariance matrix Σ, then:

$$\mathrm{Var}[f(z)] \leq \mathbb{E}\left[\nabla f^\top(z)\Sigma\nabla f(z)\right].$$

Proof. See Boucheron et al. (2013), Theorem 3.20. □

Appendix C Proofs

C.1 Proof of Proposition 1

Proof. Let $\pi \in \mathcal{S}_N$ be a permutation matrix and $Y \equiv \pi X$. The Bellman equation can be written as:

$$v(x, Y) = \max_u \{r(x, u, Y) + \beta \mathbb{E}\left[v(x', Y')\right]\} \tag{C.1}$$

$$\text{s.t. } x' = g(x, u) + \sigma w + \eta\omega \tag{C.2}$$

$$Y' = G(Y) + \Omega\pi W + \eta\omega\pi \mathbf{1}_N. \tag{C.3}$$

Assuming $\frac{\partial g(x,u)}{\partial u} \neq 0$, the Euler equation can be written as

$$\frac{r_u\big(x, u(x, Y), Y\big)}{g_u\big(x, u(x, Y)\big)} + \beta\mathbb{E}\left[r_x\big(x', u(x', Y'), Y'\big) - r_u\big(x', u(x', Y'), Y'\big)\frac{g_x\big(x', u(x', Y')\big)}{g_u\big(x', u(x', Y')\big)}\right] = 0,$$

where $r_u \equiv \frac{\partial r}{\partial u}$, $r_x \equiv \frac{\partial r}{\partial x}$, $g_x \equiv \frac{\partial g}{\partial x}$, and $g_u \equiv \frac{\partial g}{\partial u}$. Since r is permutation invariant with respect to X, we have

$$r_u\big(x, u(x, Y), Y\big) = r_u\big(x, u(x, Y), X\big)$$
$$r_x\big(x, u(x, Y), Y\big) = r_x\big(x, u(x, Y), X\big).$$

Using the fact that $G(Y) = \pi G(X)$ and $\Omega\pi = \pi\Omega$

$$Y' = \pi\big(G(X) + \Omega W + \eta\omega \mathbf{1}_N\big) = \pi X'. \tag{C.4}$$

Defining $u(x, \pi X) = u \circ \pi(x, X) \equiv u_\pi(x, X)$, the Euler equation becomes:

$$\frac{r_u\big(x, u_\pi(x, X), X\big)}{g_u\big(x, u_\pi(x, X)\big)} + \beta\mathbb{E}\left[r_x\big(x', u_\pi(x', X'), X'\big) - r_u\big(x', u_\pi(x', X'), X'\big)\frac{g_x\big(x', u_\pi(x', X')\big)}{g_u\big(x', u_\pi(x', X')\big)}\right] = 0,$$

where:

$$x' = g(x, u) + \sigma w + \eta\omega \tag{C.5}$$

$$X' = G(X) + \Omega W + \eta\omega \mathbf{1}_N. \tag{C.6}$$

Since the solution of the Euler equation is unique, then:

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

Given the permutation invariant optimal policy, u, and reward function, r, the Bellman equation can be written as:

$$v(x, Y) = r\big(x, u(x, X), X\big) + \beta \mathbb{E}\left[v(x', Y')\right],$$

where x' and Y' are described by equations (C.2) and (C.3). Using equation (C.4) and defining $v(x, \pi X) = v \circ \pi(x, X) \equiv v_\pi(x, X)$, the Bellman equation can be written as:

$$v_\pi(x, X) = r\big(x, u(x, X), X\big) + \beta \mathbb{E}\left[v_\pi(x', X')\right],$$

where x' and X' are described by equations (C.5) and (C.6). By the contraction mapping theorem, the solution of the Bellman equation is unique. Therefore:

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

□

C.2 Proof of Proposition 3

Proof. By the Chebyshev inequality:

$$\mathbb{P}\left(\left|f(z^1) - \mathbb{E}\left[f(z)\right]\right| \geq \epsilon\right) \leq \frac{\mathrm{Var}\big[f(z)\big]}{\epsilon^2}.$$

By the Gaussian Poincaré inequality (Proposition 7):

$$\mathrm{Var}\big[f(z)\big] \leq \mathbb{E}\left[\nabla f(z)^\top \Sigma \nabla f(z)\right].$$

Therefore:

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

Since a covariance matrix is symmetric and positive semi-definite, there exists an orthogonal matrix U (i.e. $U^\top = U^{-1}$) and a diagonal matrix Λ such that:

$$\Sigma = U \Lambda U^\top,$$

where Λ is the diagonal matrix of the eigenvalues of Σ. Hence:

$$\nabla f(z)^\top \Sigma \nabla f(z) = \sum_{i=1}^{N} \lambda_i \big[U^\top \nabla f(z)\big]_i^2 \leq \max_{i=1,..,N}\{\lambda_i\} \|\nabla f(z)\|^2 = \rho(\Sigma)\|\nabla f(z)\|^2,$$

where $\rho(\Sigma) \equiv \max_{i=1,..,N}\{\lambda_i\}$ is the spectral radius of Σ. Therefore,

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

□

C.3 Proof of Proposition 4

Proof. Since the price function $p(X)$ is permutation invariant and $\hat{u}$ is differentiable, the agent problem can be written as:

$$v(x, X) = \max_{u}\left\{p(X)\frac{x^{1-\mu}}{1-\mu} - \frac{\gamma}{2}u^2 + \beta\mathbb{E}\left[v(x', X')\right]\right\} \tag{C.7}$$

$$\text{s.t. } x' = (1-\delta)x + u + \sigma w + \eta\omega \tag{C.8}$$

$$X' = \left\{(1-\delta)\hat{x} + \hat{u}(\hat{x}, X) + \sigma\hat{w} + \eta\omega \text{ for all } (\hat{x}, \hat{w}) \in (X, W)\right\}. \tag{C.9}$$

The first-order condition is:

$$-\gamma u + \beta\mathbb{E}\left[\frac{\partial v(x', X')}{\partial x'}\right] = 0. \tag{C.10}$$

□

Given that the other agents' law of motion is independent of the agent of interest's choice, the envelope condition is:

$$\frac{\partial v(x', X')}{\partial x'} = P(X')(x')^{-\mu} + \gamma(1-\delta)u', \tag{C.11}$$

where u' is the optimal policy in the next period. Combining the first-order condition (C.10) and the envelope condition (C.11) yields:

$$-\gamma u(x, X) + \beta\mathbb{E}\left[P(X')(x')^{-\mu} + \gamma(1-\delta)u(x', X')\right] = 0. \tag{C.12}$$

In a symmetric equilibrium, all the agents use the same policy function, i.e., $\hat{u}(.,.) = u(.,.)$, and the laws of motion can be rewritten as:

$$x' = (1-\delta)x + u + \sigma w + \eta\omega \tag{C.13}$$

$$X' = \left\{(1-\delta)\hat{x} + u(\hat{x}, X) + \sigma\hat{w} + \eta\omega \text{ for all } (\hat{x}, \hat{w}) \in (X, W)\right\}. \tag{C.14}$$

For the case of $\mu = 0$, the Euler equation becomes:

$$-\gamma u(x, X) + \beta \mathbb{E}\left[P(X') + \gamma(1-\delta)u(x', X')\right] = 0. \tag{C.15}$$

In this case, the x drops out of the Euler equation. Hence, u does not depend on x, and the Euler equation can be rewritten as:

$$-\gamma u(X) + \beta \mathbb{E}\left[P(X') + \gamma(1-\delta)u(X')\right] = 0, \tag{C.16}$$

and the law of motion for X is described by equation (C.14).

Appendix D LQ Formulations

This section solves the special case of the LQ model. The purpose is to demonstrate the symmetry and concentration of measure when we know the closed-form solution of the problem.

D.1 Proof of Proposition 5: Linear-quadratic Gaussian control

With an LQ Gaussian model, we know that the policy will be affine. Therefore, $\hat{u}$ for other firms can be written as

$$\hat{u}(X_i, X) = H_0 + H_2 X_i + \frac{H_1}{N} \sum_{i=1}^{N} X_i \quad i = 1, ..., N, \tag{D.1}$$

for some undetermined H_0, H_1, and H_2. As shown in Proposition 4, in a permutation invariant symmetric equilibrium, $u(\hat{x}, X)$ does not depend on $\hat{x}$ and, hence, $H_2 = 0$. The symmetric constants, which we arbitrarily define as $\frac{H_1}{N}$ for all of the X vector, are the only ones that can fulfill permutation invariance with respect to the π permutations defined in equation (23). The law of motion for $\vec{x}$ is then:

$$\vec{x}' = A\vec{x} + Bu + C\vec{w} \tag{D.2}$$

where A, B, and C are defined in equations (24), (25), and (26). The reward and policy u of the firm of interest at state $\vec{x}$ take the form:

$$-\vec{x}^\top R\vec{x} - u^\top Q u, \tag{D.3}$$

where R and Q are defined in equations (27) and (28).

For a given pair of (H_0, H_1), the solution of this LQ problem has the form:

$$\vec{v}(\vec{x}) = -\vec{x}^\top P\vec{x} - d \tag{D.4}$$

$$\vec{u}(\vec{x}) = -F\vec{x}, \tag{D.5}$$

where P solves the following discounted algebraic matrix Riccati equation:

$$P = R + \beta A^\top PA - \beta^2 A^\top PB\big(Q + \beta B^\top PB\big)^{-1} B^\top PA, \tag{D.6}$$

d can be found via:

$$d = \frac{\beta}{1-\beta}\text{trace}\big(PCC^\top\big), \tag{D.7}$$

and F satisfies:

$$F = \beta\big(Q + \beta B^\top PB\big)^{-1} B^\top PA. \tag{D.8}$$

Ljungqvist and Sargent (2018, ch. 5) derive equations (D.6), (D.7), and (D.8).

D.2 Proof of Proposition 5: Permutation invariance

Now, we illustrate the permutation invariance of the optimal policy and value function of the LQ model described in Proposition 5. Define $\vec{y} \equiv \pi\vec{x}$. The value function associated with $\vec{y}$ in (31) can be written as:

$$v(\vec{y}) = -\vec{y}^\top P\vec{y} - d = -\vec{x}^\top\big(\pi^\top P\pi\big)\vec{x} - d,$$

where P solves the discounted algebraic Riccati equation:

$$P = R + \beta A^\top PA - \beta^2 A^\top PB\big(Q + \beta B^\top PB\big)^{-1} B^\top PA. \tag{D.9}$$

Sandwiching both sides of Equation (D.9) by $\pi^\top$ and π and using the fact that R is permutation invariant (i.e., $\pi^\top R\pi = R$) yields:

$$\pi^\top P\pi = R + \beta\pi^\top A^\top PA\pi - \beta^2\pi^\top A^\top PB\big(Q + \beta B^\top PB\big)^{-1} B^\top PA\pi.$$

Since the deterministic part of the laws of motion A is permutation equivariant (i.e., $A\pi = \pi A$) and the vector B remains unchanged under a permutation (i.e., $\pi B = B$), the above equation can be rewritten as:

$$\pi^\top P\pi = R + \beta A^\top\big(\pi^\top P\pi\big)A - \beta^2 A^\top\big(\pi^\top P\pi\big)B\big[Q + \beta B^\top(\pi^\top P\pi)B\big]^{-1} B^\top(\pi^\top P\pi)A. \tag{D.10}$$

Notice that $\pi^\top P\pi$ solves the same discounted algebraic Riccati equation (D.9) as P. Since the solution of equation (D.9) is unique:

$$\pi^\top P\pi = P.$$

The constant term d can be written as

$$\begin{aligned} d &= \beta(1-\beta)^{-1}\text{trace}(P\pi CC^\top\pi^\top) \\ &= \beta(1-\beta)^{-1}\text{trace}(\pi^\top P\pi CC^\top) \\ &= \beta(1-\beta)^{-1}\text{trace}(PCC^\top). \end{aligned} \tag{D.11}$$

As shown above, d remains unchanged under permutation and, therefore:

$$v(\vec{y}) = v(\pi\vec{x}) = v(\vec{x}).$$

The optimal policy $\vec{u}(\vec{y})$ can be written as

$$u(\vec{y}) = -\beta\big(Q+\beta B^\top PB\big)^{-1}B^\top PA\vec{y} = -\beta\big(Q+\beta B^\top PB\big)^{-1}B^\top(\pi^\top P\pi)A\vec{x}.$$

Since $(\pi^\top P\pi) = P$, then:

$$u(\vec{y}) = u(\pi\vec{x}) = u(\vec{x}).$$

Structure of the P Quadratic Form. When solving for Proposition 5, you can further find that the permutation invariant matrix P is of the form

$$P \equiv \left[\begin{array}{cc:c} p_1 & p_2 & \frac{p_3}{N}\mathbf{1}_N^\top \\ p_2 & 0 & \frac{p_4}{N}\mathbf{1}_N^\top \\ \hdashline \frac{p_3}{N}\mathbf{1}_N & \frac{p_4}{N}\mathbf{1}_N & \frac{p_5}{N^2}\mathbf{1}_N\mathbf{1}_N^\top \end{array}\right] \tag{D.12}$$

for some $p_1, \ldots p_5$. The lower block of this matrix is $\frac{p_5}{N^2}\mathbf{1}_N\mathbf{1}_N^\top$ (the $N\times N$ matrix of 1s is then the outer-product $\mathbf{1}_N\mathbf{1}_N^\top$.) This shows the symmetry with respect to the $i = 2, \ldots N$ states in $\vec{x}$ due to the block structure similar to that in equation (23), and the magnitude of their interaction (i.e., the lower block decreases at rate N^2 rather than N).

D.3 Proof of Proposition 5: Concentration of measure

The optimal policy for $\vec{x}\,'$ can be written as:

$$u(\vec{x}\,') = -F\vec{x}\,' = -F(A\vec{x} + Bu + C\vec{w}).$$

The optimal policy defined in Proposition 5 for a given H_0 and H_1 is $u(\vec{x}) = H_0 + \frac{H_1}{N}\sum_{i=1}^{N} X_i = H_0 + H_1\mathbb{E}[x]$, and, hence, a bounded function in N according to Definition 3. For any finite H_1 (including the equilibrium value), the gradient of $u(\vec{x}\,')$ with respect to the idiosyncratic shocks of all other firms $W_{-1} \equiv \begin{bmatrix} W_2 & \dots & W_N \end{bmatrix}$ becomes:

$$\nabla_{W_{-1}} u(\vec{x}\,') = \frac{\sigma H_1}{N}\mathbf{1}_{N-1}.$$

Therefore, the square of the norm of the gradient of the policy can be bounded by:

$$\mathbb{E}\left[\|\nabla_{W_{-1}} u(\vec{x}\,')\|^2\right] = \sigma^2 H_1^2\left(\frac{N-1}{N^2}\right) \leq \frac{\sigma^2 H_1^2}{N-1}.$$

Using Proposition 3 and conditioning on w, ω, and $\vec{x}$:

$$\mathbb{P}\left(\left|u(\vec{x}\,'^1) - \mathbb{E}\left[u(\vec{x}\,') \mid w, \omega, \vec{x}\right]\right| \geq \epsilon\right) \leq \frac{\sigma^2 H_1^2}{\epsilon^2}\frac{1}{N-1}. \tag{D.13}$$

The value function can be written as:

$$v(\vec{x}\,') = -\vec{x}\,'^{\top} P\vec{x}\,' - d. \tag{D.14}$$

Defining $X'_{-1} \equiv \begin{bmatrix} X'_2 & \dots & X'_N \end{bmatrix}^{\top}$ and given the symmetry in equations (33) and (D.12) by using the chain rule and additive structure of the idiosyncratic shocks, the gradient of the value function is:

$$\nabla_{W_{-1}} v(\vec{x}\,') = -2\sigma\left[\frac{p_3 + p_4 x'}{N} + \frac{p_5 x'}{N^2}\right]\mathbf{1}_{N-1} - \frac{2p_5\sigma}{N^2}\mathbf{1}_{N-1}\mathbf{1}_{N-1}^{\top}X'_{-1},$$

and then:

$$\begin{aligned}\|\nabla_{W_{-1}} v(\vec{x}\,')\|^2 =& \left[2\sigma\left(p_3 + p_4 x' + \frac{p_5 x'}{N}\right)\right]^2\frac{N-1}{N^2} + 8\sigma^2 p_5\left(p_3 + p_4 x' + \frac{p_5 x'}{N}\right)\mathbf{1}_{N-1}^{\top}X'_{-1}\frac{N-1}{N^3} + \\ & (2\sigma p_5)^2 X'^{\top}_{-1}\mathbf{1}_{N-1}\mathbf{1}_{N-1}^{\top}X'_{-1}\frac{N-1}{N^4}\end{aligned}$$

Since conditional on $(w, \omega, \vec{x})$, x' is not a random variable and X'_{-1} is a normal random vector

with conditional mean $\mathbb{E}\left[X'_{\text{-}1} \mid w, \omega, \vec{x}\right] = X_{\text{-}1} + \big(u(X) + \eta\omega\big)\mathbf{1}_{N-1}$ we have:

$$\begin{aligned}\mathbb{E}\left[\|\nabla_{W_{\text{-}1}} v(\vec{x}\,')\|^2 \;\middle|\; w, \omega, \vec{x}\right] =& \frac{(N-1)^2}{N^2}\Big[2\sigma\big(p_3 + p_4 x' + \frac{p_5 x'}{N}\big)\Big]^2 \frac{1}{N-1} + \\ & 8\frac{(N-1)^3}{N^3}\sigma^2 p_5\Big[p_3 + p_4 x' + \frac{p_5 x'}{N}\Big]\Big(\frac{\mathbf{1}_{N-1}^\top X_{\text{-}1}}{N-1} + u(X) + \eta\omega\Big)\frac{1}{N-1} + \\ & \frac{N-1}{N^4}(2\sigma p_5)^2 \mathbb{E}\left[{X'_{\text{-}1}}^\top \mathbf{1}_{N-1}\mathbf{1}_{N-1}^\top X'_{\text{-}1} \;\middle|\; w, \omega, \vec{x}\right]\end{aligned}$$

Considering $X'_{\text{-}1}$ is a normal random vector with conditional covariance $\operatorname{cov}\left(X'_{\text{-}1} \mid w, \omega, \vec{x}\right) = \sigma^2 \mathbf{I}_{N-1}$, we have:[20]

$$\frac{N-1}{N^4}\mathbb{E}\left[{X'_{\text{-}1}}^\top \mathbf{1}_{N-1}\mathbf{1}_{N-1}^\top X'_{\text{-}1} \;\middle|\; w, \omega, \vec{x}\right] = \Big[\frac{(N-1)^3}{N^4}\sigma^2 + \frac{(N-1)^4}{N^4}\big(\frac{\mathbf{1}_{N-1}^\top X_{\text{-}1}}{N-1} + u(X) + \eta\omega\big)^2\Big]\frac{1}{N-1}$$

Since $\frac{\mathbf{1}_{N-1}^\top X_{\text{-}1}}{N-1}$ and $u(X)$ are bounded functions in N (Definition 3), there exists K_1 such that:

$$\Big|\frac{\mathbf{1}_{N-1}^\top X_{\text{-}1}}{N-1} + u(X) + \eta\omega\Big| \leq K_1.$$

Note that $|p_5|$ and $\big|p_3 + p_4 x' + \frac{p_5 x'}{N}\big|$ are both bounded, and $\frac{(N-1)^3}{N^4}\sigma^2 \leq \sigma^2\frac{(N-1)^4}{N^4}$. Therefore, by the Cauchy-Schwarz inequality, there exists a positive C_v such that

$$\mathbb{E}\left[\|\nabla_{W_{\text{-}1}} v(\vec{x}\,')\|^2 \;\middle|\; w, \omega, \vec{x}\right] \leq \frac{\sigma^2 C_v^2}{N-1}.$$

Hence, by Proposition 3:

$$\mathbb{P}\left(\Big|v(\vec{x}\,'^1) - \mathbb{E}\left[v(\vec{x}\,') \mid w, \omega, \vec{x}\right]\Big| \geq \epsilon\right) \leq \frac{\sigma^2 C_v^2}{\epsilon^2}\frac{1}{N-1}. \tag{D.15}$$

The inequalities in (D.13) and (D.15) show that even with a single draw, we can calculate the expectation as N increases. A high-dimensional state space is a help rather than a hindrance when calculating expectations (as long as there is sufficient independence in the states) because of the concentration of measure phenomenon. Inequality (35) shows that, with a single draw of $W_{\text{-}1}$, we know the precise distribution of the deviation from the expectation, and with a large enough N, it converges to 0. We could take multiple draws if required. But since Monte Carlo integration converges very slowly in general, the gains in extra accuracy will be minor even if we increase the number of draws substantially.

[20] If $\vec{z}$ is a random vector with mean $\vec{m}$ and covariance matrix Σ, then, for any matrix Λ, $\mathbb{E}\left[\vec{z}^\top \Lambda \vec{z}\right] = \operatorname{trace}(\Lambda\Sigma) + \vec{m}^\top \Lambda \vec{m}$.

The Euler residuals corresponding to a Monte Carlo draw of W is defined as:

$$\varepsilon(X,u) = \gamma u(X) - \beta \mathbb{E}_{w,\omega}\left[P(X') + \gamma(1-\delta)u(X')\right]$$

where $\mathbb{E}_{w,\omega}[\cdot]$ corresponds to expectation over w and ω. Substituting for a linear price and linear policy $u(X') = H_0 + \frac{H_1}{N}X_1 + \frac{H_1}{N}\sum_{i=2}^{N} X_i'$

$$\varepsilon(X,u) = \gamma u(X) - \beta\Big[\alpha_0 - \frac{\alpha_1}{N}\sum_{i=1}^{N}\big(X_i + u(X)\big)\Big] - \beta\gamma(1-\delta)\Big[H_0 + \frac{H_1}{N}\sum_{i=1}^{N}\big(X_i + u(X)\big)\Big] + \beta\sigma\Big[\alpha_1 - \gamma(1-\delta)H_1\Big]\frac{1}{N}\sum_{i=2}^{N} W_i$$

From the Euler equation (19), the first three terms sum up to zero, therefore:

$$\varepsilon(X,u) = \beta\sigma\Big[\alpha_1 - \gamma(1-\delta)H_1\Big]\frac{1}{N}\sum_{i=2}^{N} W_i.$$

Utilizing the fact that $W_i \sim \mathcal{N}(0,1)$, the Euler residual is a Gaussian random variable

$$\varepsilon(X,u) \sim \mathcal{N}\Big(0, \frac{N-1}{N^2}\beta^2\sigma^2\big[\alpha_1 - \gamma(1-\delta)H_1\big]^2\Big).$$

For a linear policy, the error in forecasting the policy in the next period is:

$$u(\vec{x}'^1) - \mathbb{E}\left[u(\vec{x}') \mid w, \omega, \vec{x}\right] = \frac{H_1\sigma}{N}\sum_{i=2}^{N} W_i,$$

therefore, it is a Gaussian random variable

$$u(\vec{x}'^1) - \mathbb{E}\left[u(\vec{x}') \mid w, \omega, \vec{x}\right] \sim \mathcal{N}\Big(0, \frac{N-1}{N^2}H_1^2\sigma^2\Big).$$

D.4 Representation of the v function

If it is known that the LQ problem satisfies certainty equivalence and the value function is quadratic and permutation invariant, v can be represented as:

$$\begin{aligned} v(x,X) &= -\Big(p_1 + d + 2p_2 x + 2p_3\frac{\sum_{i=1}^{N} X_i}{N} + 2p_4 x\frac{\sum_{i=1}^{N} X_i}{N} + p_5\big(\frac{\sum_{i=1}^{N} X_i}{N}\big)^2\Big) \\ &= \rho_v\big(x, \frac{1}{N}\sum_{i=1}^{N}\phi_v(X_i)\big) \end{aligned}$$

In other words, $\phi_v : \mathbb{R} \to \mathbb{R}^1$ (i.e., $L = 1$) is the identity function $\phi_v(X) = \Big[X\Big]$, and $\rho_v : \mathbb{R} \times \mathbb{R}^1 \to \mathbb{R}$ such that:

$$\rho_v(x, y) = -(p_1 + d + 2p_2x + 2p_3y + 2p_4xy + p_5y^2).$$

Appendix E Additional Numerical Results

E.1 More network variations

Table 2 reports a more complete list of experiments, including those of Table 1.

Table 2: Linear Model Performance in Additional Experiments

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%
	Shallow (2 layers)	159	16.6	3.7e-06	7.8e-07	7.6e-07	9.4e-03	33.53%
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%
	Shallow (2 layers)	137	17.0	1.6e-06	9.9e-07	9.5e-07	1.8e-02	59.41%
	Deep(8 layers)	241	115.3	2.8e-06	1.2e-06	1.0e-06	5.2e-05	0.16%
	Thin (64 nodes)	82	12.6	1.6e-06	9.1e-07	9.2e-07	3.8e-05	0.12%
	Wide (256 nodes)	61	197.9	1.8e-06	8.7e-07	8.0e-07	4.3e-05	0.13%
ϕ(ReLU)	Baseline	107	66.8	3.7e-06	3.3e-07	3.3e-07	2.7e-05	0.09%
	L = 1	88	66.0	1.8e-05	2.3e-07	2.4e-07	2.8e-05	0.10%
	L = 2	86	66.3	1.3e-05	2.1e-07	2.2e-07	2.6e-05	0.08%
	L = 8	70	67.8	3.0e-05	5.9e-07	5.9e-07	3.3e-05	0.11%
	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%
	Shallow(ϕ : 1 layer)	58	50.4	8.7e-06	1.5e-07	1.5e-07	2.5e-05	0.08%
	Shallow(ρ : 2 layers)	89	34.0	3.1e-06	4.2e-07	4.2e-07	2.7e-05	0.09%
	Deep(ϕ : 4 layers, ρ : 8 layers)	242	165.1	2.1e-03	2.2e-03	2.1e-03	2.7e-03	8.50%
	Very Thin(ϕ, ρ : 16 nodes)	45	1.2	1.2e-05	4.9e-07	4.9e-07	3.2e-05	0.10%
	Thin(ϕ, ρ : 64 nodes)	87	17.0	1.1e-05	4.5e-07	4.5e-07	3.0e-05	0.10%
	Wide(ϕ, ρ : 256 nodes)	115	264.7	5.4e-06	4.0e-07	4.0e-07	4.1e-05	0.13%

E.2 Nonlinear demand function

The results in Table 3 confirm the conventional wisdom in the deep learning literature. First, *Very Shallow* in ϕ(ReLU), and *Very Shallow* and *Shallow* in ϕ(Moments) highlight the role of depth and generalizability. In these experiments, the MSE of the Euler residuals on both the

training and the validation data is very low. However, as seen in Test MSE (ε), the approximated policy is incapable of generalization. Second, for cases where there is nonlinearity in the function of interest, depth plays a more important role than pure capacity (number of parameters). For instance, in ϕ(Moments), *Thin* has way fewer parameters than *Shallow*, but it has more depth. *Thin* has an extraordinarily higher generalization power than *Shallow*.

Table 3: Nonlinear Model Performance

group	description	Time (s)	Params (K)	Train MSE (ε)	Test MSE (ε)	Val MSE (ε)
ϕ(Moments)	Baseline	26	49.8	6.0e-06	5.0e-06	3.8e-06
	Moments (1)	24	49.4	2.7e-05	6.5e-06	3.4e-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
	Shallow (2 layers)	35	17.0	5.8e-06	1.2e+00	4.4e-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 = 1	109	66.3	9.4e-06	1.3e-05	4.5e-06
	L = 2	73	66.6	1.0e-05	3.3e-06	2.3e-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