Model-Agnostic Dynamic Programming*

Marc de la Barrera† Tim de Silva‡

May 1, 2024

Latest Version

Abstract

Traditional dynamic programming requires a mathematical model of the transition function for the state vector. Leveraging reinforcement learning techniques, we develop a framework to solve dynamic optimization problems that does not require modeling the data-generating process (DGP) of exogenous states. Instead, the method samples realizations of these states directly from the data, allowing the modeler to be "agnostic" about the DGP. We apply our method to a canonical life cycle consumption-saving problem, solving the model without specifying the DGP for income. Using income data from the CPS, we find that the welfare loss from using a standard parametric income process relative to placing no restrictions on the DGP is small. We conclude by verifying that our method achieves a global optimum when given a known DGP and discussing directions for future work.

*We thank Jonathan A. Parker, David Thesmar, and Anthony Lee Zhang for helpful comments and suggestions. A companion `Python` package called `nndp` is available on GitHub and PyPI.

†MIT, Department of Economics, 50 Memorial Drive, Cambridge, MA 02142, mbarrera@mit.edu.

‡MIT Sloan School of Management, 100 Main St, E62-678, Cambridge, MA 02142, tdesilva@mit.edu.

1 Introduction

Many economic models formulate the decision problems of economic agents as dynamic optimization problems (see Stokey et al. 1989 for numerous examples). A dynamic optimization problem is characterized by the following ingredients: (i) a state vector that describes an agent's decision environment in the current period; (ii) a set of actions that an agent takes; (iii) a utility function that specifies the agent's payoff associated with choosing a given set of actions in a given state; and (iv) a law of motion for the state vector (i.e., a transition process), which specifies the probability distribution over states in the next period given the state and actions chosen in the current period.

The traditional approach to solving dynamic optimization problems starts with specifying the law of motion for the exogenous components of the state vector and estimating its parameters. For example, in the canonical life cycle consumption-saving problem of Gourinchas and Parker (2002), there are three states: age, assets, and income. The law of motion for these states is straightforward: age in the next period equals age in the current period plus one; assets in the next period are determined by the consumption-saving choice and the budget constraint; and income in the next period evolves exogenously according to a Markov process. The first step in the traditional approach to solving this model would be to specify a process for income (i.e., the only exogenous state), which is usually a combination of a deterministic component and a stochastic component that follows a normal-AR1 process, and estimate the parameters of this process using income data. After doing so, the traditional approach would then solve this model using standard techniques, such as value or policy function iteration.

This paper proposes a new methodology for solving dynamic programming problems that sidesteps the need to specify a law of motion for exogenous states, which we call Model-Agnostic Dynamic Programming (Agnostic DP, for short). Our motivation for devel-

oping this approach comes from the growing literature on labor income risk, which shows that administrative income data exhibit properties (e.g., excess skewness and kurtosis) that cannot be captured by the traditional income processes (Guvenen et al., 2014, 2021; Braxton et al., 2021). This evidence thus raises the question of what model of income processes should be used when solving dynamic consumption-saving problems (Guvenen et al., 2022).

Motivated by this evidence, the Agnostic DP approach that we develop in this paper does not require specifying the data-generating process (DGP) for exogenous states. Instead, we sample actual realizations of exogenous states (e.g., income in the consumption-savings example) directly and then use reinforcement learning techniques to find the optimal policy function given the unknown DGP for exogenous states. In particular, we follow Duarte et al. (2023) and parameterize the policy function using a deep neural network. We then use a stochastic gradient descent algorithm to solve for the network parameters that maximize the agent's expected lifetime utility. Unlike traditional approaches to solving dynamic programming problems, at no point in this solution process do we need to specify the law of motion for exogenous states, which allows us to remain "agnostic" about the data-generating process. The only requirement is that we have enough realizations of exogenous states from the data in order to train the network (i.e., perform stochastic gradient descent).

In Section 2, we pose a generic (finite-horizon) dynamic programming problem and describe our solution method in full generality. An agent in state s_t takes an action a_t that delivers the agent utility $u(s_t, a_t)$. We partition the state space into exogenous states k_t (e.g., time, income, or productivity) and endogenous states x_t (e.g., assets). We assume throughout that we have sample paths of exogenous states and a distribution for the initial value of the endogenous state. Given these preliminaries, we can then define a transition function of the state, $s_{t+1} = \hat{m}(s_t, a_t, k_{t+1})$, which, given a state, an action, and a future

realization of exogenous states, returns the state at $t+1$. We then show how to parameterize the policy function as a neural network, as in Duarte et al. (2022), and describe the algorithm to solve the model while simultaneously sampling k_{t+1} from data realizations. The innovation of this approach is that it does not require specifying a parametric model for how k_{t+1} relates to s_t.

Section 3 applies our method to a canonical life cycle consumption-saving problem. In this model, an agent lives for T periods and receives an exogenous stream of income. In each period, the agent makes a consumption-savings decision subject to a borrowing limit and an exogenous interest rate. To solve the model, we parametrize the policy function using a deep neural network that the current states as inputs (time, assets, and income) and outputs consumption. Since the agent (like the modeler) does not know the law of motion for income (i.e., the only exogenous state in the model), we solve the model by optimizing the over 1 million parameters of the neural network to maximize expected lifetime utility, given income realizations that are sampled directly from the Current Population Survey (CPS) based on individuals' current income and age.

Having obtained the optimal policy function without specifying the DGP for income, we then ask the following question: what is the welfare loss from imposing a parametric model of the income process? To answer this question, we solve the model using the traditional approach by specifying and estimating a standard parametric income process. We then compare the expected lifetime utility of an agent that uses this "Classic" policy function, which has an incorrect representation of the DGP for income, with that of an agent that uses the Agnostic DP policy function from our method. Surprisingly, we find that these two values are virtually identical, implying that the cost of assuming a functional form for income is negligible.

One possibility for this negligible difference is that Agnostic DP does not achieve the global optimum. We rule out this possibility by comparing our method with the classical

approach in a model in which the DGP is known. This is a useful exercise because we know that the classical approach delivers the optimal policy function when the DGP is known. However, in this case, we find that Agnostic DP is able to replicate the optimal policy function and achieve the same value function as Classic, which illustrates the reliability of our solution method.

Although the benefits of being "agnostic" about the data-generating process are small in our specific application, there are two notable benefits of Agnostic DP relative to the Classic approach. First, Agnostic DP is much simpler to code. Second, and more importantly, the Agnostic DP algorithm does not change as the number of states and actions increases, while the "curse of dimensionality" limits the Classic approach from solving more complex models. In principle, our Agnostic DP algorithm can handle as many states and actions as required, but more attention needs to be put into the architecture of the neural network as the model becomes more complex.

We conclude by discussing plans for future work. One reason why the welfare loss from assuming a functional form for income would be low is because we are using survey rather than administrative data; in future work, we plan to implement our approach using administrative earnings data. Two other directions for future work include generalizing the method to allow for infinite horizon problems and general equilibrium.

Related literature This paper adds to a nascent literature that develops methods to solve dynamic stochastic economic models using machine learning techniques (see, e.g., Duarte et al. 2022; Duarte et al. 2023; Scheidegger and Bilionis 2019; Fernández-Villaverde et al. 2023; Azinovic et al. 2022). Closest to our paper is Duarte et al. (2022), who introduce the deep reinforcement learning method for solving finite-horizon dynamic stochastic programming problems. They use this method to solve a rich model of life cycle portfolio choice, which includes many ingredients only modeled in isolation in prior work. How-

ever, their solution method is also "classical" in the sense that they need to specify the DGP for exogenous states in the model. In contrast, the contribution of our paper is to show how to leverage their solution technique to be agnostic about this DGP.

This paper is also part of an extensive literature in macroeconomics and household finance that studies household consumption, savings, and portfolio choices using life cycle models. The standard approach in this literature is to formulate a dynamic stochastic model of household behavior over the life cycle and then estimate it in two stages (e.g., Gourinchas and Parker, 2002; Catherine, 2022). In the first stage, the parameters that govern the DGP for exogenous states (e.g., income) are carefully estimated; in the second stage, any remaining parameters are estimated using indirect inference. In contrast to the first stage of this approach, our solution approach does not take a stand on the DGP for exogenous states. Our current application of this approach works with a simpler model than most of this literature, but in future work, we plan to explore its use in more complex models.

2 Model-Agnostic Approach

2.1 Framework

2.1.1 Setup

We are interested in solving a canonical dynamic programming as in Stokey et al. (1989) of the form

$$V(s_0) = \max_{\{a_t \in \Gamma(s_t)\}_{t=0}^{T}} \mathbb{E}_0 \left[\sum_{t=0}^{T} u(s_t, a_t) \middle| s_0 \right] \qquad \text{subject to}$$
$$s_{t+1} = m(s_t, a_t, \epsilon_t). \tag{1}$$

Over T periods, the agent takes *actions* a_t to maximize utility $u(s_t, a_t)$, which depends on the action taken and the *state* s_t. By convention, time t is the first element of s_t, which implies that $u(s_t, a_t)$ can depend on t in any general form. We partition the state into two elements, $s_t = (k_t, x_t)$, where k_t are *exogenous* states and x_t are *endogenous* states. The action space is constrained by $\Gamma(s_t)$, which defines the set of possible values that a_t can take, given s_t. The law of motion of the state is governed by the function $m(s_t, a_t, \epsilon_t)$, where ϵ_t is a random variable and makes the problem stochastic.

The goal is to find a policy function $a_t = \pi(s_t)$, which specifies which action to take given a current state s_t. Unless this policy can be solved analytically, it is common to parametrize this policy with some vector parameter θ and approximate π by $\hat{\pi}(s_t, \theta)$. θ can be the parameters of an interpolator or the weights in a neural network. We will consider this later case since neural networks are universal approximators and can handle many states easily. We now can define $\tilde{V}(s, \theta; \hat{\pi})$ as the expected value of using the policy function $\hat{\pi}(s, \theta)$ as

$$\tilde{V}(s_0, \theta; \hat{\pi}) = \mathbb{E}_0 \left[\sum_{t=0}^{T} u(s_t, \hat{\pi}(s_t, \theta)) \middle| s_0 \right] \qquad \text{subject to}$$
$$s_{t+1} = m(s_t, \hat{\pi}(s_t, \theta), \epsilon_t). \tag{2}$$

$\tilde{V}(s_0, \theta; \hat{\pi})$ is the expected value the agent gets when it starts at s_0, and follows the policy

$\hat{\pi}$ parametrized by θ. Through the paper, we will take $\hat{\pi}$ as given and will be looking for θ. The optimal choice for $\hat{\pi}$ is left for further research, and we will use a multi-layer neural network with a fixed number of layers. In order to get rid of the expectation term, we define $\tilde{V}(s_0, \theta; \hat{\pi}, \{\epsilon_t\}_{t=0}^T)$ as the value of the policy $\hat{\pi}(s_t, \theta)$ when the initial state is s_0 for a particular shock realization $\{\epsilon_t\}_{t=0}^T$,

$$\tilde{V}(s_0, \theta; \hat{\pi}, \{\epsilon_t\}_{t=0}^T) = \sum_{t=0}^{T} u(s_t, \hat{\pi}(s_t, \theta)) \quad \text{subject to} \quad s_{t+1} = m(s_t, \hat{\pi}(s_t, \theta), \epsilon_t)$$

Traditional dynamic programming like (1) solve for a function $V(s_0)$ that gives the value *for every* s_0. However, once we are solving a problem for a given policy function, defined by $(\hat{\pi}, \theta)$, we need a way to compare different policy functions that do not depend on the initial state s_0. Thus, we assume the initial state s_0 comes from a distribution $F(\cdot)$, and the problem we will be solving is to find the parameters θ that maximize

$$\bar{V}(\hat{\pi}) = \max_{\theta} \mathbb{E}[\tilde{V}(s_0, \theta; \hat{\pi})], \tag{3}$$

where the expectation is taken with respect to s_0.

In order to solve (3), we will leverage advances in computer science and machine learning that allow taking the gradient of $\tilde{V}(s_0, \theta; \hat{\pi})$ with respect to θ, which we denote by $\nabla_\theta \tilde{V}(s_0, \theta; \hat{\pi})$. We will replace expectations by simulations of s_0 and paths for ϵ_t. The central contribution of this paper is how we think about the law of motion m.

2.1.2 Constraining the Action Space

The policy function will be defined by a neural network, which takes as input the state s_t and returns some actions a_t. These actions are constrained by $\Gamma(s_t)$, so we need to make sure that the neural network outputs a feasible action. For this, we have to determine the

last activation layer according to the constraints of the problem and do a transformation that depends on the state. Suppose the action space is a scalar, and let $\tilde{a}(s_t)$ be the output of the neural network before we make sure it satisfies $a_t \in \Gamma(s_t)$. Then we can be in on of the following cases.

When the action space is bounded below and above, so $a_t \in [\underline{b}(s_t), \bar{b}(s_t)]$, then the neural network should output a probability $\tilde{a}(s_t) \in [0, 1]$ and then transform the action to $a(s_t) = \underline{b}(s_t) + (\bar{b}(s_t) = \underline{b}(s_t))\tilde{a}$. For that, using a sigmoid/logistic activation function is a good choice. An example of this constraint is consumption, which has to be positive and is bounded above by some value determined by assets, income, and borrowing constraints. The neural network then outputs the share of the total available consumption that is actually consumed.

Another possibility is that the action space is one-sided bounded like $a_t \geq \underline{b}(s_t)$. In such case, using a ReLU activation function, which is bounded below by zero, and setting $a(s_t) = \underline{b}(s_t) + \tilde{a}(s_t)$ would achieve the desired result. Similarly if $a_t \leq \bar{b}(s_t)$, set $a(s_t) = \bar{b}(s_t) - \tilde{a}(s_t)$. This constraint is natural in many models of labor supply where labor is not bounded above but required to be positive.

Finally, if the action is unconstrained and $a_t \in \mathbb{R}$, then any unbounded activation function suffices to satisfy $a_t \in \Gamma(s_t)$. In particular, the linear activation function is a good choice for that case.

The previous three cases considered a scalar action, but in most models, the action space is a vector, with components a_{it}. In such case, it is not always the case that we can constrain a_{ti} using the information of the state only because it depends on other actions a_{tj}. A clear example is a life cycle model with endogenous labor supply: how much an agent is allowed to consume depends on how much labor the agent supplies. We can still ensure that $a_t \in \Gamma(s_t)$, noting that labor constraints are independent of consumption, and

given labor, we can constrain consumption.

2.1.3 The Law of Motion

The focus of this paper is on the law of motion of the state m. Traditional solution methods of the problem (1) assume a functional form for m and use it to take first-order conditions and compute expectations. The proposed methodology does not require such tractability and instead treats m like a *black-box*: given a state s_t and an action a_t, returns a new state. ϵ_t shocks make this mapping stochastic. The agent does not need to *know* m, but rather it *learns* it. All we need to provide is a function that samples s_{t+1} given s_t and a_t.

If the modeler knows m, it is an input of the model. This is how most economic models start: by defining a law of motion of the state that will be estimated/calibrated. We have developed a `Python` package `nndp`, that solves problem (1) given m, u and Γ. The advantage of using neural networks is that handling more states is straightforward, given that neural networks are meant to work with many inputs and provide many outputs. We use automatic differentiation and just-in-time compilation to speed up the convergence process by leveraging `jax`.

The most interesting case is when the modeler does not know m or wants to remain agnostic about the data-generating process. The next section discusses this case.

2.2 Solution Algorithm

The standard way of solving economic models is to (i) use data to estimate a process for the law of motion of the state and (ii) solve a traditional dynamic problem model given the estimated law of motion. Our methodology proposes to bypass step (i) and instead solve the dynamic problem while feeding real data realizations. This way, the modeler remains agnostic about the data-generating process and does not constrain it at all.

We have partitioned the state s_t into exogenous states k_t (time, income, interest rates...) and endogenous states x_t (assets). Given s_0, we can solve (1) if one is able to get enough path realizations. We define the value function of following a policy $\hat{\pi}$, parametrized by θ, given a future process for exogenous states $\{k_t\}_{t=0}^T$ and initial endogenous state x_0 as

$$\begin{aligned} \hat{V}(x_0, \theta, \{k_t\}_{t=0}^T; \hat{\pi}) = \sum_{t=0}^{T} u(s_t, \hat{\pi}(s_t, \theta)) \quad \text{subject to} \\ s_{t+1} = \hat{m}(s_t, \hat{\pi}(s_t, \theta), k_{t+1}) \\ s_0 = (k_0, x_0). \end{aligned} \tag{4}$$

We have not included the constraint $a_t \in \Gamma(s_t)$ because we assume that the policy function $\hat{\pi}(s_t, \theta)$ already outputs feasible actions, as discussed in the previous section. Note that (4) is fully deterministic and differentiable with respect to θ. Algorithm 1 shows how we can find the parameter θ that maximizes (4) using stochastic gradient descent techniques.

Algorithm 1: Agnostic DP

Primitives: reward function $u(s_t, a_t)$, transition function $\hat{m}(s_t, a_t, k_{t+1})$, constraints $a_t \in \Gamma(s_t)$, and T.

Data: M paths with x_{i0} and $\{k_{it}\}_{t=0}^T$ indexed by i

Define a neural network architecture $\hat{\pi}$ that satisfies $a_t \in \Gamma(s_t)$ parameterized by θ;

Initialize the policy function parameters θ^0;

for $n < N$ **do**

  Sample $m < M$ paths, and let $\mathcal{I}^n$ be the set of paths selected. Compute the gradient $g(\theta^n) = \nabla_\theta \frac{1}{m} \sum_{i \in \mathcal{I}^n} \hat{V}(s_{i0}, \theta^n, \{k_{it}\}_{t=0}^T; \hat{\pi})$;

  Update $\theta^{n+1} = update(\theta^n, g(\theta^n))$

end

3 Application to Life Cycle Consumption-Saving Problem

In this section, we show an application of our method to a standard consumption-savings problem widely used in economics and household finance. It consists of a household with some wealth w_t that lives for T periods, receives an exogenous stream of income y_t, and can save at a constant interest rate R. The only decision that the household makes is how much to consume, so $a_t = c_t$. The value function of an agent at state $s_t = (t, y_t, w_t)$ is

$$\begin{aligned} V(t, y_t, w_t) = \max_{\{c_{t+k}\}_{k=t}^{T-1}} \mathbb{E}_t \left[\sum_{k=t}^{T-1} \beta^t \frac{c_{t+k}^{1-\gamma}}{1-\gamma} + \beta^T b(w_T) \middle| y_t \right] \\ w_{t+1} = R\left(w_t + y_t - c_t\right) \\ w_t \geq 0 \quad c_t \geq 0. \end{aligned} \tag{5}$$

We have added a bequest motive at the time of death T. We adopt De Nardi et al. (2016) functional form $b(w_T) = b_0 \frac{(w_T + b_1)^{1-\gamma}}{1-\gamma}$, in part to showcase how utility can be time-invariant once t is part of the state space. b_0 is the intensity of the bequest motive, while b_1 determines the curvature of the bequest function and, hence, the extent to which bequests are luxury goods. In the notation of (1), the reward function is given by

$$u(s_t, a_t) = \begin{cases} \beta^t \frac{c_t^{1-\gamma}}{1-\gamma} & \text{if } t < T \\ \beta^T b_0 \frac{(w_t + b_1)^{1-\gamma}}{1-\gamma} & \text{if } t = T \end{cases}.$$

For simplicity, we assume that wealth has to be positive, allowing for positive borrowing is straightforward. We partition the state into exogenous states $k_t = (t, y_t)$ and endogenous states $x_t = w_t$. Problem (5) is missing the law of motion of y_t, which we have not specified. Suppose instead that we have a panel where we observe the income realizations of individuals over their lifetime. In the example provided, we have assumed that

each household lives exactly for T periods, the appendix describes how the model could be enriched with (agnostic) random death. With that dataset, we could use Algorithm 1 to solve for $V(t, y_t, w_t)$ without making any assumption on the income process y_t. Next, we propose a method to generate that panel drawing from real income realizations.

3.1 Generating a Sampler Dataset

Not making any assumption about the data-generating process of the exogenous state comes at a cost: we need to observe many paths of income to learn its process to avoid overfitting. The universe of US tax returns (e.g., Guvenen et al., 2014) would be ideal for our solution method. Unfortunately, we don't have access to these data, so we use an alternative method to generate sample paths from real data realizations.

3.1.1 Income Paths

The CPS is a monthly U.S. labor force survey covering the period 1962 to the present and gathers information on over 65,000 households. It is the main survey to compute unemployment statistics, and it also contains information on education, labor force status, demographics, and other aspects of the U.S. population. The CPS interviews sample members eight times. Respondents are interviewed for four consecutive months, are rotated out for eight months, and then are included in the sample for another four months. In March, the Annual Social and Economic (ASEC) Supplement provides the usual monthly labor force data, but in addition, provides supplemental data on work experience, income, noncash benefits, and migration.

In particular, the ASEC Supplement has information about total income at the individual level and has a panel structure. In a given March, around 50,000 households are surveyed, and around 45,000 individuals are followed from year t to $t + 1$. We consider individuals aged from 25 to 65 to define an income path of $T = 40$ years. We deflate income using the

cpi deflator and round it to the nearest multiple of $\underline{y} = \$2,500$, and call this variable y_{it}. We define t as the number of years since the person turned 25.

In order to obtain a complete path of length T, we sample from realized transitions from k_t to k_{t+1}. We start by sampling an individual aged 25 and we observe its income y_{i0} and y_{i1}. Then we sample one observation from all individuals aged 26 with income $y_{i'1} = y_{i1}$, for whom we also observe $y_{i'2}$. Any sample is done using ASEC weights. Rounding income to the nearest multiple of $\underline{y}$ allows for a higher likelihood to find at least one observation with $y_{i'1} = y_{i1}$. With that, we generate a path of length 3 consisting of $y_{i0}, y_{i1} = y_{i'1}, y_{i'2}$. We iterate further by sampling individuals aged 27 and with income $y_{i''2} = y_{i'2}$, to generate an observation for $y_{i''3}$. Once we reach $T = 40$, we consider we have generated a new income path. It might be the case that given $k_t = (t, y_t)$, we are unable to find a match to iterate forward. If that happens, we start the path from the beginning. This algorithm allows us to generate as many balanced paths as required to train and evaluate the model. Figure 2 shows 5 paths of income sampled using this procedure. We see some persistence and, at the same time, big jumps. Our methodology ensures that every pair y_{it}, y_{it+1} has been observed in the CPS.

The algorithm generates paths where the transitions y_t, y_{t+1} are data realizations. We are assuming that only age and income are determinants of future income, a simplification that is also shared by many other models, like the one we will use in Section 3.4. We can represent any income process of the form $y_{t+1} = f(t, y_t, \epsilon_t)$, with ϵ_t being an idiosyncratic shock with any distribution. The sampling mechanism can be generalized if we condition the matching on more observable states. We could add gender in the state space k_t or industry group. The obvious caveat of expanding the state space is that it becomes harder to find matches given the CPS sample size. There are many individuals aged 35 with $50,000 income from whom we observe income at 36 (and thus we can sample one realization) than women aged 35 with $50,000 income and working in accounting.

3.1.2 Distribution of Initial Assets

In addition to income paths, we require knowledge of the distribution of initial assets of individuals at age 25. We obtain those from the Survey of Consumer Finances, which is a triennial statistical survey of the balance sheet, pension, income, and other demographic characteristics of families in the United States. We pool all years from 1989 to 2022 and consider the net worth of observations aged 25-30. After filtering, 21,689 observations remain. Since we do not consider borrowing in our model, we set any observation with negative net worth to 0, which is 21% of the sample, and winsorize at 99%. Median assets are $100,308, and mean assets are $500,286, consistent with a fat tail distribution. Figure 1 shows the distribution of wealth.

3.1.3 Sampling Paths of States

With data on income and initial assets, we can sample initial states by sampling one income realization of somebody aged 25 from the CPS and the initial assets of somebody aged 25-30. A caveat of this approach is that we can't jointly sample income and assets because they come from different datasets. It is worth noting, though, that the distribution of initial states s_0 is important insofar as we want the neural network to optimize the policy around states that are more likely to be realized. In this sense, not having a correlation between income and assets helps to sample from a more dispersed distribution. We visit more states at the cost of visiting less frequently states that are more representative of an individual at 25.

What is relevant is that now we can generate as many realizations $(x_{i0}, \{k_{it}\}_{t=0}^{T})$ as we want, which are needed to solve our problem. Given an agent at state $s_{it} = (t_{it}, y_{it}, b_{it})$, an action $a_t = c_t$, we can define the function $m(s_t, a_t, k_{t+1})$, which returns s_{t+1}. The modeler specifies the law of motion for wealth but is agnostic about the process of y_{t+1}, which

comes from k_{t+1}.

3.2 Solution Method

We parametrize the policy function with a deep neural network $\hat{\pi}(s_t, \theta)$ that takes a 3-dimensional input and the parameters we optimize over and returns a 1-dimensional output bounded from 0 to 1. The output represents the share of cash-on-hand that the household consumes.[1] We build a network with $n_{layers} = 5$ hidden layers and each layer has $n_{nodes} = 500$ nodes, totalling 1,255,001 parameters. Each internal layer has a $tanh$ activation function, which generates an output from -1 to 1, and the output activation function is a sigmoid, which generates a value from 0 to 1, which then is converted to total consumption. Figure 3 shows a graphical representation of the network.

We train the network with $N = 750$ epochs, and in each epoch we sample $m = 200$ paths $(x_{i0}, \{k_{it}\}_{t=0}^{T})$ as described in Section 3.1.3. Sampling more paths allows for a better calculation of expected values. Gradients are computed using `optax`, which is a gradient processing and optimization library for JAX (Bradbury et al. (2018)), and we use Adam optimization routine (Kingma and Ba (2015)) with a learning rate of 10^{-4} . Optimization of hyperparameters and neural network structure is out of the scope of this paper, but there is room for making the network lighter and, therefore, faster.

3.3 Results

Figure 4 shows the convergence of the value function as the neural network is trained, and the value function at the test dataset, that is, $E[\hat{V}(s_0, \theta, \{k_t\}_{t=0}^{T}; \hat{\pi}]$ across realizations of s_0 and $\{k_t\}_{t=0}^{T}$. Most of the training gains are achieved after 200 epochs, and each epoch has 200 paths, which means that the neural networks need to observe 40,000 paths of income

[1] With a positive borrowing limit, it would represent the share of available cash-on-hand.

to learn its process. Figure 4 shows the value function of an agent at age 25 that starts with 0 assets as a function of income. As expected, is increasing and convex. The function is not completely smooth because expected values are computed by sampling from income realizations. Since in the CPS, there are not many observations aged 25 earning more than $100,000, we suffer from small sample problems at this high-income realizations.

In order to understand the policy function generated by the neural network, Figure 6 shows it of an agent with 0 assets as a function of income and time. The first panel shows the policy at age 25 ($t = 0$), and the second panel shows it at age 65 ($t = T = 40$). The figure also adds the income realization distribution at those ages since the neural network optimizes better states that are visited often. As expected, a young agent with low income behaves as a hand-to-mouth, consuming the majority of it. At age 65, the agent consumes a larger fraction of their income, although those with income higher than $40,000 save some as a bequest motive.

Another way to evaluate the policy function is to plot, for an agent aged 25 with median income $\bar{y}$, which is $25,000, as assets increase. Cash on hand is $a_0 + \bar{y}$ and the green line represents consumption. The point where the solid blue line a_1 crosses the 45-degree line represents the dissaving threshold. Agents with less than $50,000 are closer to the borrowing limit and therefore save for the future, while wealthier agents dissave to increase present consumption.

3.4 The Cost of a Parameter Model for Income

In this section, we solve the model with standard techniques, which we call it "classical" method, to assess the cost of imposing a functional form for the income process. The exercise is to (i) fit an income process for y_t, (ii) solve the income-fluctuations process with classical methods, and (iii) evaluate the policy function with real data realizations.

That is, we solve a model assuming a particular data-generating process for y_t but then we evaluate it with observational data.

To this end, we assume that $\log y_t$ is the sum of a deterministic component $f(t)$ that is a function of age and a disturbance η_{it} that follows an AR(1) process with persistence ρ.

$$\log y_{it} = f(t) + \eta_{it} \tag{6}$$

$$\eta_{it} = \rho\eta_{it-1} + \epsilon_{it}, \qquad \epsilon_{it} \sim N(0, \sigma) \tag{7}$$

As is standard, we fit the deterministic component of income to a polynomial of second degree

$$f(t) = \delta_0 + \delta_1 t + \delta_2 t^2. \tag{8}$$

The model is solved using backward iteration, which is robust and ensures that we reach a global optimum given the convexity of u. We discretize a grid of assets and income using 200 points for each dimension and approximate the normal shocks using Gauss-Hermite quadrature using 11 points to take expectations.

Table 1 shows the estimated parameter values, and Figure 10 shows the age profile, which is an inverted U, as expected. We obtain a standard deviation of the income shocks of 0.71, which is higher than the standard estimation of income processes. This is primarily due to the inclusion of zeros in the dataset to estimate the AR1. To explain big drops in income, a high volatility of the idiosyncratic shocks is required.

By solving the model, we obtain $\pi^{cl}_{AR1}(t, y_t, a_t)$, which is the policy function computed using the classic method when the data generating process for income is $AR1$. We can compare it with $\pi^{ag}_{AG}(t, y_t, a_t)$ which is the solution derived in Section 3.3, which is the policy function of the neural network when it has been trained in agnostic (model-free)

data.

Figure 8 compares these two policy functions for randomly generated states at $t = 0$. Policies are fairly similar except for high values of consumption. At above \$30,000, the agnostic policy consumes more than the policy policy that assumes a functional process for income. Note that there is no reason to expect both policies to coincide because expectations of future income differ in both cases.

We want to quantify how much we are missing by specifying an income process versus letting the neural network learn it. Figure 9 shows the value of following each policy when they are evaluated using data realizations. Both policies derive fairly the same value, with insignificant differences between one and the other policy. This means that the cost of assuming a fairly simple income process is negligible.

There are two potential explanations for this unexpected result. One option is that the income process is indeed represented well by a quadratic polynomial and an AR1 residual. This is unlikely to be the case given that the deterministic trend regression has an adjusted R^2 of 0.012, while the adjusted R^2 of the regression (7) is 0.5. An alternative is that the income process is mostly driven by the idiosyncratic shocks. In this case, the benefit of learning the true DGP of income is low, given the high noise-to-info ratio of the process.

3.5 Validation of Agnostic DP Solution Algorithm

In Section 3.3, we were agnostic about the true DGP of the process and, therefore, unsure if our methodology resulted in an optimal policy. In this section, we test whether Agnostic DP can achieve the same results as classical methods when trained with data for which we know its DGP.

To this end, we simulate data from the income process that we fitted in the previous

section and use it as the input for Algorithm 1. In this case, we do know the optimal value and policy function up to the interpolation error induced in the classic solution method. Figure 11 shows how the agnostic method virtually achieves the global optimum in every state except those with bad realizations. This is normal since these are states that are less frequently visited by the neural network.

Overall, we can conclude that, at least for this simple model, training a model on agnostic data is as effective as solving a model with traditional methods that assume a functional form for the exogenous states.

4 Conclusion

In this paper, we propose a new approach for solving dynamic programming problems that does not require specifying the data-generating process for exogenous states, which we call Agnostic DP. Instead of using data to estimate a process and then solving the model in the standard way, we directly feed data realizations into the solution method, sidestepping the estimation part. Our method is more general because it allows for *any* underlying data generating process, at the cost of requiring more observations to train the model and avoid overfitting.

A clear advantage of using neural networks and gradient-based algorithms is that those naturally handle many states. We provide a `Python` package where the solution method is independent of the problem that we aim to solve. The researcher must specify a reward function $u(s_t, a_t)$, a law of motion $m(s_t, a_t, k_{t+1})$, and a neural network that takes states s_t as an input and outputs a_t, and the code does not change with the dimensionality of the states or actions.

We apply our methodology to a standard life cycle consumption-saving model. We start

by proposing a method of sampling income paths using CPS data. For every observation of a given age and income bin, we sample a future realization of all individuals with the same age and income bin. While the method is not perfect, we use it for illustrational purposes. Our contribution is orthogonal to the input data or how it was created. For future work, we aim to either train the model with real income paths obtained from administrative earnings data or train a neural network to sample income realizations while being agnostic about the DGP that generates it.

Within this application, we show how to solve a dynamic programming problem while being completely agnostic about the data-generating process for exogenous states. In this application, the model generates reasonable policy functions that are more optimized in the states that are more likely to be realized. We then discuss the cost of assuming a functional form for the exogenous state and find that this cost is low for the dataset that we have generated. And finally, we confirm that our methodology does reach a global optimum by comparing the solution we obtain with Agnostic DP when we know the DGP, and the solution obtained with traditional methods, which we know is globally optimum.

This paper is a proof of concept of the power of Agnostic DP, and opens several avenues of research. A caveat of the solution method proposed is that it is defined for finite horizon problems. The problem becomes harder to optimize as we increase the horizon, partly due to discounting. Early in life, mistakes in the policy function when old have very little effect on total utility. The methodology works for solving partial equilibrium problems. A natural extension would be to close the model in general equilibrium. It is straightforward to have aggregate variables as a state variable, but satisfying market clearing conditions is not.

Tables and Figures

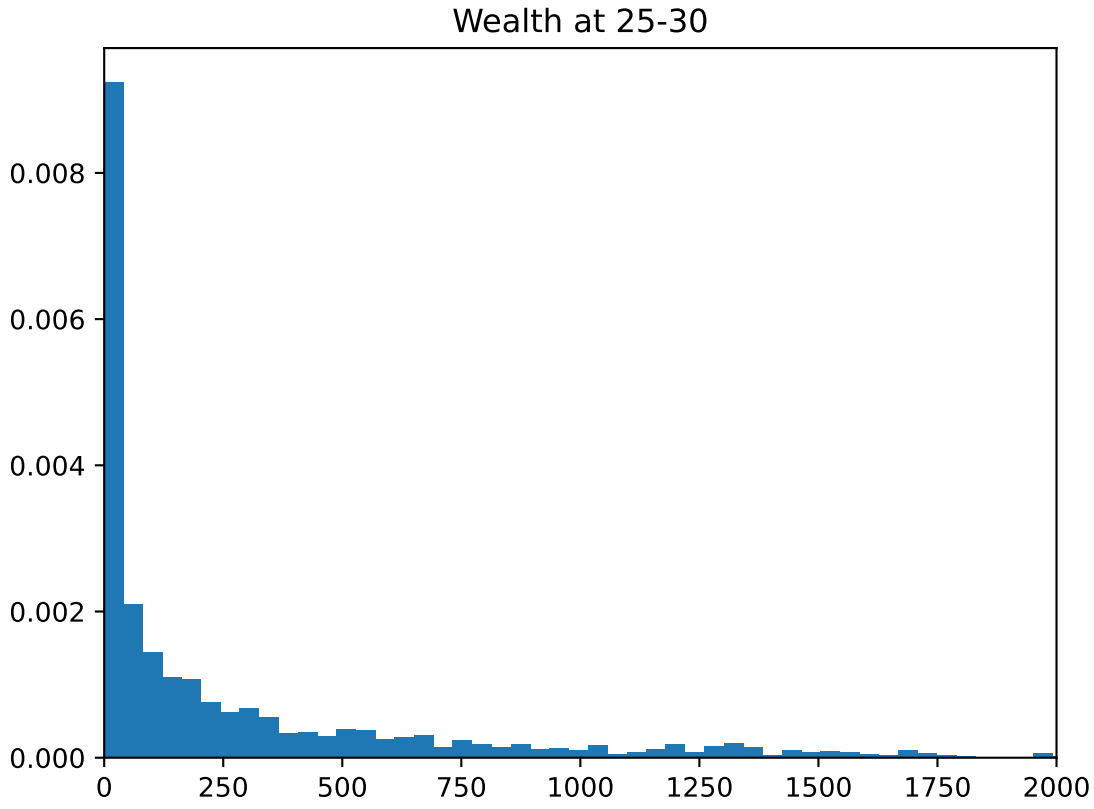

Figure 1: Wealth distribution at age 25-30 from the Survey of Consumer Finances.

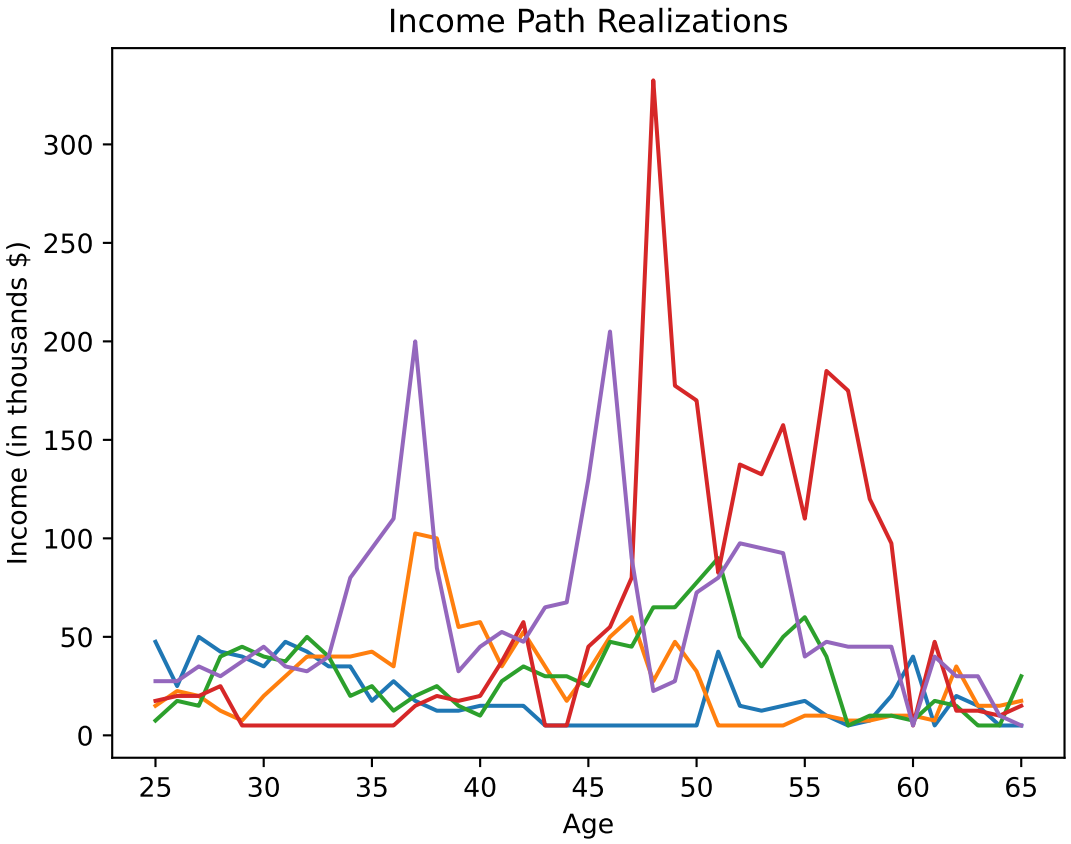

Figure 2: Synthetic Income realizations from the CPS.

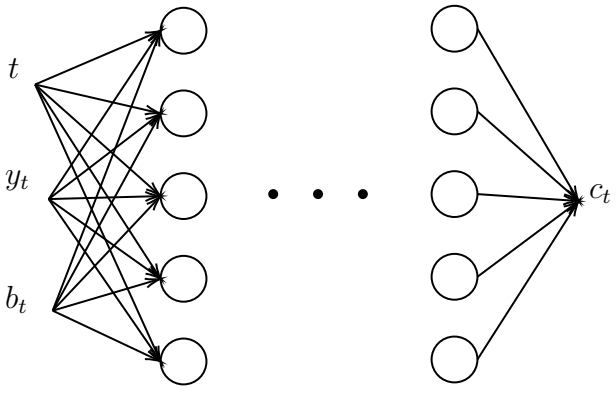

Figure 3: Neural network used to train the model, with 3 inputs, 1 output and 5 hidden layers, with 500 nodes each. Hidden layers are activated with $tanh$, and the output layer is activated using $sigmoid$

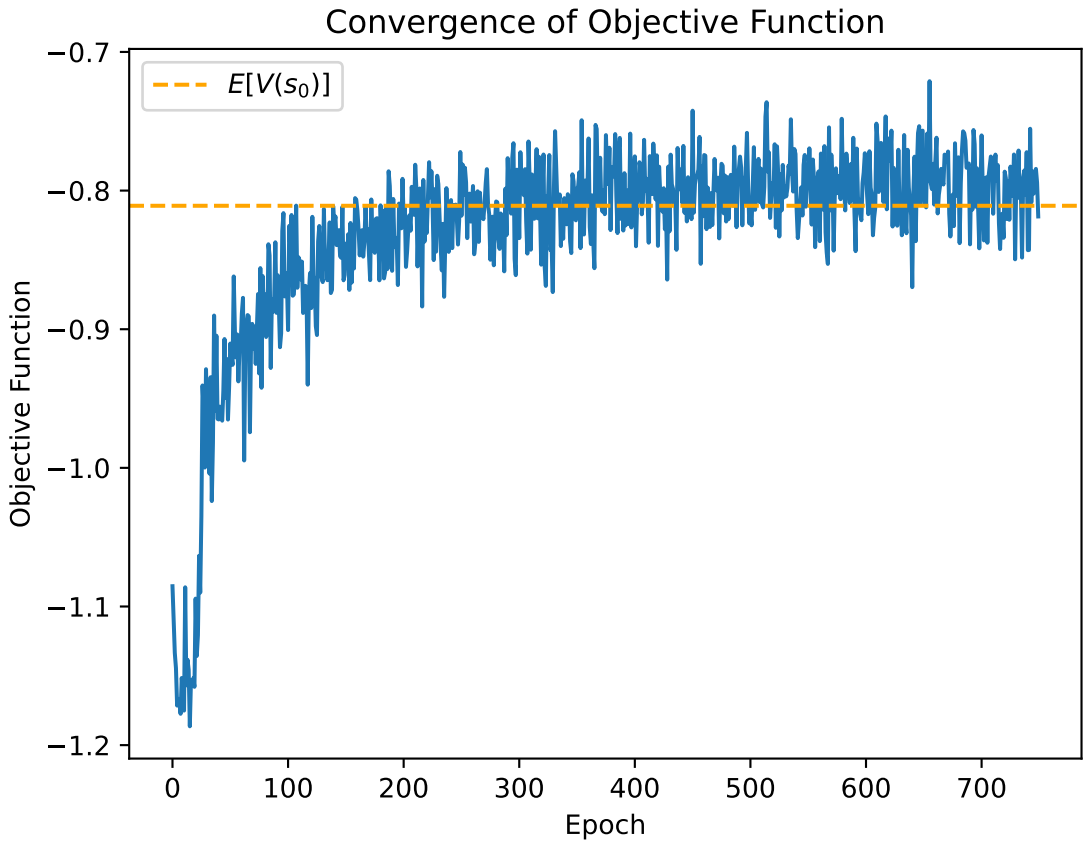

Figure 4: Convergence of the training algorithm. In each epoch, we use 200 sample paths (blue line). The orange dashed line denotes the value of the test data.

Parameter	Value
ρ	0.71
σ	0.71
δ_0	3.025
δ_1	0.035
δ_2	-0.000797

Table 1: Calibration of the income process. $\log y_{it} = \delta_0 + \delta_1 t + \delta_2^2 + \eta_{it}$ and $\eta_{it} = \rho\eta_{it-1} + \epsilon_{it}$

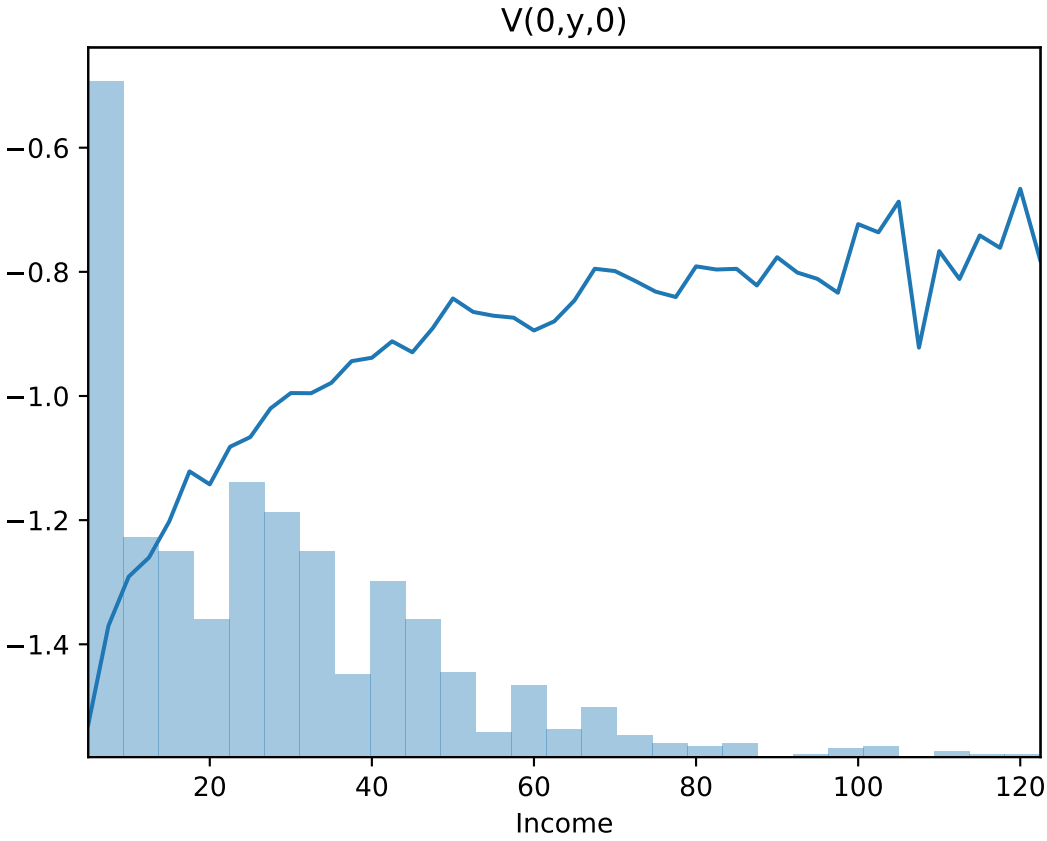

Figure 5: Value Function as a function of income of an agent at 25 with 0 assets. The histogram represents the distribution of income at that age.

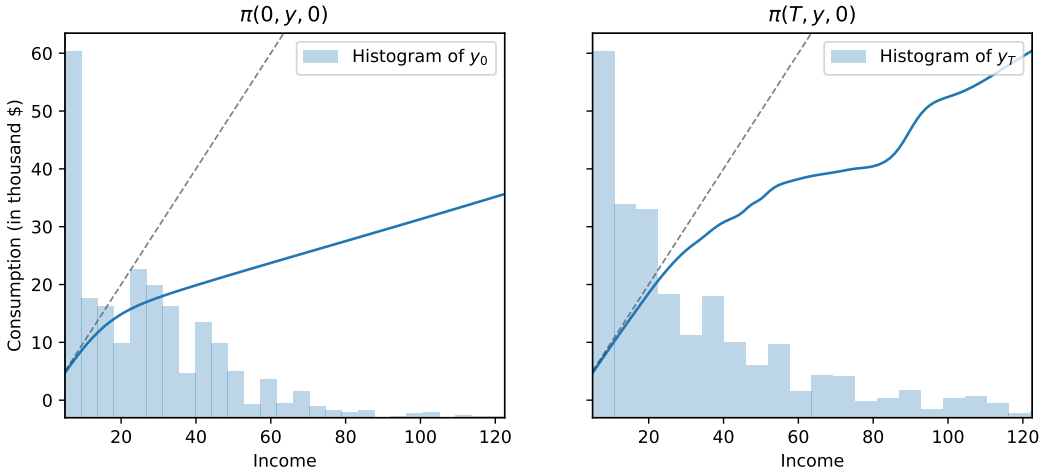

Figure 6: Policy function at t=0 (25 year old) and $t = 40$ (65 year old). The histogram represents the distribution of income at that age.

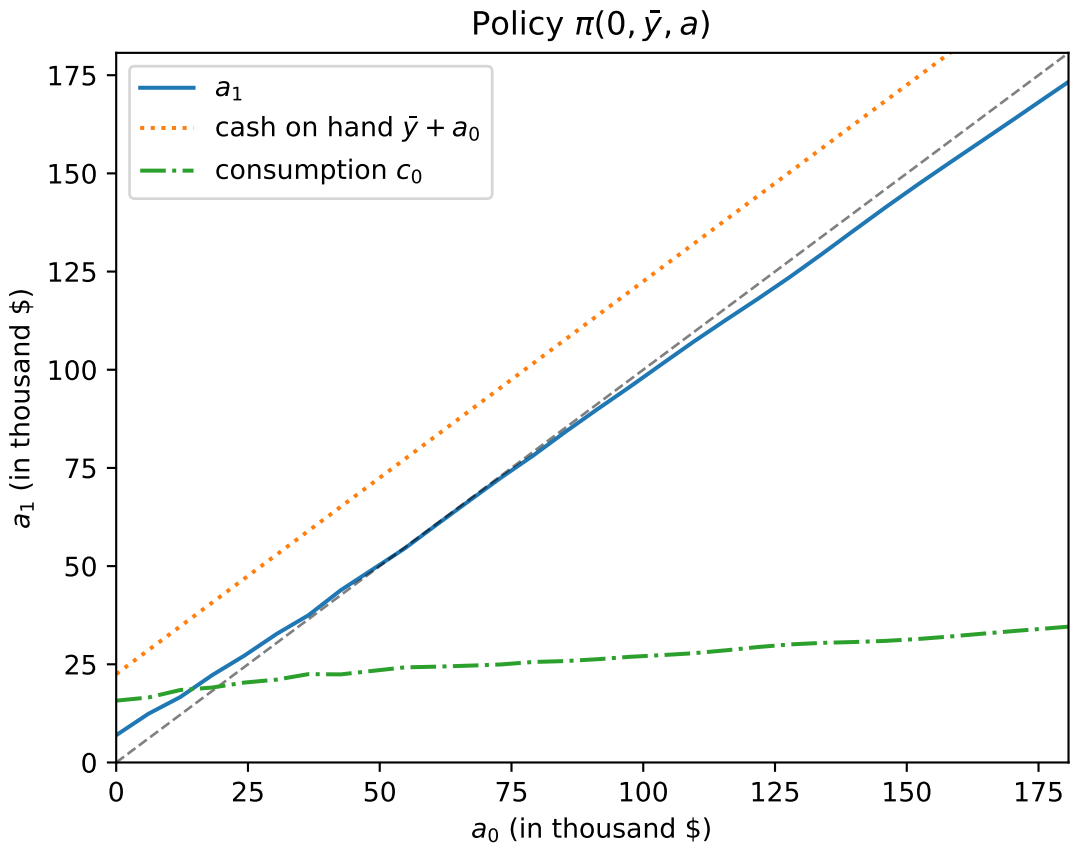

Figure 7: Policy function for an agent at 25 with median income $22,500 as a function of assets.

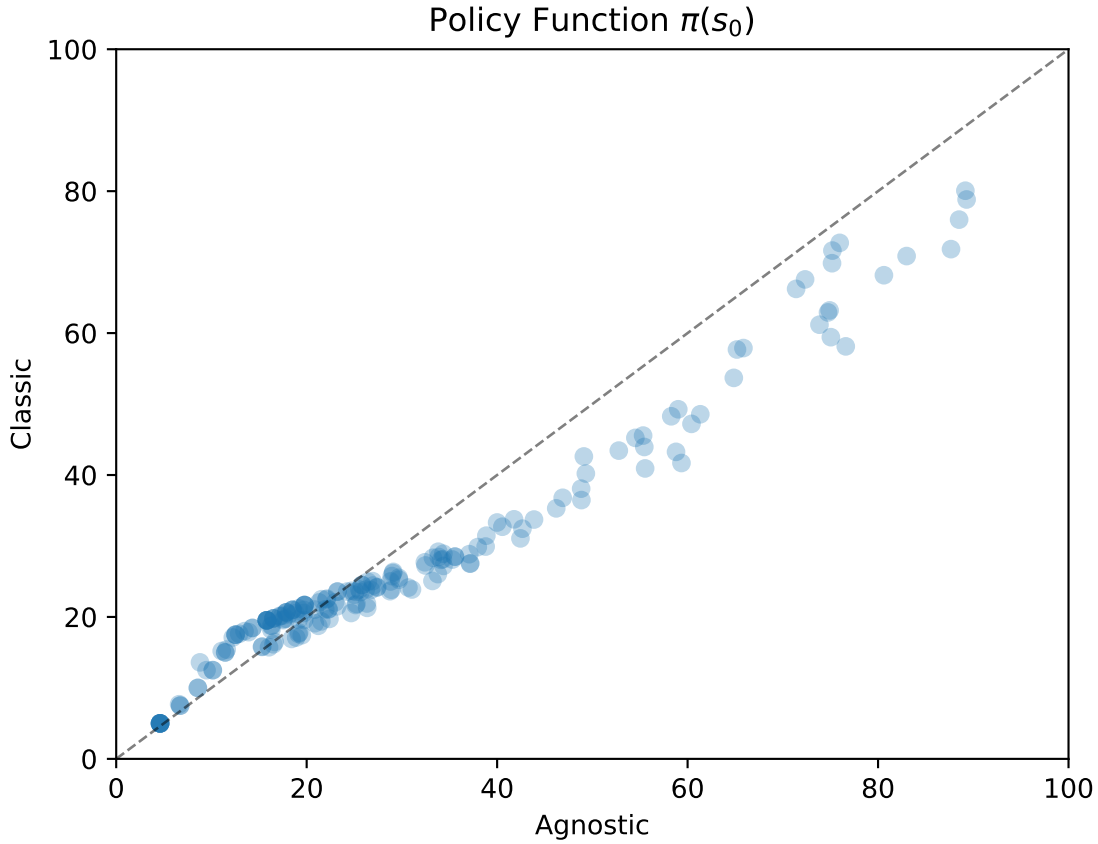

Figure 8: Policy function of the agnostic model and the classic model. Each dot represents a random initial state at age 25. Note that each policy has been trained with different income processes so they do not have to coincide.

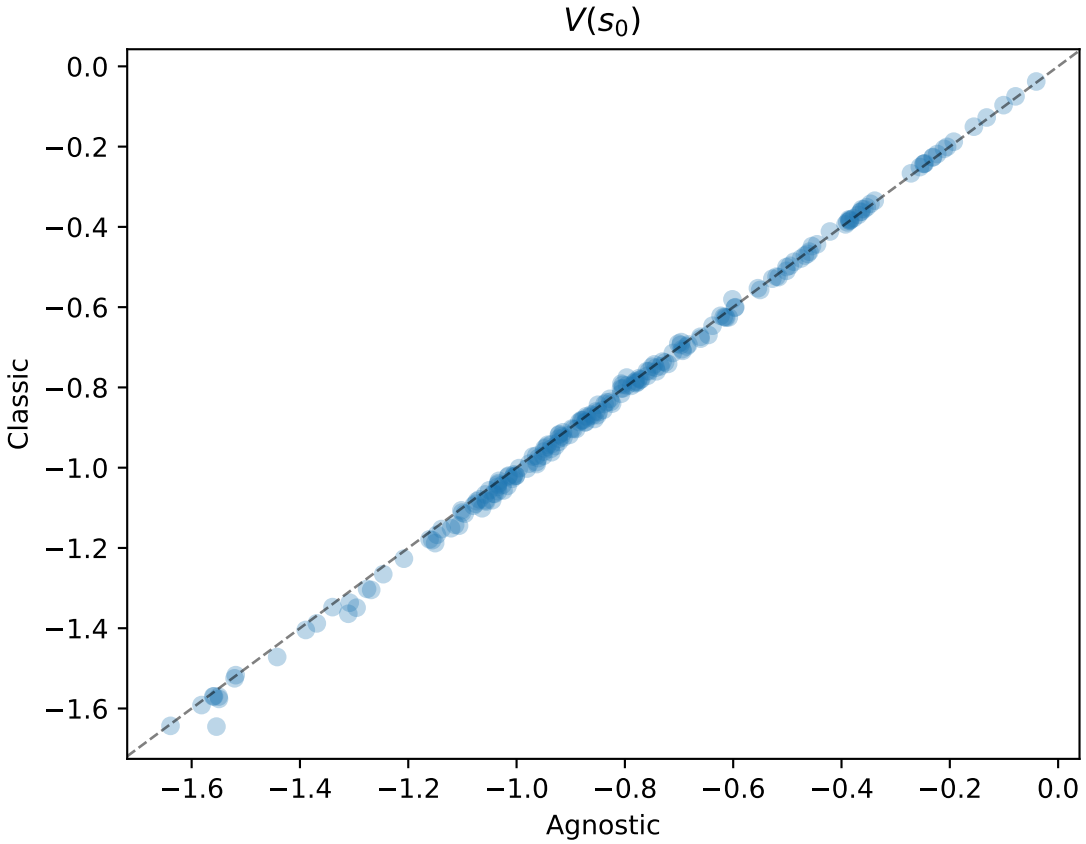

Figure 9: Value function comparison. Each dot represents $E[V(s_0)]$ for a given s_0, computed with the agnostic policy function and the classic policy function. Income paths are sampled from data realizations.

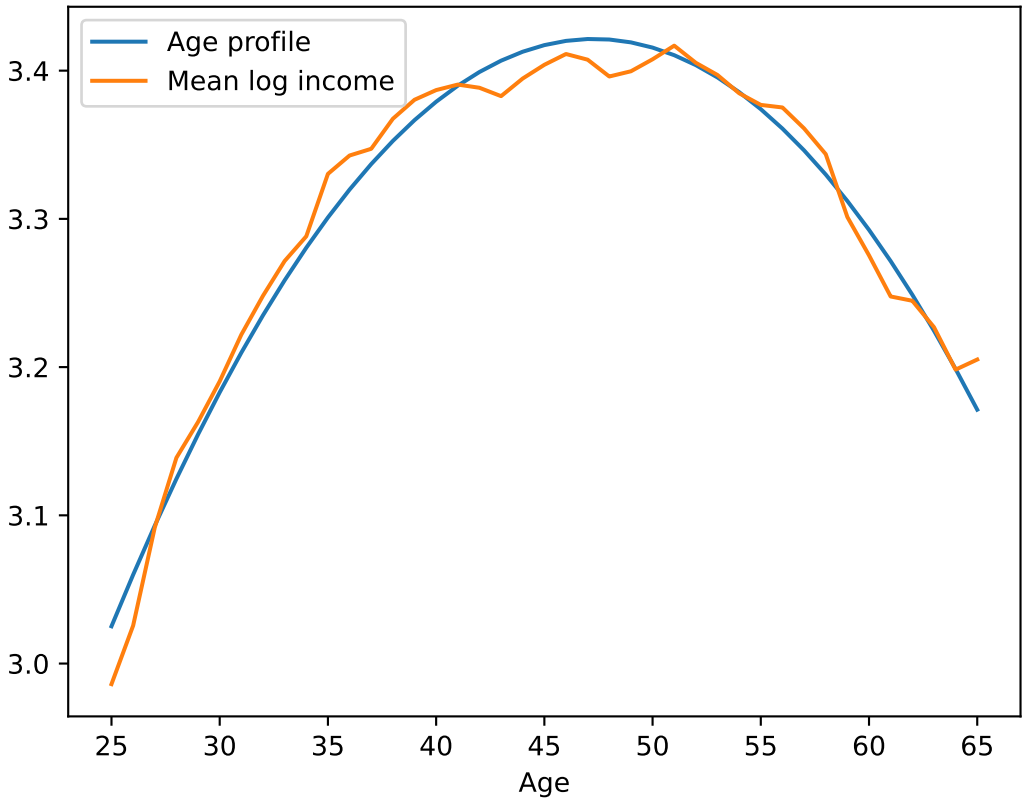

Figure 10: Age profile for the process of $\log y_t$

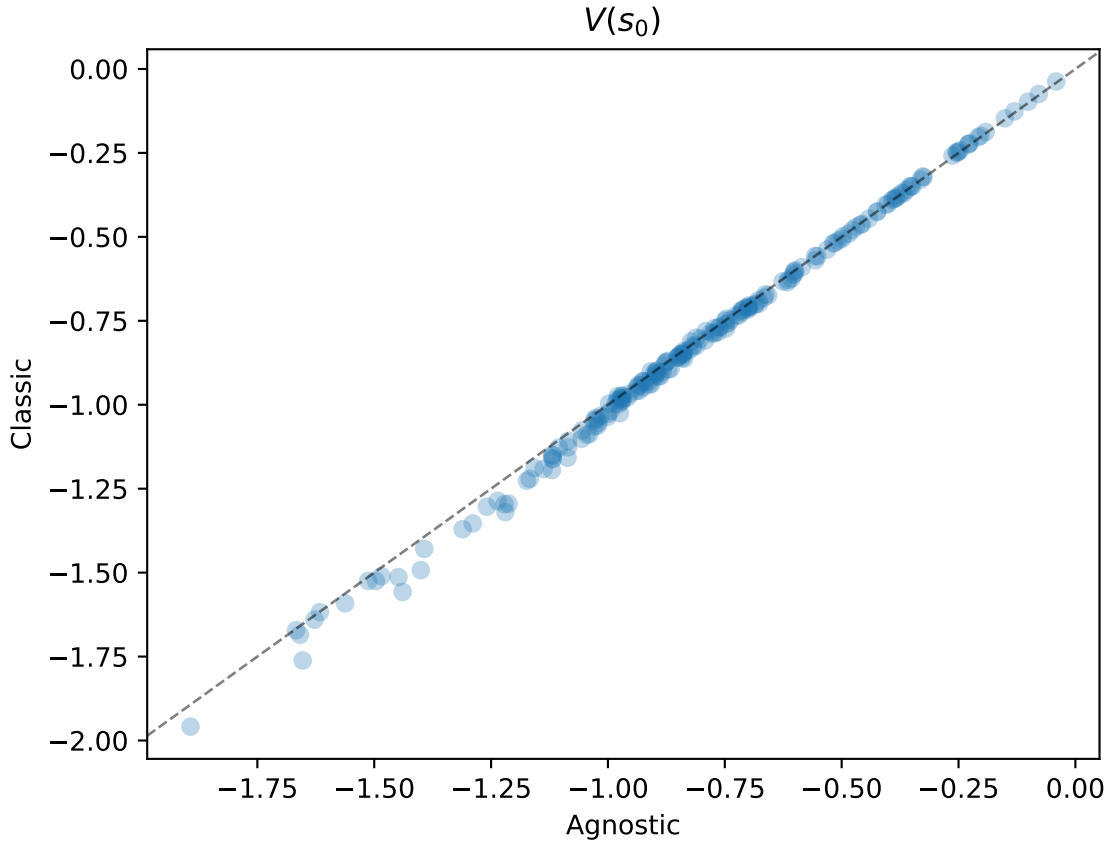

Figure 11: Value function comparison. Each dot represents $E[V(s_0)]$ for a given s_0, computed with the agnostic policy function and the classic policy function. Income realizations are sampled from the income process defined in Section 3.4

References

Azinovic, Marlon, Luca Gaegauf, and Simon Scheidegger (2022), "Deep equilibrium nets." *International Economic Review*, 63, 1471–1525.

Bradbury, James, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, and Qiao Zhang (2018), "JAX: composable transformations of Python+NumPy programs."

Braxton, J. Carter, Kyle Herkenhoff, Jonathan Rothbaum, and Lawrence D. W. Schmidt (2021), "Changing income risk across the US skill distribution: Evidence from a generalized Kalman filter." *Working Paper*.

Catherine, Sylvain (2022), "Countercyclical labor income risk and portfolio choices over the life cycle." *The Review of Financial Studies*, 35, 4016–4054.

De Nardi, Mariacristina, Eric French, and John Bailey Jones (2016), "Medicaid insurance in old age." *American Economic Review*, 106, 3480–3520.

Duarte, Victor, Diogo Duarte, and Dejanir H Silva (2023), "Machine learning for continuous-time finance."

Duarte, Victor, Julia Fonseca, Jonathan A. Parker, and Aaron Goodman (2022), "Simple Allocation Rules and Optimal Portfolio Choice Over the Lifecycle." *Working Paper*.

Fernández-Villaverde, Jesús, Samuel Hurtado, and Galo Nuño (2023), "Financial frictions and the wealth distribution." *Econometrica*, 91, 869–901.

Gourinchas, Pierre-Olivier and Jonathan A. Parker (2002), "Consumption over the life cycle." *Econometrica*, 70, 47–89.

Guvenen, Fatih, Fatih Karahan, Serdar Ozkan, and Jae Song (2021), "What Do Data on Millions of U.S. Workers Reveal About Lifecycle Earnings Dynamics?" *Econometrica*, 89, 2303–2339.

Guvenen, Fatih, Alisdair McKay, and Conor Ryan (2022), "A Tractable Income Process for Business Cycle Analysis." *Working Paper*.

Guvenen, Fatih, Serdar Ozkan, and Jae Song (2014), "The nature of countercyclical income risk." *Journal of Political Economy*, 122, 621–660.

Kingma, Diederik P. and Jimmy Ba (2015), "Adam: A method for stochastic optimization."

Scheidegger, Simon and Ilias Bilionis (2019), "Machine learning for high-dimensional dynamic stochastic economies." *Journal of Computational Science*, 33, 68–82.

Stokey, Nancy L., Robert E. Lucas, and Edward C. Prescott (1989), *Recursive Methods in Economic Dynamics*. Harvard University Press.