Machine Learning for Continuous-Time Finance

Presenter: Junping Zhu

September 4, 2025

Outline

Research question

- Dynamic programming principle(DPP) is one of the cornerstones of modern financial economics. However, it is plagued by the "curse of dimensionality" for **larger state space** $\mathcal{S}$
- PIA? DPIA! : propose a deep policy iteration algorithm

Merton Problem in the simplest setting

- Merton Problem: a set of investment-consumption problems that try to address the tradeoff between risk-taking, wealth accumulation and consumption
- Assumptions:
  - Two tradeable assets
  - No market frictions...
- Objective:

$$\max_{(\pi,C)} \mathbb{E}\left[\int_0^T e^{-\rho t} U(C_t)\,dt + e^{-\rho T} U_T(W_T)\right]$$

$$\begin{cases} \dfrac{dS_t}{S_t} = \mu\,dt + \sigma\,dB_t, \\ dW_t = \left[rW_t + \pi_t(\mu - r) - C_t\right]dt + \pi_t \sigma\,dB_t. \end{cases}$$

Generalized Framework

- Objective:

$$\max_{u_s : s \in [0,T]} \mathbb{E}\left[\int_0^T L(s, x_s, u_s)\, ds \; + \; K(T, x_T)\right]$$

$$\text{s.t.} \quad dx_t \; = \; f(t, x_t, u_t)\, dt \; + \; g(t, x_t, u_t)\, dB_t$$

- x_t is state variables and u_t is control or policy variables

Standard Procedure to solve a Merton problem

- Step1: Define Value Function $V(t,x)$

$$V(t,x) \;=\; \max_{u_s:\, s\in[t,T]} \mathbb{E}\left[\int_t^T L(s,x_s,u_s)\,ds \;+\; K(T,x_T) \;\middle|\; x_t = x\right].$$

- Step2: Establish the DPP, which links the value function at the current time to the expectation of the value function at a future time t

$$V(t,x) \;=\; \max_{u_s:\, s\in[t,t+\Delta]} \mathbb{E}\left[\int_t^{t+\Delta} L(s,x_s,u_s)\,ds \;+\; V(t+\Delta, X_{t+\Delta})\right].$$

Standard Procedure to solve a Merton problem

- Step3: Send t $\to 0$ and use the Ito lemma to obtain the Hamilton–Jacobi–Belman equation(hereafter HJB equation)

$$\max_u \{V_t + \mathcal{M}(t,x,u)V + L(t,x,u)\} = 0, \qquad V(T,x) = K(T,x)$$

where

$$\mathcal{M}(t,x,u)V = f(t,x,u)\,\partial_x V + \tfrac{1}{2}\,g(t,x,u)^2\,\partial_{xx}V.$$

- However, it is often easy to write down, but difficult to solve, an HJB equation in closed form $\to$ PIA(Policy Iteration Algorithm)

Policy Iteration Algorithm

- Idea: treat the PDE and the optimization component separately and approach the solution gradually
- Step1: Given a control u, we can solve the PDE numerically, and obtain J

$$J_t + \mathcal{M}(t,x,u)J + L(t,x,u) = 0, \qquad J(T,x) = K(T,x).$$

- Step2:Given a given function J, we can compute the maximizer numerically.

$$u^* = \arg\max_u \{\mathcal{M}(t,x,u)J + L(t,x,u)\}.$$

- Given an initial guess, and ITERATE until some stopping criterion is met.

Curse of dimensionality kicks in

- Curse1: The value function and policy live on a high-dimensional state space.(**Step 1 of S.P.**)
  - State variable in simplest Merton Problem is limited, that is dynamic budget constraint x.
  - Solu: Use deep neural networks to represent value and policy function
- Curse2: compute a multidimensional Ito's lemma, which naturally scales poorly with the number of state variables. (**Step 3 of S.P.**)
  - Solu: automatic differentiation
- Curse3: maximizing an objective function at each iteration step becomes very expensive with many controls and state variables.(**Steps of PIA**)
  - Solu: DPI(deep policy iteration)

Outline

Deep policy iteration algorithm

- A generalized framework of optimal control problems
  - An agent face a Markovian decision process with the vector of state variables $s \in \mathcal{S}$ and choose the policy(i.e control) c to maximize her lifetime expected utility:

$$V(\mathbf{s}_t) = \max_{\{\mathbf{c}_v\}} \mathbb{E}_t \left[\int_t^\infty e^{-\rho(v-t)} u(\mathbf{c}_v) dv \right]$$

$$\text{s.t.} d\mathbf{s}_t = \mathbf{f}(\mathbf{s}_t, \mathbf{c}_t)dt + \mathbf{g}(\mathbf{s}_t, \mathbf{c}_t)d\mathbf{B}_t,$$

$$\mathbf{c}_t \in \Gamma(\mathbf{s}_t), \forall t \in [0, \infty),$$

  - HJB equation(hear comes **Curse 2**)

$$0 = \max_{\mathbf{c} \in \Gamma(\mathbf{s})} \{\text{HJB}(\mathbf{s}, \mathbf{c}, V(\mathbf{s}))\}dt$$

where

$$\text{HJB}(\mathbf{s}, \mathbf{c}, V) = u(\mathbf{c}) - \rho V + (\nabla_{\mathbf{s}} V)^\top \mathbf{f}(\mathbf{s}, \mathbf{c}) + \frac{1}{2}\text{Tr}\left[\mathbf{g}(\mathbf{s}, \mathbf{c})^\top \mathbf{H}_{\mathbf{s}} V \mathbf{g}(\mathbf{s}, \mathbf{c})\right]$$

Curse2: multidimensional Ito's lemma

- In 1-D state variable setting, s follow the following SDE

$$ds_t = f(s_t)dt + g(s_t)dB_t$$

$V(s)$ denote an arbitrary function of s with continuous second-order partial derivatives. Ito's lemma states that:

$$\frac{\mathbb{E}dV}{dt}(s) = V_s f(s) + \frac{1}{2} V_{ss} g(s)^2$$

- When it comes to multiple dimension Ito's lemma states that:

$$\frac{\mathbb{E}dV}{dt}(s) = \nabla_s V(s)^\top f(s) + \tfrac{1}{2} \operatorname{Tr}\left[g(s)^\top H_s V(s) g(s)\right],$$

Curse2: multidimensional Ito's lemma

- A naive implementation of Ito's lemma would involve computing all firstand second-order partial derivatives. The paper propose the following proposition to bypass

Proposition 1. For a given $\mathbf{s}$, define the auxiliary function $F:\mathbb{R}\to\mathbb{R}$ as

$$F(\epsilon)\equiv\sum_{i=1}^{m}V\left(\mathbf{s}+\frac{\epsilon}{\sqrt{2}}\mathbf{g}_i(\mathbf{s})+\frac{\epsilon^2}{2m}\mathbf{f}(\mathbf{s})\right),$$

where $\mathbf{g}_i(\mathbf{s})$ represents column i of the matrix $\mathbf{g}(\mathbf{s})$. Then,

$$F''(0)=\frac{\mathbb{E}dV}{dt}(\mathbf{s}).$$

- Then, the HJB becomes

$$\text{HJB}(\mathbf{s},\mathbf{c},V)=u(\mathbf{c})-\rho V+F''(0)$$

Bad idea: PIA

- Now, since we have written HJB Equation, it's time to perform policy iteration algorithm(PIA)! That is, compute the PDE and optimization part separately.
- Recall HJB equation becomes

$$0 = \max_{\mathbf{c}\in\Gamma(\mathbf{s})} \{u(\mathbf{c}) - \rho V + F''(0)\}dt$$

- Optimization part(Policy Improvement)

$$\mathbf{c}(\mathbf{s}) = \operatorname*{argmax}_{\mathbf{c}\in\Gamma(\mathbf{s})}(u(\mathbf{c}) - \rho V + F''(0))$$

- PDE part(Policy Evaluation)

$$0 = u(\mathbf{c}) - \rho V + F''(0)$$

Bad idea: PIA(Vector Presentation)

- However, the state space $\mathcal{S}$ is so large(**here come Curse 1 and 3**), so the standard way becomes
  - Choose a finite subset of the state space, i.e just budget constraint
  - Parameterize the value and policy functions

$$\mathbf{c}(\mathbf{s}_i; \theta_C) = \theta_{C,i} \quad V_\pi(\mathbf{s}_i; \theta_V) = \theta_{V,i},$$

    where θ_C and θ_V are vectors under subset of the state space.
  - Give a initial guess of θ_C^0 and θ_V^0, construct a sequence $\{\theta_C^j, \theta_V^j\}_{j\in N}$ as follows until some stopping criterion is met.

$$\theta_{C,i}^j = \arg\max_{\mathbf{c}\in\Gamma(\mathbf{s}_i)} \mathrm{HJB}(\mathbf{s}_i, \mathbf{c}; \theta_V^{j-1})$$

$$\theta_{V,i}^j = \theta_{V,i}^{j-1} + \mathrm{HJB}(\mathbf{s}_i; \theta_C^j, \theta_V^{j-1})\Delta t,$$

Bad idea: PIA

- The previous PIA relies on a discretization of the state space, because it is challenging to approximate a high-dimensional nonlinear value and policy function on a computer(**Curse 1**).
- The policy improvement step, involves an optimization step for every state. This step of optimizing for every single state can be computationally very costly(**Curse 3**)

Good idea: DPIA

- Step1 (Sampling): Consider a random sample of points $\{s_i\}_{i=1}^{I}$ in the state space.
- Step2 (Policy improvement): For each state si in the mini-batch and starting from the initial guess $\mathbf{c}_{0,i} \equiv \mathbf{c}(\mathbf{s}_i; \theta_C^{j-1})$, do one step of gradient descent on $-\text{HJB}(\mathbf{s}_i, \mathbf{c}, \theta_V^{j-1})$ using a learning rate of 1. The new control for each point in the mini-batch is

$$\mathbf{c}_{1,i} = \mathbf{c}_{0,i} + \nabla_{\mathbf{c}}\text{HJB}(\mathbf{s}_i; \mathbf{c}_{0,i}, \theta_V^{j-1}).$$

Idea: here try to see which direction in c (or which better policy) improves the HJB mostly(embed economical ideas into training scheme in Machine learning)

Good idea: DPIA

- Step2 Contd.
  - Then, use $\mathbf{c}_{1,i}$ as targets to train the policy network. The objective is to find θ_C^j to minimize the following function

$$\theta_C^j = \arg\min_{\theta} \mathcal{L}(\theta), \text{ where} \mathcal{L}(\theta) = \frac{1}{2I} \sum_{i=1}^{I} ||\mathbf{c}(\mathbf{s}_i; \theta) - \mathbf{c}_{1,i}||^2.$$

  - Dynamic policy updating rule becomes

$$\theta_C^j = \theta_C^{j-1} + \eta_C \frac{1}{I} \sum_{i=1}^{I} \nabla_{\theta_C} \text{HJB}(\mathbf{s}_i, \theta_C^{j-1}, \theta_V^{j-1}).$$

Good idea: DPIA

- Step3 Idea: how "good" is the outcome policy of step 2? should we stop?(Simple Merton Problem just need 1 iteration under PIA)
- Step3 (Policy Evaluation): The Bellman target is

$$V(\mathbf{s};\theta^j) = V(\mathbf{s};\theta_V^{j-1}) + \mathrm{HJB}(\mathbf{s},\theta_C^j,\theta_V^{j-1})\Delta t.$$

  - Policy Evaluation 1

$$\theta_V^j = \theta_V^{j-1} + \eta_V \frac{\Delta t}{I}\sum_{i=1}^{I} \mathrm{HJB}(\mathbf{s}_i,\theta_C^j,\theta_V^{j-1})\nabla_{\theta_V} V(\mathbf{s}_i;\theta_V^{j-1}).$$

  - Policy Evaluation 2

$$\theta_V^j = \theta_V^{j-1} - \eta_V \frac{1}{I}\sum_{i=1}^{I} \mathrm{HJB}(\mathbf{s}_i,\theta_C^j,\theta_V^{j-1})\nabla_{\theta_V} \mathrm{HJB}(\mathbf{s}_i,\theta_C^j,\theta_V^{j-1}).$$

Outline

Outline

An example

- Consider the following objective, and **find the optimal portfolio share and consumption policy**

$$V(W, \mathbf{x}) = \max_{\{C_t, \alpha_t\}_0^\infty} \mathbb{E}_0 \left[\int_0^\infty e^{-\rho t} \frac{C_t^{1-\gamma}}{1-\gamma} dt \right],$$

with the wealth dynamics

$$dW_t = [(r_t + \alpha_t \xi_t) W_t - C_t] dt + \alpha_t W_t \sigma_{\mathbf{r}} d\mathbf{Z}_t.$$

- Under this setting, the state variables space is consituted by W wealth and x return predictors. The control variables are α_t and C_t
- Exogenous processes(r_t and ξ_t) are NOT constant.

An example

- Under CRRA, we can derive one-to-one mapping between the consumption policy and the value-function. That is ,

$$V(W,x) = \phi(x)\frac{W^{1-\gamma}}{1-\gamma} \quad , C_t = \phi(x_t)^{-\frac{1}{\gamma}} W_t$$

Test DPIA how good: Reverse engineering

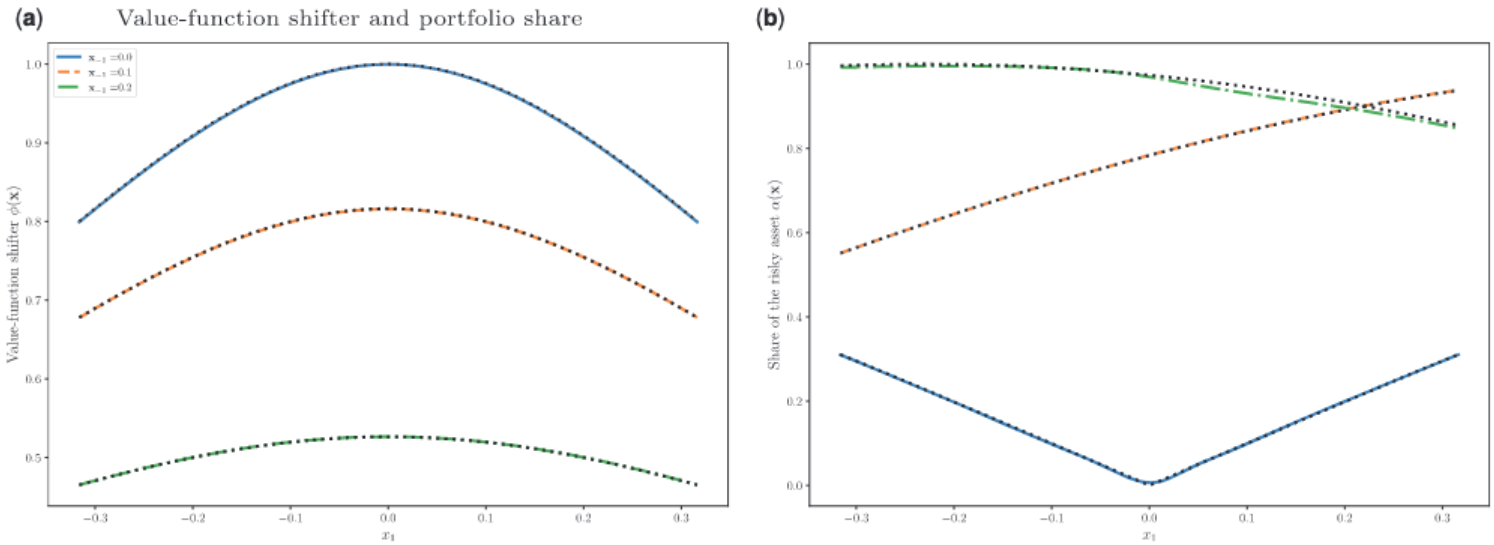
- Step1: Pre-specify the functional form $\phi(x)$ and $\alpha(x)$
  - $\phi(x)$: multivariate version of the Runge function
    $\phi(\mathbf{x}) = \frac{1}{1+\frac{25}{N}\sum_{j=1}^{N} x_j^2}$
  - $\alpha(x)$: $\alpha(\mathbf{x}) = \left|\sin\left(\sum_{j=1}^{N} x_j^2\right)\right|$
- Step2: Follow the standard procedure of Merton problems, obtain HJB and compute FOC on C and α, we can yield:

$$\xi(\mathbf{x}) = \gamma\,||\sigma_{\mathbf{r}}||^2\alpha(\mathbf{x}) - \sigma_\phi(\mathbf{x})\sigma_{\mathbf{r}}^\top,$$

$$r(\mathbf{x}) = \frac{\rho}{1-\gamma} + \frac{\gamma\,||\sigma_{\mathbf{r}}||^2}{2} - \left(\frac{\gamma\,\phi(\mathbf{x})^{-\frac{1}{\gamma}}}{1-\gamma} + \alpha(\mathbf{x})\xi(\mathbf{x}) + \sigma_\phi(\mathbf{x})\sigma_{\mathbf{r}}^\top + \frac{\mu_\phi(\mathbf{x}) + 0.5||\sigma_\phi(\mathbf{x})||^2}{1-\gamma}\right).$$

Test DPIA how good: Reverse engineering

- Step3: use the DPI method with Step2 $r(x)$ and $\xi(x)$ and compare the solution of the algorithm with the known functions $\alpha(x)$ and $C(x)$.
- The results are sufficiently accurate(ϕ_t and α_t)



A realistic example

- More tradable assets: has access to five risky assets in addition to a risk-free money market account, includes stocks, long- and medium-term nominal bonds, and long- and medium-term real bonds.
- More realistic utility: assume that the investor has recursive preferences with a risk aversion coefficient and an elasticity of intertemporal substitution (EIS)
- Returns are driven more realistic predictors(state variables), like inflation, GDP growth and other macro variables.
- More...

A realistic example Setup

- In classical Merton problem, we need to specify
  - Risky assets dynamics: such as stock(follows BM or Jump process). Model the evolution of nominal and real bonds of different maturities other than stocks.
  - State dynamics, such as how budget constraint change overtime. Here state variables x_t process can be estimated by fitting a VAR(1).
  - Investor Preference or utility functions
  - optimization problem

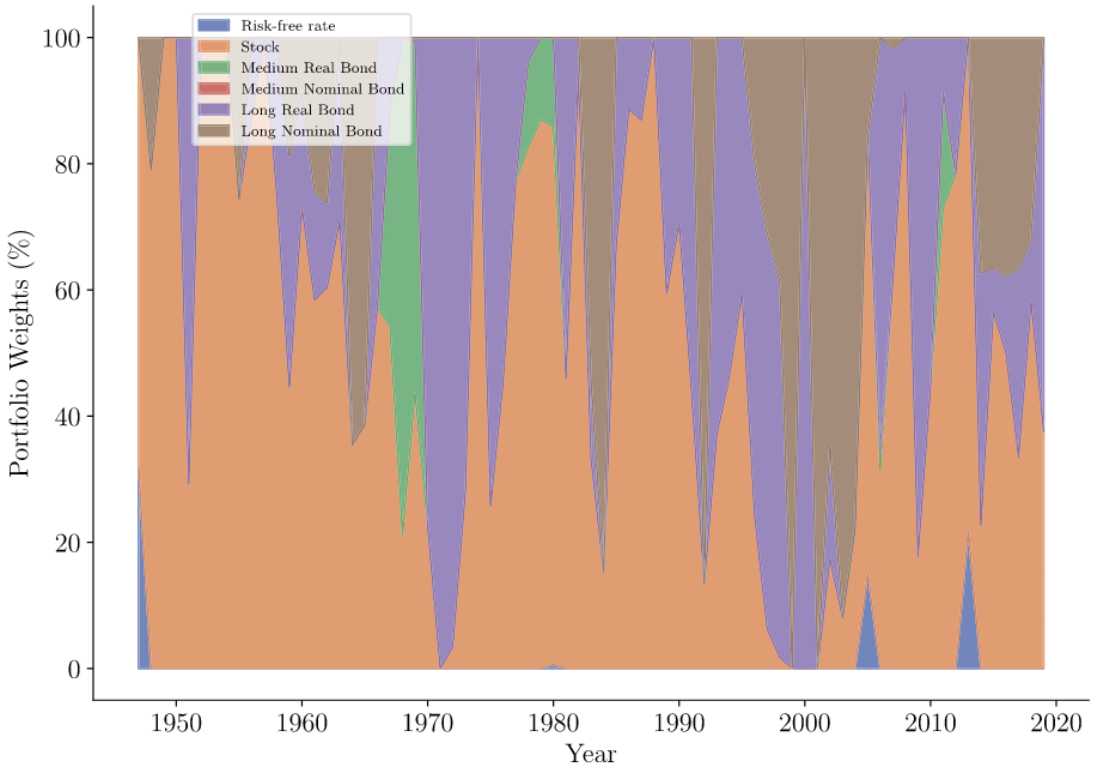
$$V(W, \mathbf{x}) = \max_{\{C_t, \alpha_t\}_0^\infty} \mathbb{E}_0 \left[\int_0^\infty f(C_t, V_t)\, dt \right].$$

subject to the wealth dynamics

$$dW_t = \left(W_t \left(r(\mathbf{x}_t) + \alpha_t^\top \xi(\mathbf{x}_t) \right) - C_t \right) dt + W_t \alpha_t^\top \sigma_R dZ_t,$$

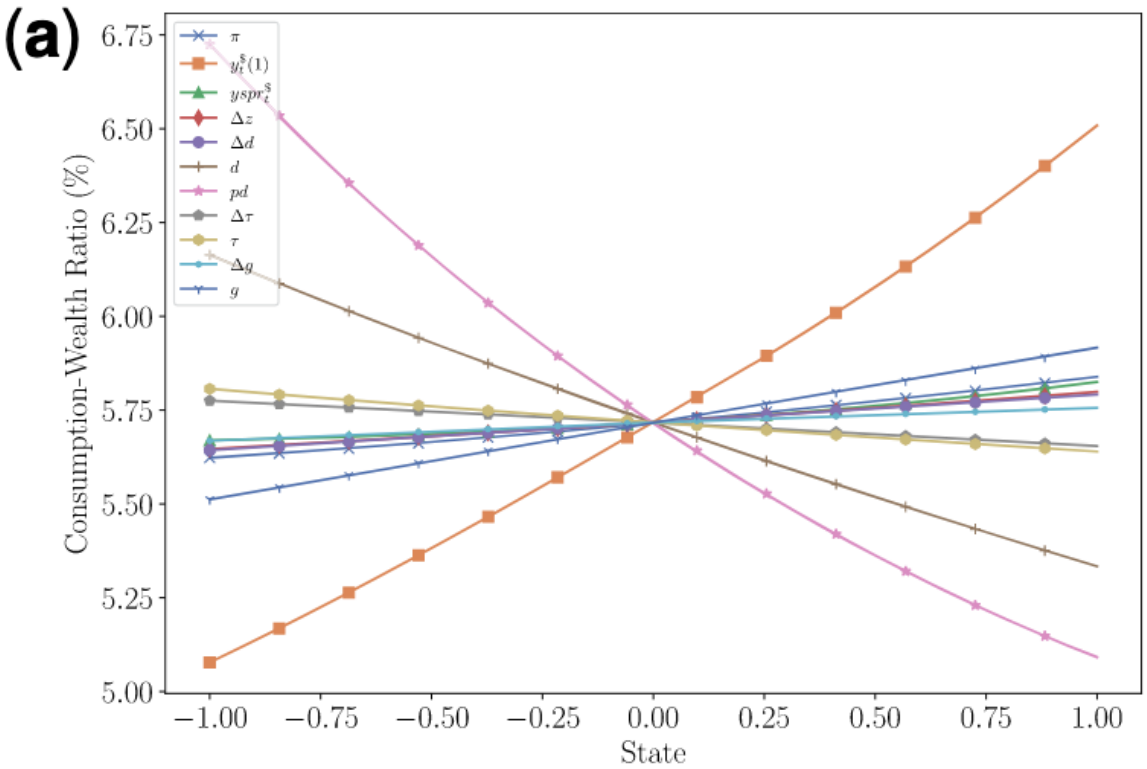
Portfolio allocation

- Market timing
- Substantial demand for inflation-protected bonds
- Substitution between stocks and bonds



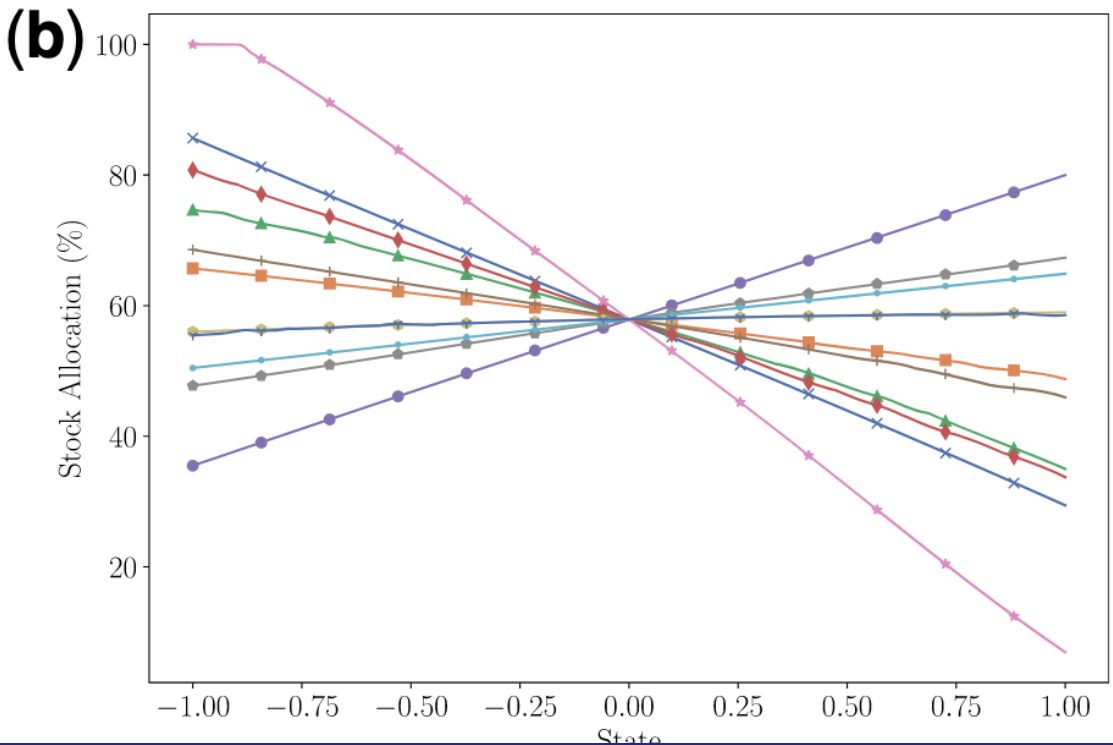
Policy functions

- Higher expected return, higher consumption-wealth ratio



Policy functions

- Higher price-dividend ratio, lower stock-wealth ratio
- Higher yield spread or the inflation rate, lower stock-wealth ratio



Outline

Model setup

- Analogous to an individual investor, manager of firm must decide how much to invest and how much cash to pay out to shareholders
- Objective:

$$V(k, z; \Phi) = \max_{\{i_t\}_0^\infty} \mathbb{E}\left[\int_0^\infty e^{-rt} D_t \, dt\right],$$

where

$$D_t = D_t^* \left(1 + \lambda \mathbf{1}_{\{D_t^* < 0\}}\right), \quad D_t^* = z_t k_t^\alpha - \left(i_t + 0.5\chi i_t^2\right) k_t.$$

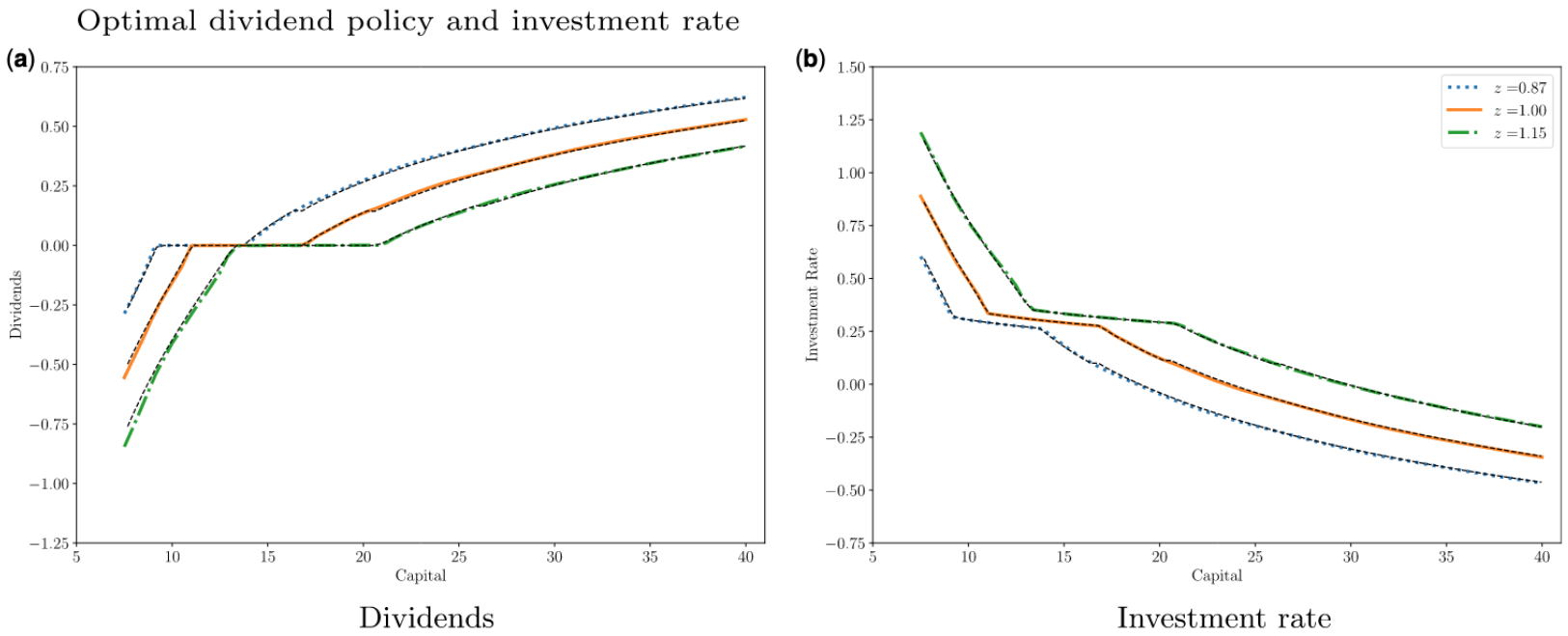
- State variables: TFP z_t and capital stock k_t and their dynamics

$$d \ln z_t = -\theta \ln z_t \, dt + \sigma \, dW_t, \quad \frac{dk_t}{k_t} = (i_t - \delta) \, dt.$$

- Policy or control variables: dividend and investment ratio
- Φ? Not state variables, not policy functions: UVF

Policy Functions

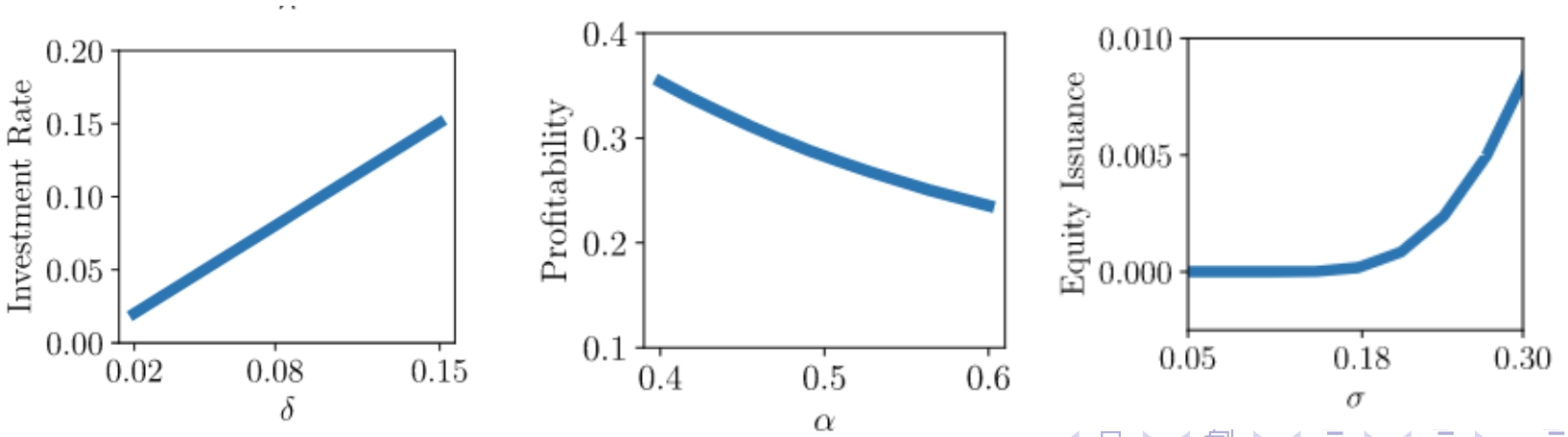
- Comparison between finite-differences and DPI
- Three different stages: equity insurance; inaction; pay out



Dividends Investment rate

Global sensitivity analysis and universal value functions

- Different parametrization Φ gives rise to different model results.
- For a large number of parameter values, sensitivity analysis can become computationally very costly and impractical. Here DPI comes
- 1. Average investment rate is sensitive to depreciation rate δ
- 2. Average profitability is sensitive to elasticity α in CD function
- 3. For high-volatility firms, equity issuance is sensitive to σ



Outline

Two-tree economy

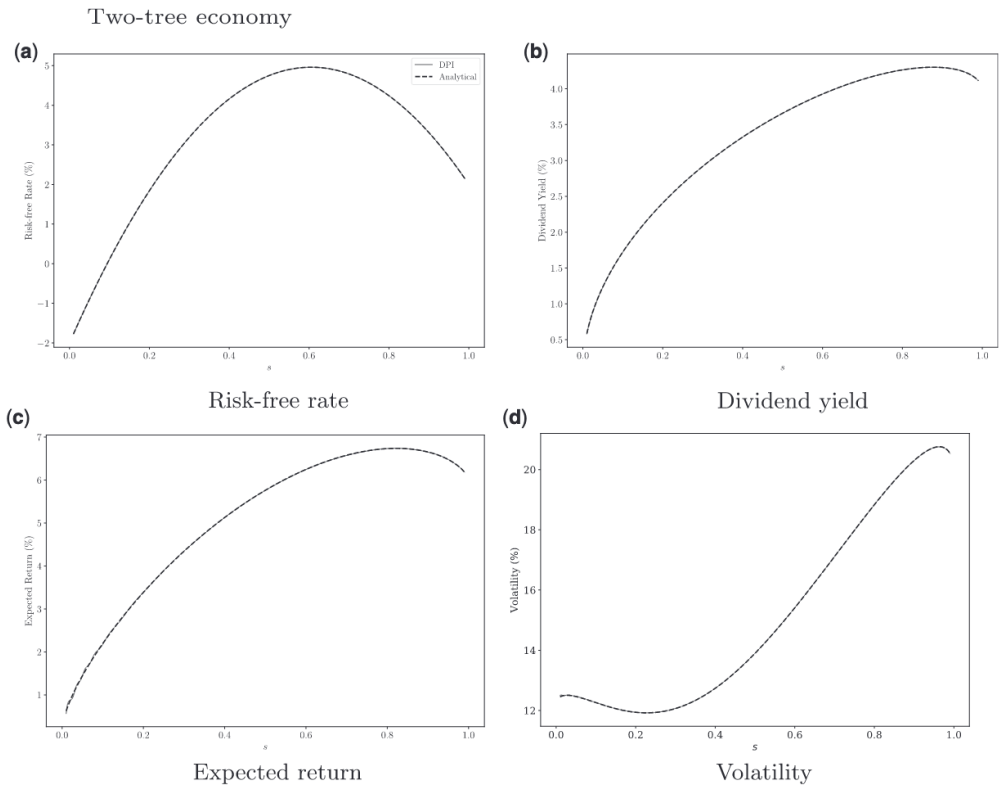
- What is "Tree"? a cash-flow source (like a firm) paying a random, continuous dividend over time.
- Under two-tree economy, Cochrane et.al(2008) derive closed-form expressions for the equilibrium objects quantities such as the short-term interest rate, dividend yield, expected return, and asset volatility
- Objective in **Infinite horizon**: a representative consumer chooses a consumption stream to maximize the lifetime expected utility

$$V = \mathbb{E}\left[\int_0^\infty e^{-\rho t} \log(C_t)\, dt\right].$$

$$C_t = D_{1t} + D_{2t}, \quad \frac{dD_{it}}{D_{it}} = \mu\, dt + \sigma\, dZ_{it}.$$

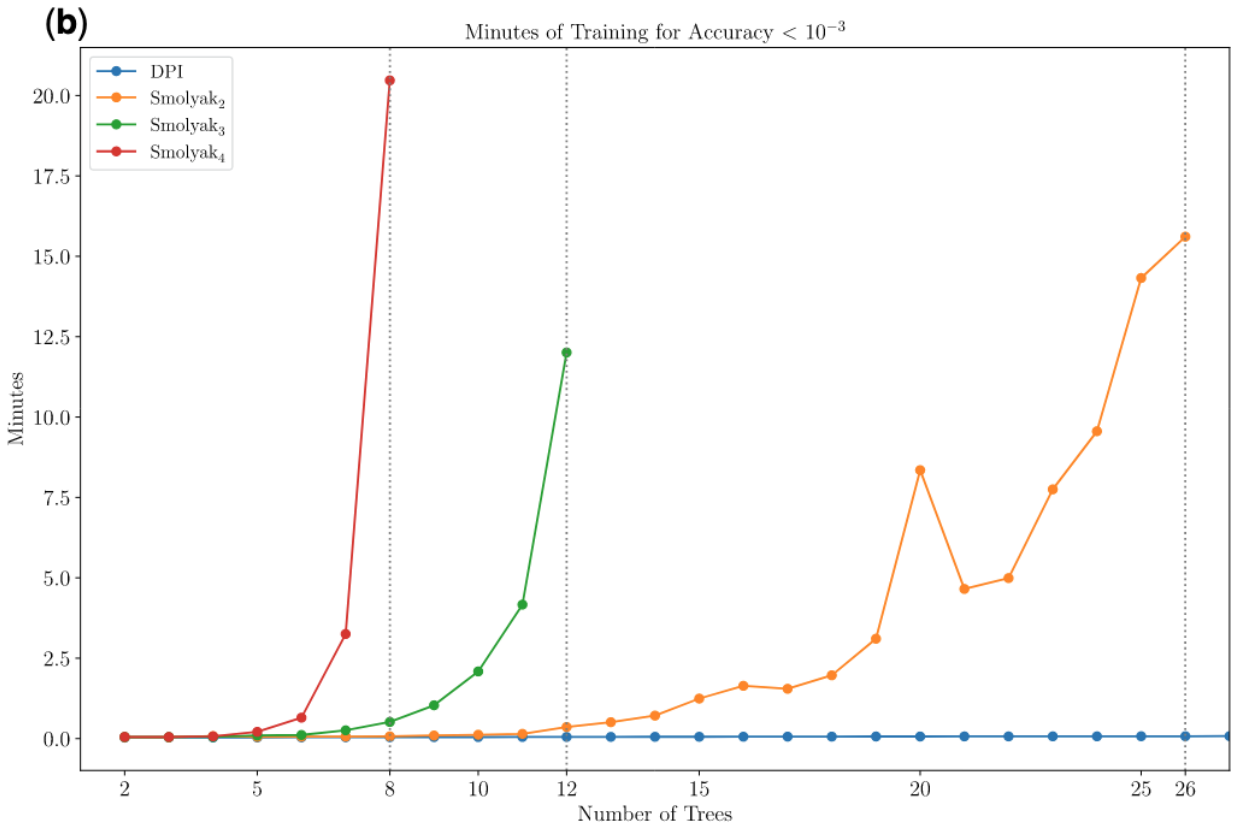
Two-tree economy

- Below quantities are determined by a single state variable, namely, the dividend share $s_t = \frac{D_{1t}}{D_{1t}+D_{2t}}$
- DPI method and analytical solution of Cochrane et al.(2008)



Lucas Orchard: N-trees

- COD: two-tree Lucas economy → Lucas orchard economy



Smolyak methods and DPI algorithm MSEs

Lucas Orchard: N-trees

- Important economic channels are overlooked when being restricted to a small number of trees
- The state is no longer one number s_t; it's essentially a whole vector of dividend shares
- **Diversified:**when the economy has many trees and the dividend share s_1 is relatively small, the behavior of the economy resembles a fully diversified economy

Conclude

- Compare to classical PIA, DPIA solves three curse of dimensionality
  - uses deep learning to represent value and policy functions
  - uses automatic differentiation to compute multidimensional Ito's lemma with little computation cost
  - uses a gradientbased version of policy iteration that dispenses root-finding methods to find the optimal control for a given state