Journal of Economic Dynamics & Control 135 (2022) 104295

Contents lists available at ScienceDirect

Journal of Economic Dynamics & Control

journal homepage: www.elsevier.com/locate/jedc

Deep learning classification: Modeling discrete labor choice[☆]

Lilia Maliar[a,b], Serguei Maliar[c,*]

[a] The Graduate Center, City University of New York, USA
[b] CEPR and Hoover Institution, Stanford University, USA
[c] Santa Clara University, USA

ARTICLE INFO

Article history:
Received 19 June 2021
Revised 20 December 2021
Accepted 21 December 2021
Available online 28 December 2021

Keywords:
Artificial intelligence
Machine learning
Deep learning
Neural network
Logistic regression
Softmax regression
Classification
Discrete choice
Indivisible labor

ABSTRACT

We introduce a deep learning classification (DLC) method for analyzing equilibrium in discrete-continuos choice dynamic models. As an illustration, we solve Krusell and Smith's (1998) heterogeneous-agent model with incomplete markets, borrowing constraint and indivisible labor choice. The novel feature of our analysis is that we construct state-contingent discontinuous decision functions that tell us when the agent switches from one employment state to another. We use deep learning not only to characterize the discrete indivisible labor choice but also to perform model reduction and to deal with multicollinearity. Our TensorFlow-based implementation of DLC is tractable in models with thousands of state variables.

© 2021 Elsevier B.V. All rights reserved.

1. Introduction

Dynamic macroeconomic models are generally built on the assumption of continuous-set choices. For example, the agent can distribute wealth in any proportion between consumption and savings or she can distribute time endowment in any proportion between work and leisure. But certain economic choices are discrete: the agent can either buy a house or not, be either employed or not, either retire or not, etc.[1] The progress in modeling discrete choices is still limited because such choices are more challenging to characterize.

In the present paper, we introduce a deep learning classification (DLC) method that can be used to solve a broad class of dynamic economic models with both continuous-set and discrete-set choices. To solve for continuous-set choices, we apply a projection-style method, specifically, we parameterize decision functions with a deep neural network, and we find the coefficients of the neural network (biases and weights) to satisfy the model's equations. Our main novelty is a classification method for constructing discrete-set choices. We define a state-contingent probability function that, for each feasible discrete

[☆] Lilia Maliar and Serguei Maliar acknowledge the support from NSF grants SES-1949413 and SES-1949430, respectively. We thank Marc Maliar for his help with writing a TensorFlow code for producing numerical solutions. We received very useful comments and suggestions from the Editor and the referees.

[*] Corresponding author.
E-mail address: maliars@stanford.edu (S. Maliar).

[1] Examples include indivisibile labor models Chang and Kim (2007); Chang et al. (2019); Hansen (1985); Prescott et al. (2009); Rogerson (1988), retirement models Iskhakov et al. (2017), sovereign default models Arellano (2008); Chatterjee et al. (2007).

https://doi.org/10.1016/j.jedc.2021.104295
0165-1889/© 2021 Elsevier B.V. All rights reserved.

choice, gives the probability that this specific choice is optimal; we parameterize the probability function with a deep neural network; and we find the network parameters to satisfy the optimality conditions for the discrete choices.

As an illustration, let us consider the image recognition problem, which us a typical classification problem in data science. For example, a machine classifies images into cats, dogs and sheep. We parameterize the probabilities of the three classes with a deep neural network. The machine is given a collection of images and is trained to minimize the cross-entropy loss (which is equivalent to maximizing the likelihood function) that ensures the correct classification of images; see Goodfellow et al. (2016) for a survey of classification methods in data science.

Our classification method for constructing discrete choices in economics is analogous to the above image-recognition analysis. For example, in a model with three indivisible labor choices, we use a deep neural network to parameterize the probabilities of being full-time employed, part-time employed and unemployed.[2] The machine is given a collection of employment choices conditional on state and is trained to maximize the likelihood function that those choices are optimal. The same idea can be applied for analyzing the models with retirement, default, house purchase, etc.

However, the analogy between the image recognition and our solution technique is not exact. Image recognition is the skill that human beings instinctly know how to do, so that software is trained on a set of images that human being first identified as matching or not matching. In our application, the software is trained to do something that human beings do not instinctly know how to do, namely, to construct matched vectors of independent and corresponding dependent variables that satisfy a set of model's equations.

The classification solution method we propose can be used to solve small-scale representative agent models. However, the power of deep learning consists in its ability to solve large-scale applications that are intractable with conventional solution methods. To illustrate these remarkable capacities of the proposed DLC method, we solve a large-scale application, specifically, a version of Krusell and Smith (1998)'s model in which the agents face indivisible labor choices.

The studied model features heterogeneous agents, incomplete markets and borrowing constraints and is computationally challenging even in the absence of discrete choices. The state space of the studied model may include thousands of state variables of heterogenous agents and is prohibitively large. To make the model tractable, Krusell and Smith (1998) construct a reduced state space of each agent by considering her own state variables and one or few aggregate moments of the wealth distribution; see Den Haan (2010) for a review of earlier techniques for reducing the state space. Furthermore, several recent papers use linearization and perturbation to simplify the analysis of equilibrium in heterogeneous-agent models, including Ahn et al. (2018); Boppart et al. (2018); Childers (2016); McKay and Reis (2016); Mertens and Judd (2017); Reiter (2010); Winberry (2018), (Bayer and Luetticke, 2020); see Reiter (2019) for a thoughtful discussion of that literature.

A distinctive feature of our DLC method is that it does not rely on moments, linearization, perturbation or any other pre-designed reduction of the state space but works with the actual state space consisting of all individual and aggregate state variables – we let deep neural network to choose how to condense large sets of state variables into much smaller sets of features. Our code is written using Google's TensorFlow platform – deep learning software that led to many ground breaking applications in data science – and is it tractable in models with thousands of state variables.

The studied indivisible labor model was analyzed in important contributions of Chang and Kim (2007) and Chang et al. (2019). The former paper extends Krusell and Smith (1998)'s analysis to include indivisible labor choice by constructing value functions of employed and unemployed agents; their approach reduces the discrete choice problem to the analysis of two continuous-choice value functions. The latter paper offers a more simple and tractable way of modeling indivisible-labor choice by discretizing the first-order conditions of the associated divisible labor model, namely, it assumes that the labor is divisible and that the agent decides on how many hours to work but if the chosen hours fall below a certain level, the agent becomes unemployed. With that assumption, the model with discrete choices is reduced to a familiar setup with continuous labor choices and occasionally binding constraints. We go a step further and solve for entirely discrete choices by using classification techniques instead of relying on some continuous choice representations. Our decision functions tell us when the agent switches from one discrete choice to another, conditional on the individual and aggregate states.

We solve three numerical examples: a version of Krusell and Smith (1998)'s model with continuous choices (i.e., divisible labor), a version of that model with continuous-discrete choices (i.e., indivisible labor) and the corresponding representative-agent model. We find that the introduction of indivisible labor helps us correct some shortcomings of the divisible labor model in reproducing the data, in particular, the data on labor markets. One improvement relatively to empirical data is that the volatility of labor increases relatively to the output; in that respect, our findings are similar to the indivisible labor framework of (Hansen, 1985; Rogerson, 1988). Another improvement is a reduction in the correlation between labor and wages which is excessively high in the representative-agent model with divisible labor; this implication is an outcome of both the assumptions of indivisible labor and heterogeneous agents. As for distributional implications, the predictions of our heterogeneous-agent model with indivisible labor are similar to those of the models studied by Chang and Kim (2007) and Chang et al. (2019). First, the assumption of indivisible labor increases the degrees of inequality, helping to bring the model closer to the data. Furthermore, unlike in the divisible labor model, the degrees of income and wealth inequalities in the indivisible labor economics are less sensitive to variations in the coefficient of risk aversion. However, we conclude that

[2] The earlier literature on indivisible labor (e.g., Rogerson (1988) and Hansen (1985)) convexifies the choice set by introducing lotteries, i.e., they assume that the agent chooses the probability to be employed and unemployed. In our analysis, we also construct probabilities but such probabilities have totally different meaning: they indicate which of the available discrete choices is most likely to be optimal and hence, is selected by the agent.

the assumption of indivisible labor alone is not sufficient to produce empirically relevant degrees of income and wealth inequality in the model. We should emphasize that this is a common outcome of the Krusell and Smith (1998) type of model (without assuming heterogeneity in discount factors).

Our DLC method is related to recent papers on deep learning, including Duarte (2021), Villa and Valaitis (2019), Fernández-Villaverde et al. (2018), Azinovic et al. (2020), Lepetyuk et al. (2020) and especially, Maliar et al. (2018, 2019, 2021). However, this literature does not analyze models with discrete choices, which is the main subject of the present paper. From the other side, there are numerous methods in econometrics for estimating discrete-choice models but these methods are limited to statistic applications; see Train (2009) for a review. An exception is an endogenous grid method with taste shocks by Iskhakov et al. (2017) that is designed to deal with discrete choices in dynamic environment; see also Iskhakov and Keane, 2021 for an application of this method for estimating a partial equilibrium model with discrete labour supply. In the context of Carroll (2005)'s analysis, these papers suggest to apply logistic smoothing to the kinks by transferring the problem into the choice probability space via the taste shocks. There is an important conceptual difference between our analysis and those papers, namely, we do not attempt to smooth the kinks but instead to accurately approximate such kinks by using the-state-of-the-art deep learning classification method.

Finally, our solution method is related to the fields of supervised, unsupervised and reinforcement learning from machine learning literature. First, nonlinear regression equations, which we estimate using artificial data, can be viewed as a generalization of canonical supervised learning; see Maliar et al. (2021) for a discussion. Second, the decision and value functions are not known in economic models, so we can also interpret our solution method as unsupervised learning; such interpretation is advocated in Azinovic et al. (2020) Third, our solution method approximates not only the decision and value functions but also the ergodic set, so it has a direct connection to reinforcement learning; see Goodfellow et al. (2016) for a review of supervised and unsupervised learning in the computer science literature including deep learning; see Sutton and Barto (2018) for a review of reinforcement learning literature, and see Powell (2008) for a review of the related field of approximate dynamic programming.

The rest of the paper is as follows: In Section 2, we set up the Krusell and Smith (1998) model with divisible labor choice; in Section 3, we solve the model with indivisible labor choice; in Section 4, we analyze the model with full- and part-time employment; in Section 5, we compare the aggregate and distributional predictions of divisible and indivisible labor models; and finally, in Section 6, we conclude.

2. Krusell–Smith model with divisible and indivisible labor choices

We consider a version of Krusell and Smith (1998)'s model with three different sets of labor choice: divisible labor in which the agent can work any number of hours, indivisible labor in which the agent chooses between employment and unemployment, and a version of indivisible labor model in which the agent can work part time and full time.

Consumer side The economy consists of a set of heterogeneous agents $i = 1, \ldots, \ell$ that are identical in fundamentals, but differ in dimensions of productivity and capital holdings. The agents experience idiosyncratic productivity shocks and the economy experiences aggregate shock. Each agent i solves

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

$$\text{s.t. } c_t^i + k_{t+1}^i = R_t k_t^i + W_t v_t^i n_t^i, \tag{2}$$

$$n_t \in N, \tag{3}$$

$$\ln v_{t+1}^i = \rho_v \ln v_t^i + \sigma_v \epsilon_t^i \text{ with } \epsilon_t^i \sim \mathcal{N}(0,1), \tag{4}$$

$$k_{t+1}^i \geq \overline{k}, \tag{5}$$

where c_t^i, n_t^i, k_t^i and v_t^i are consumption, hours worked, capital and idiosynratic labor productivity; $\beta \in (0,1)$ is the discount factor; $\rho_v \in (-1,1)$ and $\sigma_v \geq 0$; and initial condition $\left(k_0^i, v_0^i\right)$ is given. The capital choice is restricted by a borrowing limit $\overline{k} \leq 0$. The three different versions of the model are distinguished by the set of allowable labor choices N. We normalize the idiosyncratic productivity levels to one by $\sum_{i=1}^{\ell} v_t^i = 1$ for all t, so that they have no impact on the aggregate productivity level.

Production side

The production side of the economy is described by a Cobb–Douglas production function $\exp(z_t) k_t^{\alpha-1} h_t^{1-\alpha}$, where $k_t = \sum_{i=1}^{\ell} k_t^i$ is aggregate capital, $h_t = \sum_{i=1}^{\ell} v_t^i n_t^i$ is aggregate efficiency labor, and z_t is an aggregate productivity shock following a first-order autoregressive process,

$$\ln z_{t+1} = \rho_z \ln z_t + \sigma_z \epsilon_t \text{ with } \epsilon_t \sim \mathcal{N}(0,1), \tag{6}$$

where $\rho_z \in (-1, 1)$ and $\sigma_z \geq 0$. The interest rate R_t and wage W_t are given by

$$R_t = 1 - d + z_t \alpha k_t^{\alpha-1} h_t^{1-\alpha} \text{ and } W_t = z_t (1-\alpha) k_t^{\alpha} h_t^{-\alpha}, \tag{7}$$

where $d \in (0, 1]$ is the depreciation rate.

Intertemporal choice

The Kuhn–Tucker condition of the agent's problem (1)–(5) with respect to capital is

$$\mu_t^i \delta_t^i = 0, \tag{8}$$

where $\delta_t^i \equiv k_{t+1}^i - \overline{k} \geq 0$ is the distance to the borrowing limit, and $\mu_t^i \geq 0$ is the Lagrange multiplier associated with the borrowing constraint (5) which satisfies the Euler equation,

$$\mu_t^i \equiv u_1\left(c_t^i, n_t^i\right) - \beta E_t\left[u_1\left(c_{t+1}^i, n_{t+1}^i\right) R_{t+1}\right], \tag{9}$$

where u_1 denotes a first-order partial derivative of function u with respect to the first argument. Whenever $\delta_t^i > 0$, the agent is not at the borrowing limit, i.e., $k_{t+1}^i > \overline{k}$, so the Euler equation must hold with equality leading to $\mu_t^i = 0$, and whenever the Euler equation does not hold with equality, it must be that the agent is at the borrowing constraint $\delta_t^i = 0$.

The optimality conditions for labor

To characterize labor choice, we assume that the utility function in (1) takes the form

$$u(c, n) = \frac{c^{1-\gamma} - 1}{1-\gamma} + B\frac{(L-n)^{1-\eta} - 1}{1-\eta}, \tag{10}$$

where γ, $\eta > 0$ and L is the total time endowment which we normalize to L instead of the conventional normalization to 1 because it helps to calibrate the divisible and indivisible labor models to the same steady state.

We consider three versions of the model that differ in the set of allowable labor choices $n_t \in N$: i) the divisible labor model, ii) indivisible labor model and iii) full- and part-time employment model.

i)	divisible labor model	$N = [0, L]$,
ii)	indivisible labor model	$N = \{0, \overline{n}\}$,
iii)	three-state employment model	$N = \{0, \underline{n}, \overline{n}\}$,

In the divisible labor model i), the labor choice is characterized by a FOC of (1)–(5) with respect to labor, which, under the utility function (10), is

$$n_t^i = L - \left[\frac{c_i^{-\gamma} W_t v_t^i}{B}\right]^{-1/\eta}. \tag{11}$$

In the indivisible labor model ii), the agent chooses to be employed ($n_t^i = \overline{n}$) or unemployed ($n_t^i = 0$) depending on which of the two choices leads to a higher continuation value, i.e.,

$$n_t^i = \overline{n} \text{ if } V^E = \max\left\{V^E, V^U\right\} \text{ and } n_t^i = 0 \text{ otherwise.} \tag{12}$$

where V^E and V^U denote value functions of the agent in the employed and unemployed states, respectively.

Finally, in the three-state model iii), the three employment states, $n_t^i = \overline{n}$, $n_t^i = \underline{n}$ and $n_t^i = 0$, correspond to full-time unemployment, part-time employment and unemployment, respectively,

$$n_t^i = \overline{n} \text{ if } V^{FT} = \max\left\{V^U,\ V^{FT}, V^{PT}\right\},\ n_t^i = \underline{n} \text{ if } V^{PT} = \max\left\{V^U,\ V^{FT}, V^{PT}\right\} \text{ and } n_t^i = 0 \text{ otherwise,} \tag{13}$$

where V^{FT}, V^{PT} and V^U denote value functions of full-time employed, part-time employed and unemployed agents, respectively.

3. Deep learning algorithm for divisible labor model

In this section, we first describe a collection of machine learning techniques for analyzing large scale dynamic economic models. We then combine these techniques into a deep learning solution algorithm for solving large scale dynamic economic models. We finally apply the developed deep learning algorithm to solve a version of Krusell and Smith (1998)'s model with divisible labor.

3.1. The ingredients of the deep learning analysis for continuous choice

The state space of Krusell and Smith (1998)'s model includes the state variables of ℓ heterogenous agents, as well as aggregate productivity, $(\left\{k_t^i, v_t^i\right\}_{i=1}^{\ell}, z_t)$. In total, this is $2\ell + 1$ state variables; for example, with $\ell = 1000$, the state space has 2001 state variables. To deal with so large dimensionality, we rely on a combination of techniques introduced in Maliar et al. (2018, 2019, 2021), including: i) stochastic simulation that allows us to restrict attention to the ergodic set in which the solution "lives" - this technique is not specific to deep learning but it help reduce the size of the solution domain under any

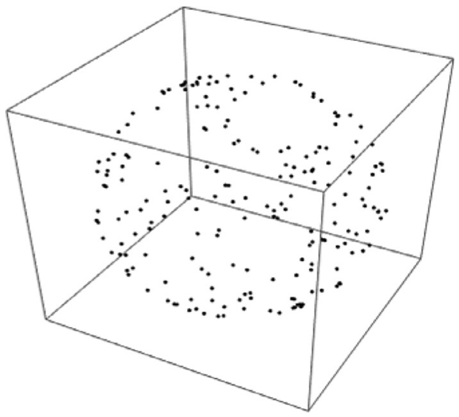

Fig. 1. Hypercube versus hypersphere.

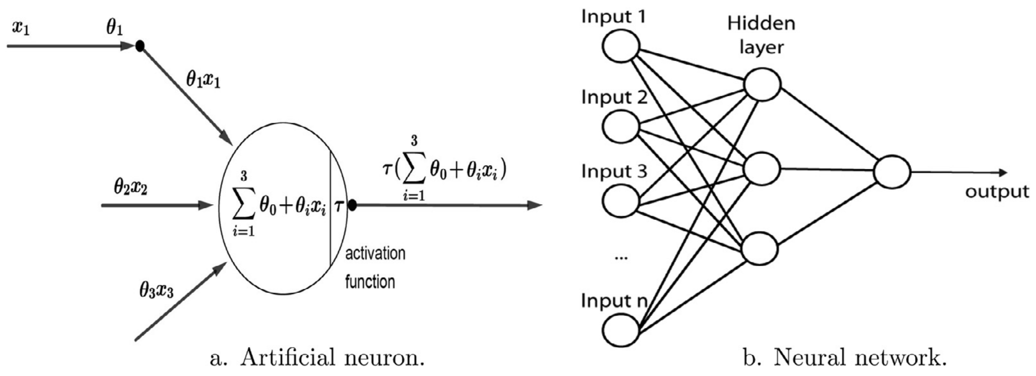

Fig. 2. Artificial neuron and neural network.

approximation function; ii) multilayer neural networks that perform model reduction and help deal with multicollinearity; iii) a (batch) stochastic gradient descent method that reduces the number of function evaluations by operating on random grids; iv) a Fischer–Burmeister function that effectively approximates the kink; v) and most importantly, "all-in-one expectation operator" that allows us to approximate high-dimensional integrals with just 2 random draws (or batches) on each iteration. We implement the numerical method using TensorFlow – a Google data science platform that is used to facilitate the remarkable data-science applications such as image and speech recognition, self driving cars, etc. Below, we describe this methodology in more details. Later, we will augment this methodology to accommodate the indivisible labor model.

Ergodic-set domain

We solve the model on simulated series (ergodic set) instead of a hypercube-style domain used by the classical projection methods (like the Smolyak method, see Maliar and Maliar, 2014). Under normally distributed shocks, stochastic simulation typically have a shape of a hypersphere (hyperoval), like the one shown in Fig. 1.

The ratio of a volume of a hypersphere to the volume of an enclosing hypercube is an infinitesimally small number in high-dimensional applications; for example, for a 30-dimensional case, this ratio is of order 10^{-14}; see Judd et al. (2011) for a discussion. Thus, by solving the model on simulated series, we restrict attention to a relatively small ergodic set in which the solution "lives" – this is the first technique that helps us deal with the curse of dimensionality.

Neural networks

We use neural networks for parameterizing decision and value functions instead of more conventional approximation families like polynomial functions. Examples of an artificial neuron and a neural network are represented in Fig. 2a and b, respectively.

In Fig. 1a, the circle represents an artificial neuron that receives 3 signals (inputs) x_1, x_2 and x_3. We construct a weighted sum of the signals $z \equiv \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3$, where θ_0 is a constant term called "bias" and θ_1, θ_2, θ_3 are called "weights". We then apply an activation function $\tau(\cdot)$ to the linear combination z to obtain output $\tau(z) = \tau(\theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3)$. The coefficients θ_i control the strength of a signal that is passed to the output.

In Fig. 1b, we combine multiple neurons into a neural network. Each neuron of the hidden layer receives a combination of inputs $1, ., n$ (state variables in terms of our model); it aggregates these signals with some weights and a bias, applies some activation function (linear or nonlinear transformation) and passes the resulting output forward. Similarly, the output neuron collects the activated signals from the neurons of hidden layer, aggregates them with some weights and bias and

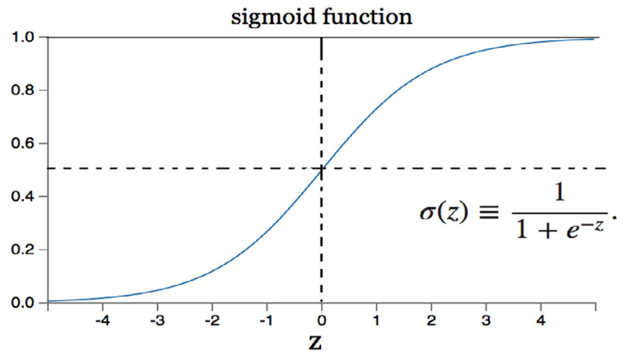

Fig. 3. Sigmoid function.

transforms them with (possibly) another activation function to produce the final output (decision and value functions in terms of our model). Thus, the neural network is a collection of nested linear and nonlinear polynomial functions to which we apply nonlinear transformations. To our purpose, it is a flexible approximating family characterized by a vector of weights and biases in the same way as a polynomial function is characterized by a vector of polynomial coefficients. However, neural networks may be more effective than polynomial functions for constructing approximations on unstructured data such as collections of simulated points.

Activation functions

The activation function that we use in our benchmark experiments is a sigmoid function $\sigma(z) = \frac{1}{1+e^{-z}} = \frac{1}{1+e^{-\theta_0+\theta_1 x_1+\theta_2 x_2+\ldots+\theta_n x_n}}$; see Fig. 3.

The sigmoid function has two convenient properties: First, its derivative can be inferred from the sigmoid function itself $\sigma'(x) = \sigma(x)(1-\sigma(x))$ without using the operator of differentiation. Second, it maps a real line into a unit interval $\sigma : \mathbb{R}^n \to [0,1]$ which makes it a natural candidate for approximating variables that are bounded between 0 and 1. There are many other activation functions used in machine learning literature: (i) heaviside step function: $\tau(z) = 1_{z\geq 0}$; (ii) tanh (hyperbolic tangent) $\tau(z) = \frac{e^z-e^{-z}}{e^z+e^{-z}}$; (iii) relu (rectified linear units): $\tau(z) = \max(0, z)$; (iv) leaky relu: $r(z) = \max(\kappa z, z)$, $\kappa \leq 0$; (v) maxout: $\tau(z) = \max(a_1 z + b_1, a_2 z + b_2, a_3 z + b_3)$. It depends on a specific problem which activation function works best.

In our deep learning algorithm, we solve for two decision functions in terms of state variables, which are hours worked $\frac{n_t^i}{L}$ and the fraction of wealth that goes to consumption $\frac{c_t^i}{w_t^i}$, where wealth is defined as $w_t^i \equiv R_t k_t^i + W_t v_t^i n_t^i$. These two decision functions are parameterized by a sigmoid function

$$\sigma\left(\zeta_0 + \varphi\left(k_t^i, v_t^i, \left\{k_t^i, v_t^i\right\}_{i=1}^{\ell}, z_t; \theta\right)\right), \tag{14}$$

where $\varphi(\cdot)$ is a multilayer neural network parameterized by a vector of coefficients θ (weights and biases), $\sigma(z) = \frac{1}{1+e^{-z}}$ is a sigmoid function which ensures that $\frac{c_t^i}{w_t^i}$ and n_t^i are bounded to be in intervals $[0,1]$ and $[0,L]$, respectively, and ζ_0 is a constant term that we use to initialize the network. In addition, we parameterize the Lagrange multiplier μ_t^i associated with the borrowing constraint using an exponential activation function

$$\exp\left(\zeta_0 + \varphi\left(k_t^i, v_t^i, \left\{k_t^i, v_t^i\right\}_{i=1}^{\ell}, z_t; \theta\right)\right). \tag{15}$$

The exponential activation function ensures that the Lagrange multiplier is always non-negative. Since the agents are identical in fundamentals, the above three $2\ell+1$–dimensional decision functions are sufficient to characterize the choices of all ℓ heterogeneous agents.

Model reduction

Our solution method aims at solving models with thousands of heterogeneous agents (and thus, state variables). The deep learning analysis can deal with such a huge state space because neural networks possess a remarkable property of model reduction. It condenses the information from a large number of inputs into a smaller number of neurons in the hidden layers, making it progressively more abstract and compact. In some sense, this procedure is similar to a photo compression or principal component transformation when a large dataset is condensed into a smaller set of principal components without losing essential information; see Judd et al. (2011) for a discussion of model reduction using principal-component analysis.

In some cases, neural networks can be more effective for approximating non-linear functions than polynomial functions. For example, if we have 2001 state variables, a second-order polynomial function has about 2 millions of polynomial coefficients and yet it is not sufficiently flexible for approximating kinks. In turn, a neural network with two hidden layers, each of which is composed of 32 neurons, has a comparable number of coefficients (weights and biases) but produces a far more accurate approximation to kinks in the context of our model.

Krusell and Smith (1998) proposed one specific model reduction method, namely, they approximate the distribution of state variables with just one moment – the mean of wealth distribution m_t. For their model, such model reduction works extremely well because it turned out that the mean is a sufficient statistic for capturing essentially all relevant information from the distribution, which reduces their state space to just four state variables (k_t^i, v_t^i, z_t, m_t).

If Krusell and Smith (1998)'s model reduction is the most efficient representation of the state space, the neural network is likely to find this reduction as well, as an outcome of training. However, the neural network will automatically consider many other possible ways of extracting the information from the distribution $\left\{k_1^i, v_1^i\right\}_{i=1}^{\ell}$ and condensing it in a relatively small set of hidden layers. The output of the hidden layers may look like moments or some other statistics – we will not always be able to understand how the machine learning handles the information in the hidden layers but this fact does not prevent us from using model reduction in applications.

Casting the model into an objective function for deep learning

We form the objective function that minimizes the Euler-residual by minimizing the squared residuals in three model's conditions,

$$\Xi(\theta) \equiv E_{(K_t,Y_t,z_t)}\left\{\left[\Psi^{FB}\left(1-\frac{c_t^i}{w_t^i}, 1-\mu_t^i\right)\right]^2 + \varpi_n\left[n_t^i - \left(L - \left[\frac{\left(c_t^i\right)^{-\gamma} W_t v_t^i}{B}\right]^{-1/\eta}\right)\right]^2 \right.$$
$$\left. + \varpi_\mu\left[\frac{\beta E_{(\Sigma_{t+1},\epsilon_{t+1})}\left[\left(c_{t+1}^i\right)^{-\gamma} R_{t+1}\middle|\Sigma_{t+1},\epsilon_{t+1}\right]}{\left(c_t^i\right)^{-\gamma}} - \mu_t^i\right]^2\right\}, \quad (16)$$

where $K \equiv \left(k^1,\ldots,k^\ell\right)$ and $Y \equiv \left(v^1,\ldots,v^\ell\right)$ are the vectors of individual state variables; z_t is aggregate productivity; $\Sigma_{t+1} \equiv \left(\epsilon_{t+1}^1,\ldots,\epsilon_{t+1}^\ell\right)$ is the vector of individual productivity shocks; ϵ_{t+1} is the aggregate productivity shock; Ψ^{FB} is a Fisher–Burmeister objective function

$$\Psi^{FB}(a,b) = a + b - \sqrt{a^2+b^2}, \quad (17)$$

whose solution to $\Psi^{FB}(a,b) = 0$ satisfies $a \geq 0$, $b \geq 0$ and $ab = 0$ and is equivalent to the solution to the Kuhn–Tucker conditions (8) but it is differentiable. Finally, ϖ_μ and ϖ_n are the two weights that reflect the relative importance of the different terms in the objective function (for unit-free terms, we can set these weights to one). The objective function is constructed so that by minimizing it with respect to the coefficients θ of neural network, approximations for decision functions for $\frac{c_t^i}{w_t^i}$, μ_t^i, n_t^i deliver us the solution.

All-in-one expectation operator

However, the constructed objective function $\Xi(\theta)$ is not convenient for numerical treatment because it contains a square of the expectation operator $\left[E_{(\Sigma_{t+1},\epsilon_{t+1})}[\cdot]\right]^2$ nested inside another expectation operator $E_{(K_t,Y_t,z_t)}[\cdot]$. Constructing two nested expectation operators is costly if feasible at all because the inner expectation operator $E_{(\Sigma_{t+1},\epsilon_{t+1})}[\cdot]$ has very high dimensionality; in particular, if $\ell = 1,000$, it amounts to a construction of 1,001 -dimensional integral. This task would be simplified enormously if we could combine the two expectation operators into one but this is not possible to do directly since by Jensen's inequality we have $E_{(K_t,Y_t,z_t)}\left[\left[E_{(\Sigma_{t+1},\epsilon_{t+1})}[\cdot]\right]^2\right] \neq E_{(K_t,Y_t,z_t)}E_{(\Sigma_{t+1},\epsilon_{t+1})}\left[[\cdot]^2\right]$.

Maliar et al. (2021) propose a simple but powerful technique, called *all-in-one* (AiO) expectation operator, that can merge the two expectation operators into one. The idea is to replace the squared expectation function $\left[E_{(\Sigma_{t+1},\epsilon_{t+1})}[\cdot]\right]^2$ under one random draw $(\Sigma_{t+1},\epsilon_{t+1})$ with a product of two expectation functions $\left[E_{(\Sigma'_{t+1},\epsilon'_{t+1})}[\cdot]\right] \times \left[E_{(\Sigma''_{t+1},\epsilon''_{t+1})}[\cdot]\right]$ under two uncorrelated random draws $\left(\Sigma'_{t+1},\epsilon'_{t+1}\right)$ and $\left(\Sigma''_{t+1},\epsilon''_{t+1}\right)$. Since the two random draws are uncorrelated, the expectation operator can be taken outside of the expectation function which allows us to re-write (16) as

$$\Xi(\theta) \equiv E_{\left(K_t,Y_t,z_t,\Sigma'_{t+1},\epsilon'_{t+1},\Sigma''_{t+1},\epsilon''_{t+1}\right)}\left\{\left[\Psi^{FB}\left(1-\frac{c_t^i}{w_t^i}, 1-\mu_t^i\right)\right]^2 + \varpi_n\left[n_t^i - \left(L - \left[\frac{\left(c_t^i\right)^{-\gamma} W_t v_t^i}{B}\right]^{-1/\eta}\right)\right]^2 \right.$$
$$\left. + \varpi_\mu\left[\frac{\beta\left[\left(c_{t+1}^i\right)^{-\gamma} R_{t+1}\middle|\Sigma'_{t+1},\epsilon'_{t+1}\right]}{\left(c_t^i\right)^{-\gamma}} - \mu_t^i\right]\left[\frac{\beta\left[\left(c_{t+1}^i\right)^{-\gamma} R_{t+1}\middle|\Sigma''_{t+1},\epsilon''_{t+1}\right]}{\left(c_t^i\right)^{-\gamma}} - \mu_t^i\right]\right\}. \quad (18)$$

Thus, we are able to represent the studied model as an expectation function (18) across a vector of random variables $\left(K_t, Y_t, z_t, \Sigma'_{t+1}, \epsilon'_{t+1}, \Sigma''_{t+1}, \epsilon''_{t+1}\right)$; see Maliar et al. (2021) for a discussion and further applications of the AiO expectation operator.

Training: gradient descent, batches and parallel computing

We use a gradient descent method to train the model, i.e., to search for a vector of coefficients θ that minimizes (18). Given that (18) is an expectation function, we can bring the gradient operator inside by writing $\nabla_\theta \Xi(\theta) = \nabla_\theta E[\xi(\omega;\theta)] = E[\nabla_\theta \xi(\omega;\theta)]$, where ∇_θ is a gradient operator, $\omega \equiv \left(K_t, Y_t, z_t, \Sigma'_{t+1}, \epsilon'_{t+1}, \Sigma''_{t+1}, \epsilon''_{t+1}\right)$ is a vector of all random variables, and $\xi(\cdot)$ is an integrand of expectation function (18) (i.e., the expression inside the curly brackets $\{\cdot\}$). The latter expectation function can be approximated by a simple average across Monte Carlo random draws $E[\nabla_\theta \xi(\omega;\theta)] \approx \frac{1}{N}\sum_{n=1}^{N} \nabla_\theta \xi(\omega_n;\theta)$, where ω_n denotes a specific realization of the vector of random variables. Thus, the gradient descent method can be implemented as

$$\theta \leftarrow \theta - \lambda \nabla_\theta \Xi(\theta) \qquad \text{with} \qquad \nabla_\theta \Xi(\theta) \approx \frac{1}{N}\sum_{n=1}^{N} \nabla_\theta \xi(\omega_n;\theta), \tag{19}$$

where θ and λ are the parameter vector and learning rate, respectively.[3] The advantage of this representation is that we implement a cheap computation of the gradient of the integrand instead of computing far more expensive gradient of the expectation function. The modern data platforms such as TensorFlow and PyTorch can compute such a gradient using a symbolic differentiation, which facilitates an the implementation of parallel computation.[4] Specifically, it allows to construct N multiple realizations of ω by using N batches (replications of the model) and to compute the expectation functions as simple averages across such batches. TensorFlow and PyTorch are designed to effectively work with batches using parallel computing, so the calculation of expectation functions is remarkably cheap. Finally, even if we use just one batch in each iteration which leads to a poor estimator of the expectation function, such an estimator is still unbiased and converges to its true value over the iteration process.

Dealing with multicollinearity

In the arguments of approximating functions (14) and (15), the state variables of agent i appear twice $\varphi\left(k_t^i, v_t^i, \left\{k_t^i, v_t^i\right\}_{i=1}^{\ell}, z_t; \theta\right)$ because they enter both as variables of agent i and as an element of the distribution. This repetition implies perfect collinearity in explanatory variables, so that the inverse problem is not well defined. Such a multicollinearity would break down a conventional least-squares method which solves the inverse problem (since an inverse of a matrix with linearly dependent rows or columns does not exist). However, neural networks are trained by using the gradient-descent method that avoids solving an inverse problem. As a result, neural networks can learn to ignore redundant colinear variables; see Maliar et al. (2021) for numerical illustrations and a discussion.[5]

3.2. Deep learning method for divisible labor model

Below, we combine the above techniques into a deep learning method for solving the divisible labor model. Our deep learning method is similar in spirit to Krusell and Smith (1998)'s analysis but is simpler conceptually as it does not alternate between approximating the individual decision functions and the law of motion for aggregate variables. We just simulate the panel of heterogeneous agents, use the resulting distributions to infer the aggregate quantities and prices and train the individual decision functions. As the machine is trained and the panel is simulated, the decision functions are refined jointly with the ergodic distribution. Since random variables are autocorrelated, the stochastic gradient is also correlated over time and hence, it is biased. To reduce the bias, we train the model on cross-sections which are sufficiently separated in time instead of using all the consecutive periods. Specifically, we train the model and simulate it for 10 or more consecutive periods before implementing the next training step.

For our numerical analysis, we assume $\alpha = 0.36$; $d = 0.08$; $\beta = 0.96$; $\rho = 0.9$; $\sigma = 0.1$; $\rho_z = 0.9$; $\sigma_z = 0.21$; and $\bar{k} = 0$ – these values are in line with the literature, e.g., Chang and Kim (2007), Reiter (2010, 2019), Chang et al. (2019). We perform training using the *ADAM* stochastic gradient descent method with the batch size of 100 and the learning rate of 0.001. We fix the number of iterations (which is also a simulation length) to be $K = 100,000$. The choice of these parameters must ensure both convergence and low running time and it reflects our experience in constructing deep learning approximations. Finally, we study numerically the role of the elasticities γ and η of the utility function by performing a sensitivity analysis.

3.3. Numerical results for the divisible labor model

In Fig. 4, we illustrate the accuracy and cost of our deep learning methods, specifically, the reduction in the loss function over the process of training (left panel) and the running time per iteration depending on the number of agents (right panel).

[3] In our code, we used a version of the gradient descent method, called ADAM, which updates each coefficient of θ at a different rate which we find to enhance the convergence compared to conventional gradient descent method.

[4] Notice that we would not be able to bring the gradient operator inside the expectation function without the AiO operator. We would have no choice but to approximate the gradient of the expectation function numerically, which may be infeasible with a large number of agents.

[5] It is possible to design a transformation that avoids a repetition of state variables in the set of arguments but it would require cumbersome and costly permutations, so we find it easier to keep the repeated arguments since their do not have a significant negative impact on the constructed neural network approximations.

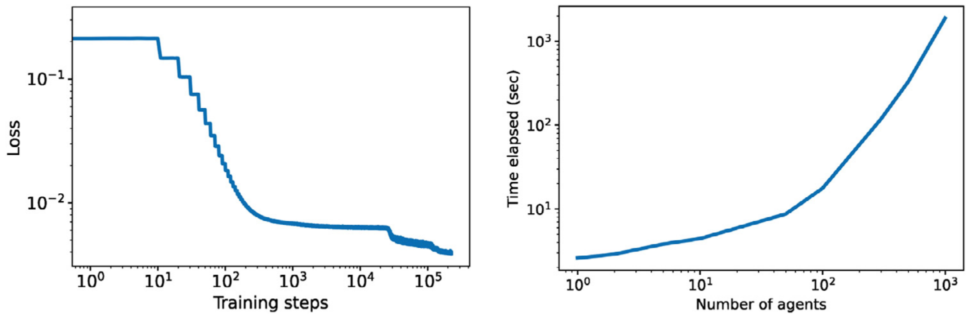

Fig. 4. Training errors and running time for divisible labor model under $\gamma = 1$ and $\eta = 1$.

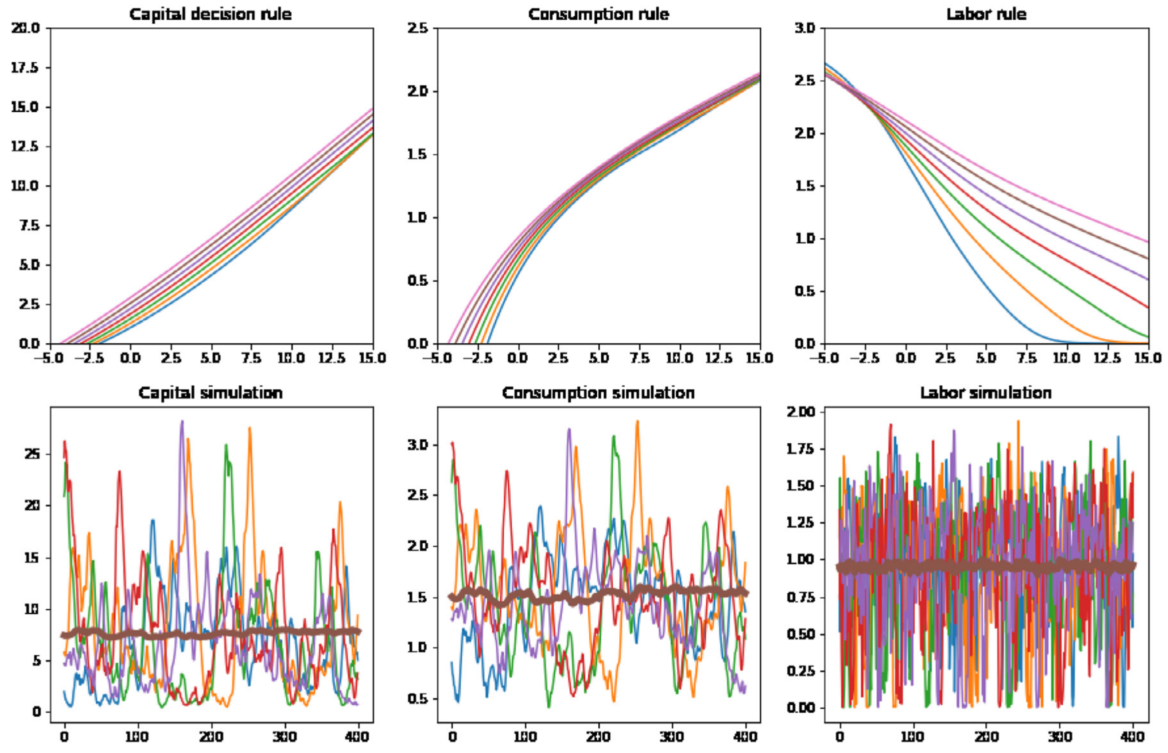

Fig. 5. Solution to divisible labor model under $\gamma = 1$ and $\eta = 1$.

From the left panel of the figure, we see that the loss function is reduced to about 10^{-3} over the training steps which implies that the unit-free residuals in the model's equation are of order of 0.1%. In turn, from the right panel, we see that the running time increases more than linearly in the logarithmic scale. That happens because large number of state variables in the model such as 1000 or 2000 involves larger memory usage, in addition to larger running time. Even though our deep learning method is capable of analyzing models with a very high dimensionality, it does not scale linearly for a very large number of agents.

In Fig. 5, we show the solution with $\ell = 300$ heterogeneous agents under $\gamma = 1$ and $\eta = 1$.

In the top row, we show the individual decision rules as a function of previous capital for seven different states of individual productivity. We see that next-period capital k^i_{t+1} and consumption c^i_t increase in the current capital k^i_t, while labor n^i_t is decreasing meaning that the agent chooses to enjoy more leisure $L - n^i_t$. The seven lines in each graph correspond to seven different productivity states representing a mean $\pm$ 1, 2 and 3 standard deviations; they show that k^i_{t+1}, c^i_t and n^i_t are all increasing with productivity v^i_t. We observe a soft kink in the consumption function in the area where the agent reaches the borrowing limit.

In the bottom row, we show simulated series for 5 agents over time (we do not show all agents to avoid the clutter). Here, we also show as the thick lines the simulation for the corresponding aggregate variables. As expected, fluctuations in individual capital of agents are significantly larger than the fluctuations in their consumption and labor. In the bottom row,

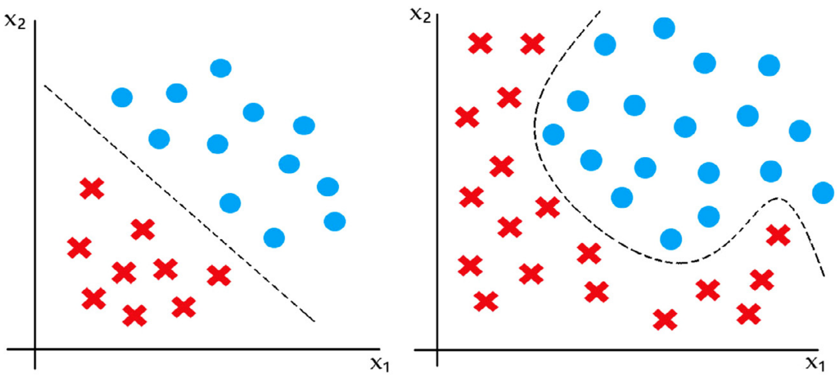

Fig. 6. Examples of binary classification.

we also show simulation for the corresponding aggregate variables k_{t+1}, c_t and n_t. The volatility of the aggregate variables is typical for the real business cycle models and is considerably lower than that of the individual variables. We will quantify the business cycle and distributive properties of the model in Section 5 after we present the version of the model with indivisible labor.

Finally, we comment on two alternative solution methods that we could have used instead of the present one and that would have simplified the construction of equilibrium. First, we could have approximated numerically just one labor decision function instead of both consumption and labor functions. This point was emphasized in Maliar and Maliar (2005), who argue that given n_t^i, we can find h_t, W_t and hence, c_t^i, in a closed form, while given c_t^i, we need a numerical procedure to construct h_t, W_t and n_t^i – hence, it is better to parameterize n_t^i than c_t^i in a similar model. Second, we could have solved for one individual decision function (for example, c_t^i) and one aggregate variable (for example, W_t) in terms of state variables since we can find n_t^i in a closed form, given c_t^i and W_t. We do not follow these approaches because they do not carry over to the model with indivisible labor.

4. DLC algorithm for indivisible labor: logistic regression

In this section, we introduce the DLC method for solving the indivisible labor model. Like in the divisible labor case, we consider the actual state space composed of the individual and aggregate state variables, and we let the neural network to perform the model reduction. To construct the policy function for discrete labor choice, we use a logistic regression algorithm – a popular technique from the field of machine learning. We first describe this techniques using a simple regression example, and we then show how the logistic regression can be used for modeling a discrete labor choice in the context of our indivisible labor analysis.

4.1. Binary classification problem in supervised learning literature

Let us consider a typical classification problem. We have a collection of ℓ data points $\left\{X^i, y^i\right\}_{i=1}^{\ell}$ where $X^i \equiv \left(1, x_1^i, x_2^i, \ldots\right)$ is a collection of dependent variables (features) and y^i is a categorical independent variable (label) that takes values 0 and 1. For example, y^i is a tumor that can be benign "0" or malignant "1", and X^i is a set of observed characteristics such as the size of the tumor and lab test. Our goal is to construct a line that separates the classes 0 and 1. In Fig. 6, we show two examples of a binary classification with two features x_1 and x_2 in which the classes 0 and 1 are represented by "**o**" and "**x**", respectively. In both cases, the goal is to constuct a dashed line that separates the known examples of the two types.

There are numerous techniques in the machine learning literature that can be used for training binary classifiers, including k-nearest neighbors, decision trees, support vector machine, logistic regression, extreme learning machine, naive Bayes. We will restrict attention to one such technique – logistic regression – which is simple, general and can be conveniently combined with our deep learning analysis.

As a first step, we form a hypothesis about the functional form of the separating line. For the left panel, it is sufficient to assume that the separating line is linear

$$H_0 : \theta_0 + \theta_1 x_1 + \theta_2 x_2 = 0,$$

but for the right panel, we must use a sufficiently flexible nonlinear separating function such as a higher-order polynomial function,

$$H_0 : \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_1^2 + \theta_4 x_1 x_2 + \theta_5 x_2^2 + \ldots = 0,$$

where $(\theta_0, \theta_1, \ldots) \equiv \theta$ are the polynomial coefficients. When $X\theta \equiv \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \ldots > 0$, we conclude that y belongs to class 1 and otherwise, we conclude that it is from class 0.

Our next step is to estimate θ coefficients. Since y is a categorical variable $y \in \{0, 1\}$, we cannot use ordinary least-squares estimator, i.e., we cannot regress y on $X\theta$. Instead, we form a logistic regression

$$H_0 : \log \frac{p}{1-p} = X\theta, \tag{20}$$

where p is the probability that a data point with characteristics $X \equiv \left(1, x_1, x_2, x_1^2, \ldots\right)$ belongs to class 1, and $\theta \equiv (\theta_0, \theta_1, \ldots, \theta_m, \ldots, \theta_M)$ is a coefficient vector. The logistic function is an excellent choice for approximating probability: First, it ensures that $p = \frac{1}{1+\exp(-X\theta)} \in (0, 1)$ for any θ and X, and hence p and $(1-p)$ can be interpreted as probabilities that a data point belongs to classes 1 and 0, respectively. Second, $p = \frac{1}{2}$ corresponds to the separation line $X\theta = 0$. Hence, when $p > \frac{1}{2}$, the data point is "above" the separating line $X\theta$, and thus, belongs to the class **1** and if $p < \frac{1}{2}$, the opposite is true. Finally, when $X\theta \to -\infty$ and $X\theta \to +\infty$, we have that $p \to 0$ and $p \to 1$, respectively.

The logistic regression (20) provides a convenient way to estimate the decision boundary coefficients θ by using a maximum likelihood estimator.[6] A probability that the data point i belongs to classes 0 and 1 can be represented with a single formula by

$$\text{Prob}(y \mid X; \theta) = p^y (1-p)^{1-y}.$$

Indeed, if $y = 1$, we have $\text{Prob}(y = 1 \mid X; \theta) = (p)^1 (1-p)^0 = p$; and if $y = 0$, we have $\text{Prob}(y = 0 \mid X; \theta) = (p)^0 (1-p)^1 = 1-p$. We search for the coefficient vector θ that maximizes the (log)likelihood of the event such that a given matrix of features $\left\{X^i\right\}_{i=1}^{\ell}$ produces the given output realizations $\left\{y^i\right\}_{i=1}^{\ell}$, i.e.,

$$\max_{\theta} \ln L(\theta) = \ln \prod_{i=1}^{\ell} \left(p\left(X^i; \theta\right)\right)^{y^i} \left(1 - p\left(X^i; \theta\right)\right)^{1-y^i} =$$
$$\sum_{i=1}^{\ell} \left[y^i \ln\left(p\left(X^i; \theta\right)\right) + \left(1 - y^i\right) \ln\left(1 - p\left(X^i; \theta\right)\right) \right], \tag{21}$$

where the probability $p\left(X^i; \theta\right) \equiv \frac{1}{1+\exp(-X^i\theta)}$ is given by a logistic function.[7] To find the maximizer, we compute the first-order conditions with respect to all coefficients θ_m for $m = 0, \ldots, M$,

$$\frac{\partial \ln L(\theta)}{\partial \theta_m} = \sum_{i=1}^{\ell} \left[\frac{y^i}{p\left(X^i; \theta\right)} \frac{\partial p\left(X^i; \theta\right)}{\partial \theta_m} - \frac{\left(1 - y^i\right)}{\left(1 - p\left(X^i; \theta\right)\right)} \frac{\partial p\left(X^i; \theta\right)}{\partial \theta_m} \right]$$
$$= \sum_{i=1}^{\ell} \left[y^i x_m^i \left(1 - p\left(X^i; \theta\right)\right) - \left(1 - y^i\right) x_m^i p\left(X^i; \theta\right) \right] = \sum_{i=1}^{\ell} \left[y^i - p\left(X^i; \theta\right) \right] x_m^i, \tag{22}$$

where x_m^i is the feature m of agent i.[8] To show the second equality in (22), we have used the remarkable property of the logistic function that its derivative can be computed directly from the logistic function itself $\frac{\partial p(z)}{\partial z} = \frac{\partial}{\partial z}\left[\frac{1}{1+\exp(-z)}\right] = p(z)(1 - p(z))$. The constructed gradient $\nabla \ln L_\theta(\theta) \equiv \left[\frac{\partial \ln L(\theta)}{\partial \theta_1}, \ldots, \frac{\partial \ln L(\theta)}{\partial \theta_M}\right]'$ can be used for implementing the gradient descent-style method $\theta \leftarrow \theta - \lambda \nabla \ln L_\theta(\theta)$ described in (19).

4.2. Binary classification of labor choice

In the divisible labor model studied in Section 2, we construct a policy function that determines the hours worked, while in the indivisible labor model studied here, we construct a decision boundary $\varphi\left(s_t^i; \theta\right) = 0$ that separates the employment and unemployment choices conditional on state $s_t^i \equiv \left(k_t^i, v_t^i, \left\{k_t^i, v_t^i\right\}_{i=1}^{\ell}, z_t\right)$. Whenever $\varphi\left(s_t^i; \theta\right) \ge 0$, the agent is employed $n_t^i = \overline{n}$ and otherwise, the agent is unemployed $n_t^i = 0$. Let us show how such a decision boundary can be constructed by using the logistic regression classification method.

Since our model has a large number of explanatory variables (state variables) as well as a highly nonlinear decision boundary, we use neural networks for approximating such boundary (instead of the polynomial function). We estimate the

6 One can be tempted to minimize the least-squares style criteria $\min_{\theta} \sum_{i=1}^{\ell} \left(y^i - \frac{1}{1+\exp(-X^i\theta)}\right)^2$ instead of maximizing the likelihood function but that would not lead to efficient training (as the sigmoid function has many local minima).

7 In the machine learning field, the problem of maximizing the likelihood function is typically formulated as the problem of minimizing the cross entropy loss, which is defined as a negative likelihood function. The two representations are obviously equivalent.

8 Interestingly, the gradient of the logistic regression is given by the same expression as the gradient of the conventional linear regression, namely, minimizing $L(\theta) = \frac{1}{2} \sum_{i=1}^{\ell} \left(y^i - X^i\theta\right)^2$ with respect to θ_j leads to partial derivatives of type $\frac{\partial L(\theta)}{\partial \theta_m} = \sum_{i=1}^{\ell} \left(y^i - X^i\theta\right) x_m^i$ for $m = 1, \ldots, M$.

coefficients of the neural network (weights and biases) by formulating a logistic regression in the way which is parallel to (20),

$$H_0 : \log \frac{p}{1-p} = \varphi(s;\theta). \tag{23}$$

We parameterize the decision functions p_t^i and $\frac{c_t^i}{w_t^i}$ by a sigmoid function in the indivisible labor model (instead of parameterization (14) of the divisible labor model):

$$\sigma\left(\zeta_0 + \varphi\left(k_t^i, v_t^i, \left\{k_t^i, v_t^i\right\}_{i=1}^{\ell}, z_t; \theta\right)\right), \tag{24}$$

where $\varphi(\cdot)$ is a multilayer neural network parameterized by a vector of coefficients θ (weights and biases), $\sigma(z) = \frac{1}{1+e^{-z}}$ is a sigmoid function which ensures that $\frac{c_t^i}{w_t^i}$ and p_t^i are bounded in the interval $[0, 1]$, respectively, and ζ_0 is a constant term that we use to initialize the network. (Here, we also parameterize the Lagrange multiplier as is shown in (15)). The function p_t^i, parameterized by (24), allows us to infer the indivisible labor choice directly, specifically, an agent is employed $n_t^i = \overline{n}$ whenever $p_t^i \geq \frac{1}{2}$ and is unemployed otherwise $n_t^i = 0$. We can then compute $h_t = \sum_{i=1}^{\ell} v_t^i n^i$ and find W_t and R_t from (7) and restore the remaining individual and aggregate variables.

Our next goal is to check if the constructed labor choices are consistent with the individual optimality conditions. To validate the individual choices, we use the decision functions p_t^i, $\frac{c_t^i}{w_t^i}$ and μ_t^i to restore the value functions for the employed and unemployed agents $V^E\left(s_t^i;\theta^E\right)$ and $V^U\left(s_t^i;\theta^U\right)$, respectively, from appropriately formulated Bellman equations like those used in Chang and Kim (2007). We next construct the labor choice $\widehat{n}_t^i$ implied by these two value functions

$$\widehat{n}_t^i = \begin{cases} \overline{n} \text{ if } V^E = \max\left\{V^E, V^U\right\}, \\ 0 \text{ otherwise.} \end{cases} \tag{25}$$

In the solution, the labor choice $\widehat{n}_t^i$ implied by the value functions must coincide with the labor choice n_t^i produced by our decision function for all i and t. If this is not the case, we proceed with training our classifier. To this purpose, we construct the categorical variable $y_t^i \in \{0, 1\}$ such that

$$y_t^i = \begin{cases} 1 \text{ if } \widehat{n}_t^i = \overline{n}, \\ 0 \text{ otherwise,} \end{cases} \tag{26}$$

and we use it to form the (log)likelihood function

$$\ln L(\theta) = \frac{1}{\ell}\sum_{i=1}^{\ell}\left[y_t^i \ln\left(p\left(s_t^i;\theta\right)\right) + \left(1-y_t^i\right)\ln\left(1-p\left(s_t^i;\theta\right)\right)\right]. \tag{27}$$

We then maximize the likelihood function (27) by using a conventional / stochastic / batch stochastic gradient descent methods described in (19), where we construct the gradient as in (22). We iterate on the decision functions p_t^i, $\frac{c_t^i}{w_t^i}$ and μ_t^i until convergence.

4.3. Determining indivisible labor: value functions versus "discretized" FOC

Note that there is an important implementational difference in the construction of the labor choice in the divisible and indivisible labor models. In the former model, the optimal labor choice must satisfy FOC (11) and hence, it can be constructed by considering just the current period variables. However, this is not true for the indivisible labor model in which the agent chooses to be employed or unemployed depending on which of the two continuation values is larger V^E or V^U.

Prescott et al. (2009) propose a cleaver approach to modeling the indivisible labor choice under which such a choice can be constructed from the current state variables without the need of constructing value functions; also, see Chang et al. (2019) for implementation of this idea in the context of Krusell and Smith (1998)'s model. Their main idea is to allow for intensive and extensive margins by "discretizing" the FOC (11). To be specific, they assume that the labor choice is divisible and is characterized by the divisible-labor FOC (11) as long as it is above a given threshold $\overline{n}_f$ but it jumps to zero whenever the labor choice falls below $\overline{n}_f$ (i.e., the agent becomes unemployed):

$$\widehat{n}_t^i = \begin{cases} L - \left[\frac{c_i^{-\gamma} W_t \exp\left(v_t^i\right)}{B}\right]^{-1/\eta} \geq \overline{n}_f, \\ 0 \text{ otherwise.} \end{cases} \tag{28}$$

We borrow from Prescott et al. (2009) and Chang et al. (2019) the idea of discretizing the FOCs of the divisible labor model, however, we go a step further and we make the labor choice entirely indivisible by assuming that n_t^i can take just two values 0 (unemployed) and $\overline{n}$ (employed):

$$\widehat{n}_t^i = \begin{cases} \overline{n} \text{ if } L - \left[\frac{c_i^{-\gamma} W_t \exp\left(v_t^i\right)}{B}\right]^{-1/\eta} \geq \overline{n}_f, \\ 0 \text{ otherwise.} \end{cases} \tag{29}$$

We believe that the above approach can be a simple and effective alternative to conventional methods that solve for indivisible labor by constructing the value functions V^E and V^U explicitly.

4.4. Solution method for the indivisible labor model

The deep learning method for the indivisible labor model is similar to the one for the divisible labor model except for the way we construct the labor choice. Concerning the labor choice, we show two options: one is based on a reconstruction of value functions and the other is based on the discretized FOC. We again simulate the panel of heterogeneous agents, and we use the resulting distributions to infer the aggregate quantities and prices. As the machine is trained and the panel is simulated, the decision functions are refined jointly with the ergodic distribution. The omitted steps coincide with those in Algorithm 1.

Algorithm 1 Deep learning for divisible labor model.

Step 0: (Initialization).
Construct initial state of the economy $\left(\left\{k_0^i, v_0^i\right\}_{i=1}^{\ell}, z_0\right)$ and parameterizethree decision functions by a neural network with three outputs
$\left\{\frac{n_t^i}{L}, \frac{c_t^i}{w_t^i}\right\} = \sigma\left(\zeta_0 + \varphi\left(k_t^i, v_t^i, \left\{k_t^i, v_t^i\right\}_{i=1}^{\ell}, z_t; \theta\right)\right)$,
$\mu_t^i = \exp\left(\zeta_0 + \varphi\left(k_t^i, v_t^i, \left\{k_t^i, v_t^i\right\}_{i=1}^{\ell}, z_t; \theta\right)\right)$,
where $w_t^i \equiv R_t k_t^i + W_t v_t^i n_t^i$ is wealth; μ_t^i isLagrange multiplier associated with the borrowing constraint; $\varphi(\cdot)$ is a neuralnetwork; $\sigma(z) = \frac{1}{1+e^{-z}}$ is a sigmoid(logistic) function; ζ_0 is a constant; θ is a vector of coefficients (biases andweights).

Step 1: (Evaluation of decision functions).
Given state $\left(k_t^i, v_t^i, \left\{k_t^i, v_t^i\right\}_{i=1}^{\ell}, z_t\right) \equiv s_t^i$,compute n_t^i, μ_t^i, $\frac{c_t^i}{w_t^i}$ fromthe neural networks, find the prices R_t and W_t; and find k_{t+1}^i from the budgetconstraint for all agents $i = 1, \ldots, \ell$.

Step 2: (Construction of Euler residuals).
Draw two random sets of individual productivity shocks $\Sigma_1 = \left(\epsilon_1^1, \ldots, \epsilon_1^{\ell}\right)$, $\Sigma_2 = \left(\epsilon_2^1, \ldots, \epsilon_2^{\ell}\right)$ and two aggregate shocks $\epsilon_{1,}$, ϵ_2,and construct the Euler residuals

$$\Xi(\theta) = \left\{\left[\Psi^{FB}\left(1 - \frac{c_t^i}{w_t^i}, 1 - \mu_t^i\right)\right]^2 + \varpi_n\left[n_t^i - \left(L - \left[\frac{\left(c_t^i\right)^{-\gamma} W_t v_t^i}{B}\right]^{-1/\eta}\right)\right]^2\right.$$
$$\left. + \varpi_\mu\left[\frac{\beta\left[\left(c_{t+1}^i\right)^{-\gamma} R_{t+1}\middle|\Sigma'_{t+1}, \epsilon'_{t+1}\right]}{\left(c_t^i\right)^{-\gamma}} - \mu_t^i\right]\left[\frac{\beta\left[\left(c_{t+1}^i\right)^{-\gamma} R_{t+1}\middle|\Sigma''_{t+1}, \epsilon''_{t+1}\right]}{\left(c_t^i\right)^{-\gamma}} - \mu_t^i\right]\right\},$$

where $\Psi^{FB}(a, b) = a + b - \sqrt{a^2 + b^2}$ is a Fischer-Burmeister function; and ϖ_n, ϖ_μ are given weights.

Step 3*:* (Training).
Train the neural network coefficients θ tominimize the residual function $\Xi(\theta)$ by using a stochastic gradient descent method $\theta \leftarrow \theta - \lambda \nabla_\theta \Xi(\theta)$ with $\nabla_\theta \Xi(\theta) \approx \frac{1}{N}\sum_{n=1}^{N} \nabla_\theta \xi(\omega_n; \theta)$, where $n = 1, \ldots, N$ denotes batches.

Step 4*:* (Simulation).
Move to $t + 1$ by using endogenous and exogenousvariables obtained in Step 3 under $\Sigma_1 = \left(\epsilon_1^1, \ldots, \epsilon_1^{\ell}\right)$ and ϵ_1 as a next-period state $\left(\left\{k_{t+1}^i, v_t^i\right\}_{i=1}^{\ell}, z_{t+1}\right)$.

The proposed algorithm can be viewed as policy iteration because it solves for the decision functions by updating the value functions as opposed to value function iteration which constructs value functions by inferring the policy functions from the given value function. In that respect, we differ from the related literature, in particular, from Chang and Kim (2007) and from Chang et al. (2019). In our numerical experiment, we implemented a version of Algorithm 2, in which we construct the indivisible labor choice from the discretized FOC (29). An important role in equilibrium play the parameter $\overline{n}$ and $\overline{n}_f$ which determine how often the agents will be employed and unemployed. We calibrate the hours of employed

Algorithm 2 Deep learning for the indivisible labor model.

Step 0: (Initialization).

Construct initial state of the economy $\left(\left\{k_0^i, v_0^i\right\}_{i=1}^{\ell}, z_0\right)$ and parameterizethe decision functions by

$\left\{p_t^i, \frac{c_t^i}{w_t^i}\right\} = \sigma\left(\zeta_0 + \varphi\left(k_t^i, v_t^i, \left\{k_t^i, v_t^i\right\}_{i=1}^{\ell}, z_t; \theta\right)\right),$

$\mu_t^i = \exp\left(\zeta_0 + \varphi\left(k_t^i, v_t^i, \left\{k_t^i, v_t^i\right\}_{i=1}^{\ell}, z_t; \theta\right)\right),$

where p_t^i is the probability of being employed $n_t^i = \overline{n}$.

Step 1: (Evaluation of decision functions).

Given state $\left(k_t^i, v_t^i, \left\{k_t^i, v_t^i\right\}_{i=1}^{\ell}, z_t\right) \equiv s_t^i$,compute $n_t^i = \overline{n}$ if $p_t^i \geq \frac{1}{2}$ and $n_t^i = 0$ if $p_t^i < \frac{1}{2}$. Compute w_t^i and $\frac{c_t^i}{w_t^i}$ from thedecision rules, find the prices R_t and and W_t; and find k_{t+1}^i from the budget constraint for all agents $i = 1, \ldots, \ell$.

Option 1: Construct value functions $V^E$and V^U and find $\widehat{n}_t^i = \begin{cases} \overline{n} \text{ if } V^E = \max\left\{V^E, V^U\right\}, \\ 0 \text{ otherwise.} \end{cases}$

Option 2: Define $\overline{n}_f$ and usethe discretized FOC to find $\widehat{n}_t^i = \begin{cases} \overline{n} \text{ if } L - \left[\frac{c_i^{-\gamma} W_t \exp\left(v_t^i\right)}{B}\right]^{-1/\eta} \geq \overline{n}_f, \\ 0 \text{ otherwise.} \end{cases}$

Define $y_t^i = \begin{cases} 1 \text{ if } \widehat{n}_t^i = \overline{n}, \\ 0 \text{ otherwise,} \end{cases}$ for each s_t^i.

Step 2: (Construction of Euler residuals).

Draw two random sets of individual productivity shocks $\Sigma_1 = \left(\epsilon_1^1, \ldots, \epsilon_1^{\ell}\right)$, $\Sigma_2 = \left(\epsilon_2^1, \ldots, \epsilon_2^{\ell}\right)$ and two aggregate shocks $\epsilon_{1,}$, ϵ_2,and construct the Euler residuals

$$\Xi(\theta) = \left\{\left[\Psi^{FB}\left(1 - \frac{c_t^i}{w_t^i}, 1 - \mu_t^i\right)\right]^2 + \varpi_n\left[y_t^i \ln\left(p\left(s_t^i; \theta\right)\right) + \left(1 - y_t^i\right) \ln\left(1 - p\left(s_t^i; \theta\right)\right)\right]^2 \right.$$
$$\left. + \varpi_\mu\left[\frac{\beta\left[\left(c_{t+1}^i\right)^{-\gamma} R_{t+1} \middle| \Sigma'_{t+1}, \epsilon'_{t+1}\right]}{\left(c_t^i\right)^{-\gamma}} - \mu_t^i\right]\left[\frac{\beta\left[\left(c_{t+1}^i\right)^{-\gamma} R_{t+1} \middle| \Sigma''_{t+1}, \epsilon''_{t+1}\right]}{\left(c_t^i\right)^{-\gamma}} - \mu_t^i\right]\right\},$$

where $\Psi^{FB}(a, b) = a + b - \sqrt{a^2 + b^2}$ is a Fischer-Burmeister function; and ϖ_n, ϖ_μ are given weights.

Step 3: **(Training).**

...

Step 4: **(Simulation).**

...

agent at $\overline{n} = 1.1$ and we set the parameter $\overline{n}_f$ at $\overline{n}_f = 0.5$, which implies that the agents who choose to work less than 0.5 in the divisible labor model become unemployed in the indivisible labor model. We set the hours of unemployed agent to a small number $n = 0.01$ because $n = 0$ leads to numerical issues in training. The resulting parameterization delivers unconditional probability of employment of about 90%, which is roughly consistent with the data.

4.5. Numerical results

In Fig. 7, we show the accuracy and cost of the deep learning classification method, specifically, the reduction of the loss function during the process of training and the time per iteration depending on the number of agents in the model. The regularities here are very similar to those observed in the divisible labor version of the model.

In Fig. 8, we illustrate the solution to the model with discrete labor choice produced by the DLC method.

The magnitude of fluctuations of consumption and capital appears to be similar in Fig. 8 but this happens because consumption and capital graphs have different units. In fact, capital is far more sensitive and volatile than consumption in the constructed solution, which is stylized feature of consumption-saving models.

The decision rules in the top row are qualitatively similar to those in the divisible labor model, however, the changes in labor happen in discrete jumps and they induce the corresponding discrete jumps in consumption and capital (the seven lines again correspond to seven individual productivity levels). The jumps in consumption happen because in equilibrium, the employed agent accepts a consumption cut to enjoy more leisure in the unemployed state. The exact moment when the agent switches from the employed to unemployed states depends on the productivity level. Also, we see that more productive agents remain employed for larger capital levels than low productive agents.

Algorithm 3 Deep learning for the model with full and partial employment.

Step 0: (Initialization).

Construct initial state of the economy $\left(\left\{k_0^i, v_0^i\right\}_{i=1}^{\ell}, z_0\right)$ and parameterizethe decision functions by

$$\left\{\frac{p_t^i(\overline{n})}{\Sigma}, \frac{p_t^i(\underline{n})}{\Sigma}, \frac{p_t^i(0)}{\Sigma}, \frac{c_t^i}{w_t^i}\right\} = \sigma\left(\zeta_0 + \varphi\left(k_t^i, v_t^i, \left\{k_t^i, v_t^i\right\}_{i=1}^{\ell}, z_t; \theta\right)\right),$$

where $p_t^i(\overline{n})$, $p_t^i(\underline{n})$ and $p_t^i(0)$ arethe probabilities to be full- and part-time employed and unemployed, respectively, and $\Sigma \equiv p_t^i(\overline{n}) + p_t^i(\underline{n}) + p_t^i(0)$ is a normalization of probability to one.

Step 1: (Evaluation of decision functions).

Given state $\left(k_t^i, v_t^i, \left\{k_t^i, v_t^i\right\}_{i=1}^{\ell}, z_t\right) \equiv s_t^i$,set $n_t^i = \overline{n}$, $n_t^i = \underline{n}$ and $n_t^i = 0$depending on which probability $p_t^i(\overline{n})$, $p_t^i(\underline{n})$ and $p_t^i(0)$ isthe largest. Compute w_t^i, $\frac{c_t^i}{w_t^i}$ from thedecision rules and find k_{t+1}^i from the budget constraint for all agents $i = 1, \ldots \ell$.

Reconstruct the value functions in the employed, parttime employed and unemployed agents V^E, V^{PT} and V^U, respectively.

Find $\widehat{n}_t^i = \begin{cases} \overline{n} \text{ if } V^E = \max\left\{V^E,\ V^{PT}, V^U\right\}, \\ \underline{n} \text{ if } V^{PT} = \max\left\{V^E,\ V^{FT}, V^U\right\}, \\ 0 \text{ otherwise.} \end{cases}$ and define $y_t^i = \begin{cases} (1,0,0) \text{ if } \widehat{n}_t^i = \overline{n}, \\ (0,1,0) \text{ if } \widehat{n}_t^i = \underline{n}, \\ (0,0,1) \text{ otherwise.} \end{cases}$ for each s_t^i.

Option 1: Construct value functions V^E, V^{PT}, V^U and find $\widehat{n}_t^i = \begin{cases} \overline{n} \text{ if } V^E = \max\left\{V^E,\ V^{PT}, V^U\right\}, \\ \underline{n} \text{ if } V^{PT} = \max\left\{V^E,\ V^{FT}, V^U\right\}, \\ 0 \text{ otherwise.} \end{cases}$

Option 2: Define $\left[\overline{n}_p, \overline{n}_f\right]$ and use discretized FOC to find $\widehat{n}_t^i = \begin{cases} \overline{n} \text{ if } L - \left[\frac{c_i^{-\gamma} W_t \exp\left(v_t^i\right)}{B}\right]^{-1/\eta} \geq \overline{n}_f \\ \underline{n} \text{ if } L - \left[\frac{c_i^{-\gamma} W_t \exp\left(v_t^i\right)}{B}\right]^{-1/\eta} \in \left[\overline{n}_p, \overline{n}_f\right] \\ 0 \text{ otherwise} \end{cases}$

Define $y_t^i = \begin{cases} (1,0,0) \text{ if } \widehat{n}_t^i = \overline{n}, \\ (0,1,0) \text{ if } \widehat{n}_t^i = \underline{n}, \\ (0,0,1) \text{ otherwise.} \end{cases}$ for each s_t^i.

Step 2: (Construction of Euler residuals).

Draw two random sets of individual productivity shocks $\Sigma_1 = \left(\epsilon_1^1, \ldots, \epsilon_1^{\ell}\right)$, $\Sigma_2 = \left(\epsilon_2^1, \ldots, \epsilon_2^{\ell}\right)$ and two aggregate shocks $\epsilon_{1,}$, ϵ_2,and construct the Euler residuals

$$\Xi(\theta) = \left\{\left[\Psi^{FB}\left(1 - \frac{c_t^i}{w_t^i}, 1 - \mu_t^i\right)\right]^2 + \frac{\varpi_n}{3} \textstyle\sum_{k=1}^3 \left[\widehat{y}_t^{i,k} \ln\left(p\left(s_t^i; \theta^{(k)}\right)\right) + \left(1 - \widehat{y}_t^{i,k}\right) \ln\left(1 - p\left(s_t^i; \theta^{(k)}\right)\right)\right]^2 \right.$$
$$\left. + \varpi_\mu \left[\frac{\beta\left[\left(c_{t+1}^i\right)^{-\gamma} R_{t+1} \middle| \Sigma'_{t+1}, \epsilon'_{t+1}\right]}{\left(c_t^i\right)^{-\gamma}} - \mu_t^i\right]\left[\frac{\beta\left[\left(c_{t+1}^i\right)^{-\gamma} R_{t+1} \middle| \Sigma''_{t+1}, \epsilon''_{t+1}\right]}{\left(c_t^i\right)^{-\gamma}} - \mu_t^i\right]\right\},$$

where $\Psi^{FB}(a, b) = a + b - \sqrt{a^2 + b^2}$ is a Fischer–Burmeister function; and ϖ_n, ϖ_μ are given weights.

Step 3: **(Training).**

…

Step 4: **(Simulation).**

…

The simulated series for the individual capital and consumption in Fig. 8 are similar to those in the divisible labor model in Fig. 5, althiugh labor fluctuates between the employed and unemployed states in discrete jumps. Finally, the fluctuations in aggregate series are again typical for the real business cycle models, except that we can observe that aggregate labor changes in small discrete increments which are a consequence of discrete jumps in individual labor.

Iskhakov et al. (2017) introduce another method for solving dynamic models with discrete-continuous choices, namely, an endogenous grid method with taste shocks, so it is interesting to compare our analysis to theirs. There are three main differences: First, our DLC method is designed to approximate sharp kinks in policy functions like those shown in the figure, while Iskhakov et al. (2017) suggest to smooth the kinks by introducing supplementary preference shocks which transform the discrete choice problem into the choice probability space. Second, in our application, our discrete-choice decisions depend only on the economy's state, while in their application, such decisions depend both on state and time, namely, the agent has to decide which time period to retire. The presence of time among the argument of the choice function feature creates "secondary" discrete shocks that propagate across time domain; such secondary shocks are absent in our applica-

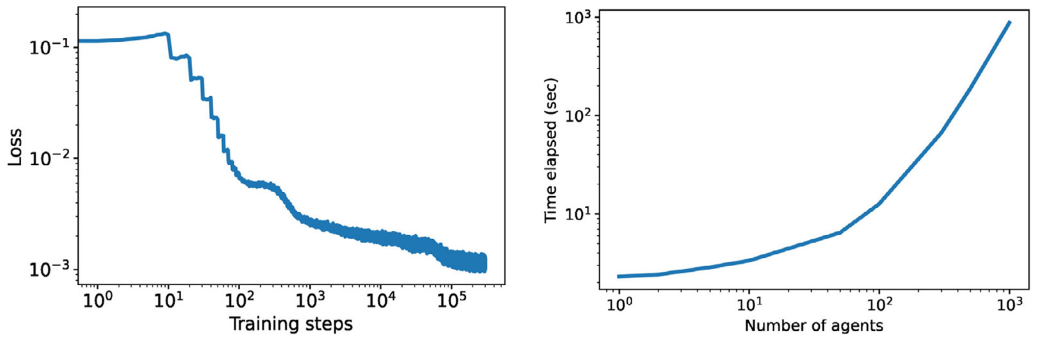

Fig. 7. Training errors and running time for indivisible labor model under $\gamma = 1$ and $\eta = 1$.

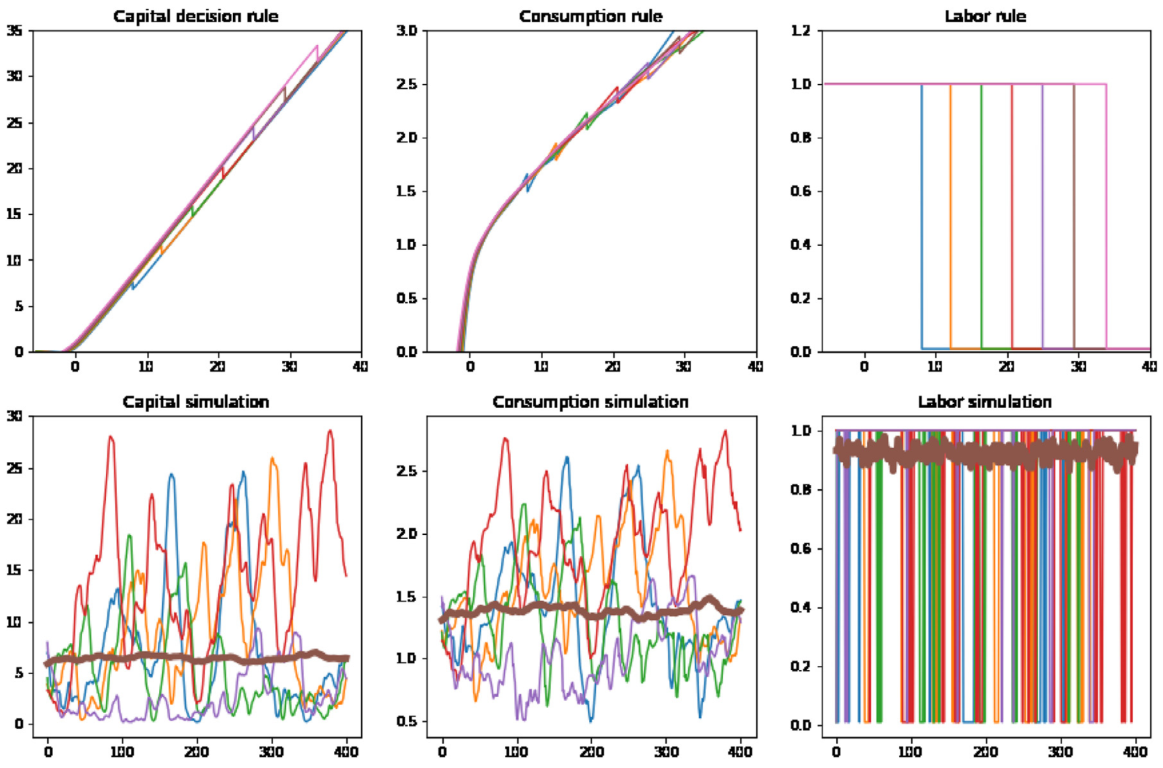

Fig. 8. Solution to indivisible labor model under $\gamma = 1$ and $\eta = 1$.

tion. Finally, the problems that we solve have much larger dimensionality than those studied in Iskhakov et al. (2017), but that paper is more challenging in another respect: they estimate the model's parameters which requires solving the model a large number of times. It would be interesting to see how our DLC method performs in the context of their application but this lies beyond the scope of the present paper.

5. Deep learning classification method for multiclass problems: softmax regression

How can logistic classification approach can be generalized for the case of more than 2 outcomes? Specifically, we are interested in the case when the agent faces three labor choices: full-time employment, part-time employment and unemployment. We again start by reviewing the multiclass classification techniques in the machine learning literature, and we then show how such techniques can be incorporated in a solution method for dynamic economic models.

5.1. Multiclass classification problem in supervised learning literature

Let us consider a multiclassification problem. We again have a collection of ℓ data points $\left\{X^i, y^i\right\}_{i=1}^{\ell}$ where $X^i \equiv \left(1, x_1^i, x_2^i, \ldots\right)$ is composed of dependent variables (features) but now y^i is a categorical independent variable (label) that

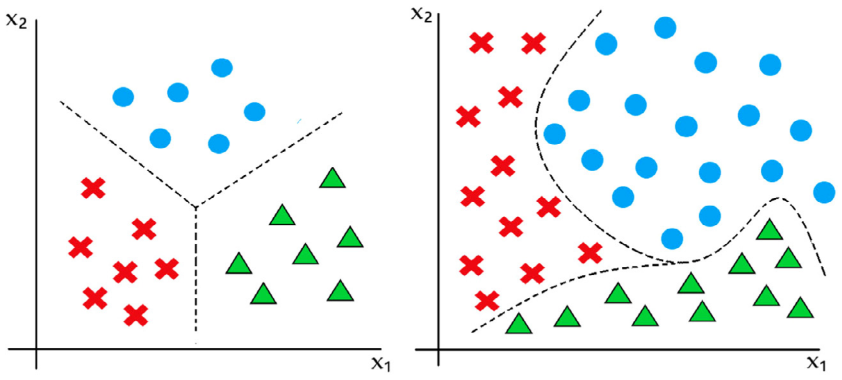

Fig. 9. Examples of multiclass classification.

takes K values. For example, y^i is a tumor that can be benign "1", malignant "2" or advanced malignant "3", and X^i is a set of observed characteristics such as the size of the tumor and lab test. Our goal is to construct the lines that separate the classes 1, 2 and 3. In Fig. 9, we show two examples of multiclass classification problem with two features x_1 and x_2 in which the classes 1, 2 and 3 are represented by "$\mathbf{x}$", "Δ" and "$\mathbf{o}$", respectively; in the left panel, the decision boundaries between the classes are linear, and in the right panel, they are nonlinear.

A popular approach in machine learning is to reformulate a multiclass classification problem as a collection of binary classification problems.[9] The key assumption behind this approach is the hypothesis of an independence of irrelevant alternatives which postulates that the agent's choice between any two alternatives is independent of the presence or absence of other alternatives.[10] In our analysis, that means that the choice between $\{\mathbf{x}\}$ and $\{\Delta\}$ is independent of the availability of $\{\mathbf{o}\}$, the choice between $\{\Delta\}$ and $\{\mathbf{o}\}$ is independent of the availability of $\{\mathbf{x}\}$ and the choice between $\{\mathbf{o}\}$ and $\{\mathbf{x}\}$ is independent of the availability of $\{\Delta\}$.

Two binary reformulations of a multiclass classification problems are the *one-versus-one* and *one-versus-rest* (or *one-versus-all*) classifiers,

$$\begin{array}{llll} one-versus-one & \ln\frac{p(\mathbf{x})}{p(\mathbf{o})}=X\theta^{(1)} & \ln\frac{p(\Delta)}{p(\mathbf{o})}=X\theta^{(2)} & \ln\frac{p(\Delta)}{p(\mathbf{x})}=X\theta^{(3)}, \\ one-versus-rest & \ln\frac{p(\mathbf{x})}{p(\mathbf{o})+p(\Delta)}=X\theta^{(1)} & \ln\frac{p(\Delta)}{p(\mathbf{o})+p(\mathbf{x})}=X\theta^{(2)} & \ln\frac{p(\mathbf{o})}{p(\Delta)+p(\mathbf{x})}=X\theta^{(3)}, \end{array}$$

where $\theta^{(1)}$, $\theta^{(2)}$ and $\theta^{(3)}$ are the regression coefficients and X is the matrix of features. Under *one-versus-one* method, we construct decision boundaries between each pair of classes by running $\frac{K(K-1)}{2}$ pairwise logistic regressions, specifically, we separate $\{\mathbf{x}\}$ from $\{\mathbf{o}\}$, we separate $\{\Delta\}$ from $\{\mathbf{o}\}$, and we separate $\{\Delta\}$ from $\{\mathbf{x}\}$, where the number of classes is $K=3$. In turn, under *one-versus-rest* approach, we construct decision boundaries between each given class and the remaining classes by running K pairwise logistic regressions, specifically, we separate $\{\mathbf{x}\}$ from $\{\mathbf{o},\Delta\}$, we separate $\{\Delta\}$ from $\{\mathbf{o},\mathbf{x}\}$, and we separate $\{\mathbf{o}\}$ from $\{\Delta,\mathbf{x}\}$. For a larger number of classes K, the *one-versus-one* method is more expensive than *one-versus-rest* method but it is more robust to unbalanced data sets. Again, there are numerous techniques in machine learning literature for constructing and training multiclass classifiers, including k-nearest neighbors, decision trees, naive Bayes, random forest, gradient boosting, neural networks, extreme learning machine.

To train the constructed multiclass classifiers, we may omit one of three regressions by imposing the restriction that the probabilities are added to one. For the one-versus-one classifier, the first two regressions imply $p(\mathbf{x})=p(\mathbf{o})\exp\left(X\theta^{(1)}\right)$ and $p(\Delta)=p(\mathbf{o})\exp\left(X\theta^{(2)}\right)$ so that $p(\mathbf{o})\left(1+\exp\left(X\theta^{(1)}\right)+\exp\left(X\theta^{(2)}\right)\right)=1$. In turn, for the one-versus-rest classifier, in the first regression, we replace $p(\mathbf{o})+p(\Delta)$ with $1-p(\mathbf{x})$ and in the second regression, we replace $p(\mathbf{o})+p(\mathbf{x})$ with $1-p(\Delta)$. Consequently, we can re-write two classifiers as

$$\begin{array}{llll} one-versus-one & p(\mathbf{x})=\exp\left(X\theta^{(1)}\right)p(\mathbf{o}) & p(\Delta)=\exp\left(X\theta^{(2)}\right)p(\mathbf{o}) & p(\mathbf{o})=\frac{1}{1+\exp\left(X\theta^{(1)}\right)+\exp\left(X\theta^{(2)}\right)}, \\ one-versus-rest & p(\mathbf{x})=\frac{1}{1+\exp(-X\theta^{(1)})} & p(\Delta)=\frac{1}{1+\exp(-X\theta^{(2)})} & p(\mathbf{o})=1-p(\mathbf{x})-p(\Delta). \end{array}$$

Note that in the above expressions, we treat the normalizing class $\{\mathbf{o}\}$ differently from the other two classes $\{\Delta,\mathbf{x}\}$. There is also a symmetric version of the *one-versus-rest* method in which all K classes are treated identically by estimating K unnormalized one-versus-rest logistic regressions $\ln p(\mathbf{x})=X\theta^{(1)}$, $\ln p(\Delta)=X\theta^{(2)}$, $\ln p(\mathbf{o})=X\theta^{(3)}$ and by normalizing the

[9] One may think of an alternative approach in which we construct an interval $[a,b)$ and one decision boundary $H_0: X\theta=0$ which separates the data into multiple classes; for example, if $X\theta<a$, we choose 0, if $X\theta\in[a,b)$ we choose 1 and if $X\theta>b$ we choose 2. But this approach requires strong monotonicity and is tractable only in very special cases.

[10] It is possible to find some real-life situations in which this assumption is violated, for example, the voting preferences between two politicians can be affected by the presence or absence of another politician which is nobody preferred alternative but which everyone prefers to avoid the victory of the least preferred candidate.

exponential function ex-post by their sum. This classifier is called softmax and it is a generalization of a logistic function to multiple dimensions,

$$one-versus-rest\ softmax \quad p(\mathbf{x}) = \tfrac{1}{\Sigma}\exp\left(X\theta^{(1)}\right) \quad p(\Delta) = \tfrac{1}{\Sigma}\exp\left(X\theta^{(2)}\right) \quad \ln p(\mathbf{o}) = \tfrac{1}{\Sigma}\exp\left(X\theta^{(3)}\right),$$

where $\Sigma = \exp\left(X\theta^{(1)}\right) + \exp\left(X\theta^{(2)}\right) + \exp\left(X\theta^{(3)}\right)$. The symmetric treatment is particularly convenient in the context of our deep learning analysis because it allows us to use a neural network with K symmetric outputs in which the softmax function in the last layer selects the largest probability from K probabilities constructed.

The log-likelihood function for the softmx classifier is similar to the one for the binary classifier (21) except that we also do a summation over K of possible outcomes,

$$\max_{\theta_1,\dots,\theta_K} \ln L(\theta_1,\dots,\theta_K) = \frac{1}{K\ell}\sum_{k=1}^{K}\sum_{i=1}^{\ell}\left[y^{i,k}\ln\left(p\left(X^i;\theta^k\right)\right) + \left(1-y^{i,k}\right)\ln\left(1-p\left(X^i;\theta^k\right)\right)\right], \tag{30}$$

where $y^{i,k}$ is a categorical variable constructed so that $y^{i,k}=1$ if observation i belongs to class k and it is zero otherwise. Again, we maximize the constructed likelihood function (30) by using a gradient descent style method, $\theta \leftarrow \theta - \lambda\nabla \ln L_\theta(\theta)$, described in (19).

5.2. Multiclass classification of labor choice

We next extend our indivisible labor heterogeneous-agent model with two employment choices $\{0,\overline{n}\}$ to three employment choices $\{0,\underline{n},\overline{n}\}$. In the latter case, we parameterize not one but three decision boundaries that separate the three employment choices, so instead of the parameterization (14), we use a sigmoid function to parameterize four functions $\frac{p_t^i(\overline{n})}{\Sigma}$, $\frac{p_t^i(\underline{n})}{\Sigma}$, $\frac{p_t^i(0)}{\Sigma}$, $\frac{c_t^i}{w_t^i}$, specifically:

$$\sigma\left(\zeta_0 + \varphi\left(k_t^i, v_t^i, \left\{k_t^i, v_t^i\right\}_{i=1}^{\ell}, z_t; \theta\right)\right), \tag{31}$$

where $\varphi(\cdot)$ is a multilayer neural network parameterized by a vector of coefficients θ (weights and biases), $\Sigma \equiv p_t^i(\overline{n}) + p_t^i(\underline{n}) + p_t^i(0)$ normalizes the probabilities to one; $\sigma(z) = \frac{1}{1+e^{-z}}$ is a sigmoid function which ensures that $\frac{c_t^i}{w_t^i}$ and $\frac{p_t^i(\overline{n})}{\Sigma}$, $\frac{p_t^i(\underline{n})}{\Sigma}$ and $\frac{p_t^i(0)}{\Sigma}$ are bounded in the interval $[0,1]$, and ζ_0 is a constant term that we use to initialize the network. (In addition, we also parameterize the Lagrange multiplier as shown in (15)). The output of the logistic function $\max\left\{\frac{p_t^i(\overline{n})}{\Sigma}, \frac{p_t^i(\underline{n})}{\Sigma}, \frac{p_t^i(0)}{\Sigma}\right\}$ allows us to infer the indivisible labor choice, specifically, an agent is employed $n_t^i = \overline{n}$, part-time employed $n_t^i = \underline{n}$ and unemployed $n_t^i = 0$ whenever its maximum is equal to $\frac{p_t^i(\overline{n})}{\Sigma}$, $\frac{p_t^i(\underline{n})}{\Sigma}$ and $\frac{p_t^i(0)}{\Sigma}$, respectively. We can then compute $h_t = \sum_{i=1}^{\ell} v_t^i n^i$ and find W_t and R_t from (7) and restore the remaining individual and aggregate variables.

Our next goal is to check if the constructed labor choices are consistent with the individual optimality conditions. To validate the individual choices, we use the decision functions $\frac{p_t^i(\overline{n})}{\Sigma}$, $\frac{p_t^i(\underline{n})}{\Sigma}$, $\frac{p_t^i(0)}{\Sigma}$, $\frac{c_t^i}{w_t^i}$ and μ_t^i to recover the value functions for employed, part-time employed and unemployed agents, V^E, V^{PT} and V^U, respectively, using the appropriately formulated Bellman equations; see Chang and Kim (2007). We then construct the labor choice $\widehat{n}_t^i$ implied by such value functions,

$$\widehat{n}_t^i = \begin{cases} \overline{n} \text{ if } V^E = \max\left\{V^E,\ V^{PT}, V^U\right\}, \\ \underline{n} \text{ if } V^{PT} = \max\left\{V^E,\ V^{FT}, V^U\right\}, \\ 0 \text{ otherwise.} \end{cases} \tag{32}$$

In the solution, the labor choice implied by the value function $\widehat{n}_t^i$ must coincide with the labor choice produced by our decision function n_t^i for all i,t. If this is not the case, we proceed to training of our classifier. To this purpose, we construct the categorical variable $y_t^i \equiv \left(y_t^{i,1}, y_t^{i,2}, y_t^{i,3}\right)$ such that

$$y_t^i = \begin{cases} (1,0,0) \text{ if } \widehat{n}_t^i = \overline{n}, \\ (0,1,0) \text{ if } \widehat{n}_t^i = \underline{n}, \\ (0,0,1) \text{ otherwise.} \end{cases} \tag{33}$$

We then formulate the (log)likelihood function

$$\ln L\left(\theta^{(1)},\theta^{(2)},\theta^{(3)}\right) = \frac{1}{3\ell}\sum_{k=1}^{3}\sum_{i=1}^{\ell}\left[\widehat{y}_t^{i,k}\ln\left(p\left(s_t^i;\theta^{(k)}\right)\right) + \left(1-\widehat{y}_t^{i,k}\right)\ln\left(1-p\left(s_t^i;\theta^{(k)}\right)\right)\right]. \tag{34}$$

We train the model to maximize the likelihood function (34) by using a conventional / stochastic / batch stochastic gradient descent method described in (19) where the gradient can be constructed as in (22). We iterate on the decision functions $p_t^i(\overline{n}), p_t^i(\underline{n}), p_t^i(0)$, $\frac{c_t^i}{w_t^i}$ and μ_t^i until convergence.

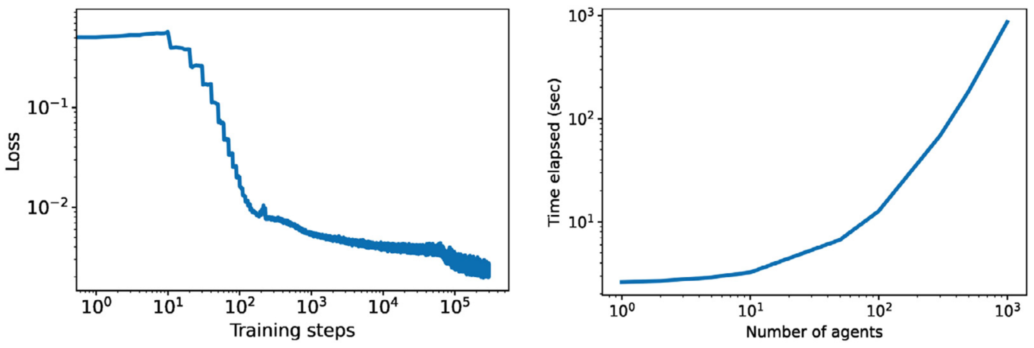

Fig. 10. Training errors and running time for three-state employment model under $\gamma = 1$ and $\eta = 1$.

5.3. *Determining three-state labor: value functions versus "discretized" FOC*

Chang and Kim (2007) consider a related heterogeneous-agent model with three states but they allow for intensive and extensive margins. In contrast, we assume an entirely discrete choice between the three employment states:

$$\widehat{n}_t^i = \begin{cases} \overline{n} \text{ if } L - \left[\frac{c_i^{-\gamma} W_t \exp\left(v_t^i\right)}{B}\right]^{-1/\eta} \geq \overline{n}_f \\ \underline{n} \text{ if } L - \left[\frac{c_i^{-\gamma} W_t \exp\left(v_t^i\right)}{B}\right]^{-1/\eta} \in \left[\overline{n}_p, \overline{n}_f\right] \\ 0 \text{ otherwise} \end{cases} \tag{35}$$

Thus, we assume that the agent chooses full-time employment, $n_t^i = \overline{n}$, whenever her labor choices implied by the FOC of the divisible labor model (11) is above a threshold $\overline{n}_f$; she chooses part-time employment, $n_t^i = \underline{n}$, whenever it belongs to the interval $\left[\overline{n}_p, \overline{n}_f\right]$; and she chooses unemployment whenever it falls below the part-time employment threshold $\overline{n}_p$.

5.4. *Solution method for the model with full- and part-time employment*

Below, we describe the deep learning method for the indivisible labor model with three employment states (the omitted steps coincide with those in Algorithm 1). Again, we allow for two options in the construction of labor choice: one is based on a reconstruction of value functions and the other is based on the discretized FOC.

We calibrate $\overline{n} = 1.1$ and $\underline{n} = 0.55$. Again, these choices together with the interval $\left[\overline{n}_p, \overline{n}_f\right]$ determines how often the agents will be employed and unemployed. We calibrate these parameters at $\overline{n}_p = 0.25$ and $\overline{n}_f = 0.75$ which means that the agents who would choose to work less than 0.25 in the divisible labor model will become unemployed in the indivisible labor model, the agent who would choose the hours worked in the range $[0.25, 0.75]$ would be part time employed and the agent who would choose the hours worked larger than $\overline{n}_f = 0.75$ would be full time employed. We again set the hours of unemployed agent to a small number $n = 0.01$ because $n = 0$ produces numerical issues in training. The resulting parameterization delivers unconditional probability of employment of about 90%, which is roughly consistent with the data.

5.4.1. *Numerical results*

In Fig. 10, we illustrate the training procedure and the time per iteration, and we observe that the tendencies are similar to those in the previously considered models.

In Fig. 11, we show the numerical solution to the model with three employment states produced by the DLC method.

The decision rules in the top row now experience two jumps instead of one jump in Fig. 8 because there are switches, one is between the full and partial employment, and the other is between the partial employment and unemployment (for each of the seven productivity levels). For more productive agents, the switches occur for higher levels of wealth than for less productive agents. This is because the opportunity cost of leisure is higher for more productive agents.

The simulated series for the individual capital and consumption in the bottom row are similar to those in the previous figure but the individual labor switches in two discrete steps between full employment, partial employment and unemployment. Finally, in the bottom row, we show the fluctuations of the aggregate variables (see the thick lines) where we can distinguish how discrete changes in individual labor are transformed into discrete changes in aggregate labor.

6. Assessing the role of heterogeneity and labor choice

We discuss the aggregate and distributional implications of indivisible labor choice in Sections 5.1 and 5.2, respectively.

6.1. *Aggregate implications*

In Table 1, we provide selected business cycle statistics for the studied divisible and indivisible labor economies, as well as for the associated representative-agent model with divisible labor. where the variables y_t, c_t, n_t, h_t, i_t and

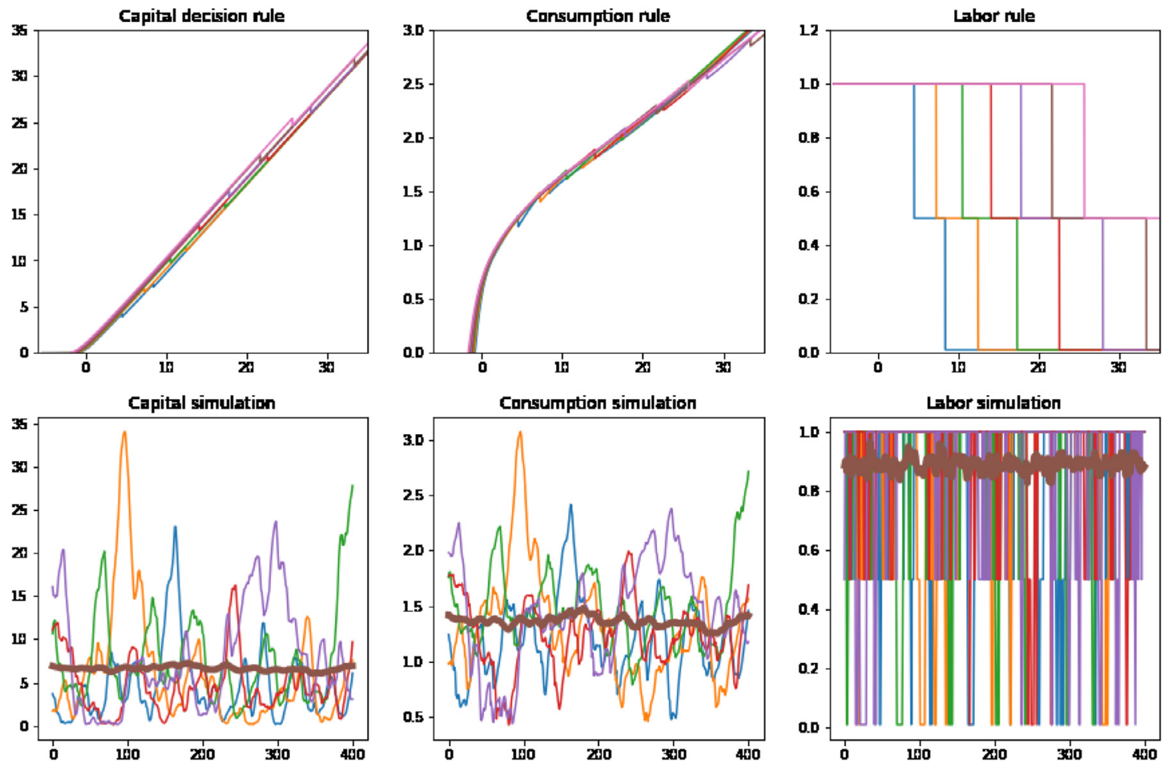

Fig. 11. Solution to the three-state employment model under $\gamma = 1$ and $\eta = 1$.

Table 1
Selected business cycle statistics.

	RA			HA Divisible labor			HA Indivisible labor			HA Full/Part time			US data$^{(*)}$
Elasticity η	$\frac{1}{5}$	1	5	$\frac{1}{5}$	1	5	$\frac{1}{5}$	1	5	$\frac{1}{5}$	1	5	
$std(y)$	0.046	0.044	0.035	0.045	0.033	0.043	0.037	0.040	0.031	0.034	0.039	0.036	0.021
$\frac{std(c)}{std(y)}$	0.784	0.872	0.922	0.725	0.898	0.860	0.880	0.903	0.856	0.913	0.862	0.858	0.45
$\frac{std(n)}{std(y)}$	0.318	0.153	0.044	0.648	0.616	0.143	0.931	0.627	0.078	1.044	0.654	0.225	–
$\frac{std(h)}{std(y)}$	0.318	0.153	0.044	0.769	0.389	0.100	0.685	0.402	0.032	0.761	0.439	0.134	0.82
$\frac{std(i)}{std(y)}$	2.08	1.75	1.67	1.53	1.88	1.72	2.09	1.79	1.83	2.16	1.87	1.73	2.41
$\frac{std(y/n)}{std(y)}$	0.789	0.885	0.960	0.935	1.07	0.953	1.06	1.02	1.00	1.09	0.972	0.975	0.50
$corr(c,y)$	0.931	0.953	0.951	0.778	0.876	0.942	0.829	0.911	0.926	0.807	0.895	0.940	0.69
$corr(n,y)$	0.754	0.783	0.897	0.419	0.193	0.387	0.410	0.283	0.008	0.433	0.369	0.223	–
$corr(h,y)$	0.754	0.783	0.897	0.647	0.377	0.425	0.457	0.321	0.014	0.461	0.367	0.244	0.86
$corr(i,y)$	0.897	0.877	0.836	0.951	0.817	0.886	0.805	0.826	0.866	0.772	0.849	0.886	0.90
$corr(\frac{y}{h},h)$	0.964	0.994	1.00	.779	0.824	0.990	0.584	0.805	0.997	0.500	0.778	0.974	0.08

Note: The statistics are computed across 100 simulations, each of which is length 1000 periods; RA and HA refer to representative- and heterogeneous-agent models; $^{(*)}$ source: Chang and Kim (2007).

y_t/h_t are, respectively, the aggregate output, consumption, labor, efficiency labor, investment and productivity, where $y_t = \exp(z_t)k_t^{\alpha}h_t^{1-\alpha}$, c_t, n_t and h_t are constructed by aggregating the corresponding individual quantities and $i_t = k_{t+1} - k_t(1-d)$.

The business cycle statistics of all studied economies are typical for the real business cycle models. The volatility of output is somewhat higher than that in the US economy because the individual shocks contribute to the volatility of aggregate variables. To adjust for this effect, we report the ratio of volatilities of other variables relative to that of output.

It is well known that the representative-agent divisible-labor model is capable of accounting for stylized features of consumption-saving behavior over the US business cycle, however, has difficulties in reproducing the labor market statistics. In particular, it underpredicts the volatility of labor and overpredicts the correlation between labor and output and that between the labor and wage.

Concerning the first problem, the seminal works of (Hansen, 1985; Rogerson, 1988) showed that the introduction of indivisible labor helps increase the volatility of labor. Under the assumption of complete markets, agents trade employment insurances, and the economy behaves as if the utility is linear in labor, which magnifies labor fluctuations compared to the economy where agents are risk averse with respect to labor choice. Chang et al. (2019) consider the model with intensive

Table 2
Distributional implications of the studied models.

	HA Divisible labor			HA Indivisible labor			HA Full/Part time			US data[(*)]
Elasticity η	$\frac{1}{5}$	1	5	$\frac{1}{5}$	1	5	$\frac{1}{5}$	1	5	
Income distribution										
Top 1%	0.015	0.036	0.032	0.033	0.032	0.029	0.031	0.032	0.030	0.14
Top 20%	0.270	0.398	0.374	0.385	0.368	0.358	0.371	0.371	0.359	0.38
Top 40%	0.520	0.636	0.610	0.630	0.606	0.595	0.612	0.610	0.594	0.62
Bottom 40%	0.254	0.193	0.214	0.193	0.214	0.224	0.205	0.211	0.226	0.19
Bottom 20%	0.082	0.071	0.083	0.068	0.081	0.088	0.072	0.079	0.089	0.08
Gini	0.190	0.327	0.292	0.320	0.288	0.272	0.300	0.294	0.271	0.53
Wealth distribution										
Top 1%	0.013	0.038	0.043	0.042	0.043	0.042	0.037	0.042	0.041	0.28
Top 20%	0.243	0.422	0.461	0.453	0.446	0.447	0.407	0.447	0.445	0.76
Top 40%	0.470	0.666	0.708	0.705	0.691	0.696	0.648	0.690	0.689	0.95
Bottom 40%	0.319	0.169	0.137	0.135	0.148	0.143	0.181	0.149	0.150	0.02
Bottom 20%	0.133	0.059	0.043	0.041	0.047	0.045	0.064	0.048	0.048	−0.01
Gini	0.111	0.365	0.420	0.416	0.401	0.405	0.344	0.400	0.398	0.76

Note: The statistics are computed across 100 simulations, each of which is length 1000 periods; RA and HA refer to representative- and heterogeneous-agent models; [(*)] source: Chang et al. (2019).

and extensive margins and find that this effect is quantitatively important, namely, the volatility of labor ranges between 30% and 60% of the output. Our analysis for the models with indivisible labor led to similar predictions for the benchmark case $\eta = 1$, however for $\eta = 5$, the volatility is excessively low; Chang et al. (2019) do not get so low volatility because they do not consider so high degrees of risk aversion, as we do.

Concerning excessively high correlation of labor variables, we can see in the table that the assumption of indivisible labor reduces the correlation $corr(n, y)$ from 0.7–0.8 to 0.4–0.2 relative to the representative-agent model. A similar reduction is observed for the correlation between efficiency labor and output. The reduction in correlation is due to the assumption of heterogeneity: in the representative-agent economy, a higher wage induces higher labor efforts, whereas in the heterogeneous-agent model, the efforts depend also on the individual productivity and the level of wealth which can offset some of the wage effect; see Maliar and Maliar (2003) for a related discussion.

6.2. Distributional implications

In Table 2, we summarize the distributional statistics produced by the heterogeneous-agent models.

Overall, the distributional implications of the studied models with indivisible labor are similar to those of the heterogeneous-agent model with intensive and extensive margins studied in Chang et al. (2019). A robust distributional implication of this class of models is that they underpredict the degree of inequality relative to the US economy; see, e.g., Aiyagari (1994) for the corresponding statistics on the US economy data. We see that the introduction of indivisible labor helps us mitigate this problem and increase the degrees of inequality, in particular, the share of wealth that belongs to the top one percent of the population. Unlike the divisible labor model, the indivisible labor models are characterized by the degrees of income and wealth inequalities that do not significantly depend on the inverse of elasticity of intertemporal substitution of labor. However, the assumption of indivisible labor alone is not sufficient to produce the empirically relevant degrees of income and wealth inequalities as in the data.

7. Conclusion

This paper shows how to use deep learning classification approach borrowed from data science for modeling discrete choices in dynamic economic models. A combination of the state-of-the-art machine learning techniques makes the proposed method tractable in problems with very high dimensionality – hundreds and even thousands of heterogeneous agents. We investigate just one example – discrete labor choice – but the proposed deep learning classification method has a variety of potential applications such as sovereign default models, models with retirement, and models with indivisible commodities, in particular, housing.

Appendix A

In this appendix, we show sensitivity results with respect to the inverse of intertemporal elasticity of substitution of labor η for the models with divisible labor, indivisible labor model and full- and part-time employment (Figs. A1- A6).

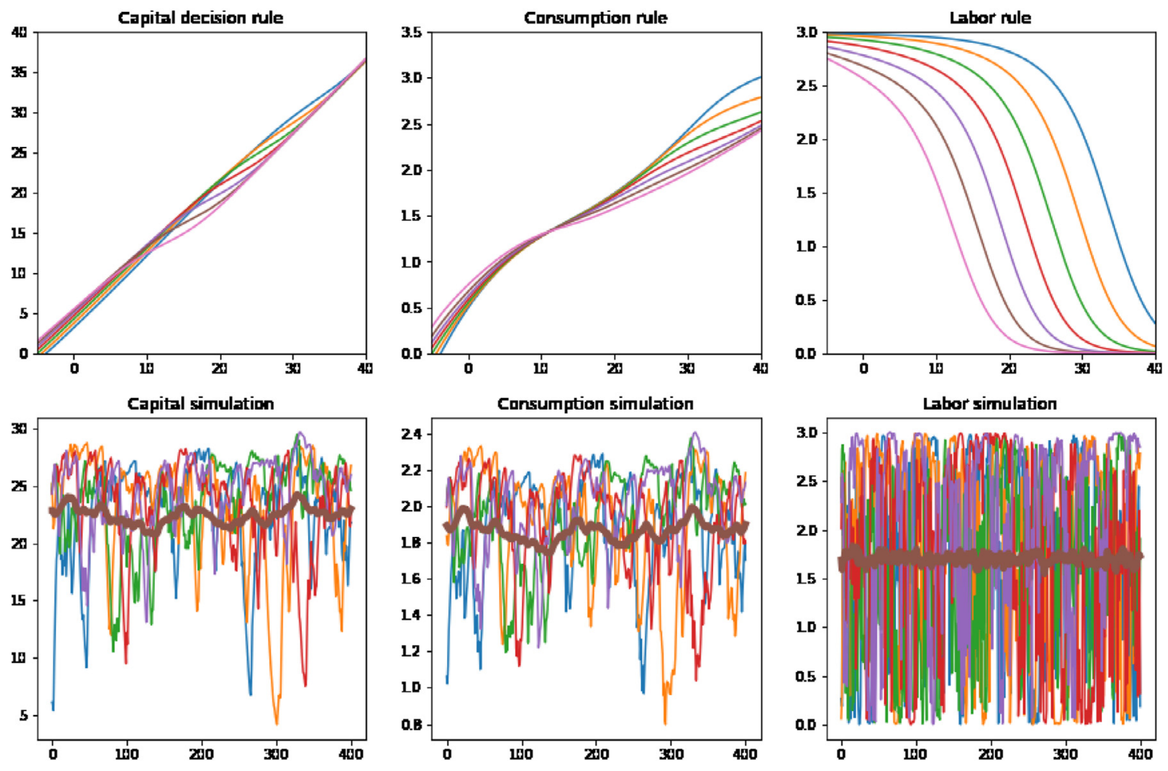

Fig. A1. Solution to divisible labor model under $\gamma = 1$ and $\eta = 1/5$.

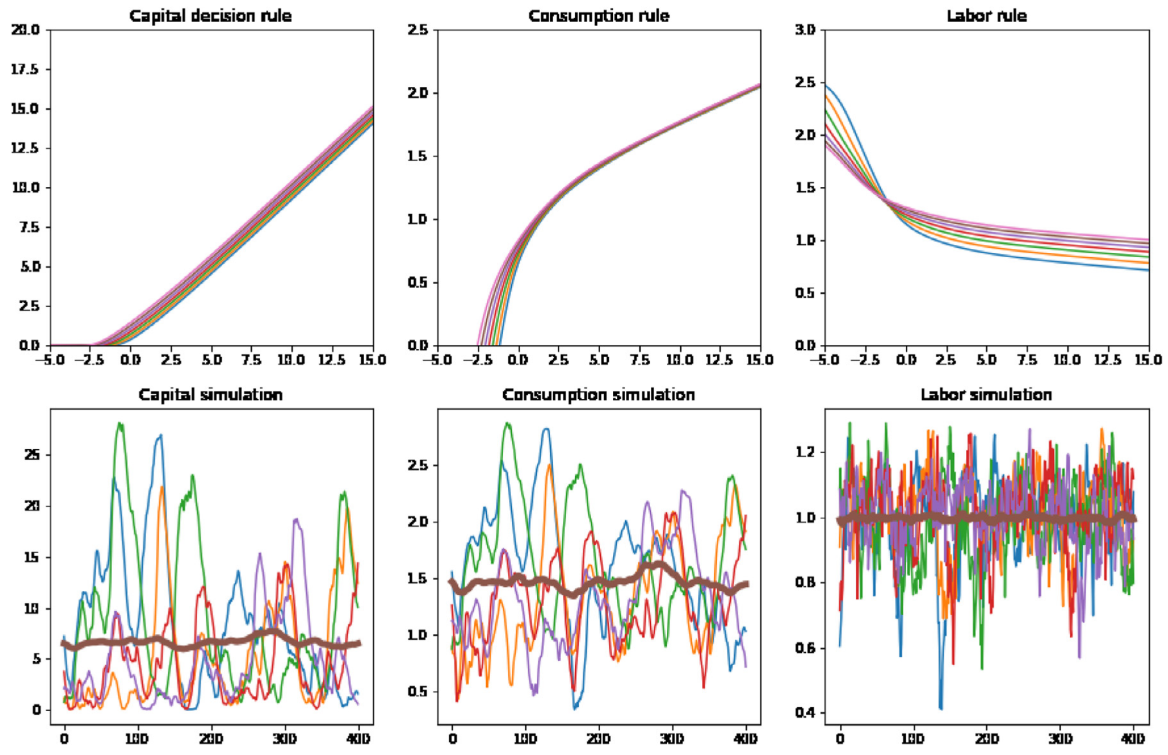

Fig. A2. Solution to divisible labor model under $\gamma = 1$ and $\eta = 5$.

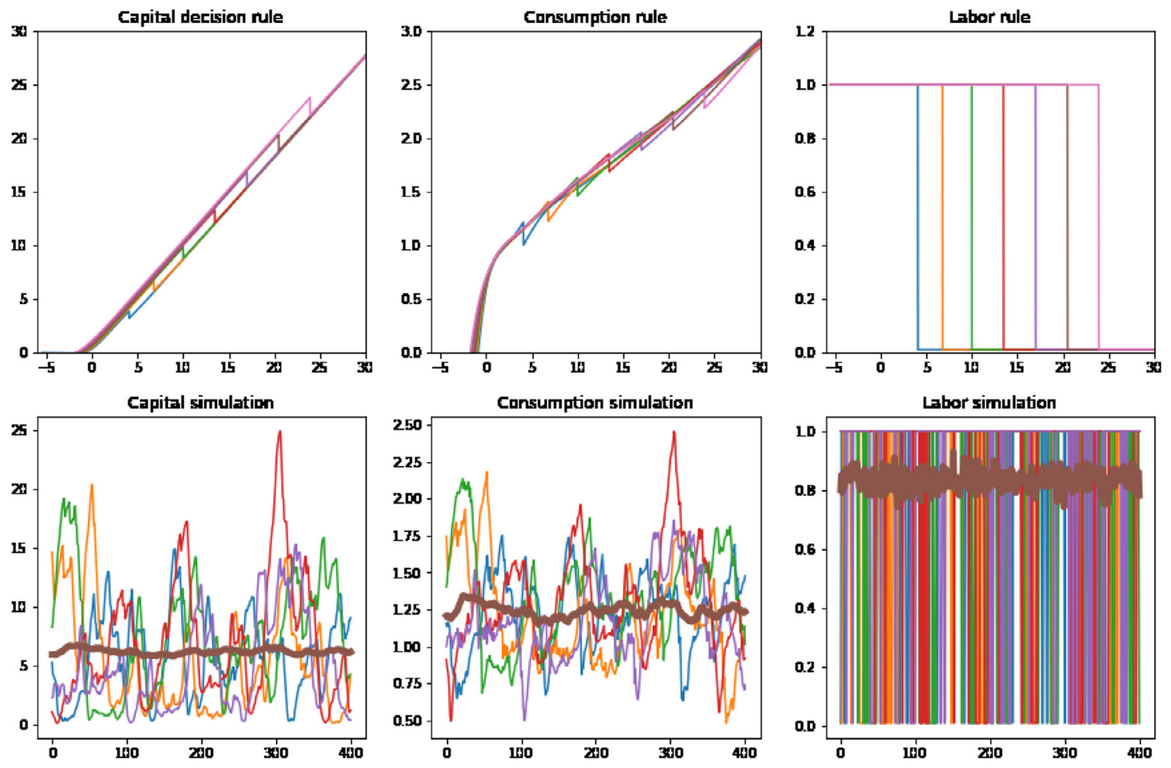

Fig. A3. Solution to indivisible labor model under $\gamma = 1$ and $\eta = 1/5$.

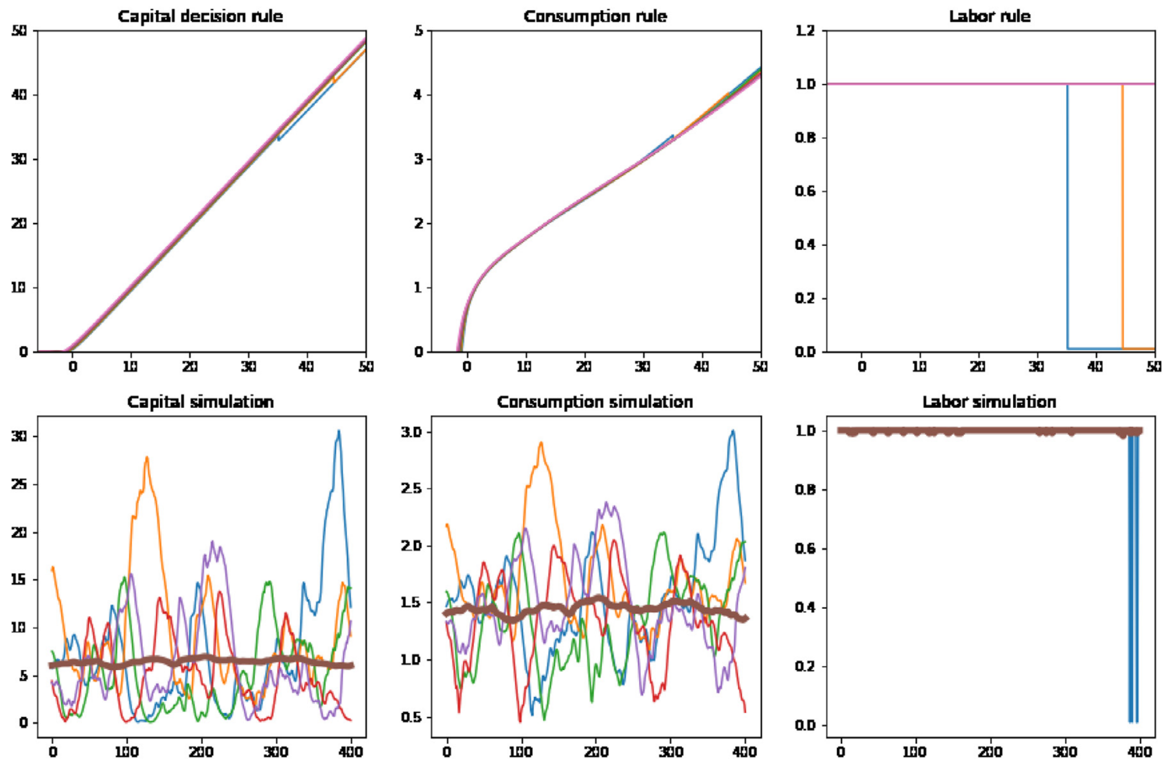

Fig. A4. Solution to indivisible labor model under $\gamma = 1$ and $\eta = 5$.

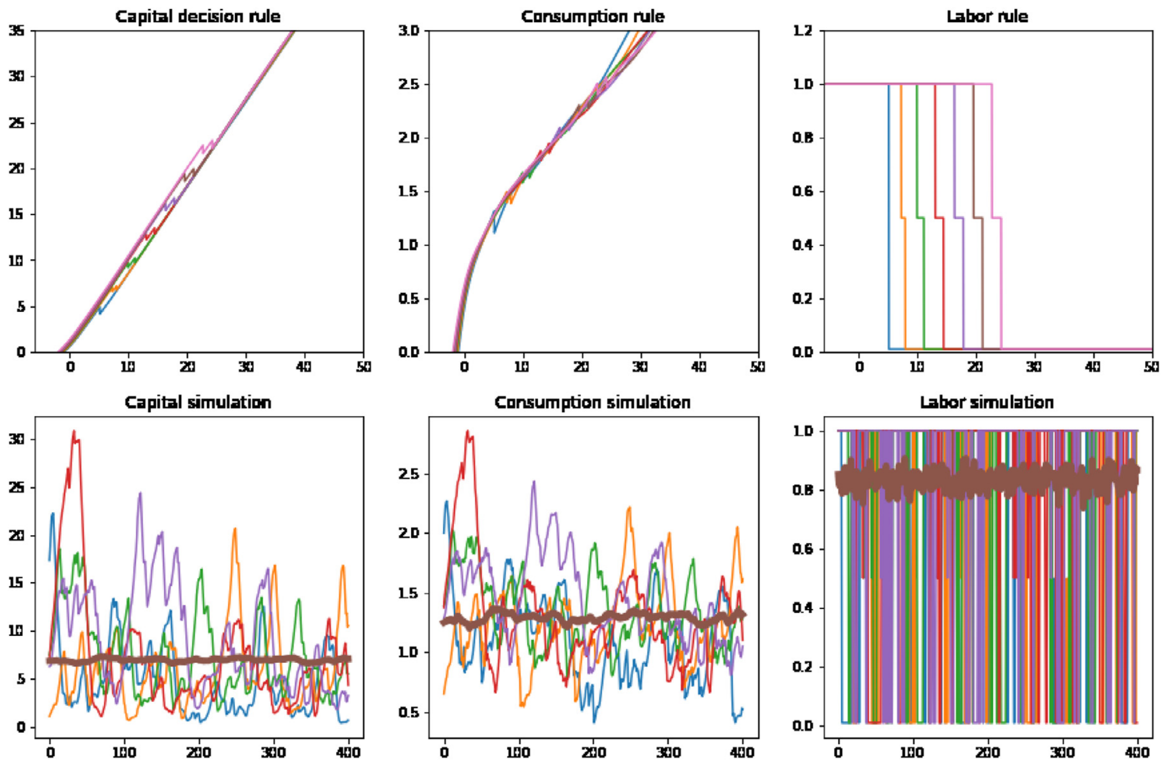

Fig. A5. Solution to the three-state employment model under $\gamma = 1$ and $\eta = 1/5$.

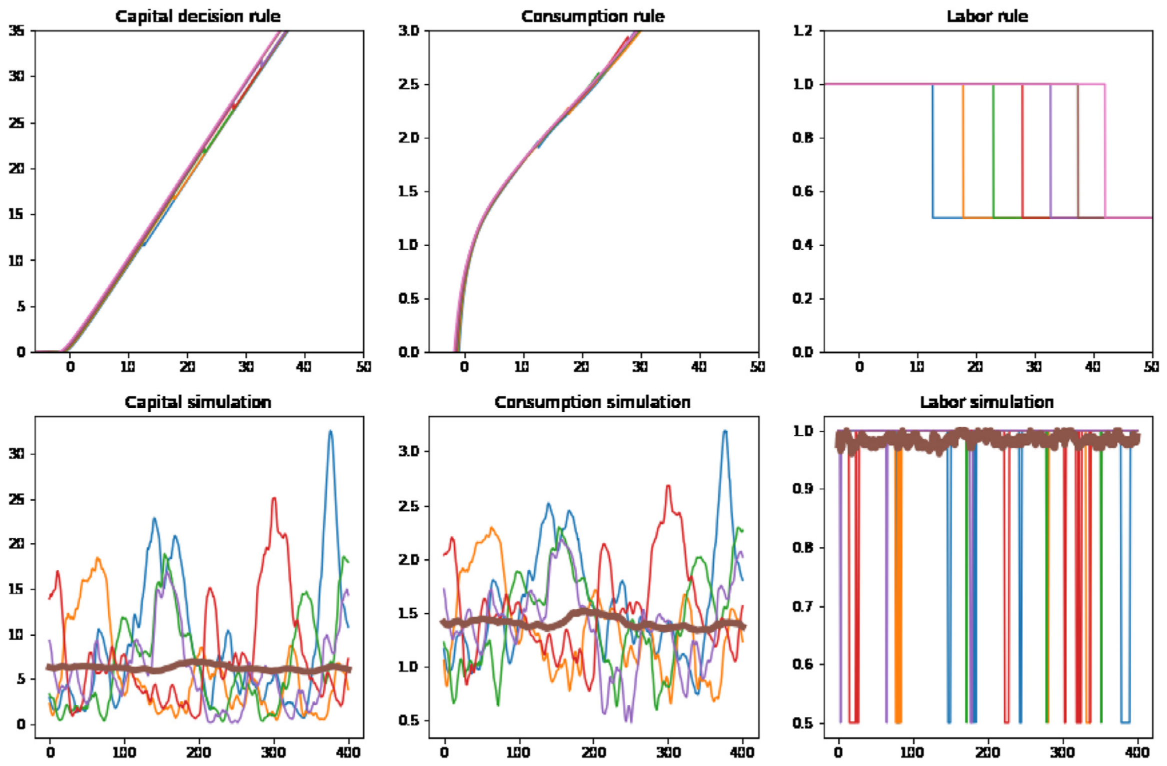

Fig. A6. Solution to the three-state employment model under $\gamma = 1$ and $\eta = 5$.

References

Ahn, S., Kaplan, G., Moll, B., Winberry, T., Wolf, C., 2018. When inequality matters for macro and macro matters for inequality. NBER Macroecon. Annu. 32 (1), 1–75.
Aiyagari, R., 1994. Uninsured idiosyncratic risk and aggregate saving. Q. J. Econ. 109 (3), 659–684.
Arellano, C., 2008. Default risk and income fluctuations in emerging economies. Am. Econ. Rev. 98 (3), 690–712.
Azinovic, M., Luca, G., Scheidegger, S., 2020. Deep equilibrium nets. SSRN: https://ssrn.com/abstract=3393482.
Bayer and Luetticke (2020). Solving discrete time heterogeneous agent models with aggregate risk and many idiosyncratic states by perturbation, Quantitative Economics, 11/4, 1253–1288.
Boppart, T., Krusell, P., Mitman, K., 2018. Exploiting MIT shocks in heterogeneous-agent economies: the impulse response as a numerical derivative. J. Econ. Dyn. Control 89 (C), 68–92.
Carroll, K., 2005. The method of endogenous grid points for solving dynamic stochastic optimal problems. Econ. Lett. 91, 312–320.
Chang, Y., Kim, S., 2007. Heterogeneity and aggregation: implications for labor-market fluctuations. Am. Econ. Rev. 97 (2007), 1939–1956.
Chang, Y., Kim, S.-B., Kwon, K., Rogerson, R., 2019. 2018 Klein lecture: individual and aggregate labor supply in heterogeneous agent economies with intensive and extensive margins. Int. Econ. Rev. 60 (1), 3–24.
Chatterjee, S., Corbae, D., Nakajima, M., Ríos-Rull, J.V., 2007. A quantitative theory of unsecured consumer credit with risk of default. Econometrica 75 (November), 1525–1589.
Childers, D., 2016. On the Solution and Application of Rational Expectations Models with Function-valued States. Meeting Papers vol. 807. Society for Economic Dynamics.
Duarte, V. (2021). Machine Learning for Continuous-Time Finance, manuscript.
Den Haan, W., 2010. Comparison of solutions to the incomplete markets model with aggregate uncertainty. J. Econ. Dyn. Control 34, 4–27.
Fernández-Villaverde, J., Hurtado, S., Nuño, G., 2018. Financial frictions and the wealth distribution. Manuscript.
Goodfellow, I., Bengio, Y., Courville, A., 2016. Deep Learning. Massachusetts Institute Technology Press.
Hansen, G., 1985. Indivisible labor and the business cycle. J. Monet. Econ. 16, 309–328.
Iskhakov, F., Jørgensen, T., Rust, J., Schjerning, B., 2017. The endogenous grid method for discrete-continuous dynamic choice models with (or without) taste shocks. Quant. Econ. 8 (2), 317–365. (lead article)
Iskhakov, F., Keane, M., 2021. Effects of taxes and safety net pensions on life-cycle labor supply, savings and human capital: the case of australia. J. Econom. 223 (2), 401–432.
Judd, K.L., Maliar, L., Maliar, S., 2011. Numerically stable and accurate stochastic simulation approaches for solving dynamic models. Quant. Econ. 2, 173–210.
Krusell, P., Smith, A., 1998. Income and wealth heterogeneity in the macroeconomy. J. Polit. Econ. 106, 868–896.
Lepetyuk, V., Maliar, L., Maliar, S., 2020. When the U.S. catches a cold, canada sneezes: a lower-bound tale told by deep learning. J. Econ. Dyn. Control 117 103926. Forthcoming
Maliar, L., Maliar, S., 2003. The representative consumer in the neoclassical growth model with idiosyncratic shocks. Rev. Econ. Dyn. 6, 362–380.
Maliar, L., Maliar, S., 2005. Parameterized expectations algorithm: how to solve for labor easily. Comput. Econ. 25, 269–274.
Maliar, L., Maliar, S., 2014. Numerical methods for large scale dynamic economic models. In: Schmedders, K., Judd, K. (Eds.), Handbook of Computational Economics, vol. 3. Elsevier Science, Amsterdam, pp. 325–477. Chapter 7
Maliar, L., Maliar, S., Winant, P., 2018. Deep learning for solving dynamic economic models. CEF-2018 presentation, https://www.youtube.com/watch?v=u7_CdytTEe8&feature=youtu.be.
Maliar, L., Maliar, S., Winant, P., 2019. Will Artificial Intelligence Replace Computational Economists Any Time Soon? CEPR Working Paper DP 14024.
Maliar, L., Maliar, S., Winant, P., 2021. Deep learning for solving dynamic economic models. J. Monet. Econ. 122 (C), 76–101.
McKay, A., Reis, R., 2016. The role of automatic stabilizers in the u.s. business cycle. Econometrica 84, 141–194.
Mertens, T. M., Judd, K. L., 2017. Solving an incomplete markets model with a large cross-section of agents. Manuscript.
Powell, W. (2010). Approximate dynamic programming. Wiley, A. John Wiley & Sons.
Prescott, E.C., Rogerson, R., Wallentius, J., 2009. Lifetime aggregate labor supply with endogenous workweek length. Rev. Econ. Dyn. 12, 23–36.
Reiter, M., 2010. Approximate and Almost-exact Aggregation in Dynamic Stochastic Heterogeneous-agent Models. IHS Working Paper 258.
Reiter, M., 2019. Solving heterogeneous agent models with non-convex optimization problems: linearization and beyond. Manuscript.
Rogerson, R., 1988. Indivisible labor, lotteries and equilibrium. J. Monet. Econ. 21, 3–16.
Sutton R. and A. Barto (2018). Reinforcement learning: an introduction. The MIT Press, Cambridge, Massachusetts, London, England.
Train, K. (2009). Discrete choice methods with simulation, Cambridge university press.
Villa, A., Valaitis, V., 2019. Machine learning projection methods for macro-finance models. Manuscript.
Winberry, T., 2018. A method for solving and estimating heterogeneous agent macro models. Quant. Econ. 9 (3), 1123–1151.