Macroeconomic Models with Heterogenous Agents

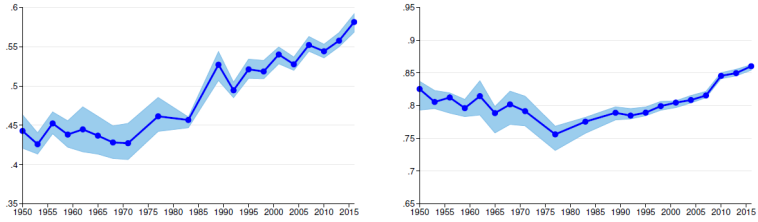
Numerical Methods, Deep-Learning Based Methods and Outlook to Data-Driven Models

Bin Cheng

February 2023

Motivation

- Heterogeneity:
  - Income and Wealth Inequality

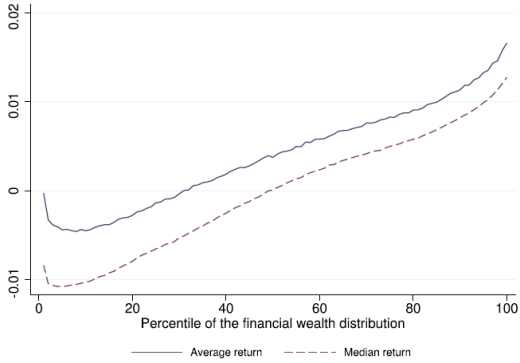


(a) Income (b) Wealth

Figure: Gini coefficients for income and wealth with confidence bands from Kuhn et al. (2020)

Motivation

- Heterogeneity:
  - The correlation between financial wealth and its return



Panel A: Return to financial wealth

Figure: Panel A shows the relation between average (solid line) and median (dashed line) return to financial wealth and financial wealth percentiles pooling data for 2005–2015 from Fagereng et al. (2020)

Motivation

- Heterogeneity:
  - Representative Agents Models Vs Heterogeneous Agents Models
  - Dimensions: Wealth, Locations, Occupations
  - Advantage: Micro Data and Macro Models!

Caveat: Missing Intercept Problem (Skip)

- Arise when considering the case from cross-section to aggregates: Autor-Dorn-Hanson (2013) "import competition explains one-quarter of the contemporaneous aggregate decline in US manufacturing employment"
- What's the effect of X_t on Y_t?:
  - x_{it} : government spending (G) in region i in year t
  - y_{it} : GDP in region i in year t
  - $X_t = \frac{1}{N}\sum_{i=1}^{N} x_{it}$: aggregate government spending
  - $Y_t = \frac{1}{N}\sum_{i=1}^{N} y_{it}$: aggregate GDP
- Assume GDP in region i satisfies

$$y_{it} = \alpha + \beta x_{it} + \gamma X_t + \varepsilon_{it}$$

- True aggregate relation :

$$Y_t = \alpha + (\beta + \gamma) X_t$$

- Cross Variation+Time-fixed Effect $\Rightarrow$ No x-sectional variation in aggregate X_t soaked into intercept (import) $\Rightarrow$ yields $\Delta Y_t = \beta \times \Delta X_t$ not $\Delta Y_t = (\beta + \gamma) \times \Delta X_t$

$$y_{it} = \tilde{\alpha}_t + \beta x_{it} + \varepsilon_{it}, \quad \tilde{\alpha}_t := \alpha + \gamma X_t$$

Outline

1. Heterogeneous-agent models:
   - Discrete-time (Bellman equation)
   - Continuous-time (HJB equations)
   - Present vs Future
2. Numerical solution of HJB equations
   - Stationary HJB Formulation in Moll et al. (2021)
   - Finite Difference Approximation
   - Upwind Scheme
3. Neural Network Based Solutions of HJB equations
   - Overview
   - PINNs and PDE
   - PINNs and HJB
   - Freedom of PINNs
4. Outlook to Data-Driven Models
   - Inverse Problem

1. HA Models: Discrete-time Bellman equation (Skip)

- A canonical formulation of deterministic optimal control problem:

$$V(x_0) = \max_{\{\alpha_t\}_{t=0}^{\infty}} \sum_{t=0}^{\infty} \beta^t r(x_t, \alpha_t)$$

subject to

$$x_{t+1} = g(x_t, \alpha_t) \text{ and } \alpha_t \in A.$$

- Components:
  - β : discount factor.
  - $x \in \mathcal{X}$: state variable (vector)
  - $\alpha \in \mathcal{A}$: control variable (vector)
  - $r : \mathcal{X} \times \mathcal{A} \to \mathcal{R}$: reward function

1. HA Models: Discrete-time Bellman equation (Skip)

- Bellman equation:

$$V(x) = \max_{\alpha} \{r(x, \alpha) + \beta V(x') \quad \text{s.t.} \quad x' = g(x, \alpha)\}$$

- Heuristic derivation- A recursive notation: Consider an optimal strategy $\{\alpha_t^*\}_{t=0}^{\infty}$

$$\begin{aligned} V(x_0) &= \sum_{t=0}^{\infty} \beta^t r(x_t, \alpha_t^*) \\ &= r(x_0, \alpha_0^*) + \sum_{t=1}^{\infty} \beta^t r(x_t, \alpha_t^*) \\ &= r(x_0, \alpha_0^*) + \beta \sum_{t=0}^{\infty} \beta^t r(x_{t+1}, \alpha_{t+1}^*) \\ &= r(x_0, \alpha_0^*) + \beta V(x_1) \end{aligned}$$

- Components:
  - x' : tomorrow's state

1. HA Models: Hamilton-Jacobi-Bellman Equations (Skip)

- A canonical formulation of deterministic optimal control problem:

$$v(x_0) = \max_{\{\alpha(t)\}_{t\geq 0}} \int_0^\infty e^{-\rho t} r(x(t), \alpha(t)) dt$$

subject to

$$\dot{x}(t) = f(x(t), \alpha(t)) \quad \text{and} \quad \alpha(t) \in A$$

- Components:
  - ρ : discount factor.
  - $x \in \mathcal{X}$: state variable (vector)
  - $\alpha \in \mathcal{A}$: control variable (vector)
  - $r : \mathcal{X} \times \mathcal{A} \to \mathcal{R}$: reward function

1. HA Models: Hamilton-Jacobi-Bellman Equations (Skip)

- HJB equation:

$$\rho v(x) = \max_{\alpha \in A} r(x, \alpha) + v'(x) \cdot f(x, \alpha)$$

- Heuristic derivation- A continuous-time generalization of the Bellman equation:

$$v\left(x_t\right) = \max_{\alpha_t} \Delta u\left(\alpha_t\right) + e^{-\rho \Delta} v\left(x_{t+\Delta}\right)$$

Take $e^{-\rho \Delta} = 1 - \rho \Delta, \Delta \to 0$ and obtain

$$v\left(x_t\right) = \max_{\alpha_t} \Delta u\left(\alpha_t\right) + (1 - \rho \Delta) v\left(x_{t+\Delta}\right)$$

$\Rightarrow$

$$\rho v\left(x_t\right) = \max_{\alpha_t} u\left(\alpha_t\right) + (1 - \Delta \rho) \frac{v\left(x_{t+\Delta}\right) - v\left(x_t\right)}{x_{t+\Delta} - x_t} \frac{x_{t+\Delta} - x_t}{\Delta}$$

$\Rightarrow$

$$\rho v\left(x_t\right) = \max_{ct} u\left(\alpha_t\right) + v'\left(x_t\right) \dot{x}_t$$

1. HA Models: Present vs Tomorrow (Skip)

- Neoclassical Growth Model in continuous time:

$$v(k_0) = \max_{\{c(t)\}_{t\geq 0}} \int_0^\infty e^{-\rho t} u(c(t)) dt$$

subject to

$$\dot{k}(t) = F(k(t)) - \delta k(t) - c(t)$$

yields

$$\rho v(k) = \max_c u(c) + v'(k)(F(k) - \delta k - c)$$

- NGM in discrete time:

$$v(k) = \max_c u(c) + \beta v(k') \quad k' = F(k) + (1-\delta)k - c$$

- FOC in continuous time:

$$u'(c) = v'(k)$$

- FOC in discrete time:

$$u'(c) = \beta v'(k')$$

2. Numerical Solutions of HJB equations: Moll et al. (2021)

- A continuum of individuals heterogeneous in wealth a and income y.
- Standard preferences:

$$\mathbb{E}_0 \int_0^\infty e^{-\rho t} u(c_t)\, dt$$

- Wealth Evolution:

$$\dot{a}_t = y_t + r_t a_t - c_t$$

- Borrowing Constraint:

$$a_t \geq \underline{a}$$

- Income Process:
  1. a two-state Poisson process: $z_t \in \{z_1, z_2\}$, with $z_2 > z_1$.
  2. a continuous diffusion process: $dz_t = \mu(z_t)\, dt + \sigma(z_t)\, dW_t$

2. Numerical Solutions of HJB equations: Stationary HJB formulation

- Hamilton-Jacobi-Bellman equation for individual choices:

$$\rho v(a,z) = \max_c u(c) + \partial_a v(a,z)(z + ra - c) + \mu(z)\partial_z v(a,z) + \frac{\sigma^2(z)}{2}\partial_{zz} v(a,z)$$

- Kolmogorov Forward equation for evolution of distribution:

$$0 = -\partial_a[s(a,z)g(a,z)] - \partial_z[\mu(z)g(a,z)] + \frac{1}{2}\partial_{zz}\left[\sigma^2(z)g(a,z)\right]$$

- Fixed Bond Supply:

$$1 = \int_0^\infty \int_{\underline{a}}^\infty g(a,z)dadz$$

- Market Clearing:

$$0 = \int_0^\infty \int_{\underline{a}}^\infty ag(a,z)dadz =: S(r)$$

2. Numerical Solutions of HJB equations: Finite Difference Approximation

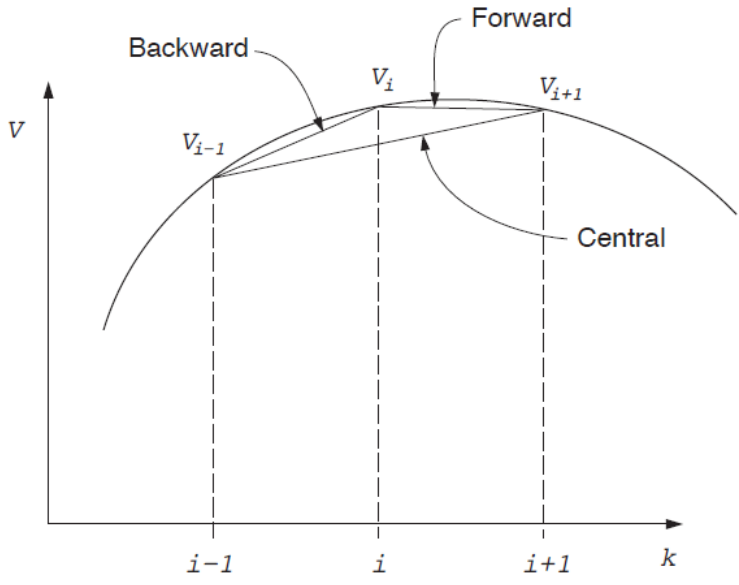
- Intuition: Differential equation $\Rightarrow$ linear system via Discretization.
- Scheme:
  - $\mathcal{X} = \{x_1, x_2, \ldots, x_I\}$ with $\Delta x = x_i - x_{i-1}$
  - $v(x) \in \{v(x_1), v(x_2), \ldots, v(x_I)\}$
  - $v'(x_i)$?

$$v'(x_i) \approx \frac{v_i - v_{i-1}}{\Delta x} \text{ backward difference}$$

$$v'(x_i) \approx \frac{v_{i+1} - v_i}{\Delta x} \text{ forward difference}$$

$$v'(x_i) \approx \frac{v_{i+1} - v_{i-1}}{2\Delta x} \text{ central difference}$$

2. Numerical Solutions of HJB equations: Graphical Illustration



2. Numerical Solutions of HJB equations: Upwind Scheme

- FD approximation to HJB:

$$\frac{v_{i,j}^{n+1} - v_{i,j}^{n}}{\Delta} + \rho v_{i,j}^{n+1} = u\left(c_{i,j}^{n}\right) + \partial_a v_{i,j}^{n+1}\left[z_j + ra_i - c_{i,j}^{n}\right] + \mu_j \partial_z v_{i,j}^{n+1} + \frac{\sigma_j^2}{2}\partial_{zz} v_{i,j}^{n+1}$$

- Discretization on a-direction:

$$\partial_{a,B} v_{i,j} = \frac{v_{i,j} - v_{i-1,j}}{\Delta a}$$
$$\partial_{a,F} v_{i,j} = \frac{v_{i+1,j} - v_{i,j}}{\Delta a}$$

- Discretization on z-direction:

$$\partial_{z,B} v_{i,j} = \frac{v_{i,j} - v_{i,j-1}}{\Delta z}$$
$$\partial_{z,F} v_{i,j} = \frac{v_{i,j+1} - v_{i,j}}{\Delta z}$$
$$\partial_{zz} v_{i,j} = \frac{v_{i,j+1} - 2v_{i,j} + v_{i,j-1}}{(\Delta z)^2}$$

2. Numerical Solutions of HJB equations: Upwind Scheme

Question :

- Which FD approximation – backward or forward – should we use?
- Best solution: "Upwind Scheme"
  - forward difference whenever drift of state variable positive
  - backward difference whenever drift of state variable negative
- FD approximation to HJB under "Upwind Scheme":

$$\frac{v_{i,j}^{n+1} - v_{i,j}^{n}}{\Delta} + \rho v_{i,j}^{n+1} = u\left(c_{i,j}^{n}\right) + \partial_{a,F} v_{i,j}^{n+1}\left[z_j + ra_i - c_{i,j,F}^{n}\right]^{+} + \partial_{a,B} v_{i,j}^{n+1}\left[z_j + ra_i - c_{i,j,B}^{n}\right]^{-}$$
$$+ \partial_{z,F} v_{i,j}^{n+1} \mu_j^{+} + \partial_{z,B} v_{i,j}^{n+1} \mu_j^{-} + \frac{\sigma_j^2}{2} \partial_{zz} v_{i,j}^{n+1}$$

reduce to: $\frac{1}{\Delta}\left(v^{n+1} - v^{n}\right) + \rho v^{n+1} = u^{n} + \mathbf{A}^{n} v^{n+1}$

2. Numerical Solutions of HJB equations: Tridiagonal Matrix A^n

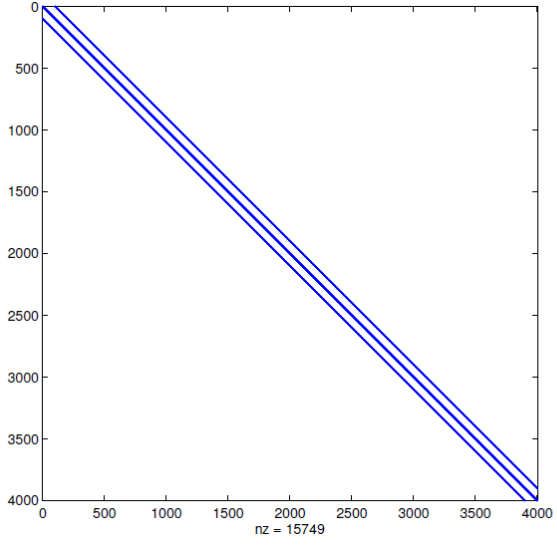


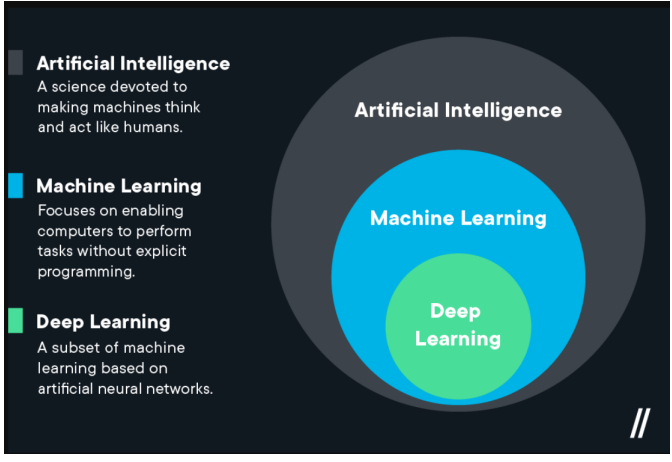
Figure: Transition Matrix

2. Numerical Solutions of HJB equations: Additional Comment

- Boundary Condition: $0 = \partial_z v(a, \underline{z}) = \partial_z v(a, \bar{z})$ and $\partial_a v(\underline{a}, z) \geq u'(z + r\underline{a}), \quad \forall z.$
- Algorithm Complexity: Simple and Hard

3. Neural Network Solutions of HJB equations: Overview

- Deep Learning: A subset of ML and more data-intensive.



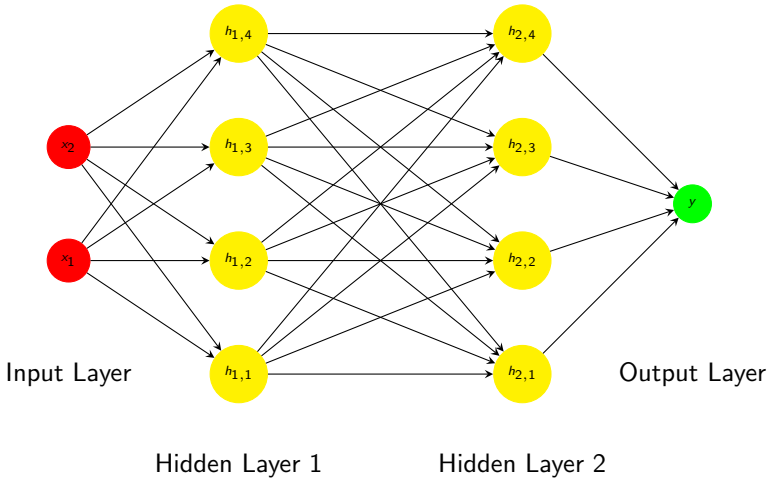
3. Neural Network Solutions of HJB equations: Overview

- AlexNet in ImageNet: Convolutional Neural Networks

Figure: Iconic Progress: Accuracy of 84.7% over the previous best result of 73.8%

3. Neural Network Solutions of HJB equations: NN Architecture[1]

- ChatGPT: Transformer-based neural network.



[1](Latex code generated by ChatGPT)

3. Neural Network Solutions of HJB equations: NN Math Formulation[2]

$$\text{Input Layer: } \mathbf{x} = [x_1, x_2]^T$$
$$\text{Hidden Layer 1: } \mathbf{h}^{(1)} = \sigma(\mathbf{W}^{(1)}\mathbf{x} + \mathbf{b}^{(1)})$$
$$\text{Hidden Layer 2: } \mathbf{h}^{(2)} = \sigma(\mathbf{W}^{(2)}\mathbf{h}^{(1)} + \mathbf{b}^{(2)})$$
$$\text{Output Layer: } y = \sigma(\mathbf{W}^{(3)}\mathbf{h}^{(2)} + b^{(3)})$$

Here, $\mathbf{x}$ is the input vector, $\mathbf{h}^{(1)}$ and $\mathbf{h}^{(2)}$ are the hidden layer activation vectors, y is the output, σ is the activation function, $\mathbf{W}^{(1)}$, $\mathbf{W}^{(2)}$, and $\mathbf{W}^{(3)}$ are the weight matrices, $\mathbf{b}^{(1)}$, $\mathbf{b}^{(2)}$, and $b^{(3)}$ are the bias vectors for each layer, and T denotes the transpose operation. The superscripts (1) and (2) denote the first and second hidden layers, respectively. The activation function σ is typically a non-linear function, such as the sigmoid or ReLU functions.

[2](Latex code generated by ChatGPT)

3. Neural Network Solutions of HJB equations: Physics-Informed Neural Networks

- A neural network is given by a compositional function,

$$u(\mathbf{x}, \Theta) = f\left(\ldots f\left(f\left(\mathbf{x}W^1\right)W^2\right)\ldots W^L\right)$$

  where $\mathbf{x}$ is the input, $\Theta = \left\{W^1, \ldots, W^L\right\}$ is the set of model parameters, $f(\cdot)$ is the non-linear activation function and L is the number of layers.

- The optimization problem is given by

$$\min_{\Theta} L(\Theta),$$

  where the loss $L(\Theta)$ is some function of $u(x, \Theta)$.

3. Neural Network Solutions of HJB equations: PINNs and PDE

- PINNs Defitions:
  1. Respecting physical laws [3]
  2. Solve supervised learning tasks

[3] Newton's laws of motion, Maxwell's laws of electromagnetism , PDEs

3. Neural Network Solutions of HJB equations: PINNs and PDE

- General PDEs:

$$\begin{aligned} \mathcal{F}(u(x);\gamma) &= f(x) \quad && x \in \Omega \\ \mathcal{B}(u(x)) &= g(x) \quad && x \in \partial\Omega \\ \mathcal{D}\left(u\left(x_i\right)\right) &= d\left(x_i\right) \quad && i \in D \end{aligned}$$

where

- $\mathcal{F}(\cdot)$ is a differential operator on the domain[4].
- $\mathcal{B}(\cdot)$ is a differential operator on the boundary of the domain.
- $\mathcal{D}$ is a differential operator on dataset[5].

[4] Here γ represents unknow or known parameters, as in sphere of forward and inverse problem

[5] Prior knowledge

3. Neural Network Solutions of HJB equations: PINNs and PDE

- Supervised Approach:
  1. Generate a dataset of solutions, i.e. dataset consisting of $x, u(x)$ based on model parameter Θ.
  2. Train NN $u_\theta(x)$ to learn mapping $x :\rightarrow u(x)$[6].
  3. Learning Objective:

$$\min_\theta \mathcal{L}(u_\theta(x), u(x))$$

where total loss function $\mathcal{L}$ is a weighted sum of three sub-loss function $\mathcal{L}_\mathcal{F}, \mathcal{L}_\mathcal{B}.\mathcal{L}_\mathcal{D}$ associated with three PDE equations as:

$$\begin{aligned}\mathcal{L} := \omega_\mathcal{F}\mathcal{L}_\mathcal{F} + \omega_\mathcal{B}\mathcal{L}_\mathcal{B} + \omega_\mathcal{D}\mathcal{L}_D &= \omega_\mathcal{F}\frac{1}{N_\mathcal{F}}\sum_{i=1}^{N_\mathcal{F}}|\mathcal{F}(u_\theta(x_i);\gamma) - f(x_i)|^2 \\ &+ \omega_\mathcal{B}\frac{1}{N_\mathcal{B}}\sum_{i=1}^{N_\mathcal{B}}|\mathcal{B}(u_\theta(x_i);\gamma) - g(x_i)|^2 \\ &+ \omega_\mathcal{D}\frac{1}{N_\mathcal{D}}\sum_{i=1}^{N_\mathcal{D}}|\mathcal{D}(u_\theta(x_i);\gamma) - d(x_i)|^2\end{aligned}$$

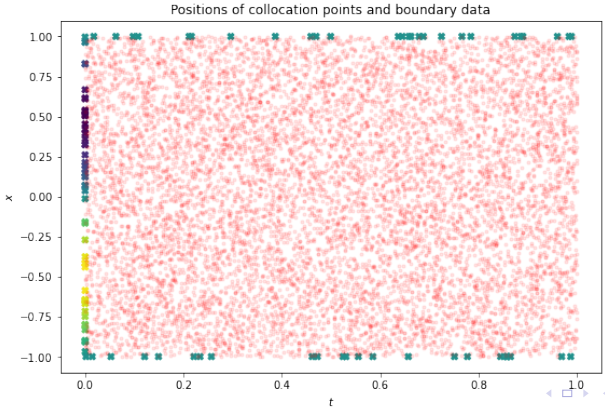
[6]Key: trained NN generalize to new inputs x

3. Neural Network Solutions of HJB equations: PINNs and PDE

- Burgers equation:

$$\begin{aligned} \partial_t u + u\,\partial_x u - (0.01/\pi)\,\partial_{xx} u = 0, \qquad & (t,x) \in (0,1] \times (-1,1), \\ u(0,x) = -\sin(\pi x), \qquad & x \in [-1,1], \\ u(t,-1) = u(t,1) = 0, \qquad & t \in (0,1]. \end{aligned}$$

- Comment: Equations using only $\mathcal{F}$ on Ω and $\mathcal{B}$ on $\partial\Omega$.

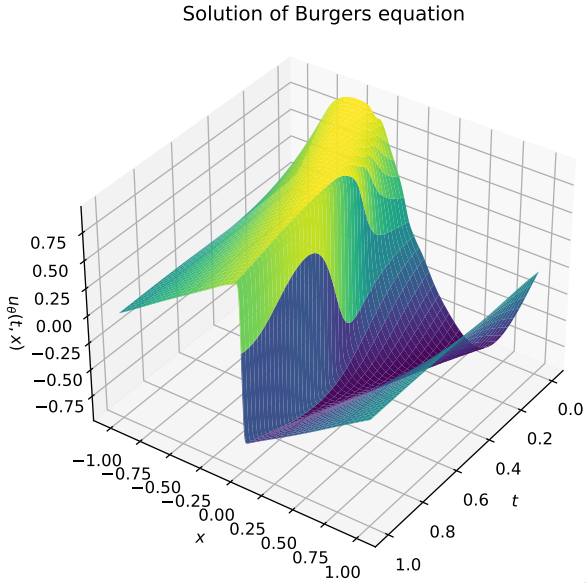


3. Neural Network Solutions of HJB equations: PINNs and PDE

- A feedforward neural network (FNN):
  1. the input is scaled elementwise to lie in the interval $[-1, 1]$ for $\{t_i, x_i\}_{i=1}^N$,
  2. followed by 8 fully connected layers each containing 20 neurons and each followed by a hyperbolic tangent activation function,
  3. one fully connected output layer for $u(\cdot)$.
  4. total trainable parameters are 3021[7].
- Auto-differentiation: Use Tensorflow.GradientTape to compute $\partial_t u_\theta$, $\partial_x u_\theta$ and $\partial_{xx} u_\theta$

[7] Time: 1min

3. Neural Network Solutions of HJB equations: PINNs and PDE



3. Neural Network Solutions of HJB equations: PINNs and HJB

- Recall HJB in Heterogenous Agent Model:

$$\rho v(a,z) = \max_c u(c) + \partial_a v(a,z)(z + ra - c) + \mu(z)\partial_z v(a,z) + \frac{\sigma^2(z)}{2}\partial_{zz} v(a,z)$$

- Reformulated as:

$$-\rho v + H\left(\partial_a v\right) + (z + ra)\partial_a v + \mu(z)\partial_z v + \frac{\sigma^2(z)}{2}\partial_{zz} v = 0$$

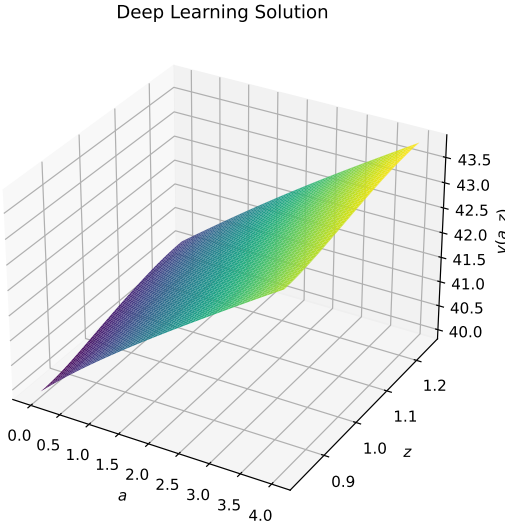
  where the Hamiltonian H is given by

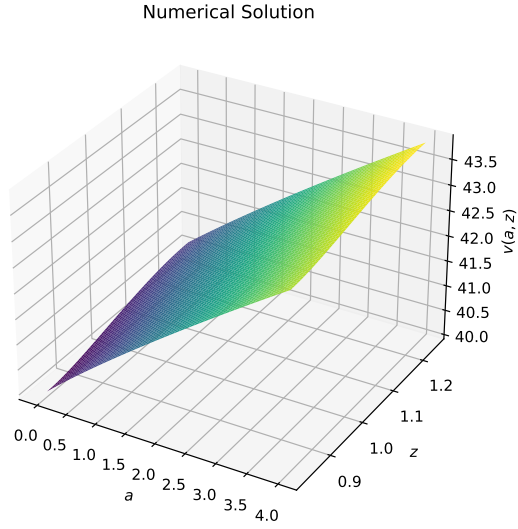
$$H(p) = \max_{c \geq 0}(-pc + u(c)).$$

- Hamiltonian H obtain optima at

$$c = u'^{-1}(\partial_a v)$$

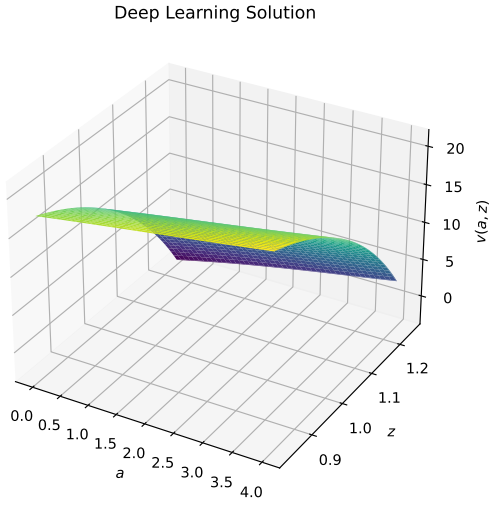
3. Neural Network Solutions of HJB equations: PINNs and HJB

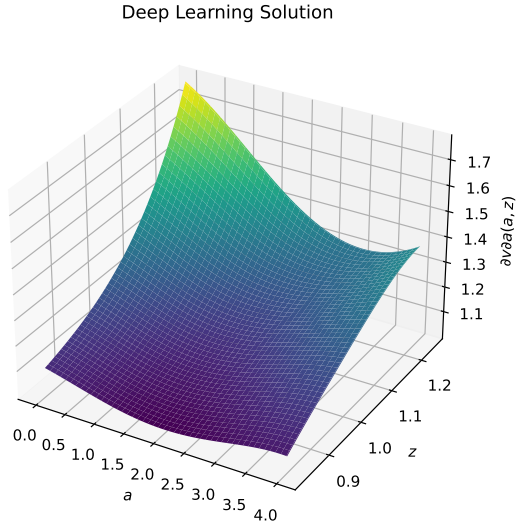




3. Neural Network Solutions of HJB equations: Freedom of PINNs

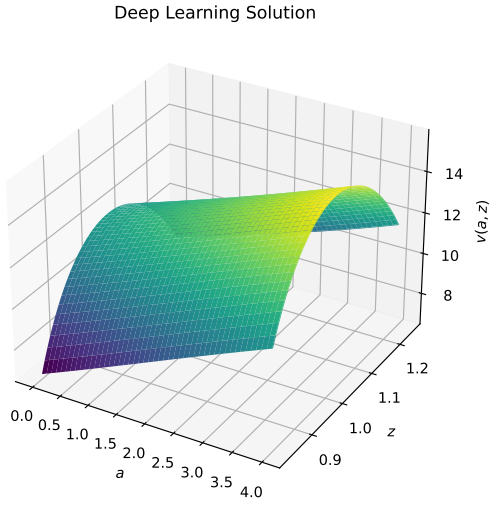
- Remove $0 = \partial_z v(a, \underline{z}) = \partial_z v(a, \bar{z})$

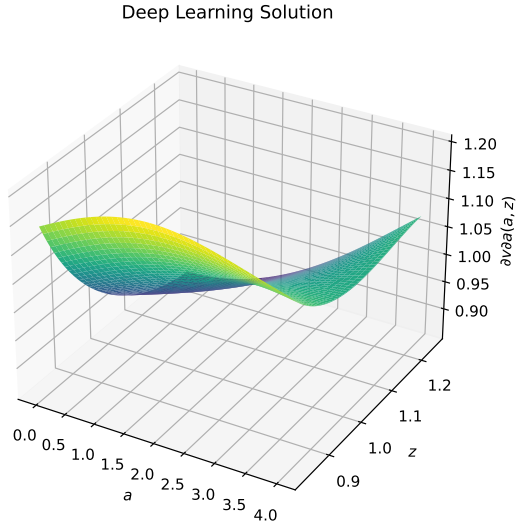




3. Neural Network Solutions of HJB equations: Freedom of PINNs

- Remove $0 = \partial_z v(a, \underline{z}) = \partial_z v(a, \bar{z})$ and $\partial_a v(\underline{a}, z) \geq u'(z + r\underline{a})$





4. Outlook to Data-Driven Models: Inverse Problem

- PDE with unknown parameters λ, parabolic Eikonal equation:

$$-\partial_t u + \sup_{|c|\leq 1} c \cdot \nabla u = -\partial_t u + |\nabla u| = \lambda^{-1}$$
$$u(T, x) = 0$$
$$u(t, -1) = u(t, 1) = 0$$

  with unknown parameter λ.
- Explicit solution: $u^*(t, x) = \lambda^{-1} \min\{1 - t, 1 - |x|\}$.

4. Outlook to Data-Driven Models: Inverse Problem

- Feed NN with data generated by explicit solution and start training of $\hat{\lambda}, u_\theta$ to approximate λ and u at the same time.
- Example:
  1. set $\lambda^* = 3$ and generate $u_{data}(t,x) = \lambda^{*-1} \min\{1-t, 1-|x|\}$
  2. Recall $\mathcal{D}$ operator, representing prior knowledge.

$$\mathcal{D}\left(u\left(x_i\right)\right) = d\left(x_i\right) i \in D$$

  3. Define loss function based on u_{data} as

$$\omega_{\mathcal{D}} \frac{1}{N_{\mathcal{D}}} \sum_{i=1}^{N_{\mathcal{D}}} \left|\mathcal{D}\left(u_\theta\left(x_i\right); \lambda\right) - u_{data}\left(x_i\right)\right|^2$$

where $\mathcal{D}$ is an identity operator and yield

$$\omega_{\mathcal{D}} \frac{1}{N_{\mathcal{D}}} \sum_{i=1}^{N_{\mathcal{D}}} \left|u_\theta\left(x_i; \lambda\right) - u_{data}\left(x_i\right)\right|^2$$

.

4. Outlook to Data-Driven Models: Inverse Problem