Deep Learning for Solving Economic Models

Jesús Fernández-Villaverde[1] and Galo Nuño[2]

August 21, 2023

[1]University of Pennsylvania

[2]Banco de España

Functional equations

Functional equations

- A large class of problems in economics search for a function d that solves a *functional equation*:

$$\mathcal{H}(d) = \mathbf{0}$$

- More formally:

  1. Let J^1 and J^2 be two functional spaces and let $\mathcal{H} : J^1 \to J^2$ be an operator between these two spaces.
  2. Let $\Omega \subseteq \mathbb{R}^l$.
  3. Then, we need to find a function $d : \Omega \to \mathbb{R}^m$ such that $\mathcal{H}(d) = \mathbf{0}$.

- Points to remember:

  1. Regular equations are particular examples of functional equations.
  2. $\mathbf{0}$ is the space zero, different in general that the zero in the reals.
  3. This formalism deals both with market equilibrium and social planner problems.

Example I: decision rules

- Take the basic stochastic neoclassical growth model:

$$\max \mathbb{E}_0 \sum_{t=0}^{\infty} \beta^t u(c_t)$$
$$c_t + k_{t+1} = e^{z_t} k_t^{\alpha} + (1-\delta) k_t, \ \forall \ t > 0$$
$$z_t = \rho z_{t-1} + \sigma \varepsilon_t, \ \varepsilon_t \sim \mathcal{N}(0,1)$$

- The first-order condition:

$$u'(c_t) = \beta \mathbb{E}_t \left\{ u'(c_{t+1}) \left(1 + \alpha e^{z_{t+1}} k_{t+1}^{\alpha-1} - \delta\right) \right\}$$

Example I: decision rules

- There is a decision rule (a.k.a. policy function) that gives the optimal choice of consumption and capital tomorrow given the states today:

$$d = \begin{cases} d^1(k_t, z_t) = c_t \\ d^2(k_t, z_t) = k_{t+1} \end{cases}$$

- Then:

$$\mathcal{H} = u'\left(d^1(k_t, z_t)\right)$$
$$-\beta \mathbb{E}_t \left\{ u'\left(d^1\left(d^2(k_t, z_t), z_{t+1}\right)\right) \left(1 + \alpha e^{z_{t+1}} \left(d^2(k_t, z_t)\right)^{\alpha-1} - \delta\right) \right\} = 0$$

- If we find d, and a transversality condition is satisfied, we are done!

Example II: conditional expectations

- Let us go back to our Euler equation:

$$u'(c_t) - \beta \mathbb{E}_t \left\{ u'(c_{t+1}) \left(1 + \alpha e^{z_{t+1}} k_{t+1}^{\alpha-1} - \delta\right) \right\} = 0$$

- Define now:

$$d = \begin{cases} d^1(k_t, z_t) = c_t \\ d^2(k_t, z_t) = \mathbb{E}_t \left\{ u'(c_{t+1}) \left(1 + \alpha e^{z_{t+1}} k_{t+1}^{\alpha-1} - \delta\right) \right\} \end{cases}$$

- Why? Example: ZLB.

- Then:

$$\mathcal{H}(d) = u'\left(d^1(k_t, z_t)\right) - \beta d^2(k_t, z_t) = \mathbf{0}$$

Example III: value functions

- There is a recursive problem associated with the previous sequential problem:

$$V(k_t, z_t) = \max_{k_{t+1}} \{u(c_t) + \beta \mathbb{E}_t V(k_{t+1}, z_{t+1})\}$$

$$c_t = e^{z_t} k_t^{\alpha} + (1-\delta) k_t - k_{t+1}, \ \forall \ t > 0$$

$$z_t = \rho z_{t-1} + \sigma \varepsilon_t, \ \varepsilon_t \sim \mathcal{N}(0,1)$$

- Then:

$$d(k_t, z_t) = V(k_t, z_t)$$

and

$$\mathcal{H}(d) = d(k_t, z_t) - \max_{k_{t+1}} \{u(c_t) + \beta \mathbb{E}_t d(k_{t+1}, z_{t+1})\} = \mathbf{0}$$

Deep learning to solve functional equations

- General idea: substitute $d(\mathbf{x})$ by $d^n(\mathbf{x}, \theta)$ where θ is an $n-$dim vector of coefficients to be determined.

- We can "learn" this function with deep learning (notice: notation with just one layer):

$$d \cong d^n(\mathbf{x}; \theta) = \theta_0 + \sum_{m=1}^{M} \theta_m \phi(z_m)$$

  where $\phi(\cdot)$ is an arbitrary activation function and:

$$z_m = \sum_{n=0}^{N} \theta_{n,m} x_n$$

- We need to sample the desired function and minimize a loss function.

Outline

- Natural sampling: simulated paths of the economy.
- Natural loss function: the implied error in the equilibrium/optimality conditions, as loss function.
- Natural minimization algorithm: Stochastic gradient descent.
- However, you want to think about this more as a framework than a concrete set of instructions.
- Let us go over different applications. Enumeration, no exhaustive list.

Examples of code

1. An LQ optimal control problem:

   https://github.com/Mekahou/Fun-Stuff/blob/main/codes/linear%20quadratic%20DP%20DNN/3.%20LQ_DP_DNN_Training_Main.ipynb

2. A Neoclassical growth model (discrete time):

   https://colab.research.google.com/drive/1jbSti3LkxASZg04Bkod0EBJFXdf6xu9_?usp=sharing

3. A Neoclassical growth model (continuous time):

   https://colab.research.google.com/drive/1rFfqbJYn26_nm-CS21Fbw_QV7ccM8Xl1T?usp=sharing

4. An OLG model:

   https://github.com/sischei/DeepEquilibriumNets

5. A Krusell-Smith and a financial frictions HA code:

   https://github.com/jesusfv/financial-frictions

Dynamic programming

Solving high-dimensional dynamic programming problems (discrete time)

- Our goal is to solve the Bellman equation globally:

$$
\begin{aligned}
V(\mathbf{x}) &= \max_{\boldsymbol{\alpha}} r(\mathbf{x}, \boldsymbol{\alpha}) + \beta * \mathbb{E}V(\mathbf{x}') \\
\text{s.t. } \mathbf{x}' &= f(\mathbf{x}, \boldsymbol{\alpha}) \\
G(\mathbf{x}, \boldsymbol{\alpha}) &\leq \mathbf{0} \\
H(\mathbf{x}, \boldsymbol{\alpha}) &= \mathbf{0}
\end{aligned}
$$

- Notice that framework is very general. Usually we will not have all these constraints.
- Think about the cases where we have many state variables. Trade-offs in terms of speed vs. scalability.

Solving high-dimensional dynamic programming problems (continuous time)

- We can also write the equivalent continuous-time Hamilton-Jacobi-Bellman (HJB) equation globally:

$$\rho V(\mathbf{x}) = \max_{\boldsymbol{\alpha}} r(\mathbf{x}, \boldsymbol{\alpha}) + \nabla_x V(\mathbf{x}) f(\mathbf{x}, \boldsymbol{\alpha}) + \frac{1}{2} tr(\sigma(\mathbf{x}))^T \Delta_x V(\mathbf{x}) \sigma(\mathbf{x}))$$
$$\text{s.t. } G(\mathbf{x}, \boldsymbol{\alpha}) \leq \mathbf{0}$$
$$H(\mathbf{x}, \boldsymbol{\alpha}) = \mathbf{0}$$

- Solution algorithm is absolutely the same.
- When discrete vs. continuous?

Neural networks

- We define four neural networks:
  1. $\tilde{V}(\mathbf{x}; \boldsymbol{\Theta}^V) : \mathbb{R}^N \to \mathbb{R}$ to approximate the value function $V(\mathbf{x})$.
  2. $\tilde{\alpha}(\mathbf{x}; \boldsymbol{\Theta}^\alpha) : \mathbb{R}^N \to \mathbb{R}^P$ to approximate the policy function $\boldsymbol{\alpha}$.
  3. $\tilde{\mu}(\mathbf{x}; \boldsymbol{\Theta}^\mu) : \mathbb{R}^N \to \mathbb{R}^{L_1}$, and $\tilde{\lambda}(\mathbf{x}; \boldsymbol{\Theta}^\lambda) : \mathbb{R}^N \to \mathbb{R}^{L_2}$ to approximate the Karush-Kuhn-Tucker (KKT) multipliers $\boldsymbol{\mu}$ and $\boldsymbol{\lambda}$.
- To simplify notation, we accumulate all weights in the matrix $\boldsymbol{\Theta} = (\boldsymbol{\Theta}^V, \boldsymbol{\Theta}^\alpha, \boldsymbol{\Theta}^\mu, \boldsymbol{\Theta}^\lambda)$.
- We could think about the approach as just one large neural network with multiple outputs.

Error criterion I (discrete time)

- The Bellman error:

$$err_B(\mathbf{x};\boldsymbol{\Theta}) \equiv r(\mathbf{x},\tilde{\alpha}(\mathbf{s};\boldsymbol{\Theta}^{\alpha})) + \beta * \mathbb{E}\tilde{V}(\mathbf{x}';\boldsymbol{\Theta}^{V}) - \tilde{V}(\mathbf{x};\boldsymbol{\Theta}^{V})$$

- The policy function error:

$$err_{\alpha}(\mathbf{x};\boldsymbol{\Theta}) \equiv \frac{\partial r(\mathbf{x},\tilde{\alpha}(\mathbf{x};\boldsymbol{\Theta}^{\alpha})}{\partial \boldsymbol{\alpha}} + \beta * D_{\alpha}f(\mathbf{x},\tilde{\alpha}(\mathbf{x};\boldsymbol{\Theta}^{\alpha}))^{T}\mathbb{E}\nabla_{x}\tilde{V}(\mathbf{x}';\boldsymbol{\Theta}^{V}) - D_{\alpha}G(\mathbf{x},\tilde{\alpha}(\mathbf{x};\boldsymbol{\Theta}^{\alpha}))^{T}\tilde{\mu}(\mathbf{x};\boldsymbol{\Theta}^{\mu}) - D_{\alpha}H(\mathbf{x},\tilde{\alpha}(\mathbf{x};\boldsymbol{\Theta}^{\alpha}))\tilde{\lambda}(\mathbf{x};\boldsymbol{\Theta}^{\lambda}),$$

where $D_{\alpha}G \in \mathbb{R}^{L_1 \times M}$, $D_{\alpha}H \in \mathbb{R}^{L_2 \times M}$, and $D_{\alpha}f \in \mathbb{R}^{N \times M}$ are the submatrices of the Jacobian matrices of G, H, and f respectively containing the derivatives with respect to α.

Error criterion I (continuous time)

- The HJB error:

$$err_{HJB}(\mathbf{x};\boldsymbol{\Theta}) \equiv r(\mathbf{x},\tilde{\alpha}(\mathbf{x};\boldsymbol{\Theta}^{\alpha})) + \nabla_x \tilde{V}(\mathbf{x};\boldsymbol{\Theta}^{V}) f(\mathbf{x},\tilde{\alpha}(\mathbf{x};\boldsymbol{\Theta}^{\alpha})) + \\ + \frac{1}{2} tr[\sigma(\mathbf{x})^T \Delta_x \tilde{V}(\mathbf{x};\boldsymbol{\Theta}^{V})\sigma(\mathbf{x})] - \rho \tilde{V}(\mathbf{x};\boldsymbol{\Theta}^{V})$$

- The policy function error:

$$err_{\alpha}(\mathbf{x};\boldsymbol{\Theta}) \equiv \frac{\partial r(\mathbf{x},\tilde{\alpha}(\mathbf{x};\boldsymbol{\Theta}^{\alpha}))}{\partial \boldsymbol{\alpha}} + D_{\alpha} f(\mathbf{x},\tilde{\alpha}(\mathbf{x};\boldsymbol{\Theta}^{\alpha}))^T \nabla_x \tilde{V}(\mathbf{x};\boldsymbol{\Theta}^{V}) \\ - D_{\alpha} G(\mathbf{x},\tilde{\alpha}(\mathbf{x};\boldsymbol{\Theta}^{\alpha}))^T \tilde{\mu}(\mathbf{x};\boldsymbol{\Theta}^{\mu}) - D_{\alpha} H(\mathbf{x},\tilde{\alpha}(\mathbf{x};\boldsymbol{\Theta}^{\alpha}))\tilde{\lambda}(\mathbf{x};\boldsymbol{\Theta}^{\lambda}),$$

where $D_{\alpha}G \in \mathbb{R}^{L_1 \times M}$, $D_{\alpha}H \in \mathbb{R}^{L_2 \times M}$, and $D_{\alpha}f \in \mathbb{R}^{N \times M}$ are the submatrices of the Jacobian matrices of G, H, and f respectively containing the derivatives with respect to α.

Error criterion II (discrete and continuous time)

- The constraint error is itself composed of the primal feasibility errors:

$$err_{PF_1}(\mathbf{x};\boldsymbol{\Theta}) \equiv \max\{0, G(\mathbf{x}, \tilde{\alpha}(\mathbf{x};\boldsymbol{\Theta}^{\alpha}))\}$$
$$err_{PF_2}(\mathbf{x};\boldsymbol{\Theta}) \equiv H(\mathbf{x}, \tilde{\alpha}(\mathbf{x};\boldsymbol{\Theta}^{\alpha}))$$

  the dual feasibility error:

$$err_{DF}(\mathbf{x};\boldsymbol{\Theta}) = \max\{0, -\tilde{\mu}(\mathbf{x};\boldsymbol{\Theta}^{\mu}\}$$

  and the complementary slackness error:

$$err_{CS}(\mathbf{x};\boldsymbol{\Theta}) = \tilde{\mu}(\mathbf{x};\boldsymbol{\Theta})^{T} G(\mathbf{x}, \tilde{\alpha}(\mathbf{x};\boldsymbol{\Theta}^{\alpha}))$$

- We combine these four errors by using the squared error as our loss criterion:

$$\mathcal{E}(\mathbf{x};\boldsymbol{\Theta}) \equiv \left|\left|err_{B/HJB}(\mathbf{x};\boldsymbol{\Theta})\right|\right|_2^2 + \left|\left|err_{\alpha}(\mathbf{x};\boldsymbol{\Theta})\right|\right|_2^2 + \left|\left|err_{PF_1}(\mathbf{x};\boldsymbol{\Theta})\right|\right|_2^2 +$$
$$+ \left|\left|err_{PF_2}(\mathbf{x};\boldsymbol{\Theta})\right|\right|_2^2 + \left|\left|err_{DF}(\mathbf{x};\boldsymbol{\Theta})\right|\right|_2^2 + \left|\left|err_{CS}(\mathbf{x};\boldsymbol{\Theta})\right|\right|_2^2$$

Training

- We train our neural networks by minimizing the above error criterion through minibatch gradient descent over points drawn from the ergodic distribution of the state vector.
- The efficient implementation of this last step is the key to the success of the algorithm.
- We start by initializing our network weights and we perform K learning steps called epochs, where K can be chosen in a variety of ways.
- For each epoch, we draw I points from the state space by simulating from the ergodic distribution.
- Then, we randomly split this sample into B minibatches of size S. For each minibatch, we define the minibatch error,by averaging the loss function over the batch.
- Finally, we perform minibatch gradient descent for all network weights, with η_k being the learning rate in the k-th epoch.

The continuous-time neoclassical growth model I

- We start with the continuous-time neoclassical growth model because it has closed-form solutions for the policy functions, which allows us to focus our attention on the analysis of the value function approximation.
- We can then back out the policy function from this approach and compare it to the results of the next step in which we approximate the policy functions themselves with a neural net.
- A single agent deciding to either save in capital or consume with a HJB equation :

$$\rho V(k) = \max_c U(c) + V'(k)[F(k) - \delta * k - c]$$

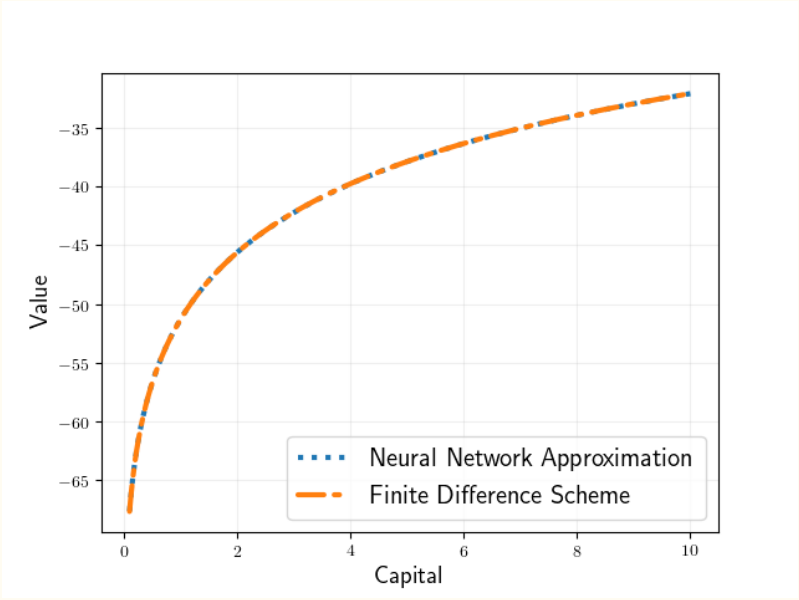
- Notice that $c = (U')^{-1}(V'(k))$. With CRRA utility, this simplifies further to $c = (V'(k))^{-\frac{1}{\gamma}}$.
- We set $\gamma = 2$, $\rho = 0.04$, $F(k) = 0.5 * k^{0.36}$, $\delta = 0.05$.

The continuous-time neoclassical growth model II

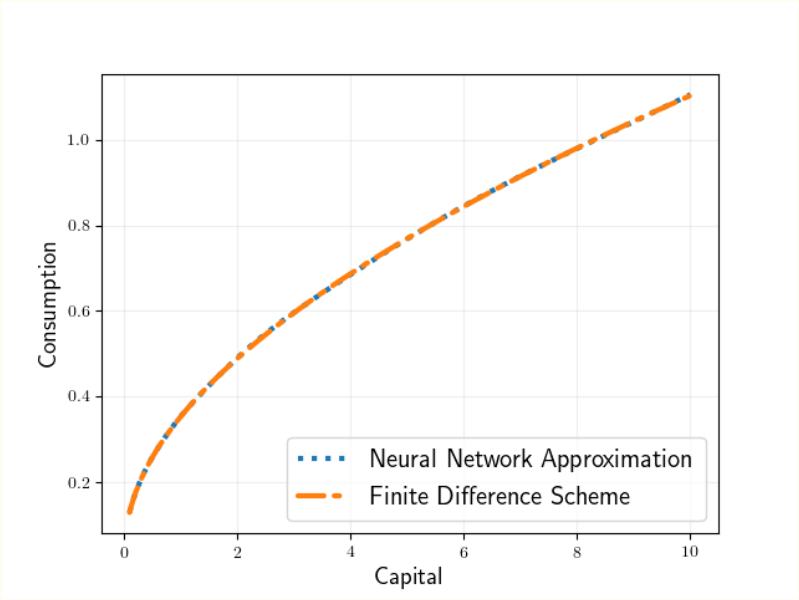
- We approximate the value function $V(k)$ with a neural network, $\tilde{V}(k;\Theta)$ with an "HJB error":

$$err_{HJB} = \rho\tilde{V}(k;\Theta) - U\left((U')^{-1}\left(\frac{\partial\tilde{V}(k;\Theta)}{\partial k}\right)\right) - \frac{\partial\tilde{V}(k;\Theta)}{\partial k}\left[F(k) - \delta * k - (U')^{-1}\left(\frac{\partial\tilde{V}(k;\Theta)}{\partial k}\right)\right]$$

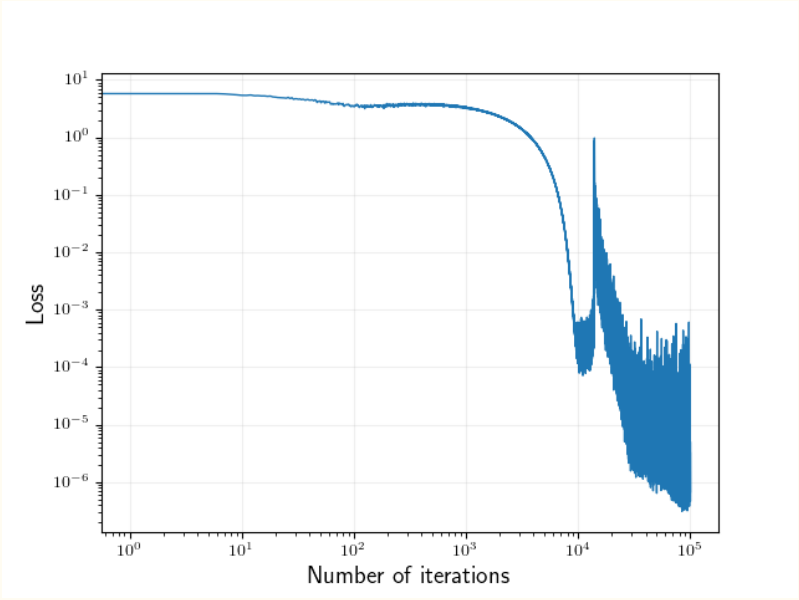
- Details:
  1. 3 layers.
  2. 8 neurons per layers.
  3. $\tanh(x)$ activation.
  4. Normal initialization $\mathcal{N}\left(0, 4\sqrt{\frac{2}{n_{input}+n_{output}}}\right)$ with input normalization.



(a) Value with closed-form policy



(c) Consumption with closed-form policy



(e) HJB error with closed-form policy

Approximating the policy function

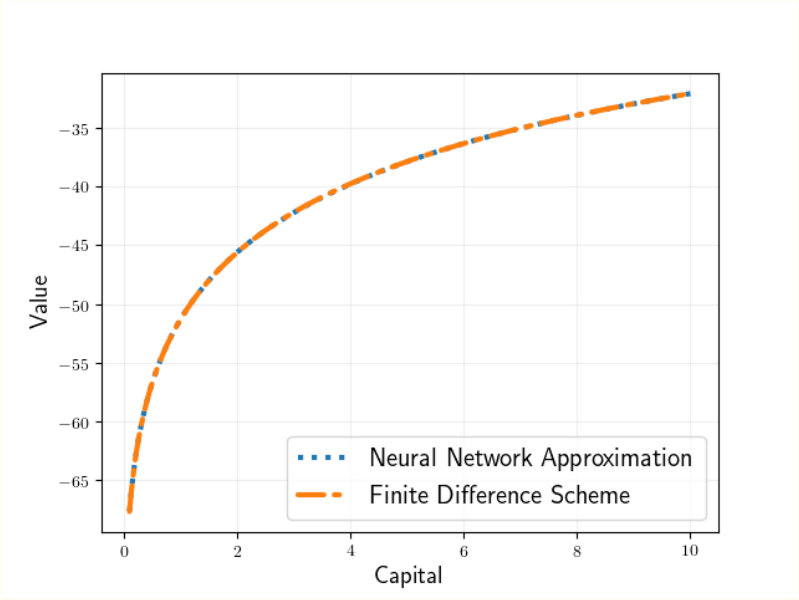
- Let us not use the closed-form consumption policy function but rather approximate said policy function directly with a policy neural network $\tilde{C}(k;\Theta^C)$.

- The new HJB error:

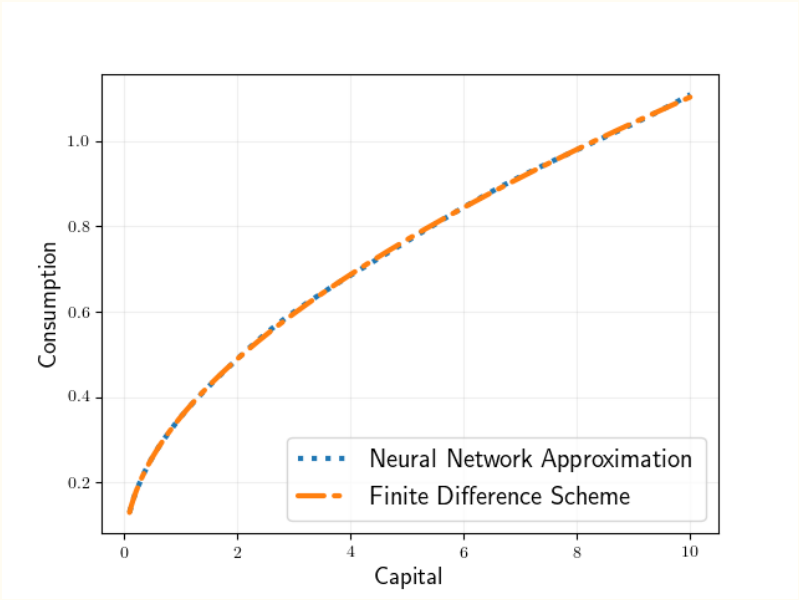
$$err_{HJB} = \rho\tilde{V}(k;\Theta^V) - U\left(\tilde{C}(k;\Theta^C)\right) - \frac{\partial\tilde{V}(k;\Theta^V)}{\partial k}\left[F(k) - \delta * k - \tilde{C}(k;\Theta^C)\right]$$

- Now we have a policy function error:

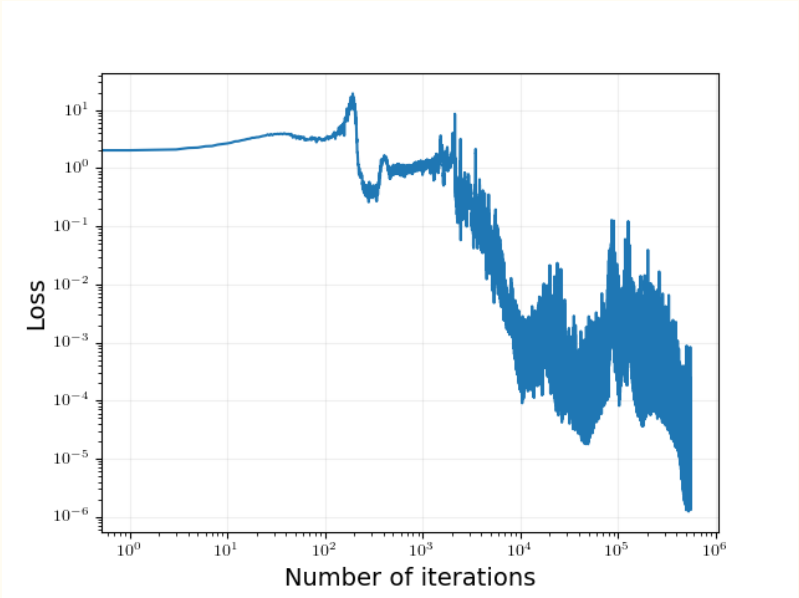
$$err_C = (U')^{-1}\left(\frac{\partial\tilde{V}(k;\Theta^V)}{\partial k}\right) - \tilde{C}(k;\Theta^C)$$



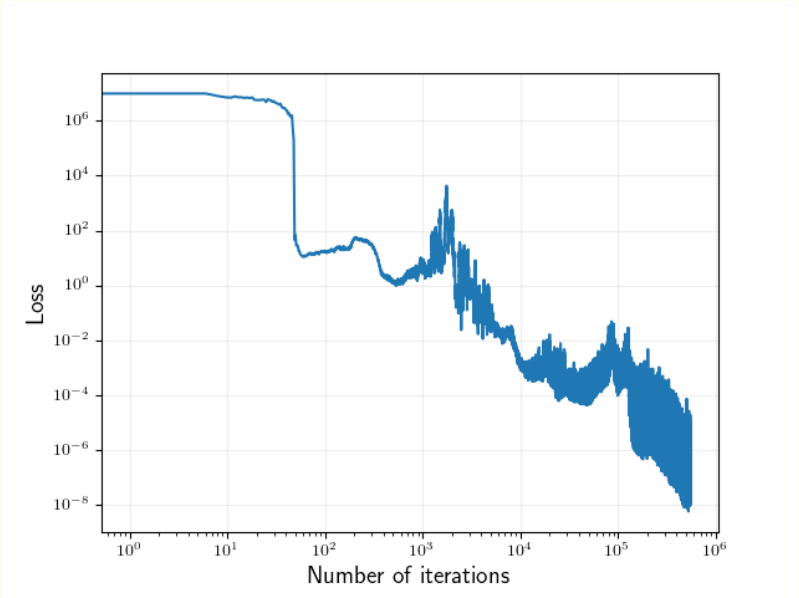
(b) Value with policy approximation



(d) Consumption with policy approximation



(f) HJB error with policy approximation



(g) Policy error with policy approximation

Models with heterogeneous agents

The challenge

- To compute and take to the data models with heterogeneous agents, we need to deal with:
  1. The distribution of agents G_t.
  2. The operator $H(\cdot)$ that characterizes how G_t evolves:

     $$G_{t+1} = H(G_t, S_t)$$

     or

     $$\frac{\partial G_t}{\partial t} = H(G_t, S_t)$$

     given the other aggregate states of the economy S_t.
- How do we track G_t and compute $H(G_t, S_t)$?

A common approach

- If we are dealing with N discrete types, we keep track of $N-1$ weights.
- If we are dealing with continuous types, we extract a finite number of features from G_t:
  1. Moments.
  2. Q-quantiles.
  3. Weights in a mixture of normals...
- We stack either the weights or features of the distribution in a vector μ_t.
- We assume μ_t follows the operator $h(\mu_t, S_t)$ instead of $H(G_t, S_t)$.
- We parametrize $h(\mu_t, S_t)$ as $h^j(\mu_t, S_t; \theta)$.
- We determine the unknown coefficients θ such that an economy where μ_t follows $h^j(\mu_t, S_t; \theta)$ replicates as well as possible the behavior an economy where G_t follows $H(\cdot)$.

Example: Basic Krusell-Smith model

- Two aggregate variables: aggregate productivity shock and household distribution $G_t(a, z)$ where:

$$\int G_t(a, z)da = K_t$$

- We summarize $G_t(a, \cdot)$ with the log of its mean: $\mu_t = \log K_t$ (extending to higher moments is simple, but tedious).

- We parametrize $\underbrace{\log K_{t+1}}_{\mu_{t+1}} = \underbrace{\theta_0(s_t) + \theta_1(s_t) \log K_t}_{h^j(\mu_t, s_t; \theta)}$.

- We determine $\{\theta_0(s_t), \theta_1(s_t)\}$ by OLS run on a simulation.

Problems

- No much guidance regarding feature and parameterization selection in general cases.
  - Yes, keeping track of the log of the mean and a linear functional form work well for the basic model. But, what about an arbitrary model?
  - Method suffers from "curse of dimensionality": difficult to implement with many state variables or high N/higher moments.
  - Lack of theoretical foundations (Does it converge? Under which metric?).

How can deep learning help?

- Deep learning addresses challenges:
  1. How to extract features from an infinite-dimensional object efficiently.
  2. How to parametrize the non-linear operator mapping how distributions evolve.
  3. How to tackle the "curse of dimensionality."
- Given time limitations, today I will discuss the last two points.
- In our notation of $y = f(\mathbf{x})$:
  1. $y = \mu_{t+1}$.
  2. $\mathbf{x} = (\mu_t, S_t)$.

Example: Financial frictions and the wealth distribution

Motivation

- Recently, many papers have documented the nonlinear relations between financial variables and aggregate fluctuations.
- For example, Jordà et al. (2016) have gathered data from 17 advanced economies over 150 years to show how output growth, volatility, skewness, and tail events all seem to depend on the levels of leverage in an economy.
- Similarly, Adrian et al. (2019a) have found how, in the U.S., sharply negative output growth follows worsening financial conditions associated with leverage.
- Can a fully nonlinear DSGE model account for these observations?
- To answer this question, we postulate, compute, and estimate a continuous-time DSGE model with a financial sector, modeled as a representative financial expert, and households, subject to uninsurable idiosyncratic labor productivity shocks.

The firm

- Representative firm with technology:

$$Y_t = K_t^{\alpha} L_t^{1-\alpha}$$

- Competitive input markets:

$$w_t = (1-\alpha) K_t^{\alpha} L_t^{-\alpha}$$
$$rc_t = \alpha K_t^{\alpha-1} L_t^{1-\alpha}$$

- Aggregate capital evolves:

$$\frac{dK_t}{K_t} = (\iota_t - \delta)\, dt + \sigma dZ_t$$

- Instantaneous return rate on capital dr_t^k:

$$dr_t^k = (rc_t - \delta)\, dt + \sigma dZ_t$$

The expert

- Representative expert (financial intermediary) with preferences:

$$\mathbb{E}_0\left[\int_0^\infty e^{-\widehat{\rho}t}\log(\widehat{C}_t)dt\right]$$

- Expert rents capital K_t to firms and issues risk-free debt B_t at rate r_t to households.
- Financial friction: expert cannot issue state-contingent claims and must absorb all risk from capital.
- Expert's net wealth $N_t = K_t - B_t$ evolves:

$$\begin{aligned} dN_t &= K_t dr_t^k - r_t B_t dt - \widehat{C}_t dt \\ &= \left[\left(r_t + \omega_t\left(rc_t - \delta - r_t\right)\right)N_t - \widehat{C}_t\right]dt + \sigma\omega_t N_t dZ_t \end{aligned}$$

  where $\omega_t \equiv \frac{K_t}{N_t}$ is the leverage ratio.

- K_t follows:

$$dK_t = dN_t + dB_t$$

Households I

- Continuum of infinitely-lived households with unit mass with preferences:

$$\mathbb{E}_0 \left[\int_0^\infty e^{-\rho t} \frac{c_t^{1-\gamma} - 1}{1-\gamma} dt \right]$$

- Heterogeneous in wealth a_m and labor productivity z_m for $m \in [0, 1]$.

- z_t units of labor valued at wage w_t evolves stochastically following a Markov chain:

  1. $z_t \in \{z_1, z_2\}$, with $z_1 < z_2$ and ergodic mean 1.
  2. Jump intensity from state 1 to state 2: λ_1 (reverse intensity is λ_2).

- Distribution of households $G_t(a, z)$.

Households II

- Households save $a_t \geq 0$ in the riskless debt issued by experts with an interest rate r_t:

$$da_t = (w_t z_t + r_t a_t - c_t)\, dt = s\,(a_t, z_t, K_t, G_t)\, dt$$

- Optimal choice: $c_t = c\,(a_t, z_t, K_t, G_t)$.

- Total consumption by households:

$$C_t \equiv \int c\,(a_t, z_t, K_t, G_t)\, dG_t\,(a, z)$$

- $G_t\,(a, z)$ has a Radon-Nikodym derivative $g_t(a, z)$ that follows the KF equation:

$$\frac{\partial g_{it}}{\partial t} = -\frac{\partial}{\partial a}\left(s\,(a_t, z_t, K_t, G_t)\, g_{it}(a)\right) - \lambda_i g_{it}(a) + \lambda_j g_{jt}(a), \;\; i \neq j = 1, 2$$

  where $g_{it}(a) \equiv g_t(a, z_i)$, $i = 1, 2$.

Market clearing

1. Total amount of labor rented by the firm is equal to labor supplied:

$$L_t = \int z dG_t = 1$$

   Then, total payments to labor are given by w_t.

2. Total amount of debt of the expert equals the total households' savings:

$$B_t \equiv \int a dG_t (da, dz)$$

   with $dB_t = (w_t + r_t B_t - C_t) dt$.

3. By the resource constraint, $dK_t = \left(Y_t - \delta K_t - C_t - \widehat{C}_t\right) dt + \sigma K_t dZ_t$, we get:

$$\iota_t = \frac{Y_t - C_t - \widehat{C}_t}{K_t}$$

Equilibrium

An equilibrium in this economy is composed by a set of prices $\left\{w_t, rc_t, r_t, r_t^k\right\}_{t\geq 0}$, quantities $\left\{K_t, N_t, B_t, \widehat{C}_t, c_{mt}\right\}_{t\geq 0}$, and a density $\left\{g_t\left(\cdot\right)\right\}_{t\geq 0}$ such that:

1. Given w_t, r_t, and g_t, the solution of the household m's problem is $c_t = c\left(a_t, z_t, K_t, G_t\right)$.
2. Given r_t^k, r_t, and N_t, the solution of the expert's problem is $\widehat{C}_t$, K_t, and B_t.
3. Given K_t, firms maximize their profits and input prices are given by w_t and rc_t.
4. Given w_t, r_t, and c_t, g_t is the solution of the KF equation.
5. Given g_t and B_t, the debt market clears.

Characterizing the equilibrium I

- First, we proceed with the expert's problem. Because of log-utility:

$$\widehat{C}_t = \widehat{\rho} N_t$$

$$\omega_t = \widehat{\omega}_t = \frac{rc_t - \delta - r_t}{\sigma^2}$$

- We can use the equilibrium values of rc_t, L_t, and ω_t to get the wage:

$$w_t = (1 - \alpha) K_t^{\alpha}$$

the rental rate of capital:

$$rc_t = \alpha K_t^{\alpha - 1}$$

and the risk-free interest rate:

$$r_t = \alpha K_t^{\alpha - 1} - \delta - \sigma^2 \frac{K_t}{N_t}$$

Characterizing the equilibrium II

- Expert's net wealth evolves as:

$$dN_t = \underbrace{\left(\alpha K_t^{\alpha-1} - \delta - \widehat{\rho} - \sigma^2\left(1 - \frac{K_t}{N_t}\right)\frac{K_t}{N_t}\right) N_t dt}_{\mu_t^N(B_t,N_t)} + \underbrace{\sigma K_t}_{\sigma_t^N(B_t,N_t)} dZ_t$$

- And debt as:

$$dB_t = \left((1-\alpha)\, K_t^{\alpha} + \left(\alpha K_t^{\alpha-1} - \delta - \sigma^2 \frac{K_t}{N_t}\right) B_t - C_t\right) dt$$

- Nonlinear structure of law of motion for dN_t and dB_t.

- We need to find:

$$C_t \equiv \int c\left(a_t, z_t, K_t, G_t\right) g_t\left(a, z\right) dadz$$

$$\frac{\partial g_{it}}{\partial t} = -\frac{\partial}{\partial a}\left(s\left(a_t, z_t, K_t, G_t\right) g_{it}(a)\right) - \lambda_i g_{it}(a) + \lambda_j g_{jt}(a), \;\; i \neq j = 1, 2$$

The DSS

- No aggregate shocks ($\sigma = 0$), but we still have idiosyncratic household shocks.
- Then:

$$r \quad = \quad r_t^k = rc_t - \delta = \alpha K_t^{\alpha-1} - \delta$$

and

$$dN_t = \left(\alpha K_t^{\alpha-1} - \delta - \widehat{\rho}\right) N_t dt$$

- Since in a steady state the drift of expert's wealth must be zero, we get:

$$K = \left(\frac{\widehat{\rho} + \delta}{\alpha}\right)^{\frac{1}{\alpha-1}}$$

and:

$$r = \widehat{\rho} < \rho$$

- The value of N is given by the dispersion of the idiosyncratic shocks (no analytic expression).

How do we find aggregate consumption?

- As in Krusell and Smith (1998), households only track a finite set of n moments of $g_t(a, z)$ to form their expectations.
- No exogenous state variable (shocks to capital encoded in K). Instead, two endogenous states.
- For ease of exposition, we set $n = 1$. The solution can be trivially extended to the case with $n > 1$.
- More concretely, households consider a *perceived law of motion* (PLM) of aggregate debt:

$$dB_t = h(B_t, N_t)\, dt$$

where

$$h(B_t, N_t) = \frac{\mathbb{E}[dB_t | B_t, N_t]}{dt}$$

A new HJB equation

- Given the PLM, the household's Hamilton-Jacobi-Bellman (HJB) equation becomes:

$$\begin{aligned}\rho V_i(a,B,N) &= \max_c \frac{c^{1-\gamma}-1}{1-\gamma} + s\frac{\partial V_i}{\partial a} + \lambda_i\left[V_j(a,B,N) - V_i(a,B,N)\right] \\ &\quad + h(B,N)\frac{\partial V_i}{\partial B} + \mu^N(B,N)\frac{\partial V_i}{\partial N} + \frac{\left[\sigma^N(B,N)\right]^2}{2}\frac{\partial^2 V_i}{\partial N^2}\end{aligned}$$

  $i \neq j = 1,2$, and where

$$s = s(a,z,N+B,G)$$

- We solve the HJB with a first-order, implicit upwind scheme in a finite difference stencil.
- Sparse system. Why?
- Alternatives for solving the HJB? Meshfree, FEM, deep learning, ...

An algorithm to find the PLM

1) Start with h_0, an initial guess for h.

2) Using current guess h_n, solve for the household consumption, c_m, in the HJB equation.

3) Construct a time series for B_t by simulating by J periods the cross-sectional distribution of households with a constant time step Δt (starting at DSS and with a burn-in).

4) Given B_t, find N_t, K_t, and:

$$\widehat{\mathbf{h}} = \left\{ \widehat{h}_1, \widehat{h}_2 ..., \widehat{h}_j \equiv \frac{B_{t_j+\Delta t} - B_{t_j}}{\Delta t}, ..., \widehat{h}_J \right\}$$

5) Define $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_J\}$, where $\mathbf{x}_j = \left\{x_j^1, x_j^2\right\} = \left\{B_{t_j}, N_{t_j}\right\}$.

6) Use $\left(\widehat{\mathbf{h}}, \mathbf{X}\right)$ and a universal nonlinear approximator to obtain h_{n+1}, a new guess for h.

7) Iterate steps 2)-6) until h_{n+1} is sufficiently close to h_n.

A universal nonlinear approximator

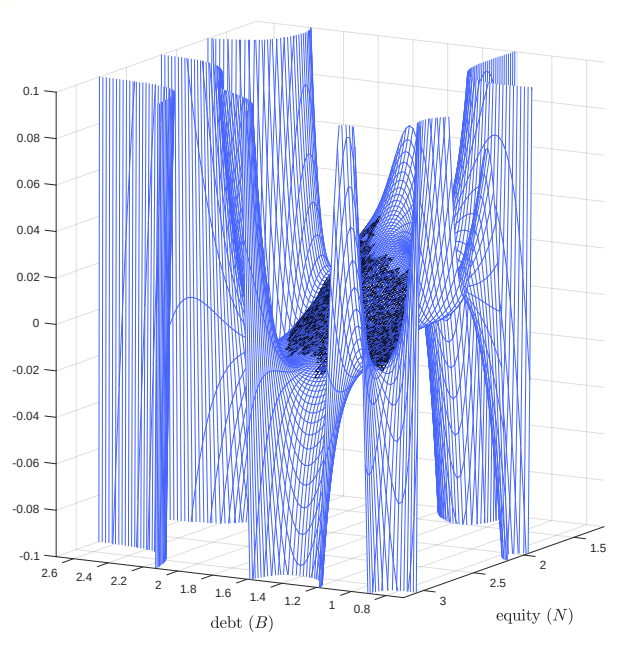
- We approximate the PLM with a neural network (NN):

$$h(\mathbf{x};\theta) = \theta_0^1 + \sum_{q=1}^{Q} \theta_q^1 \phi\left(\theta_{0,q}^2 + \sum_{i=1}^{D} \theta_{i,q}^2 x^i\right)$$

  where $Q = 16$, $D = 2$, and $\phi(z) = \log(1 + e^z)$.

- θ is selected as:

$$\theta^* = \arg\min_{\theta} \frac{1}{2} \sum_{j=1}^{J} \left\| h(\mathbf{x}_j;\theta) - \widehat{h}_j \right\|^2$$


0.1
0.08
0.06
0.04
0.02
0
-0.02
-0.04
-0.06
-0.08
-0.1
2.6
2.4
2.2
2
1.8
1.6
1.4
1.2
1
0.8
debt (B)
3
2.5
2
1.5
equity (N)

Estimation with aggregate variables I

- $D+1$ observations of Y_t at fixed time intervals $[0, \Delta, 2\Delta, .., D\Delta]$:

$$Y_0^D = \{Y_0, Y_\Delta, Y_{2\Delta}, ..., Y_D\}.$$

- More general case: sequential Monte Carlo approximation to the Kushner-Stratonovich equation (Fernández-Villaverde and Rubio Ramírez, 2007).
- We are interested in estimating a vector of structural parameters Ψ.
- Likelihood:

$$\mathcal{L}_D\left(Y_0^D|\Psi\right) = \prod_{d=1}^{D} p_Y\left(Y_{d\Delta}|Y_{(d-1)\Delta};\Psi\right),$$

where

$$p_Y\left(Y_{d\Delta}|Y_{(d-1)\Delta};\Psi\right) = \int f_{d\Delta}(Y_{d\Delta}, B)dB.$$

given a density, $f_{d\Delta}(Y_{d\Delta}, B)$, implied by the solution of the model.

Estimation with aggregate variables II

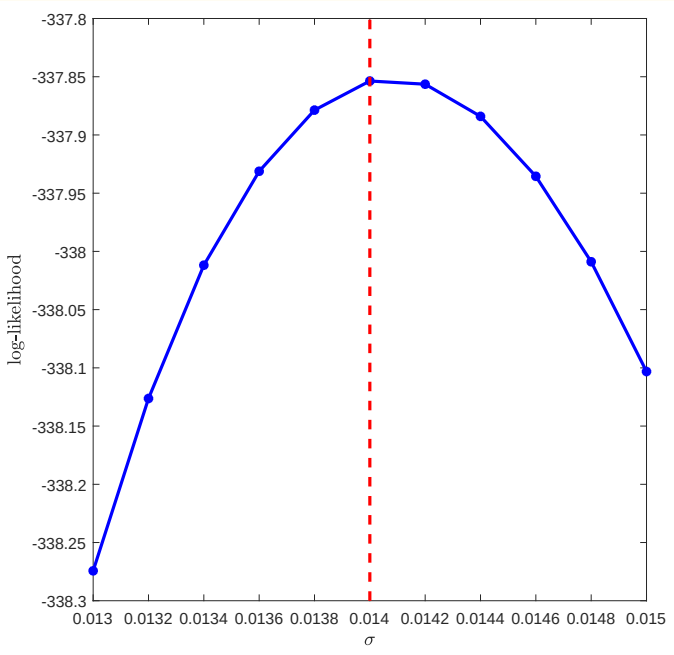
- After finding the diffusion for Y_t, $f_t^d(Y,B)$ follows the Kolmogorov forward (KF) equation in the interval $[(d-1)\Delta, d\Delta]$:

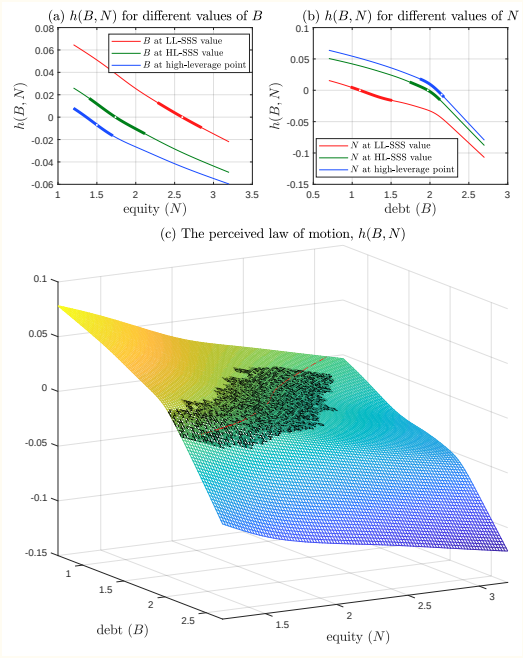
$$\begin{aligned}\frac{\partial f_t}{\partial t} &= -\frac{\partial}{\partial Y}\left[\mu^Y(Y,B)f_t(Y,B)\right] - \frac{\partial}{\partial B}\left[h(B, Y^{\frac{1}{\alpha}} - B)f_t^d(Y,B)\right] \\ &\quad + \frac{1}{2}\frac{\partial^2}{\partial Y^2}\left[\left(\sigma^Y(Y)\right)^2 f_t(Y,B)\right]\end{aligned}$$

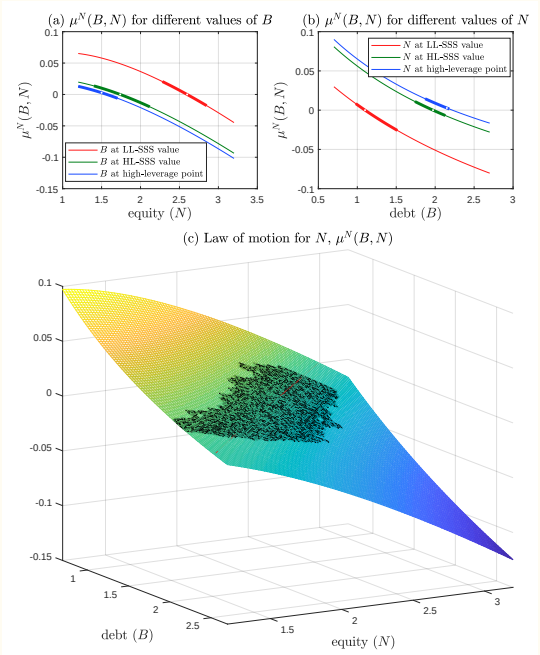
- The operator in the KF equation is the adjoint of the infinitesimal generator of the HJB.
- Thus, the solution of the KF equation amounts to transposing and inverting a sparse matrix that has already been computed.
- Our approach provides a highly efficient way of evaluating the likelihood once the model is solved.
- Conveniently, retraining of the neural network is easy for new parameter values.

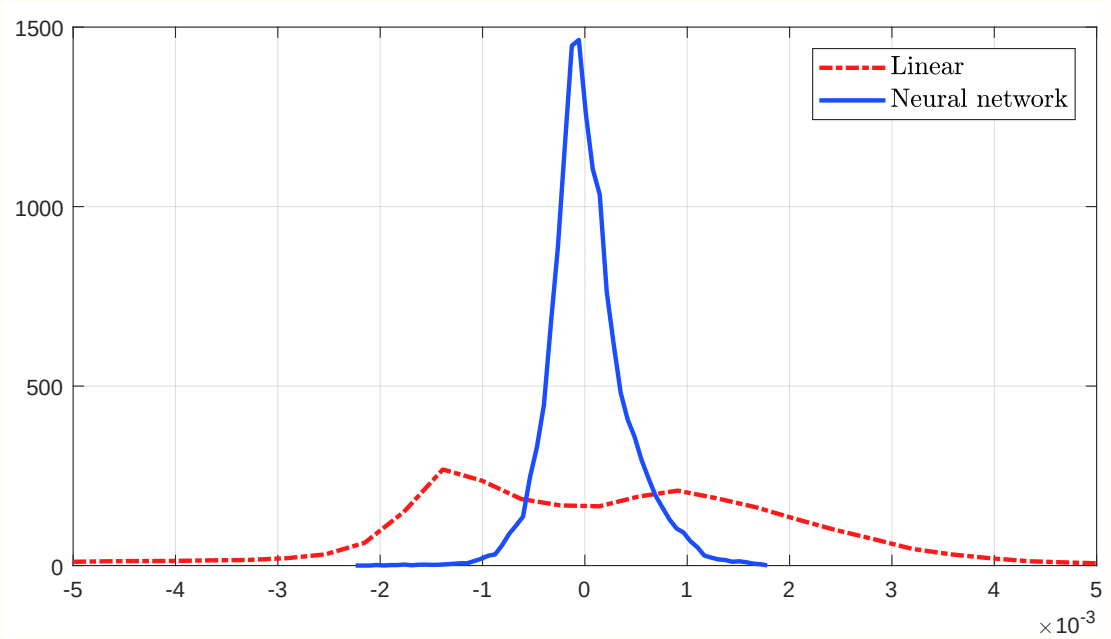
Parametrization

Parameter	Value	Description	Source/Target
α	0.35	capital share	standard
δ	0.1	yearly capital depreciation	standard
γ	2	risk aversion	standard
ρ	0.05	households' discount rate	standard
λ_1	0.986	transition rate u.-to-e.	monthly job finding rate of 0.3
λ_2	0.052	transition rate e.-to-u.	unemployment rate 5 percent
y_1	0.72	income in unemployment state	Hall and Milgrom (2008)
y_2	1.015	income in employment state	$\mathbb{E}(y) = 1$
$\widehat{\rho}$	0.0497	experts' discount rate	$K/N = 2$


-337.8
-337.85
-337.9
-337.95
-338
-338.05
-338.1
-338.15
-338.2
-338.25
-338.3
log-likelihood
0.013
0.0132
0.0134
0.0136
0.0138
0.014
0.0142
0.0144
0.0146
0.0148
0.015
σ


(a) $h(B,N)$ for different values of B
B at LL-SSS value
B at HL-SSS value
B at high-leverage point
$h(B,N)$
equity (N)
(b) $h(B,N)$ for different values of N
N at LL-SSS value
N at HL-SSS value
N at high-leverage point
$h(B,N)$
debt (B)
(c) The perceived law of motion, $h(B,N)$
debt (B)
equity (N)


(a) $\mu^N(B,N)$ for different values of B
$\mu^N(B,N)$
equity (N)
B at LL-SSS value
B at HL-SSS value
B at high-leverage point
(b) $\mu^N(B,N)$ for different values of N
$\mu^N(B,N)$
debt (B)
N at LL-SSS value
N at HL-SSS value
N at high-leverage point
(c) Law of motion for N, $\mu^N(B,N)$
debt (B)
equity (N)


1500
1000
500
0
-5
-4
-3
-2
-1
0
1
2
3
4
5
$\times 10^{-3}$
Linear
Neural network