

# Deep learning for solving dynamic economic models

Lilia Maliar, Serguei Maliar, Pablo Winant

2024 年 6 月 4 日

汇报人：张培元

# Traditional Methods

- Traditional Numerical Methods:
  - Value-based Iteration
  - Policy-based Iteration
- Optimizations of Traditional Methods:
  - Endogenous Grid Method
  - Perturbation Method
  - High-Performance Computing
- Existing Problem:
  - The Curse of Dimensionality

# Why use AI Method?

- Artificial intelligence (AI) has remarkable applications (recognition of images and speech, facilitation of computer vision, operation of self-driving cars)
- AI is good at dealing with optimization problems (most HAM models are also dealing with it)
- There is the possibility of isomorphism between them.

There are mainly three applications:

- Solving discrete-time models (Methodology)
  - Utilizing deep learning combined with a large amount of simulated data for iterative work
  - Reference:
    - HAM - Maliar et al. (JME, 2021)
    - OLG - Azinovic et al. (IER, 2022)
    - Discrete choices - Maliar and Maliar (2022, JEDC)
- Solving continuous-time models (Mathematics)
  - HAM-DP to MFG-sys, and solving the BFSDEs.
    - $\text{BFSDEs} = \text{HJB} + \text{KF}(\text{KP})$
    - Mean Field Games (2006, Jean-Michel Lasry and Pierre-Louis Lions; Huang-Malhamé-Caines)
    - Income and Wealth Distribution in Macroeconomics: A Continuous-Time Approach (2020, RES, Yves Achdou, Jiequn Han, Jean-Michel Lasry, Pierre-Louis-Lions, Benjamin Moll)
  - Reference:
    - DeepBSDE - HAN et al. (PNAS, 2018) to DeepHAM
    - Jesus Fernandez-Villaverde
    - Jonathan Payne: ME, DeepSAM
- Performing estimation or calibration
  - Calibration, Estimation.

- AI is a device that:
  - has a goal
  - perceives its environment
  - takes actions and achieve its goal (in general)
- This looks exactly like an economic agent!
  - goal = max (utility)
  - environment = state variables
  - action = controls/policies
- Main contents:
  - A class of dynamic Markov economic models with time-invariant decision functions
  - Maximize Utility; Minimize Euler/Bellman residual
  - Take a toy model and Krusell Smith as example

## Theory Part

# Base settings

- An basic HAM Model - Settings

- An exogenous shock  $m_{t+1}$ , follows a Markov process and by an i.i.d process  $\epsilon_t$  with a transition function M:

$$m_{t+1} = M(m_t, \epsilon_t)$$

- An endogenous state  $s_{t+1}$  is driven by the shock  $m_t$  and policy function  $x_t$  with a transition function S

$$s_{t+1} = S(m_t, s_t, x_t, m_{t+1})$$

- The policy function satisfies the constraint(X), in the form

$$x_t \in X(m_t, s_t)$$

- The state and the policy determine the period utility  $r(m_t, s_t, x_t)$ . And the agent maximizes discounted lifetime utility

$$\max_{\{x_t, s_{t+1}\}_{t=0}^{\infty}} E_0 \left[ \sum_{t=0}^{\infty} \beta^t r(m_t, s_t, x_t) \right]$$

- $\beta \in [0, 1)$  is the discount factor and  $E_0$  is an expectation function across future shocks conditional on the initial state.

# The goal - decision rules

- i) Definition

- An optimal decision rule is a policy function  $\psi : R^{nm} \times R^{ns} \rightarrow R^{nx}$  such that

$$x_t = \psi(m_t, s_t) \in X(m_t, s_t)$$

- for all  $t$  and the sequence  $\{s_{t+1}, x_t\}_{t=0}^{\infty}$
- Maximizes the life time utility for any initial condition

- 2) Extensiveness

- A parametric decision rule is a member of a family of functions  $\psi(\cdot; \theta)$
- $\theta$  is a real parameter vector which condition by  $\theta \in \Theta$
- A different  $\theta$  means a different decision rule



## Method 1 - Maximize lifetime utility

- Value function

$$V(m_0, s_0) = \max_{\{x_t, s_{t+1}\}_0^\infty} E_{\{\epsilon_1, \dots, \epsilon_T\}} \left[ \sum_0^T \beta^t r(m_t, s_t, x_t) \right]$$

- Maximize lifetime utility

- Replacing the infinite-horizon problem with a finite-horizon problem  $T < \infty$
- Simulate time series solution forward under a fixed decision rule  $\psi(\cdot; \theta)$  and a future shocks  $(\epsilon_1, \epsilon_2, \dots, \epsilon_T)$ , then evaluate the lifetime reward:

$$V^T(m_0, s_0; \theta) = E_{\{\epsilon_1, \dots, \epsilon_T\}} \left[ \sum_0^T \beta^t r(m_t, s_t, \psi(m_t, s_t; \theta)) \right]$$

- Different initial points  $(m_0, s_0)$  may have different solution
- Then randomly drawn  $(m_0, s_0)$  from the range, so the maximization objective:

$$\Xi(\theta) = E_{(m_0, s_0)} \{ E_{\{\epsilon_1, \dots, \epsilon_T\}} \left[ \sum_0^T \beta^t r(m_t, s_t, \psi(m_t, s_t; \theta)) \right] \}$$

- The goal is to search a decision rule  $\psi(\cdot; \theta)$  that solves  $\max_{\theta \in \Theta} \Xi(\theta)$ .

## Method 2 - Minimize Euler-residual

- Euler equation

- Euler equations are a set of equations written in the form:

$$E_{\epsilon}[f_j(m, s, x, m', s', x')] = 0, j = 1, \dots, J$$

- With transition functions  $s' = S(m, s, x, m')$ ;  $m' = M(m, \epsilon)$ ;  $x \in X(m, s)$
- If there is a decision rule  $\psi(\cdot; \theta)$  which makes Euler residual = 0, then the model is considered solved

- Euler-residual

- Select a decision rule  $\psi(\cdot; \theta)$  and define a distribution of state variable  $(m, s)$ , calculate the expected squared residuals in the Euler equations:

$$\Xi(\theta) = E_{(m,s)} \left[ \sum_{j=1}^J v_j (E_{\epsilon}[f_j(m, s, \psi(m, s; \theta), m', s', \psi(m', s'; \theta))])^2 \right]$$

- where  $v_j$  is a vector of weights on  $J$  optimality conditions
- The goal is to construct a decision rule  $\psi(\cdot; \theta)$  that solves  $\min_{\theta \in \Theta} \Xi(\theta)$ .

## Method 3 - Minimize Bellman-residual

- Bellman equation

$$V(m, s) = \max_{x, s'} \{r(m, s, x) + \beta E_{\epsilon}[V(m', s')]\}$$

- If there is a decision rule  $\psi(\cdot; \theta)$  which makes the left and right sides of Bellman equation be equal, then the model is considered solved.

- Bellman-residual

- Same in Euler-residual:

$$\Xi(\theta) = E_{(m, s)} [V(m, s) - \max_{x, s'} \{r(m, s, x) + \beta E_{\epsilon}[V(m', s')]\}]^2$$

- How to deal with the inside  $\max_{x, s'}(\cdot)$ ?

- Three approaches

- FOC:  $r_x(m, s, x) + \beta \{E_{\epsilon}[V_{s'}(m', s')]\} \frac{\partial s'}{\partial x} = 0$
- Envelope condition:  $V_s(m, s) = r_s(m, s, x)$
- Direct optimization:  $\max_{x, s'} \{r(m, s, x) + \beta E_{\epsilon}[V(m', s')]\}$

- Use FOC method. Why? (Easy to construct and solve)

- Bellman-residual - with FOC to solve inside max

- Parametrize a value function  $V(\cdot; \theta_1)$  and decision rule  $\psi(\cdot; \theta_2)$  and define a distribution of state variable  $(s, m)$ .
- For given  $\theta = (\theta_1, \theta_2)$ , the squared residuals in the Bellman equations are given by:

$$\Xi(\theta) = E_{(m,s)} \{ V(m, s; \theta_1) - r(m, s, x) - \beta E_{\epsilon} [V(m', s'; \theta_1)] \}^2 + \\ \nu E_{(m,s)} \{ (r_x(m, s, x) + \beta \{ E_{\epsilon} [V_{s'}(m', s'; \theta_1)] \} \frac{\partial s'}{\partial x})^2 \}$$

- where  $\nu > 0$  is a vector of exogenous relative weights of equations in the two objectives.
- The goal is to construct a decision rule  $\psi(\cdot; \theta)$  that solves  $\min_{\theta \in \Theta} \Xi(\theta)$ .

# All in once (AIO)

- Why all in once?

- More i.i.d randoms, the faster state spaces increase.
- For example, in maximize lifetime utility
  - random sequence of future shocks  $\Sigma = (\epsilon_1, \epsilon_2, \dots, \epsilon_T)$
  - random initial state  $(m_0, s_0)$
  - There is a two nested expectation:  $E_{(m,s)}[E_{\Sigma}[\cdot]]$
  - Which also in Euler and Bellman methods.
- Approximating them, one after the other, is costly, especially in high dimensional applications.  $((m_0, s_0) \times (\epsilon_1, \epsilon_2, \dots, \epsilon_T))$

- What can AIO do?

- To reduce the cost of nested integration, introduce AIO expectation operator that combines the two expectation operators into one.
- Directly use one times draw  $(m_0, s_0, \epsilon_1, \epsilon_2, \dots, \epsilon_T)$  instead of  $(m_0, s_0) \times (\epsilon_1, \epsilon_2, \dots, \epsilon_T)$
- Which only need  $n$  draws instead of  $n \times n'$

## Method 1-3 with AIO

- Maximize lifetime utility:

randomly draw  $(m_0, s_0, \epsilon_1, \epsilon_2, \dots, \epsilon_T)$

$$\Xi(\theta) = E_{\omega}[\xi(\omega; \theta)] = E_{(m_0, s_0, \epsilon_1, \dots, \epsilon_T)} \left[ \sum_0^T \beta^t r(m_t, s_t, \psi(m_t, s_t; \theta)) \right]$$

- Minimize Euler-residual:

randomly draw  $(m, s, \epsilon_1, \epsilon_2)$  and use  $E_{\epsilon_1}[f(\epsilon_1)]E_{\epsilon_2}[f(\epsilon_2)] = E_{(\epsilon_1, \epsilon_2)}[f(\epsilon_1)f(\epsilon_2)]$

$$\Xi(\theta) = E_{\omega}[\xi(\omega; \theta)] = E_{(m, s, \epsilon_1, \epsilon_2)} \left\{ \sum_{j=1}^J v_j [f_j(m, s, x, m', s', x')|_{\epsilon=\epsilon_1}] [f_j(m, s, x, m', s', x')|_{\epsilon=\epsilon_2}] \right\}$$

- Minimize Bellman-residual:

randomly draw  $(m, s, \epsilon_1, \epsilon_2)$  and use  $E_{\epsilon_1}[f(\epsilon_1)]E_{\epsilon_2}[f(\epsilon_2)] = E_{(\epsilon_1, \epsilon_2)}[f(\epsilon_1)f(\epsilon_2)]$

$$\begin{aligned} \Xi(\theta) = & E_{(m, s, \epsilon_1, \epsilon_2)} \{ [V(m, s; \theta_1) - r(m, s, x) - \beta V(m', s'; \theta_1)]|_{\epsilon=\epsilon_1} \times \\ & [V(m, s; \theta_1) - r(m, s, x) - \beta V(m', s'; \theta_1)]|_{\epsilon=\epsilon_2} \} \\ & + v[r_x(m, s, x) + \beta V_{s'}(m', s'; \theta_1)|_{\epsilon=\epsilon_1} \frac{\partial s'}{\partial x}] \times [r_x(m, s, x) + \beta V_{s'}(m', s'; \theta_1)|_{\epsilon=\epsilon_2} \frac{\partial s'}{\partial x}] \} \end{aligned}$$

# Neural Network

## Advantages

- Linearly scalable, the number of parameters grows linearly with dimensionality.
- Robust to multicollinearity and can automatically perform model reduction.
- Well suited for fitting highly nonlinear environments including kinks, discontinuities, discrete choices, and switching.

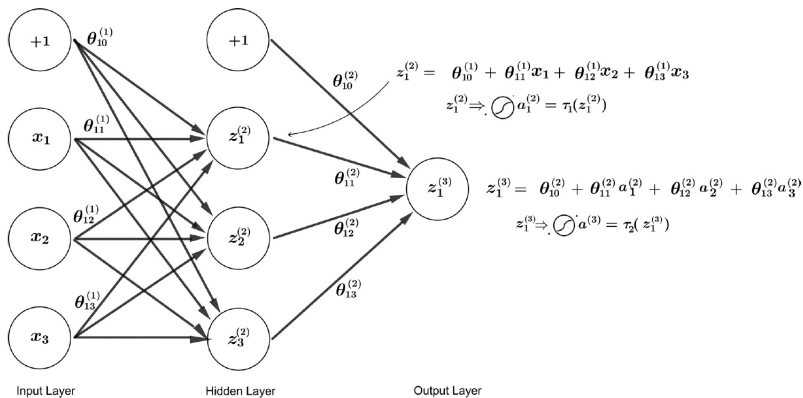


图: A neural network with one hidden layer.

## Data generation

- Stochastic simulation methods solve the model only in the area of the state space in which the solution "lives".
- Using this method in data generation makes most of the data "good".

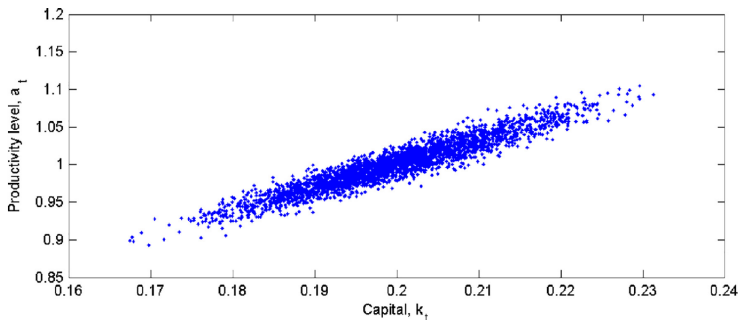


图: Ergodic set of a neoclassical growth model.



## Application Part

# A toy model

- Agent solves

$$\begin{aligned} \max_{\{c_t, w_{t+1}\}_0^\infty} E_0 \left[ \sum_{t=0}^{\infty} \beta u(c_t) \right] \\ \text{s.t. } w_{t+1} = r(w_t - c_t) + e^{y_t} \\ c_t \leq w_t \end{aligned}$$

- where  $c_t$ ,  $w_t$ ,  $y_t$  and  $k_t = w_t - c_t$  are consumption, cash-on-hand, labor productivity, interest rate, wage and next- period capital. Given  $(y_0, w_0)$
- The individual productivity evolves as

$$y_{t+1} = \rho y_t + \sigma \epsilon_t \quad \epsilon_t \sim N(0, 1)$$

- Replication: [https://github.com/cdjnzpy/JME\\_MM\\_TM](https://github.com/cdjnzpy/JME_MM_TM)

# Krusell Smith model

- The economy consists of a set of heterogeneous agents  $i = 1, \dots, \ell$  that are identical in fundamentals but differ in productivity and capital.
- Each agent  $i$  solves

$$\max_{\{c_t^i, k_{t+1}^i\}_0^\infty} E_0 \left[ \sum_{t=0}^{\infty} \beta^t u(c_t^i) \right]$$

$$\text{s.t. } w_{t+1}^i = R_{t+1}(w_t^i - c_t^i) + W_{t+1} e^{y_{t+1}^i}$$

$$c_t^i \leq w_t^i$$

- where  $c_t^i, w_t^i, y_t^i, R_t, W_t$  and  $k_t^i = w_t^i - c_t^i$  are consumption, cash-on-hand, labor productivity, interest rate, wage and next- period capital. Given  $(y_0^i, w_0^i)$
- The individual productivity evolves as

$$y_{t+1}^i = \rho_y y_t^i + \sigma_y \epsilon_t^i \text{ with } \epsilon_t^i \sim N(0, 1)$$

- The production side of the economy is described by a Cobb-Douglas production function

$$F(k) = z_t k_t^\alpha l_t^{1-\alpha}$$

- with  $\alpha \in (0, 1)$ ,  $z_t$  is an aggregate productivity shock, initial  $z_0$  is given

$$z_{t+1} = \rho z_t + \sigma \epsilon_t \text{ with } \epsilon_t \sim N(0, 1)$$

- The equilibrium prices are (with  $l_t = 1$ ):

$$R_t = 1 - \delta + z_t \alpha k_t^{\alpha-1} \left[ \sum_{i=1}^I \exp(y_t^i) \right]$$

$$W_t = z_t (1 - \alpha) k_t^\alpha \left[ \sum_{i=1}^I \exp(y_t^i) \right]$$

- where  $k_t = \sum_{i=1}^I (k_t^i)$  is aggregate capital, and  $\delta$  is depreciation rate.
- Utility function:

$$u(c) = \frac{c^{1-\gamma} - 1}{1-\gamma}$$

# State and Data Generation

## • State space

- Different from a single agent optimization, KS model involves the changes of aggregate. So it need to solve every agents in economy.
- Agent  $i$  state:  $w_t^i, y_t^i$ .
- Agg states:  $\text{distribution}(\{y_t^j, w_t^j\}_{j=1}^\ell)$ , Agg shock( $z_t$ ).
- In total, the input dimension of Neural Network is  $2\ell + 3$  which  $s_t^j = \{y_t^j, w_t^j\}_{j=1}^\ell, z_t, y_t^j, w_t^j\}$ .

## • Data generation (Stochastic simulation methods)

- After training, we can get  $(\{y_t^j, w_t^j\}_{j=1}^\ell)$  and  $(z_t)$  of economy.
- Draw next shocks from random space( $\epsilon_{t+1}$ )
- Using the transition equation ( $w' = W$ ) and shocks ( $\epsilon_{t+1}$ ), update the agent states  $w_{t+1}$  and shocks  $s_{t+1}, z_{t+1}$ .
- Use the new data  $(\{y_{t+1}^j, w_{t+1}^j\}_{j=1}^\ell, z_{t+1})$  as the sample for the next training.

# Neural Network(NN)

- Construct three NNs

- Consumption to wealth ratio:

$$\frac{c_t^j}{w_t^j} = \sigma(\zeta_0 + \eta(y_t^j, w_t^j, D_t, z_t; v)) = \psi(\cdot; \theta)$$

- Unit-free Lagrange multiplier:

$$h_t^j = \exp(\zeta_0 + \eta(y_t^j, w_t^j, D_t, z_t; v)) = h(\cdot; \theta)$$

- Value function

$$V_t^j = \zeta_0 + \eta(y_t^j, w_t^j, D_t, z_t; v) = V(\cdot; \theta)$$

- where  $\eta(\cdot)$  is NN,  $\zeta_0$  is adjustment coefficient,  $D_t = \{y_t^j, w_t^j\}_{j=1}^\ell$  is distribution,  $\sigma(x)$  is Sigmoid function,  $\theta = \{\zeta_0, v\}$ .
- Different NNs will be combined and applied to different solution methods(1-3).

# Method 1 - Maximize lifetime utility

- Loss function (Maximize = Minimize(-Loss))

$$\Xi(\theta) = E_{\omega}[\xi(\omega; \theta)] = E_{(Y_0, W_0, z_0, \Sigma, \sigma)} \left[ \sum_0^T \beta^t u(c_t) \right]$$

- Use the NN of  $\frac{c_t^i}{w_t^j} = \psi(\cdot; \theta)$

## 算法 Maximize LifeTime Utility

```

1: Initialize neural network parameters  $\theta_1$  randomly
2: Draw initial economy with  $\{y_1^i, w_1^i\}_{i=1}^\ell, z_1$ 
3: for  $k = 1$  to  $Epochs$  do
4:   for  $t = 1$  to  $T$  do
5:      $\psi(\cdot; \theta_k) \rightarrow \{c_t^i\}_{i=1}^\ell$  and  $\{k_{t+1}^i = w_t^i - c_t^i\}_{i=1}^\ell$ 
6:     Draw:  $\Sigma = \{\sigma_i^t\}_{i=1}^\ell \rightarrow \{y_{t+1}^i\}_{i=1}^\ell$  and  $\sigma = \sigma_z^t \rightarrow z_{t+1}$ 
7:     Upload  $\rightarrow R_{k+1}, W_{k+1}$ 
8:     Upload:  $\{y_{t+1}^i, w_{t+1}^i\}_{i=1}^\ell, z_{t+1}$ 
9:   end for
10:   $\nabla_\theta \leftarrow \frac{1}{\ell} \sum_{i=1}^\ell [\min_\theta [-\sum_0^T \beta^t u(c_t^i)]]$ 
11:   $\theta_{k+1} \leftarrow \theta_k - learning\_rate \times \nabla_\theta$ 
12:  if  $\|\theta_{k+1} - \theta_k\| < tol$  then
13:    Quit Training Epochs
14:  end if
15: end for

```



# Method 1 - Results

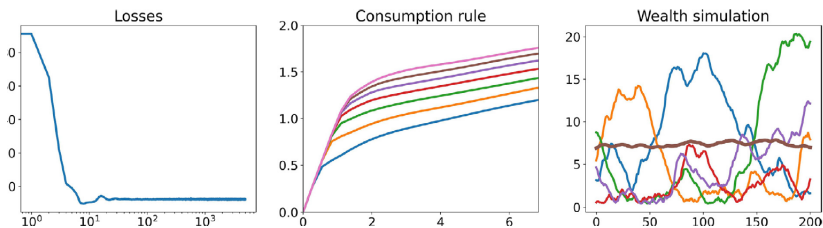


图: Lifetime utility in Krusell and Smith (1998) model.

## Method 2 - Minimize Euler-residual

- Euler Equation:

$$u'(c_t^i) = \beta R_{t+1} E_\epsilon[u'(c_{t+1}^i)]$$

- Minimize Euler-residual

$$\min\{\beta R_{t+1} E_\epsilon[u'(c_{t+1}^i)] - u'(c_t^i)\}$$

$$\min\left\{1 - \frac{\beta R_{t+1} E[u'(c')]}{u'(c)}\right\}$$

- Fischer-Burmeister (FB) function:

- The solution to Euler Eq. can be characterized by Kuhn-Tucker conditions
- $X \geq 0$ ,  $Y \geq 0$  and  $XY = 0$
- Rewrite Kuhn Tucker conditions that as the FB function

$$\Psi^{FB}(x, y) = x + y - \sqrt{x^2 + y^2}$$

- here the Kuhn-Tucker conditions are  $w_t^i - c_t^i \geq 0$  and  $|u'(c) - \beta R_{t+1} E[u'(c')]| \geq 0$ . Thus:

$$x = 1 - \frac{c_t^i}{w_t^i}; y = 1 - h_t^i$$

- with  $h_t^i = \frac{\beta R_{t+1}^i E[u'(c_{t+1}^i)]}{u'(c_t^i)}$

- Loss function (Minimize)

$$\begin{aligned} \Xi(\theta) = E_{\omega}[\xi(\omega; \theta)] = E_{(Y_t, W_t, z_t, \Sigma_1, \Sigma_2, \epsilon_1, \epsilon_2)} \{ & [\Psi(1 - \frac{c_t^i}{w_t^i}, 1 - h_t^i)]^2 \\ & + v[\frac{\beta R_{t+1} u'(c_{t+1}^i)|_{\Sigma=\Sigma_1, \epsilon=\epsilon_1}}{u'(c_t^i)} - h_t^i][\frac{\beta R_{t+1} u'(c_{t+1}^i)|_{\Sigma=\Sigma_2, \epsilon=\epsilon_2}}{u'(c_t^i)} - h_t^i] \} \end{aligned}$$

- where  $v$  is weight and  $\Psi(\cdot)$  is FB function
- Use the NN of  $\frac{c_t^i}{w_t^i} = \psi(\cdot; \theta)$
- Use the NN of  $h_t^i = h(\cdot; \theta)$

## 算法 Minimize Euler-residual

```

1: Initialize neural network parameters  $\theta_1^\psi, \theta_1^h$  randomly
2: Draw initial economy with  $\{y_1^i, w_1^i\}_{i=1}^\ell, z_1$ 
3: for  $k = 1$  to  $Epochs$  do
4:   if  $k \% 2 == 0$  then ▷ Training epoch
5:      $\psi(\cdot; \theta_k) \rightarrow \{c_k^i\}_{i=1}^\ell$  and  $\{k_{k+1}^i\}_{i=1}^\ell$ 
6:      $h(\cdot; \theta_k) \rightarrow \{h_k^i\}_{i=1}^\ell$ 
7:     Draw:  $\Sigma_1, \Sigma_2 \rightarrow [\{y_{k+1}^i\}_{i=1}^\ell]_1, [\{y_{k+1}^i\}_{i=1}^\ell]_2$  and  $\sigma_1, \sigma_2 \rightarrow z_{k+1}^1, z_{k+1}^2$ 
8:     Upload  $\rightarrow R_{k+1}^{1,2}, W_{k+1}^{1,2}$ 
9:      $\xi(\omega; \theta) = f_1(c, h) + \nu f_2(c, h, R)$  ▷ Part of Loss and used params
10:     $\nabla_\theta \leftarrow \frac{1}{\ell} \sum_{i=1}^\ell [\min_\theta \xi(\omega; \theta)]$ 
11:     $\theta_{k+1}^{\psi, h} \leftarrow \theta_k^{\psi, h} - learning\_rate \times \nabla_\theta$ 
12:   else ▷ Simulation epoch
13:     Get:  $\{c_k^i\}_{i=1}^\ell, \{k_{k+1}^i\}_{i=1}^\ell$ 
14:     Draw:  $\Sigma = \{\sigma_i^t\}_{i=1}^\ell \rightarrow \{y_{t+1}^i\}_{i=1}^\ell$  and  $\sigma = \sigma_z^t \rightarrow z_{t+1}$ 
15:     Upload  $\rightarrow R_{k+1}, W_{k+1}, \{y_{k+1}^i, w_{k+1}^i\}_{i=1}^\ell, z_{k+1}$ 
16:   end if
17:   if  $\|\theta_{k+1} - \theta_k\| < tol$  then
18:     Quit Training Epochs
19:   end if
20: end for

```

## Method 2 - Results

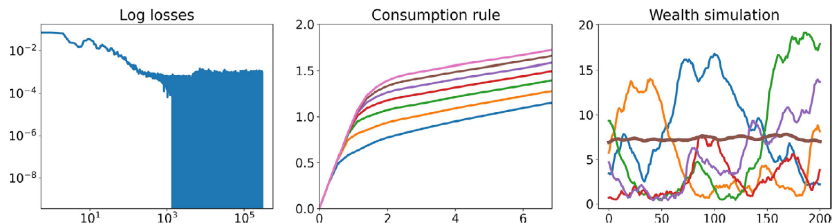


图: Euler-equation method in Krusell and Smith (1998) model.

## Method 3 - Minimize Bellman-residual

- Loss function:

$$\begin{aligned}\Xi(\theta) = & E_{(m,s,\epsilon_1,\epsilon_2)} \{ [V(m,s;\theta_1) - r(m,s,x) - \beta V(m',s';\theta_1)|_{\epsilon=\epsilon_1}] \times \\ & [V(m,s;\theta_1) - r(m,s,x) - \beta V(m',s';\theta_1)|_{\epsilon=\epsilon_2}] \\ & + v[r_x(m,s,x) + \beta V_{s'}(m',s';\theta_1)|_{\epsilon=\epsilon_1} \frac{\partial s'}{\partial x}] \times [r_x(m,s,x) + \beta V_{s'}(m',s';\theta_1)|_{\epsilon=\epsilon_2} \frac{\partial s'}{\partial x}] \}\end{aligned}$$

- Focus on FOC Part:

- We have FOC:

$$u'(c_t^j) + \beta \{ E_\epsilon [V_{w_{t+1}^j}(w_{t+1}^j, y_{t+1}^j, z_{t+1}^j)] \} \frac{\partial w_{t+1}^j}{\partial c_t^j} = 0$$

- In s.t.  $w_{t+1}^j = R_{t+1}(w_t^j - c_t^j) + W_{t+1} e^{y_{t+1}^j}$ , we have  $\frac{\partial w_{t+1}^j}{\partial c_t^j} = -R_{t+1}$
- Thus:

$$Loss_{FOC} = [u'(c_t^j) - \beta V_{w_{t+1}^j}(s_{t+1}^j) R_{t+1}]_{\epsilon=\epsilon_1} [u'(c_t^j) - \beta V_{w_{t+1}^j}(s_{t+1}^j) R_{t+1}]_{\epsilon=\epsilon_2} = 0$$

## Method 3 - Minimize Bellman-residual

- With K-T conditions:

- $w_t^i - c_t^i \geq 0$
- $[u'(c_t^i) - \beta V_{w_{t+1}^i}(s_{t+1}^i)R_{t+1}] \geq 0$
- $(w_t^i - c_t^i) \times Loss_{FOC} = 0$

- With FB function:

- $x = 1 - \frac{c_t^i}{w_t^i}; y = 1 - h_t^i$
- with  $h_t^i = \frac{\beta R_{t+1} E_{\epsilon}[V_{w_{t+1}^i}(s_{t+1}^i)]}{u'(c_t^i)}$

## Method 3 - Minimize Bellman-residual

- Loss function (Minimize)

$$\begin{aligned} \Xi(\theta) = E_{\omega}[\xi(\omega; \theta)] = E_{(Y_t, W_t, z_t, \Sigma_1, \Sigma_2, \epsilon_1, \epsilon_2)} \{ & [V(s_t^j; \theta) - u(c_t^j) - \beta V(s_{t+1}^j; \theta)|_{\Sigma=\Sigma_1, \epsilon=\epsilon_1}] \\ & \times [V(s_t^j; \theta) - u(c_t^j) - \beta V(s_{t+1}^j; \theta)|_{\Sigma=\Sigma_2, \epsilon=\epsilon_2}] + v[\Psi(1 - \frac{c_t^j}{w_t^j}, 1 - h_t^j)]^2 \\ & + v_h[\frac{\beta R_{t+1} \frac{\partial V(s_{t+1}^j; \theta)}{\partial w_{t+1}^j} |_{\Sigma=\Sigma_1, \epsilon=\epsilon_1}}{u'(c_t^j)} - h_t^j] \times [\frac{\beta R_{t+1} \frac{\partial V(s_{t+1}^j; \theta)}{\partial w_{t+1}^j} |_{\Sigma=\Sigma_2, \epsilon=\epsilon_2}}{u'(c_t^j)} - h_t^j] \} \end{aligned}$$

- where  $v, v_h$  are weight for different part.
- Use the NN of  $\frac{c_t^j}{w_t^j} = \psi(\cdot; \theta)$
- Use the NN of  $h_t^j = h(\cdot; \theta)$
- Use the NN of  $V(s_t^j) = V(\cdot; \theta)$



## 算法 Minimize Bellman-residual

```

1: Initialize neural network parameters  $\theta_1^\psi, \theta_1^h, \theta_1^V$  randomly
2: Draw initial economy with  $\{y_1^i, w_1^i\}_{i=1}^\ell, z_1$ 
3: for  $k = 1$  to  $Epochs$  do
4:   if  $k \% 2 == 0$  then ▷ Training epoch
5:     Get:  $\{c_k^i\}_{i=1}^\ell, \{k_{k+1}^i\}_{i=1}^\ell, \{h_k^i\}_{i=1}^\ell$ 
6:      $V(\cdot; \theta_k) \rightarrow \{V_k^i\}_{i=1}^\ell$ 
7:     Draw:  $\Sigma_1, \Sigma_2 \rightarrow [\{y_{k+1}^i\}_{i=1}^\ell]_1, [\{y_{k+1}^i\}_{i=1}^\ell]_2$  and  $\sigma_1, \sigma_2 \rightarrow z_{k+1}^1, z_{k+1}^2$ 
8:     Upload  $\rightarrow R_{k+1}^{1,2}, W_{k+1}^{1,2}$ 
9:      $\xi^B(\omega; \theta) = f_1(V, c) + v f_2(c, h)$  ▷ Bellman-residual
10:     $\xi^F(\omega; \theta) = v_h f_3(V, c, w, h, R)$  ▷ FOC-residual
11:     $\nabla_\theta^{B,F} \leftarrow \frac{1}{\ell} \sum_{i=1}^\ell [\min_\theta \xi^{B,F}(\omega; \theta)]$ 
12:     $\theta_{k+1}^V \leftarrow \theta_k^V - learning\_rate \times \nabla_\theta^B$  ▷ Upload NN  $V$ 
13:     $\theta_{k+1}^{\psi,h} \leftarrow \theta_k^{\psi,h} - learning\_rate \times \nabla_\theta^F$  ▷ Upload NN  $\psi, h$ 
14:   else ▷ Simulation epoch
15:     Same as Euler, upload  $\rightarrow R_{k+1}, W_{k+1}, \{y_{k+1}^i, w_{k+1}^i\}_{i=1}^\ell, z_{k+1}$ 
16:   end if
17:   if  $\|\theta_{k+1} - \theta_k\| < tol$  then
18:     Quit Training Epochs
19:   end if
20: end for

```

## Method 3 - Results

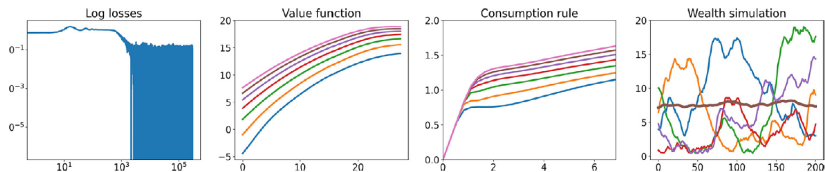



图: Bellman-equation method in Krusell and Smith (1998) model.

# Deep Learning Method - Results

- The output is consistent with the numerical solution
- Much faster than numerical method

$\ell$	$std(y)$	$corr(y, c)$	$Gini(k)$	$Bottom\ 40\%$	$Top\ 20\%$	$Top\ 1\%$	$Time, sec.$	$R^2$
1	1.69	0.862	-	-	-	-	522	0.9837
5	1.69	0.681	0.403	0.143	0.446	0.034	678	0.9910
10	1.64	0.671	0.443	0.115	0.469	0.037	805	0.9934
50	1.64	0.681	0.447	0.113	0.473	0.036	1721	0.9898
100	1.66	0.708	0.430	0.123	0.460	0.036	3297	0.9936
500	1.63	0.699	0.438	0.119	0.467	0.037	21,823	0.9965
1000	1.66	0.707	0.430	0.118	0.465	0.037	43,241	0.9977

 Selected statistics for Krusell and Smith (1998) model.

谢谢大家

## Future Work

# Some Frictions

- Only Euler method is best at training.
- On all range:

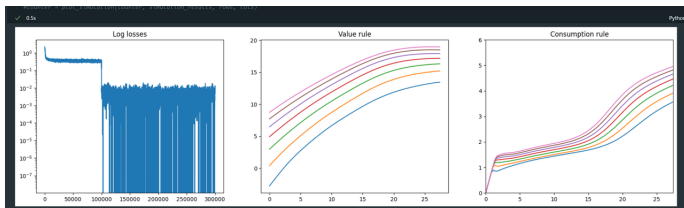


图: Fully Training for Bellman Method.

# More General Deep Learning Method

- Bellman Equation

$$V(m, s) = \max_{x, s'} \{r(m, s, x) + \beta E_{\epsilon}[V(m', s')]\}$$

- Min & Max Problem:

$$Loss = \min_{x, s'} \{V(m, s) - \max_{x, s'} \{r(m, s, x) + \beta E_{\epsilon}[V(m', s')]\}\}$$

- GAN like NNs:

- In Generative NN, Fixed Value NN, and train Policy NN to:

$$\max_{x, s'} \{r(m, s, x) + \beta E_{\epsilon}[\hat{V}(m', s')]\}$$

- In Adversarial NN, Fixed Policy NN, and train Value NN to:

$$\min_{x, s'} \{V(m, s) - \hat{r}(m, s, x) - \beta E_{\epsilon}[\hat{V}(m', s')]\}$$

- More RL like:

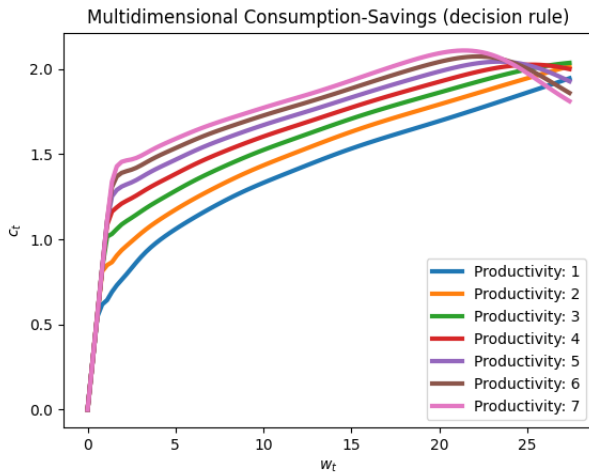
- Parallel Training and Sampling
  - Sampling = Simulation
  - Env. more like a RL env. (state, action, reward)

- How General?
- On methodology:
  - No need for Euler Eq
  - Only Bellman eq needed
  - More stable
- On models:
  - HAM continous choices (Infty, OLG)
  - HAM discrete choices (Infty, OLG)
- On trainings:
  - Flexible Initialization
  - Wider Range

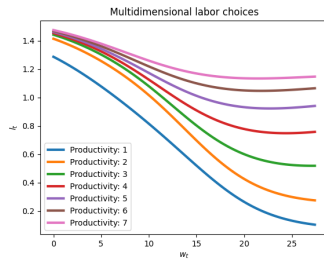
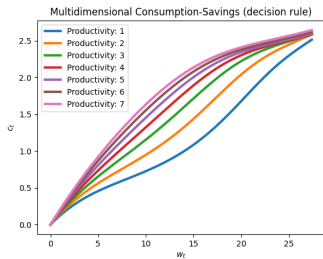


## Some Results

- The JME MM problem:

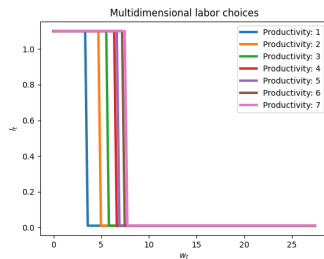
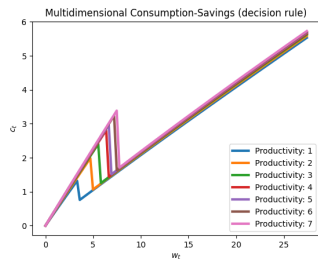


## • Labor choices Model:



Policies

## • Labor choices Model (Discrete):



Policies

- Small OLG:

