

Machine learning, artificial intelligence and deep learning methods for dynamic economic models: **Introduction**

Lilia Maliar, Serguei Maliar

November 2022

Minicourse

Summary of minicourse

- Artificial intelligence (AI) has many impressive applications, including
 - playing chess and Go,
 - self-driving cars, computer vision,
 - speech recognition.
- We demonstrate that dynamic economic models can be effectively analyzed by using the same break-ground AI techniques and the same combinations of hardware and software as those used by data scientists for dealing with their impressive applications.
- Using AI approach, we will be able to solve high-dimensional models with thousands of state variables that were intractable under conventional solution methods.

Plan of minicourse

Day 1.

1. Introduction to machine learning.
2. Multilayer neural networks. Deep learning.
3. Hands-on session: a python and Tensor ow deep learning code for solving a neoclassical growth model.

Day 2.

1. Unsupervised learning (clustering, dimensionality reduction).
2. Applications of unsupervised learning for economic dynamics.
3. Smolyak sparse grids analysis.
4. Hands-on session: bToTEM central banking model of the Bank of Canada for projection and policy analysis.

Day 3.

1. Introduction to reinforcement learning
2. Reinforcement learning for solving large scale heterogeneous-agent models.
3. Hands-on session: a python and Tensor ow deep learning code for solving a heterogeneous agent Krusell-Smith model.
4. Conclusion.

Definitions of machine learning

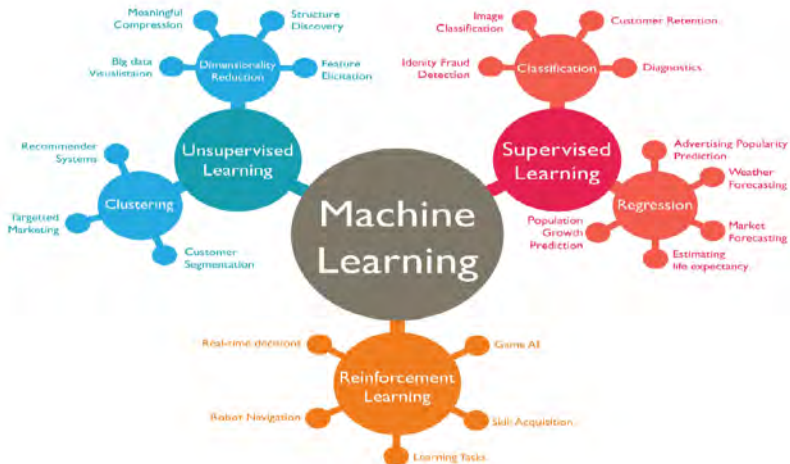
- Machine learning (ML) is a collection of algorithms and models that computer systems use to perform a specific task without being explicitly programmed, relying on patterns and inference instead.
- The name "machine learning" was coined in 1959 by Arthur Samuel (creator of first checkers-playing platform).
- Tom M. Mitchell provided a widely quoted, more formal definition of machine learning:
"A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P if its performance at tasks in T , as measured by P , improves with experience E ."
- For example, the task T is to recognize dogs in images, P is the fraction of correct classifications, and E is the amount of data (images) used for learning.

Machine learning categories

Machine learning (ML) can be classified into three broad categories:

- Supervised learning
- Unsupervised learning
- Reinforcement learning

Machine learning categories in a diagram



Source: <https://towardsdatascience.com/machine-learning-types-2-c1291d4f04b1>

Supervised learning

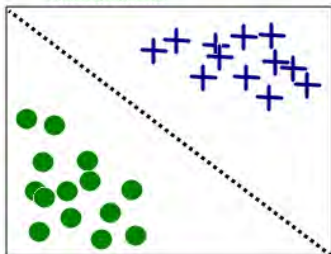
Supervised learning: The computer is presented with example inputs and their desired outputs, labeled by a “teacher”, and the goal is to learn a general rule that maps inputs to outputs.

Supervised learning can be subdivided into two types:

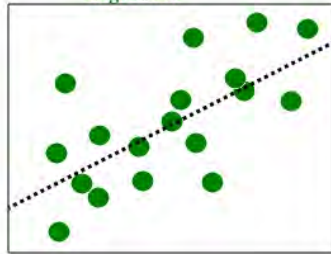
- **Classification:** Inputs are divided into two or more classes, and the learner must produce a model that assigns unseen inputs to one or more of these classes.
 - Spam filtering is an example of classification, where the inputs are email (or other) messages and the classes are “spam” and “not spam”.
- **Regression:** It is also a supervised learning problem, but the outputs are continuous rather than discrete.
 - For example, predicting the stock prices using historical data.

Illustration of classification and regression

Classification



Regression



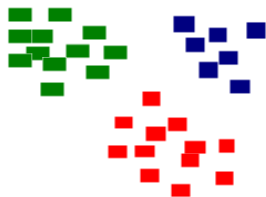
Unsupervised learning

Unsupervised learning: No labels are given to the learning algorithm, leaving it on its own to find structure in its input, for example, discovering hidden patterns in data.

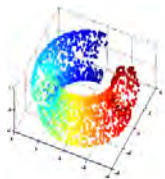
Examples of unsupervised learning:

- **Clustering:** Divide a set of inputs into groups which are not known beforehand.
- **Dimensionality reduction:** Simplify inputs by mapping them into a lower-dimensional space.

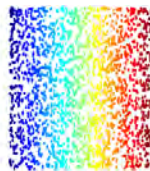
Examples of unsupervised learning



a) Clustering



(a)



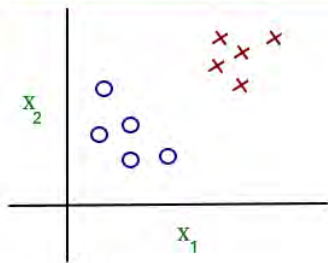
(b)

b) Dimensionality reduction

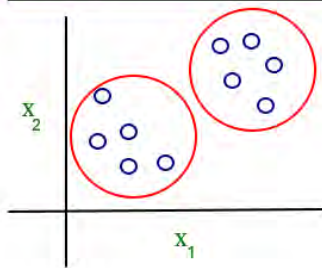
Supervised versus unsupervised learning

The data in supervised learning is labelled, while the data in unsupervised learning is not labeled.

Supervised Learning



Unsupervised Learning



Examples of learning algorithms

Question: Which of the problems below are best addressed by using supervised or unsupervised learning algorithms?

1. Given 50 articles written by male authors, and 50 articles written by female authors, learn to predict the gender of a new manuscript's author (when the identity of this author is unknown).
2. Examine a large collection of emails that are known to be spam email, to discover if there are sub-types of spam mail.
3. Given historical data of children's ages and heights, predict children's height as a function of their age.
4. Take a collection of 1000 essays written on the US Economy, and find a way to automatically group these essays into a small number of groups of essays that are somehow "similar" or "related".

Semi-supervised learning

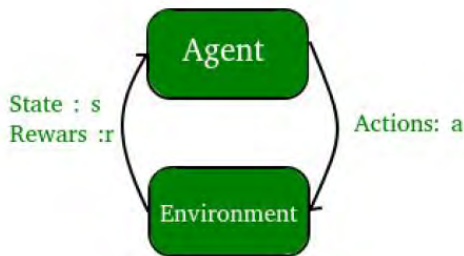
- **Semi-supervised learning:** Problems where you have a large amount of input data and only some of the data is labeled, are called semi-supervised learning problems.
- These problems sit in between both supervised and unsupervised learning.
- For example, a photo archive where only some of the images are labeled, (e.g. dog, cat, person) and the majority are unlabeled.

Reinforcement learning

- **Reinforcement learning** is concerned with how software agents ought to take actions in an environment so as to maximize some notion of cumulative reward.
- Due to its generality, reinforcement learning is studied in many other disciplines, such as game theory, control theory, operations research, information theory, simulation-based optimization, multi-agent systems, statistics and genetic algorithms.
- In machine learning, the environment is typically represented as a Markov Decision Process (MDP).
- Many reinforcement learning algorithms use dynamic programming techniques.

Reinforcement learning (cont.)

- A computer program interacts with a dynamic environment in which it must perform a certain goal (such as driving a vehicle or playing a game against an opponent).
- The program is provided feedback in terms of rewards and punishments as it navigates its problem space.



- *Google Alpha Zero player uses reinforcement learning to play Go game and other games.*

<https://www.youtube.com/watch?v=V1eYniJ0Rnk>

Online versus offline learning

- In computer science, **online** machine learning is a method of machine learning in which data becomes available in a sequential order and is used to update our best predictor for future data at each step, as opposed to **offline** batch learning techniques which generate the best predictor by learning on the entire training data set at once.
- Online learning is a common technique used in areas of machine learning where it is computationally infeasible to train over the entire dataset, requiring the need of out-of-core algorithms.
- It is also used in situations where it is necessary for the algorithm to dynamically adapt to new patterns in the data, or when the data itself is generated as a function of time, e.g., stock price prediction.

Terminology of machine learning versus econometrics

Machine learning systems learn how to combine input to produce useful predictions on never-before-seen data.

- **Feature** – A feature is an input variable—the x variable in simple linear regression. Sophisticated machine learning project could use millions of features (x_1, \dots, x_p).
- **Target (label)** – A label is the thing we're predicting—the y variable in simple linear regression. The label could be the future price of wheat, the animal shown in a picture, the meaning of an audio clip, or just about anything.
- **Model** – A model defines the relationship between features and label, e.g, a spam detection model associate certain features with "spam" .
- **Training** – Training means learning the model. You show the model labeled examples and enable it to gradually learn the relationships between features and label by minimizing **a loss function**.
- **Prediction** – Once our model is ready, you can feed it a set of inputs and it will provide a predicted output (label).

Linear regression example

Example: Consider a linear regression equation

$$y = X\theta + \varepsilon.$$

(We call regression coefficients θ instead of β —the standard notation in machine learning).

- **A set of features** is X .
- **Target (label)** is y .
- **Model** is $y = X\theta + \varepsilon$.
- **Loss function** is squared sum of residuals \Rightarrow *you want to have zero loss ideally*

$$\min_{\beta} \varepsilon'\varepsilon = (y - X\theta)'(y - X\theta).$$

- **Training** is means to find $\hat{\theta}$ that minimizes the loss function:

$$\theta_{OLS} = (X'X)^{-1} X'y.$$

- **Prediction** is the target (label) $y = \tilde{X}\theta_{OLS}$ that corresponds to some new features \tilde{X} .

Machine learning algorithms

There are algorithms and techniques which are used to accomplish ML tasks including:

- Linear Regression
- Nonlinear Regression
- Logistic Regression
- Decision Tree
- Support vector machines
- Naive Bayes
- K nearest neighbors
- K-Means
- Ensemble methods like Random Forest, etc.

The ML field is huge, so we will not try to survey all the algorithms but focus on those algorithms that we find useful for constructing solutions to dynamic economic models.

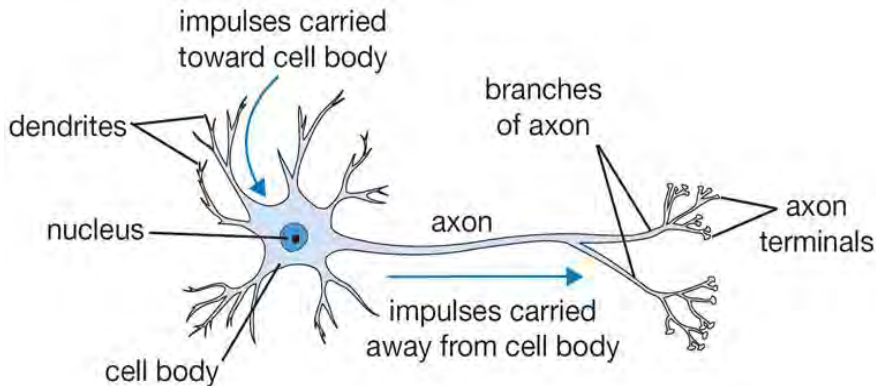
Artificial intelligence versus machine learning

Machine learning is related to **artificial intelligence** and **deep learning**.

- In computer science, **artificial intelligence (AI)**, sometimes called machine intelligence, is intelligence demonstrated by machines, in contrast to the natural intelligence displayed by humans.
- Colloquially, the term "artificial intelligence" is often used to describe machines (or computers) that mimic "cognitive" functions that humans associate with the human mind, such as "learning" and "problem solving".
- **Machine learning** is fundamentally operational definition rather than definition of AI in cognitive terms.
- It follows Alan Turing's proposal in his paper "Computing Machinery and Intelligence", in which the question "Can machines think?" is replaced with the question "Can machines do what we (as thinking entities) can do?".

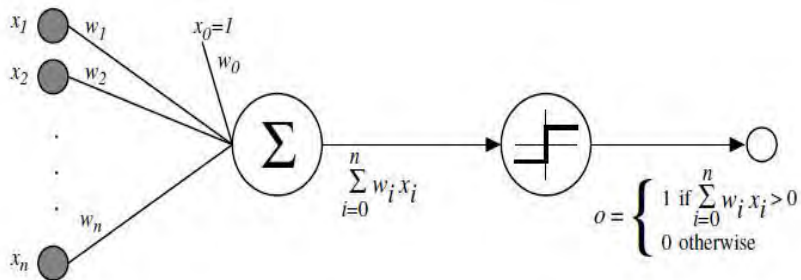
Deep learning

Deep learning is associated with multilayer neural networks and motivated by biological neuron.



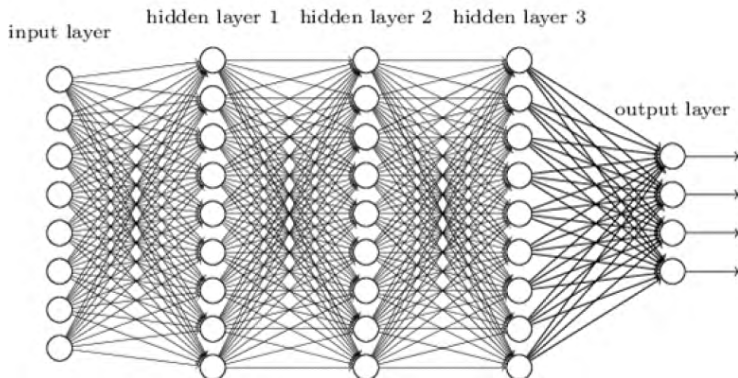
Deep learning (cont.)

Artificial neuron: perceptron

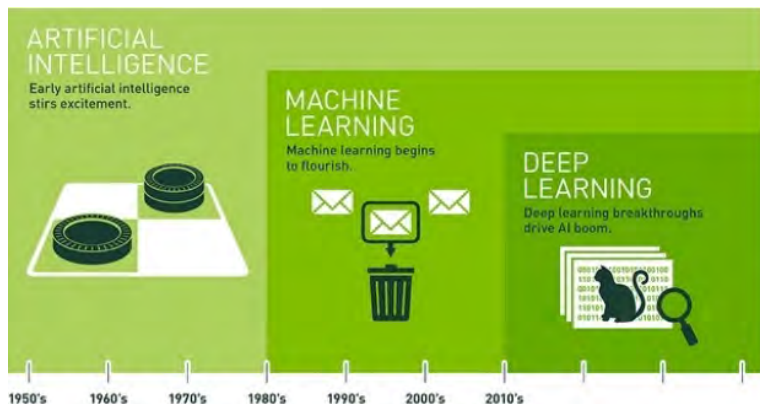


Deep learning (cont.)

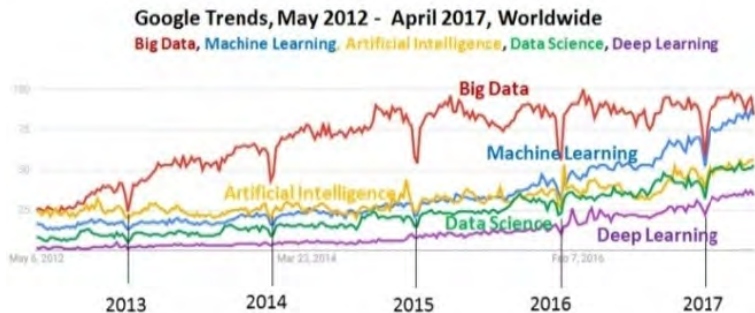
Deep neural network



Artificial intelligence, machine learning and deep learning



The rise of machines



Are the machines really intelligent?

- Alan Turing (1950) proposed **Turing test** to determine whether or not computer(machine) can think intelligently like human.
Imagine a game of three players having two humans and one computer, an interrogator(as human) is isolated from other two players. The interrogator tries to figure out which one is human and which one is computer by asking questions from both of them.
- John Searle (1980) proposed **Chinese room argument** that suggests that the Turing test could not be used to determine “whether or not a machine is considered as intelligent like humans”.
Imagine a machine that has a huge database containing questions and answers in Chinese. When a interrogator ask the question, the machine is locating the question in the database and returning the corresponding answer to the interrogator. The whole scenario would seems like that human is returning the answer but the machine has no understanding of those questions and answers and therefore we can't consider such machine as intelligent.

⇒ **We will not be concerned whether machine is intelligent or not but with how it can help solve economic models.**

Dynamic economic models, conventional solution methods and the curse of dimensionality

- A one-agent stochastic growth model:

$$\max_{\{k_{t+1}, c_t\}_{t=0}^{\infty}} E_0 \sum_{t=0}^{\infty} \beta^t u(c_t)$$

$$\text{s.t. } c_t + k_{t+1} = (1 - \delta) k_t + z_t f(k_t),$$

$$\ln z_{t+1} = \rho \ln z_t + \epsilon_{t+1}, \quad \epsilon_{t+1} \sim \mathcal{N}(0, \sigma^2),$$

initial condition (k_0, z_0) is given;

$f(\cdot)$ = production function;

c_t = consumption; k_{t+1} = capital; z_t = productivity level;

β = discount factor; δ = depreciation rate of capital;

ρ = autocorrelation coefficient of the productivity level;

σ = standard deviation of the productivity shock ϵ_{t+1} .

Concept of solution

- *Objective:* to find a recursive Markov solution in which the decisions on next-period capital and consumption are made according to some time invariant state-contingent functions

$$k' = K(k, z).$$

- Note that if the capital decision function K is known, the consumption decision function follows from the budget constraint:

$$C(k, z) = (1 - \delta)k_t + z_t f(k_t) - K(k, z).$$

Euler equation

- We construct a solution that satisfies the first-order condition (Euler equation):

$$u_1(c_t) = \beta E_t [u_1(c_{t+1}) (1 - \delta + z_{t+1} f_1(k_{t+1}))],$$

where the transition equations are

$$\begin{aligned} c_t + k_{t+1} &= (1 - \delta) k_t + z_t f(k_t). \\ \ln z_{t+1} &= \rho \ln z_t + \epsilon_{t+1}, \quad \epsilon_{t+1} \sim \mathcal{N}(0, \sigma^2), \end{aligned}$$

Residuals in the Euler equation

- The exact solution $k_{t+1} = K(k_t, z_t)$ implies zero residuals in the Euler equation for each state (k_t, z_t) :

$$R(k_t, z_t) = u_1(c_t) - \beta E_t [u_1(c_{t+1}) (1 - \delta + z_{t+1} f_1(k_{t+1}))] \equiv 0,$$

where the future variables are given by

$$\begin{aligned}c_t &= (1 - \delta) k_t + z_t f(k_t) - k_{t+1}, \\c_{t+1} &= (1 - \delta) k_{t+1} + z_{t+1} f(k_{t+1}) - k_{t+2}, \\ \ln z_{t+1} &= \rho \ln z_t + \epsilon_{t+1}, \quad \epsilon_{t+1} \sim \mathcal{N}(0, \sigma^2),\end{aligned}$$

Approximate solution to the Euler equation

- Thus, we can construct approximate solution $k_{t+1} = \widehat{K}(k_t, z_t; \theta)$ to minimize squared residuals in the Euler equation:

$$\widehat{R}(k_t, z_t; \theta) = \{u_1(c_t) - \beta E_t[u_1(c_{t+1})(1 - \delta + z_{t+1}f_1(k_{t+1}))]\}^2,$$

where the variables c_t , c_{t+1} and z_{t+1} are given by

$$\begin{aligned}c_t &= (1 - \delta)k_t + z_t f(k_t) - k_{t+1}, \\c_{t+1} &= (1 - \delta)k_{t+1} + z_{t+1}f(k_{t+1}) - \widehat{K}(k_{t+1}, z_{t+1}; \theta), \\ \ln z_{t+1} &= \rho \ln z_t + \epsilon_{t+1}, \quad \epsilon_{t+1} \sim \mathcal{N}(0, \sigma^2),\end{aligned}$$

Global projection-style Euler equation method

This method minimizes average or maximum residuals on the grid

A global projection-style Euler equation method in line with Judd (1992).

Step 1 Choose functional form $\widehat{K}(\cdot, \theta)$ for representing K , where θ is the coefficients vector. Choose a grid $\{k_m, z_m\}_{m=1, \dots, M}$ on which \widehat{K} is constructed.

Step 2 Choose nodes, ϵ_j , and weights, ω_j , $j = 1, \dots, J$, for approximating integrals. Compute next-period productivity $z'_{m,j} = z'_m \exp(\epsilon_j)$ for all j, m .

Step 3 Solve for θ that minimizes the squared sum of residuals on the grid:

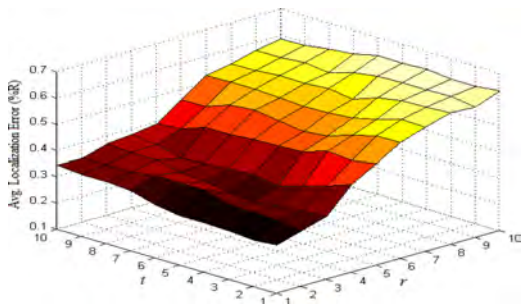
$$\min_{\theta} \Xi(\theta) = \frac{1}{M} \sum_{m=1}^M \left\{ u_1(c_m) \square \beta \sum_{j=1}^J \omega_j \cdot [u_1(c'_{m,j}) (1 \square \delta + z'_{m,j} f_1(k'_m))] \right\}^2,$$
$$c_m = (1 \square \delta) k_m + z_m f(k_m) \square k'_m$$
$$c'_{m,j} = (1 \square \delta) k'_m + z'_{m,j} f(k'_m) \square k''_{m,j}$$
$$k'_m = \widehat{K}(k_m, z_m; \theta)$$
$$k''_m = \widehat{K}(k'_m, z'_m; \theta)$$

Note that $\min_{\theta} \Xi(\theta)$ is the usual static optimization problem.

Implementation details of the projection algorithm

To implement the algorithm, we need to make the following choices:

1. A grid of points $\{k_m, z_m\}_{m=1, \dots, M}$ on which K is approximated, for example, 10 points for k and 10 points for z , which gives $10^2 = 100$ grid points in total



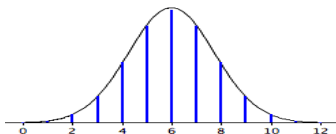
2. A method for interpolation off the grid, for example, splines.

Implementation details of the projection algorithm (cont.)

3. Numerical integration methods for approximating conditional expectation function $E_t [u_1 (c_{t+1}) (1 - \delta + z_{t+1} f_1 (k_{t+1}))]$ typically using weighted average

$$E_t [G (\epsilon_{t+1})] = \int_{-\infty}^{\infty} G (\epsilon) w (\epsilon) d\epsilon \approx \sum_{j=1}^J \omega_j G (\epsilon_j)$$

for example, 10 future shocks $\epsilon_1, \dots, \epsilon_{10}$ weighted by w_1, \dots, w_{10}

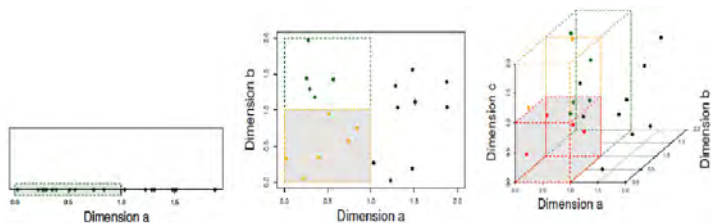


4. Solver that finds a minimizer θ to $\min_{\theta} \Xi (\theta)$, for example, a version of gradient descent method $\theta_{\ell+1} = \theta_{\ell} - \alpha \frac{\partial J(\theta_{\ell})}{\partial \theta}$ where ℓ is iteration and α is the Newton step, direction step, etc.

This techniques methods work remarkably well for small problems but becomes intractable when the dimensionality increases.

Curses of dimensionality in high dimensional problems

1. The number of grid points increases exponentially with the number of state variables, for example, with d state variables, we have 10^d grid points if we have 10 grid points in each dimension



2. Spline interpolation is intractable in high dimensions.
3. The number of integration nodes increases exponentially with number of shocks. For example, if we have $\epsilon_1^1, \dots, \epsilon_{10}^1$ for shock 1 and $\epsilon_1^2, \dots, \epsilon_{10}^2$ for shock 2, we get $10^2 = 100$ integration nodes.
4. The cost of solving $\min_{\theta} \Xi(\theta)$ also increases rapidly.

Dealing with curse of dimensionality in computational economics

There is a number of approaches in computational economics for alleviating the curse of dimensionality; see Chapter 7 of Handbook of Computational Economics by Maliar and Maliar (2014):

- Smolyak sparse grids,
- Sparse polynomial interpolation,
- Low discrepancy sequences,
- Non-product monomial integration,
- Derivative-free fixed point iteration method, etc.

We will not study these approaches today but focus on data science methods that offer an alternative to conventional computational economics.

Machine learning approaches for dealing with the curse of dimensionality

Unsupervised learning

- Simulation
- Clustering
- Dimensionality reduction
- Principle component analysis
- Regularization

Supervised (deep) learning

- multilayer neural network
- stochastic optimization

We will see that these methods make it possible to study dynamic models with thousands of state variables intractable up to now.

Machine learning, artificial intelligence and deep learning methods for dynamic economic models: **Numerical Methods for Macro Models**

Lilia Maliar, Serguei Maliar

November 2022

Minicourse

Model with elastic labor supply: a divisible-labor version

We consider a standard growth *model with elastic labor supply*. The agent solves:

$$\begin{aligned} \max_{\{k_{t+1}, c_t, \ell_t\}_{t=0, \dots, \infty}} E_0 \left\{ \sum_{t=0}^{\infty} \beta^t u(c_t, \ell_t) \right\} \\ \text{s.t. } c_t + k_{t+1} = (1 - \delta) k_t + \theta_t f(k_t, \ell_t), \\ \ln \theta_{t+1} = \rho \ln \theta_t + \sigma \epsilon_{t+1}, \quad \epsilon_{t+1} \sim \mathcal{N}(0, 1), \end{aligned}$$

where initial condition (k_0, θ_0) is given;

$f(\cdot)$ = production function;

c_t = consumption; k_{t+1} = capital; θ_t = productivity level;

β = discount factor; δ = depreciation rate of capital;

ρ = autocorrelation coefficient of the productivity level;

σ = standard deviation of the productivity shock ϵ_{t+1} .

Model with elastic labor supply: a divisible-labor version (cont.)

- Assume that the agents value leisure.
1 = total time endowment,
 l_t = leisure,
 ℓ_t = working hours.
- The agent can choose any number of working hours between 0 and 1.

$$\ell_t + l_t = 1.$$

- $u(c_t, l_t)$ = the momentary utility (strictly increasing, and concave).
- A common assumption is the CRRA utility function:

$$u(c_t, l_t) = \frac{(c_t^v l_t^{1-v})^{1-\sigma}}{1-\sigma},$$

v = share of consumption; σ = coefficient of relative risk aversion.

- If $\sigma = 1$, then $u(c_t, l_t) = \ln c_t + A \ln l_t$.

Time invariant decision functions

- Our goal is to solve for a recursive Markov equilibrium in which the decisions on next-period capital, consumption and labor are made according to some time invariant state contingent functions

$$k' = K(k, \theta), \quad c = C(k, \theta), \quad \ell = \mathcal{L}(k, \theta).$$

- A version of model in which the agent does not value leisure and supplies to the market all her time endowment is referred to as a *model with inelastic labor supply*.
- Such model is obtained by replacing $u(c_t, \ell_t)$ and $f(k_t, \ell_t)$ with $u(c_t)$ and $f(k_t)$, respectively.

First-order conditions

We assume that a solution to the model is interior and satisfies budget constraint

$$c_t + k_{t+1} = (1 - \delta) k_t + \theta_t f(k_t, \ell_t)$$

and the first-order conditions (FOCs)

$$u_1(c_t, \ell_t) = \beta E_t \{u_1(c_{t+1}, \ell_{t+1}) [1 - \delta + \theta_{t+1} f_1(k_{t+1}, \ell_{t+1})]\}, \quad (1)$$

$$u_2(c_t, \ell_t) = u_1(c_t, \ell_t) \theta_t f_2(k_t, \ell_t). \quad (2)$$

- FOC (1) is the *Euler equation* or inter-temporal FOC (relates variables of different periods).
- FOC (2) is intra-temporal FOC (relates variables within the same period).

Usual equations versus functional equations

- We need to find a solution to functional equations.
- What does it mean functional equation? $y + 2x = 7$ and $2y + 3x = 5$ is a system of 2 usual equation and solution to them are some numbers x, y .
- A solution to functional equation is a function, i.e., x, y are functions.
- What f is the solution to $f(x) = x$?
- We approximate functions K , C , and \mathcal{L} that solve a system of functional equations numerically.
- Let us consider a projection-style method in line with Judd (1992) that approximates these functions to satisfy the FOCs on a grid of points.

Global projection-style Euler equation method

(EEM): A global projection-style Euler equation method.

Step 1 Choose functional form $\widehat{K}(\cdot, b)$ for representing K , where b is the coefficients vector. Choose a grid $\{k_m, \theta_m\}_{m=1, \dots, M}$ on which \widehat{K} is constructed.

Step 2 Choose nodes, ϵ_j , and weights, ω_j , $j = 1, \dots, J$, for approximating integrals. Compute next-period productivity $\theta'_{m,j} = \theta_m^\rho \exp(\epsilon_j)$ for all j, m .

Step 3 Solve for b that approximately satisfies the model's equations:

$$\begin{aligned}u_1(c_m, l_m) &= \beta \sum_{j=1}^J \omega_j \cdot [u_1(c'_{m,j}, l'_{m,j}) (1 - \delta + \theta'_{m,j} f_1(k'_{m,j}, l'_{m,j}))], \\u_2(c_m, l_m) &= u_1(c_m, l_m) \theta_m f_2(k_m, l_m), \\c_m &= (1 - \delta) k_m + \theta_m f(k_m, l_m) - k'_m \\u_2(c'_{m,j}, l'_{m,j}) &= u_1(c'_{m,j}, l'_{m,j}) \theta'_{m,j} f_2(k'_{m,j}, l'_{m,j}), \\c'_{m,j} &= (1 - \delta) k'_m + \theta'_{m,j} f(k'_{m,j}, l'_{m,j}) - k''_{m,j}\end{aligned}$$

We have $2J + 3$ equations and $3J + 3$ unknowns $k'_m, c_m, l_m, \{k''_{m,j}, c'_{m,j}, l'_{m,j}\}_{j=1}^J$

Discussion of Step 3

We use **stationarity**: the same decision function $\widehat{K}(\cdot, b)$ is used both at t and $t + 1$:

- in the current period: $k'_m = \widehat{K}(k_m, \theta_m; b)$;
- in J possible future states: $k''_{m,j} = \widehat{K}(k'_m, \theta'_{m,j}; b)$, where future shocks are $\theta'_{m,j} = \theta_m^\rho \exp(\epsilon_j)$.

$$u_1(c_m, \ell_m) = \beta \sum_{j=1}^J \omega_j \left[u_1(c'_{m,j}, \ell'_{m,j}) + \delta + \theta'_{m,j} f_1(k'_m, \ell'_{m,j}) \right],$$

$$u_2(c_m, \ell_m) = u_1(c_m, \ell_m) \theta_m f_2(k_m, \ell_m),$$

$$c_m = (1 - \delta) k_m + \theta_m f(k_m, \ell_m) - k'_m$$

$$u_2(c'_{m,j}, \ell'_{m,j}) = u_1(c'_{m,j}, \ell'_{m,j}) \theta'_{m,j} f_2(k'_m, \ell'_{m,j}),$$

$$c'_{m,j} = (1 - \delta) k'_m + \theta'_{m,j} f(k'_m, \ell'_{m,j}) - \widehat{K}(k'_m, \theta'_{m,j}; b)$$

$2J + 3$ equations and $2J + 3$ unknowns: $c_m, \ell_m, k'_m, \{c'_{m,j}, \ell'_{m,j}\}_{j=1}^J$.

- The coefficients b are obtained by fitting $\widehat{K}(k_m, \theta_m; b)$ to k'_m .

Unidimensional grid points and basis functions

- To solve the model, we discretize the state space in Step 1 into a finite set of grid points $\{k_m, \theta_m\}_{m=1, \dots, M}$.
- Our construction of a multidimensional grid begins with unidimensional grid points and basis functions.
- The simplest possible choice is a family of ordinary polynomials and a grid of uniformly spaced points.
- However, many other choices are possible.
- In particular, a useful alternative is a family of Chebyshev polynomials and a grid composed of extrema of Chebyshev polynomials.
- Such polynomials are defined in the interval $[-1, 1]$, and thus, the model's variables such as k and θ must be rescaled to be inside this interval prior to any computation.

Why Chebyshev? Problems with interpolation

- Interpolation does not always work even for well-behaved functions.
- Does $p_n(x)$ converge to $f(x)$ as we use more points? No!
- Consider a so-called Runge function

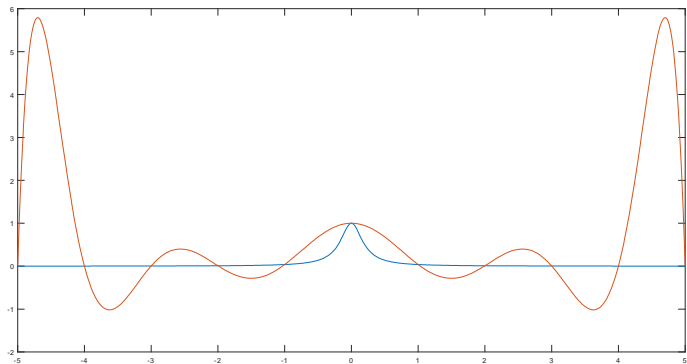
$$f(x) = \frac{1}{1+x^2},$$
$$x_i = -5, -4, \dots, 3, 4, 5$$

i.e., 11 uniformly sampled points.

- Use Lagrange interpolation (fit degree 10 polynomial).
- *Conclusion:* under $(n \square 1)$ interpolation at n uniformly spaced points, $p_n(x)$ gets worse as we use more points.

Problems with Interpolation (cont.)

Runge function



Highly oscillatory. Hard to control away from measured points.

Chebyshev Polynomials

- $D = [a, b] = [-1, 1]$
- $w(x) = \sqrt{1 - x^2}$
- $T_n(x) = \cos(n \arccos(x))$
- Recurrence formula:

$$T_0(x) = 1$$

$$T_1(x) = x$$

$$T_2(x) = 2x^2 - 1$$

$$T_3(x) = 4x^3 - 3x$$

$$T_4(x) = 8x^4 - 8x^2 + 1$$

...

$$T_{n+1}(x) = 2x T_n(x) - T_{n-1}(x).$$

- This recurrence formula follows from the recurrence relation for cosines:

$$\cos((n+1)\theta) = 2 \cos(\theta) \cos(n\theta) - \cos((n-1)\theta).$$

Chebyshev versus ordinary polynomials

Figure 1a. Ordinary polynomials.

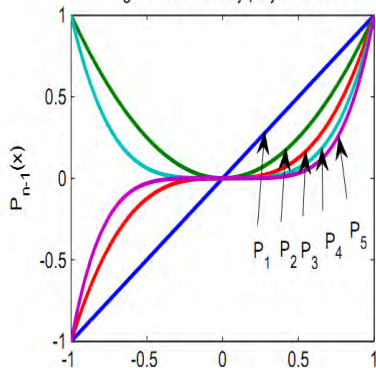
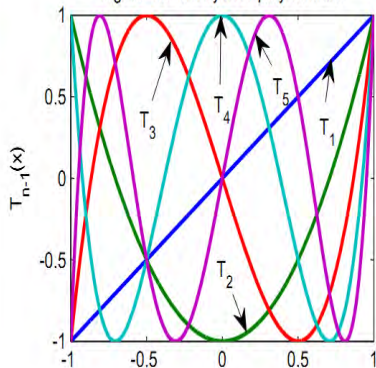


Figure 1b. Chebyshev polynomials.



Orthogonal Property

General orthogonal polynomials

- In linear algebra, we use orthogonal basis. Why?
- We want to choose orthogonal basis in functional space.
- Space: polynomials over domain D
- Weighting function: $w(x) > 0$
- Inner product of two functions f and g : $\langle f, g \rangle \equiv \int_D f(x)g(x)w(x)dx$

Definition

$\{\phi_i\}$ is a family of orthogonal polynomials w.r.t $w(x)$ iff

$$\langle \phi_i, \phi_j \rangle = 0, \quad i \neq j$$

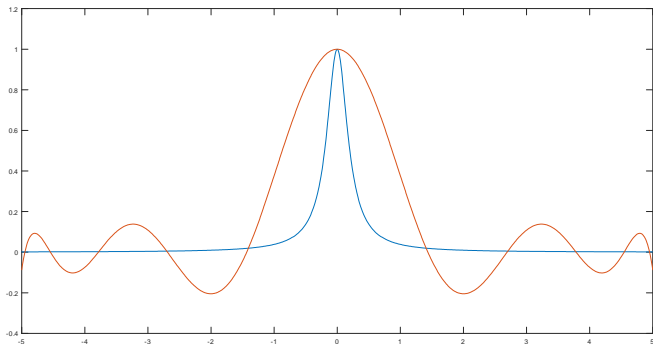
Orthogonal property verified.

- Again, $T_{n+1}(x) = 2x T_n(x) - T_{n-1}(x)$.
- For example, $T_3(x) = 4x^3 - 3x$, and
 $T_4(x) = 8x^4 - 8x^2 + 1 = 8x^4 - 8x^2 + 1$.
- Check the orthogonality $\int_{-1}^1 T_i(x)T_j(x)w(x)dx = 0$:

$$\begin{aligned} 0 &= \int_{-1}^1 T_0(x)T_1(x)w(x)dx \\ &= \int_{-1}^1 1 \cdot x \cdot (1 - x^2)^{1/2} dx = \int_{-1}^1 x \cdot (1 - x^2)^{1/2} \frac{dx^2}{2} \\ &= \frac{1}{2} \int_{y=x^2}^1 (1 - y)^{1/2} dy = 0. \end{aligned}$$

Approximating the Runge function with Chebyshev polynomials

- Using Chebyshev nodes to approximate the Runge function improves approximation (but the result is not completely good).



Chebyshev polynomials for approximation

Let us illustrate the use of Chebyshev polynomials for approximation by way of example.

Example

Let $f(x)$ be a function defined on an interval $[-1, 1]$, and let us approximate this function with a Chebyshev polynomial function of degree two, i.e.,

$$f(x) \approx \hat{f}(x; b) = b_1 + b_2x + b_3(2x^2 - 1).$$

We compute $b \equiv (b_1, b_2, b_3)$ so that $\hat{f}(\cdot; b)$ and f coincide in three extrema of Chebyshev polynomials, namely, $\{-1, 0, 1\}$,

$$\hat{f}(-1; b) = b_1 + b_2 \cdot (-1) + b_3(2 \cdot (-1)^2 - 1) = f(-1)$$

$$\hat{f}(0; b) = b_1 + b_2 \cdot 0 + b_3(2 \cdot 0^2 - 1) = f(0)$$

$$\hat{f}(1; b) = b_1 + b_2 \cdot 1 + b_3(2 \cdot 1^2 - 1) = f(1).$$

Example

(cont.) This leads us to a system of three linear equations with three unknowns that has a unique solution

$$\begin{aligned} \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix} &= \begin{bmatrix} 1 & \square 1 & 1 \\ 1 & 0 & \square 1 \\ 1 & 1 & 1 \end{bmatrix}^{-1} \begin{bmatrix} f(\square 1) \\ f(0) \\ f(1) \end{bmatrix} \\ &= \begin{bmatrix} \frac{1}{4} & \frac{1}{2} & \frac{1}{4} \\ \square \frac{1}{2} & 0 & \frac{1}{2} \\ \frac{1}{4} & \square \frac{1}{2} & \frac{1}{4} \end{bmatrix} \begin{bmatrix} f(\square 1) \\ f(0) \\ f(1) \end{bmatrix} = \begin{bmatrix} \frac{f(\square 1)}{4} + \frac{f(0)}{2} + \frac{f(1)}{4} \\ \square \frac{f(\square 1)}{2} + \frac{f(1)}{2} \\ \frac{f(\square 1)}{4} \square \frac{f(0)}{2} + \frac{f(1)}{4} \end{bmatrix}. \end{aligned}$$

- It is possible to use Chebyshev polynomials with other grids, but the grid of extrema (or zeros) of Chebyshev polynomials is a perfect match.

Multidimensional grid points and basis functions

- In Step 1 of the Euler equation algorithm, we must specify a method for approximating, representing, and interpolating two-dimensional functions.
- A tensor-product method constructs multidimensional grid points and basis functions using all possible combinations of unidimensional grid points and basis functions.
- As an example, let us approximate the capital decision function K .
- First, we take two grid points for each state variable, namely, $\{k_1, k_2\}$ and $\{\theta_1, \theta_2\}$, and we combine them to construct two-dimensional grid points, $\{(k_1, \theta_1), (k_1, \theta_2), (k_2, \theta_1), (k_2, \theta_2)\}$.
- Second, we take two basis functions for each state variable, namely, $\{1, k\}$ and $\{1, \theta\}$, and we combine them to construct two-dimensional basis functions $\{1, k, \theta, k\theta\}$.
- Third, we construct a flexible functional form for approximating K ,

$$\widehat{K}(k, \theta; b) = b_1 + b_2k + b_3\theta + b_4k\theta. \quad (3)$$

Multidimensional grid points and basis functions (cont.)

- Finally, we identify the four unknown coefficients $(b_1, b_2, b_3, b_4) \equiv b$ such that $K(k, \theta)$ and $\widehat{K}(k, \theta; b)$ coincide exactly in the four grid points constructed.
- That is, we write $\mathcal{B}b = w$, where

$$\mathcal{B} = \begin{bmatrix} 1 & k_1 & \theta_1 & k_1\theta_1 \\ 1 & k_1 & \theta_2 & k_1\theta_2 \\ 1 & k_2 & \theta_1 & k_2\theta_1 \\ 1 & k_2 & \theta_2 & k_2\theta_2 \end{bmatrix}, \quad b = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \end{bmatrix}, \quad w = \begin{bmatrix} K(k_1, \theta_1) \\ K(k_1, \theta_2) \\ K(k_2, \theta_1) \\ K(k_2, \theta_2) \end{bmatrix}.$$

- If \mathcal{B} has full rank, then coefficients vector b is uniquely determined by $b = \mathcal{B}^{-1}w$.
- The obtained approximation can be used to interpolate the capital decision function in each point off the grid.

Multidimensional methods: Lagrange Interpolation (cont.)

Odd interpolation example

- Interpolation nodes:

$$\{P_1, P_2, P_3, P_4\} \equiv \{(1, 0), (-1, 0), (0, 1), (0, -1)\}$$

- Use linear combinations of $\{1, x, y, xy\}$.
- Data: $z_i = f(P_i), i = 1, 2, 3, 4$.
- Interpolation form $f(x, y) = a + bx + cy + dxy$.
- For example, $P_1 = (x, y) = (1, 0)$.
 $z_1 = f(1, 0) = \theta_0 + \theta_1 \cdot 1 + \theta_2 \cdot 0 + \theta_3 \cdot 1 \cdot 0 = a + b$.
- Defining conditions form the singular system

$$\begin{pmatrix} 1 & 1 & 0 & 0 \\ 1 & -1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 0 & -1 & 0 \end{pmatrix} \begin{pmatrix} \theta_1 \\ \theta_2 \\ \theta_3 \\ \theta_4 \end{pmatrix} = \begin{pmatrix} z_1 \\ z_2 \\ z_3 \\ z_4 \end{pmatrix}.$$

Numerical integration

- For integration, we consider first a simple two-node Gauss-Hermite quadrature method that approximates an integral of a function of a Normally distributed variable $\epsilon \sim \mathcal{N}(0, \sigma^2)$ with a weighted average of just two values $\epsilon_1 = \alpha\sigma$ and $\epsilon_2 = \sigma$ that happen with probability $\omega_1 = \omega_2 = \frac{1}{2}$, i.e.,

$$\int_{-\infty}^{\infty} G(\epsilon) w(\epsilon) d\epsilon \approx G(\epsilon_1) \omega_1 + G(\epsilon_2) \omega_2 = \frac{1}{2} [G(\alpha\sigma) + G(\sigma)],$$

where G is a bounded continuous function, and w is a density function of a Normal distribution, i.e.,

$$u_1(c_m, \ell_m) = \frac{1}{2} \beta \left\{ u_1(c'_{m,\sigma}, \ell'_{m,\sigma}) \left[1 - \delta + \theta'_{m,\sigma} f_1(k'_m(b), \ell'_{m,\sigma}) \right] \right. \\ \left. + u_1(c'_{m,\alpha\sigma}, \ell'_{m,\alpha\sigma}) \left[1 - \delta + \theta'_{m,\alpha\sigma} f_1(k'_m(b), \ell'_{m,\alpha\sigma}) \right] \right\}$$

- Another example is a three-node Gauss-Hermite quadrature method, which uses nodes $\epsilon_1 = 0$, $\epsilon_2 = \sigma\sqrt{\frac{3}{2}}$, $\epsilon_3 = -\sigma\sqrt{\frac{3}{2}}$ and weights $\omega_1 = \frac{2\sqrt{\pi}}{3}$, $\omega_2 = \omega_3 = \frac{\sqrt{\pi}}{6}$.

Numerical integration (cont.)

- It is also a possibility to approximate integrals using Monte Carlo integration, e.g., Parameterized Expectation Algorithm (PEA) by den Haan and Marcet (1990).
- We can make J random draws and approximate an integral with a simple average of the draws,

$$\int_{-\infty}^{\infty} G(\epsilon) w(\epsilon) d\epsilon \approx \frac{1}{J} \sum_{j=1}^J G(\epsilon_j).$$

Numerical integration (cont.)

Let us compare the above integration methods using an example.

Example

Consider a quadratic function $G(\epsilon) = b_1 + b_2\epsilon + b_3\epsilon^2$, where $\epsilon \sim \mathcal{N}(0, \sigma^2)$.

(i) An exact integral is $\mathcal{I} \equiv \int_{-\infty}^{\infty} (b_1 + b_2\epsilon + b_3\epsilon^2) w(\epsilon) d\epsilon = b_1 + b_3\sigma^2$;

(ii) A two-node Gauss-Hermite quadrature integration method yields $\mathcal{I} \approx \frac{1}{2} (b_1 + b_2(\sqrt{\sigma}) + b_3(\sqrt{\sigma})^2) + \frac{1}{2} (b_1 + b_2\sigma + b_3\sigma^2) = b_1 + b_3\sigma^2$;

(iii) A one-node Gauss-Hermite quadrature integration method yields $\mathcal{I} \approx b_1$;

(iv) A Monte Carlo integration method yields

$$\mathcal{I} \approx b_1 + b_2 \left[\frac{1}{J} \sum_{j=1}^J \epsilon_j \right] + b_3 \left[\frac{1}{J} \sum_{j=1}^J \epsilon_j^2 \right].$$

(v) Quasi Monte Carlo methods for integration - the error bounds are provided in Rust (1987).

Numerical integration (cont.)

- Note that the quadrature method with two nodes delivers the exact value of the integral.
- Even with just one node, the quadrature method can deliver accurate integral if G is close to linear (which is often the case in real business cycle models), i.e., $b_3 \approx 0$.
- To assess the accuracy of Monte Carlo integration, let us use $\sigma = 0.01$, which is consistent with the magnitude of fluctuations in real business cycle models.
 - Let us concentrate just on the term $\frac{1}{J} \sum_{j=1}^J \epsilon_j$ for which the expected value and standard deviation are $E \left[\frac{1}{J} \sum_{j=1}^J \epsilon_j \right] = 0$ and $std \left[\frac{1}{J} \sum_{j=1}^J \epsilon_j \right] = \frac{\sigma}{\sqrt{J}}$, respectively.
 - The standard deviation depends on the number of random draws: with one random draw, it is 0.01 and with 1,000,000 draws, it is $\frac{0.01}{\sqrt{1000000}} = 10^{-5}$.

Numerical integration (cont.)

- The last number represents an (expected) error in approximating the integral and restricts the overall accuracy of solutions that can be attained by a solution algorithm using Monte Carlo integration.
- Why is Monte Carlo integration inefficient in this context?
- This is because we compute expectations as do econometricians, who do not know the true density function of the data-generating process and have no choice but to estimate such a function from noisy data using a regression.
- However, when solving an economic model, we do know the process for shocks. Hence, we can construct the "true" density function and we can use such a function to compute integrals very accurately, which is done by the Gauss-Hermite quadrature method.
- This is done in Judd, Maliar and Maliar (2011) who develop generalized stochastic simulation method (GSSA) that attains high accuracy by combining stochastic simulation for constructing the domain and accurate deterministic integration methods.

Optimization methods

- To solve nonlinear equations with respect to the unknown parameters vectors b^k , b^c , b^ℓ .
- This can be done with Newton-style optimization methods.
- Such methods compute first and second derivatives of an objective function with respect to the unknowns and move in the direction of gradient descent until a solution is found.

Gradient descent method for finding the coefficients

- Let us define the residual in the Euler equation:

$$R(k_m, \theta_m; b) = \left\{ u_1(c_m, \ell_m) - \beta \sum_{j=1}^J \omega_j [u_1(c'_{m,j}, \ell'_{m,j}) - (1 - \delta) u_1(c_m, \ell_m) + \theta'_m f_1(k'_m, \ell'_{m,j})] \right\}$$

$$u_2(c_m, \ell_m) = u_1(c_m, \ell_m) \theta_m f_2(k_m, \ell_m),$$

$$c_m = (1 - \delta) k_m + \theta_m f(k_m, \ell_m) \frac{k'_m}{k_m},$$

$$u_2(c'_{m,j}, \ell'_{m,j}) = u_1(c'_{m,j}, \ell'_{m,j}) \theta'_{m,j} f_2(k'_m, \ell'_{m,j}),$$

$$c'_{m,j} = (1 - \delta) k'_m + \theta'_{m,j} f(k'_m, \ell'_{m,j}) \frac{k'_m}{\widehat{K}(k'_m, \theta'_{m,j}; b)}$$

- We must solve for b that minimizes the residuals in the Euler equation $R(k_m, \theta_m; b)$ on the set of grid points $\{k_m, \theta_m\}$.

Derivative free iteration methods

- Newton methods are fast and efficient in small problems but become increasingly expensive when the number of unknowns increases.
- In high-dimensional applications, we may have thousands of parameters in approximating functions, and the cost of computing derivatives may be prohibitive.
- In such applications, derivative-free optimization methods are an effective alternative.
- A useful choice is a fixed-point iteration method that finds a root of $x = F(x)$ by constructing a sequence $x^{(i+1)} = F(x^{(i)})$.

Optimization methods (cont.)

- We illustrate this method using an example.

Example

Consider an equation $x^3 - x - 1 = 0$. Let us rewrite this equation as $x = (x + 1)^{1/3}$ and construct a sequence $x^{(i+1)} = (x^{(i)} + 1)^{1/3}$ starting from $x^{(0)} = 1$. This yields a sequence $x^{(1)} = 1.26$, $x^{(2)} = 1.31$, $x^{(3)} = 1.32, \dots$ which converges to a solution.

- The advantage of fixed-point iteration is that it can iterate in this simple manner on objects of any dimensionality, for example, on a vector of the polynomial coefficients.
- The cost of this procedure does not grow considerably with the number of the polynomial coefficients.

Nonconvergence of fixed-point iteration

- The shortcoming of fixed point iteration is that it does not always converge.

Example

If we wrote the above equation as $x = x^3 + 1$ and implemented fixed-point iteration $x^{(i+1)} = (x^{(i)})^3 + 1$, we would obtain a sequence that diverges to $+\infty$ starting from $x^{(0)} = 1$.

- Damping (partial updating) sometimes can help restore convergence.

Example

We can try $x^{(i+1)} = (1 - \xi)x^{(i)} + \xi \left[(x^{(i)})^3 + 1 \right]$ for some $\xi \in (0, 1)$. If updating is slow $\xi \ll 1$, it typically converges.

Fixed-point iteration method for finding the coefficients

- Let us define the residual in the Euler equation:
- Assume some functions $k' = K(k, \theta)$, $c = C(k, \theta)$, $\ell = \mathcal{L}(k, \theta)$ in the right side and use the system of model's equation to obtain the new functions \widehat{K} , \widehat{C} , $\widehat{\mathcal{L}}$

$$\begin{aligned}
 u_1(c_m, \ell_m) &= \beta \sum_{j=1}^J \omega_j \left[u_1(c'_{m,j}, \ell'_{m,j}) - (1 - \delta + \theta'_{m,j} f_1(k'_m, \ell'_{m,j})) \right], \\
 u_2(c_m, \ell_m) &= u_1(c_m, \ell_m) \theta_m f_2(k_m, \ell_m), \\
 c_m &= (1 - \delta) k_m + \theta_m f(k_m, \ell_m) - k'_m \\
 u_2(c'_{m,j}, \ell'_{m,j}) &= u_1(c'_{m,j}, \ell'_{m,j}) \theta'_{m,j} f_2(k'_m, \ell'_{m,j}), \\
 c'_{m,j} &= (1 - \delta) k'_m + \theta'_{m,j} f(k'_m, \ell'_{m,j}) - \widehat{K}(k'_m, \theta'_{m,j}; b)
 \end{aligned}$$

- Update the decision functions $\widehat{K} = (1 - \lambda) K + \lambda \widehat{K}$, $\widehat{C} = (1 - \lambda) C + \lambda \widehat{C}$ and $\widehat{\mathcal{L}} = (1 - \lambda) \mathcal{L} + \lambda \widehat{\mathcal{L}}$.
- $\lambda \in (0, 1)$ is like the learning rate.

Evaluating accuracy of solutions

- Our solution procedure has two stages. In Stage 1, a method attempts to compute a numerical solution to a model.
- Provided that it succeeds, we proceed to Stage 2, in which we subject a candidate solution to a tight accuracy check.
- We specifically construct a set of points $\{k_i, \theta_i\}_{i=1, \dots, I}$ that covers an area in which we want the solution to be accurate, and we compute unit-free residuals in the model's equations:

$$\mathcal{R}^{BC}(k_i, \theta_i) = \frac{(1 - \delta)k_i + \theta_i f(k_i, \ell_i)}{c_i + k'_i} - 1,$$

$$\mathcal{R}^{EE}(k_i, \theta_i) = \beta E \left\{ \frac{u_1(c'_i, \ell'_i)}{u_1(c_i, \ell_i)} [1 - \delta + \theta'_i f_1(k'_i, \ell'_i)] \right\} - 1,$$

$$\mathcal{R}^{MUL}(k_i, \theta_i) = \frac{u_1(c_i, \ell_i) \theta_i f_2(k_i, \ell_i)}{u_2(c_i, \ell_i)} - 1,$$

where \mathcal{R}^{BC} , \mathcal{R}^{EE} and \mathcal{R}^{MUL} are the residuals in the budget constraint, Euler equation, and FOC for the marginal utility of leisure.

Evaluating accuracy of solutions

- In the exact solution, residuals are zero, so we judge the quality of approximation by how far these residuals are away from zero.
- We should never evaluate residuals on points used for computing a solution in Stage 1 (in particular, for some methods the residuals in the grid points are zeros by construction) but we do so on a new set of points constructed for Stage 2.
- We consider two alternative sets of I points:
 - a fixed rectangular grid
 - a stochastic simulation.
- We report two accuracy measures, namely, the average and maximum absolute residuals across both the optimality conditions and I test points in \log_{10} units, for example, $\mathcal{R}^{BC}(k_i, \theta_i) = \square 2$ means that a residual of $10^{\square 2} = 1\%$, and $\mathcal{R}^{BC}(k_i, \theta_i) = \square 4.5$ means $10^{\square 4.5} = 0.00316\%$.
- Judd, Maliar and Maliar (Econometrica, 2017) show how to construct bounds on approximation errors from the residuals.

Challenges of economic modeling

A curse of dimensionality

- It turned out that not only analytical but also numerical solutions can be expensive (or infeasible) to obtain for many models of interest.
- Curse of dimensionality: the complexity of a problem grows exponentially with the size:
 - assume that there are N capital stocks;
 - take 10 grid points for each capital stock;
 - we obtain 10^N grid points for N capital stocks, e.g., $N = 10 \Rightarrow 10^{10}$ grid points!
- *This is true for approximation, integration and optimization!*
- Economic models can easily become intractable even with supercomputers.

We will show how to use machine learning to solve huge models unthinkable up to now (2000 state variables).

Machine learning, artificial intelligence and deep learning methods for dynamic economic models:

Smolyak method

Lilia Maliar, Serguei Maliar

November 2022

Minicourse

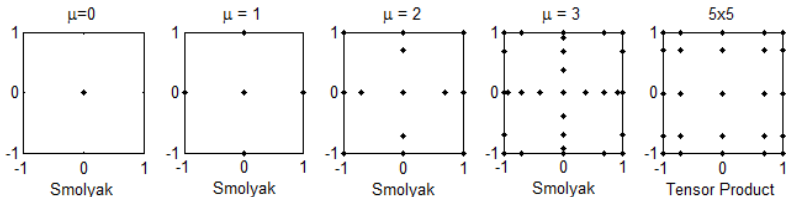
Introduction

Based on: Kenneth L. Judd, Lilia Maliar, Serguei Maliar and Rafael Valero, (2014). “Smolyak Method for Solving Dynamic Economic Models: Lagrange Interpolation, Anisotropic Grid and Adaptive Domain”, Journal of Economic Dynamic and Control 44(C), 92-123.

Introduction

Examples of Smolyak grids

under the approximation levels $\mu = 0, 1, 2, 3$ for the two-dimensional case.



Introduction

Our results: toward more efficient Smolyak interpolation

1. *Efficient construction of Smolyak polynomials.*
 - The nested-set construction of Smolyak polynomials is inefficient: it first creates a long list of repeated elements and then eliminates the repeated elements from the list.
 - We construct Smolyak polynomials using disjoint sets \implies **we avoid costly repetitions of elements.**
2. *A Lagrange-style technique for computing coefficients.*
 - The conventional Smolyak method computes polynomial coefficients using a formula with a large number of nested loops.
 - We compute the coefficients by precomputing a solution to the inverse problem \implies **a simple, general and cheap technique.**
3. *Anisotropic grid: different approximation levels for different variables.*
 - The conventional Smolyak method is symmetric (with the same number of grids and polynomial functions for all variables).
 - We develop an anisotropic version of the Smolyak method \implies **we can vary the quality of approximation across variables.**

Introduction

Our results: adapting Smolyak method to economic applications

4. *Adaptive domain.*

- The Conventional Smolyak method constructs grid points in a normalized multidimensional hypercube $[\square 1, 1]^d$.
- **We show how to effectively adapt the Smolyak hypercube domain to the high-probability set of the given model.**

5. *Computer codes:*

- We provide a simple and reliable MATLAB code for the Smolyak method.

Introduction

Two different classes of sparse grid techniques in the literature:

1. Global polynomial approximation (we fall into this category); *Smolyak (1963), Delvos (1982), Wasilkowski and Woźniakowski (1999) and Berthelmann et al. (2000), etc.*
2. Piecewise approximations based on families of local basis functions; *Griebel (1998), Bungartz and Griebel, (2004), Ma and Zabaras (2009), P üger (2010), Heinkenschloss and Kouri (2012), etc.*

Advantages of global methods:

- a global polynomial approximation is continuously differentiable everywhere.
- our complexity grows polynomially, while the complexity of piecewise basis functions grows exponentially.

However, piecewise approximations allow to refine approximations locally while our possibilities of local refinement are limited.

Conventional Smolyak grid using nested sets

Unidimensional nested sets

- Construct sets of points $i = 1, 2, \dots$ that satisfy two conditions:
 - *Condition 1* Sets $i = 1, 2, \dots$ have $m(i) = 2^{i-1} + 1$ points for $i \geq 2$ and $m(1) \equiv 1$.
 - *Condition 2* Each subsequent set $i + 1$ contains all points of the previous set i . Such sets are called *nested*.
- There are many ways to construct the sets of points, satisfying Conditions 1 and 2.
- As an example, let us consider grid points $\left\{ \square 1, \frac{\square 1}{\sqrt{2}}, 0, \frac{1}{\sqrt{2}}, 1 \right\}$ in the interval $[\square 1, 1]$ and create 3 nested sets of points:
 - $i = 1 : S_1 = \{0\};$
 - $i = 2 : S_2 = \{\square 1, 0, 1\};$
 - $i = 3 : S_3 = \left\{ \square 1, \frac{\square 1}{\sqrt{2}}, 0, \frac{1}{\sqrt{2}}, 1 \right\}.$

Conventional Smolyak grid using nested sets

Tensor products of unidimensional nested sets

	$S_{i_1} \setminus S_{i_2}$	$i_2 = 1$	$i_2 = 2$	$i_2 = 3$
		0	$\square 1, 0, 1$	$\square 1, \frac{\square 1}{\sqrt{2}}, 0, \frac{1}{\sqrt{2}}$
$i_1 = 1$	0	(0, 0)	(0, $\square 1$), (0, 0), (0, 1)	(0, $\square 1$), (0, $\frac{\square 1}{\sqrt{2}}$), (0, 0), (0, 1)
$i_1 = 2$	$\square 1$ 0 1	($\square 1, 0$) (0, 0) (1, 0)	($\square 1, \square 1$), ($\square 1, 0$), ($\square 1, 1$) (0, $\square 1$), (0, 0), (0, 1) (1, $\square 1$), (1, 0), (1, 1)	($\square 1, \square 1$), ($\square 1, \frac{\square 1}{\sqrt{2}}$), ($\square 1, 0$), (0, $\square 1$), (0, $\frac{\square 1}{\sqrt{2}}$), (0, 0), (0, 1), (1, $\square 1$), (1, $\frac{\square 1}{\sqrt{2}}$), (1, 0), (1, 1)
$i_1 = 3$	$\square 1$ $\frac{\square 1}{\sqrt{2}}$ 0 $\frac{1}{\sqrt{2}}$ 1	($\square 1, 0$) ($\frac{\square 1}{\sqrt{2}}, 0$) (0, 0) ($\frac{1}{\sqrt{2}}, 0$) (1, 0)	($\square 1, \square 1$), ($\square 1, 0$), ($\square 1, 1$) ($\frac{\square 1}{\sqrt{2}}, \square 1$), ($\frac{\square 1}{\sqrt{2}}, 0$), ($\frac{\square 1}{\sqrt{2}}, 1$) (0, $\square 1$), (0, 0), (0, 1) ($\frac{1}{\sqrt{2}}, \square 1$), ($\frac{1}{\sqrt{2}}, 0$), ($\frac{1}{\sqrt{2}}, 1$) (1, $\square 1$), (1, 0), (1, 1)	($\square 1, \square 1$), ($\square 1, \frac{\square 1}{\sqrt{2}}$), ($\square 1, 0$), ($\frac{\square 1}{\sqrt{2}}, \square 1$), ($\frac{\square 1}{\sqrt{2}}, \frac{\square 1}{\sqrt{2}}$), ($\frac{\square 1}{\sqrt{2}}, 0$), (0, $\square 1$), (0, $\frac{\square 1}{\sqrt{2}}$), (0, 0), (0, 1), ($\frac{1}{\sqrt{2}}, \square 1$), ($\frac{1}{\sqrt{2}}, \frac{\square 1}{\sqrt{2}}$), ($\frac{1}{\sqrt{2}}, 0$), (1, $\square 1$), (1, $\frac{\square 1}{\sqrt{2}}$), (1, 0), (1, 1)

Conventional Smolyak grid using nested sets

Smolyak sparse grid

- Smolyak (1963) rule used to select tensor products:

$$d \leq i_1 + i_2 \leq d + \mu,$$

where $\mu \in \{0, 1, 2, \dots\}$ is the approximation level, and d is the dimensionality (in our case, $d = 2$).

- In terms of the above table, the sum of indices of a column i_1 and a row i_2 , must be between d and $d + \mu$.
- Let $\mathcal{H}^{d,\mu}$ denote the Smolyak grid for a problem with dimensionality d and approximation level μ .

Conventional Smolyak grid using nested sets

Smolyak sparse grid: $d = 2$.

- If $\mu = 0 \implies 2 \leq i_1 + i_2 \leq 2$. The only cell that satisfies this restriction is $i_1 = 1$ and $i_2 = 1 \implies$ the Smolyak grid has just one grid point

$$\mathcal{H}^{2,0} = \{(0,0)\}.$$

- If $\mu = 1 \implies 2 \leq i_1 + i_2 \leq 3$. The 3 cells that satisfy this restriction: (a) $i_1 = 1, i_2 = 1$; (b) $i_1 = 1, i_2 = 2$; (c) $i_1 = 2, i_2 = 1$, and the corresponding 5 Smolyak grid points are

$$\mathcal{H}^{2,1} = \{(0,0), (\square 1,0), (1,0), (0,\square 1), (0,1)\}.$$

- If $\mu = 2 \implies 2 \leq i_1 + i_2 \leq 4$. There are 6 cells satisfy this restriction \implies 13 Smolyak grid points:

$$\mathcal{H}^{2,2} = \left\{ (\square 1,1), (0,1), (1,1), (\square 1,0), (0,0), (1,0), (\square 1,\square 1), (0,\square 1), (1,\square 1), \left(\frac{\square 1}{\sqrt{2}},0\right), \left(\frac{1}{\sqrt{2}},0\right), \left(0,\frac{\square 1}{\sqrt{2}}\right), \left(0,\frac{1}{\sqrt{2}}\right) \right\}.$$

Conventional Smolyak polynomials using nested sets

Let $\mathcal{P}^{d,\mu}$ denote a Smolyak polynomial function in dimension d , with approximation level μ ,

$$\mathcal{P}^{d,\mu}(x_1, \dots, x_d; b) = \sum_{\max(d, \mu+1) \leq |i| \leq d+\mu} (\square 1)^{d+\mu-\square |i|} \binom{d \square 1}{d + \mu \square |i|} p^{|i|}(x_1, \dots, x_d),$$

where $p^{|i|}(x_1, \dots, x_d)$ is the sum of $p^{i_1, \dots, i_d}(x_1, \dots, x_d)$ with $i_1 + \dots + i_d = |i|$ defined as

$$p^{i_1, \dots, i_d}(x_1, \dots, x_d) = \sum_{\ell_1=1}^{m(i_1)} \dots \sum_{\ell_d=1}^{m(i_d)} b_{\ell_1 \dots \ell_d} \ell_1(x_1) \dots \ell_d(x_d),$$

where $m(i_1), \dots, m(i_d) =$ number of basis functions in dimensions $1, \dots, d$; $m(i) \equiv 2^{i-1} + 1$ for $i \geq 2$ and $m(1) \equiv 1$; $\ell_1(x_1), \dots, \ell_d(x_d) =$ unidimensional basis functions; $\ell_d = 1, \dots, m(i_d)$; and $b_{\ell_1 \dots \ell_d}$ are polynomial coefficients.

Conventional formula for Smolyak polynomial coefficients

- Let us build multidimensional Smolyak grid points and basis functions using unidimensional Chebyshev polynomials and their extrema.
- Then, there is an explicit formula for $b_{\ell_1 \dots \ell_d}$:

$$b_{\ell_1 \dots \ell_d} = \frac{2^d}{(m(i_1) \square 1) \cdots (m(i_d) \square 1)} \cdot \frac{1}{c_{\ell_1} \cdots c_{\ell_d}} \times \sum_{j_1=1}^{m(i_1)} \cdots \sum_{j_d=1}^{m(i_d)} \frac{\ell_1 \zeta_{j_1} \cdots \ell_d \zeta_{j_d} \cdot f(\zeta_{j_1}, \dots, \zeta_{j_d})}{c_{j_1} \cdots c_{j_d}}, \quad (1)$$

where $\zeta_{j_1}, \dots, \zeta_{j_d}$ are grid points in dimensions j_1, \dots, j_d ; $c_j = 2$ for $j = 1$ and $j = m(i_d)$; $c_j = 1$ for $j = 2, \dots, m(i_d) \square 1$.

- If along any dimension d , $m(i_d) = 1 \implies$ this dimension is dropped from computation, i.e., $m(i_d) \square 1$ and $c_{j_d} = c_1$ are set to 1.
- Formula (1) is well known in the related literature; it is used, e.g., in Malin et al. (2011).

Inefficiency of conventional Smolyak interpolation

- **Inefficiency:** First, we create a list of tensor products with many repeated elements and then, we eliminate the repetitions.
- **Repetitions of grid points.**
 - $\mathcal{H}^{2,1}$: $(0,0)$ is listed 3 times \implies must eliminate 2 grid points out of 7.
 - $\mathcal{H}^{2,2}$: must eliminate 12 repeated points out of 25 points.
 - *But grid points must be constructed just once (fixed cost), so repetitions are not so important for the cost.*
- **Repetitions of basis functions.**
 - $\mathcal{P}^{2,1}$ lists 7 basis functions from sets $\{1\}$, $\{1, {}_2(x), {}_3(x)\}$, $\{1, {}_2(y), {}_3(y)\}$ and eliminates 2 repeated functions $\{1\}$ by assigning a weight ($\square 1$) to $p^{|2|}$.
 - $\mathcal{P}^{2,2}$: must eliminate 12 repeated basis functions out of 25.
 - *Smolyak polynomials must be constructed many times (in every grid point, integration node and time period) and each time we suffer from repetitions*
- **The number of repetitions increases in μ and $d \implies$ important for high-dimensional applications.**

Smolyak method with Lagrange interpolation

We now present an alternative variant of the Smolyak method.

- First, instead of nested sets, we use disjoint sets, which allows us to avoid repetitions.
- Second, we find the coefficients using Lagrange-style interpolation. This technique works for any basis function and not necessarily orthogonal ones. Most of the computations can be done up-front (precomputed).
- Our version of the Smolyak method will be more simple and intuitive and easier to program.

Step 1. Smolyak grid using disjoint sets

Unidimensional grid points using disjoint sets

- We construct the Smolyak grid using disjoint sets.
- We consider grid points $\left\{ \square 1, \frac{\square 1}{\sqrt{2}}, 0, \frac{1}{\sqrt{2}}, 1 \right\}$ in the interval $[\square 1, 1]$ and create 3 unidimensional sets of elements (grid points), A_1, A_2, A_3 , which are disjoint, i.e., $A_i \cap A_j = \{\emptyset\}$ for any i and j .
 - $i = 1 : A_1 = \{0\};$
 - $i = 2 : A_2 = \{\square 1, 1\};$
 - $i = 3 : A_3 = \left\{ \frac{\square 1}{\sqrt{2}}, \frac{1}{\sqrt{2}} \right\}.$

Step 1. Smolyak grid using disjoint sets

Tensor products of unidimensional disjoint sets of points

		$i_2 = 1$	$i_2 = 2$	$i_2 = 3$
	$A_{i_1} \setminus A_{i_2}$	0	$\square 1, 1$	$\frac{\square 1}{\sqrt{2}}, \frac{1}{\sqrt{2}}$
$i_1 = 1$	0	(0, 0)	(0, $\square 1$), (0, 1)	$(0, \frac{\square 1}{\sqrt{2}}), (0, \frac{1}{\sqrt{2}})$
$i_1 = 2$	$\square 1$ 1	$(\square 1, 0)$ (1, 0)	$(\square 1, \square 1), (\square 1, 1)$ (1, $\square 1$), (1, 1)	$(\square 1, \frac{\square 1}{\sqrt{2}}), (\square 1, \frac{1}{\sqrt{2}})$ $(1, \frac{\square 1}{\sqrt{2}}), (1, \frac{1}{\sqrt{2}})$
$i_1 = 3$	$\frac{\square 1}{\sqrt{2}}$ $\frac{1}{\sqrt{2}}$	$(\frac{\square 1}{\sqrt{2}}, 0)$ $(\frac{1}{\sqrt{2}}, 0)$	$(\frac{\square 1}{\sqrt{2}}, \square 1), (\frac{\square 1}{\sqrt{2}}, 1)$ $(\frac{1}{\sqrt{2}}, \square 1), (\frac{1}{\sqrt{2}}, 1)$	$(\frac{\square 1}{\sqrt{2}}, \frac{\square 1}{\sqrt{2}}), (\frac{\square 1}{\sqrt{2}}, \frac{1}{\sqrt{2}})$ $(\frac{1}{\sqrt{2}}, \frac{\square 1}{\sqrt{2}}), (\frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}})$

We select elements that belong to the cells with the sum of indices of a column and a row, $i_1 + i_2$, between d and $d + \mu$. This leads to the same Smolyak grids as before. However, in our case, no grid points are repeated.

Smolyak grid using disjoint sets

Smolyak sparse grid

- We use the same Smolyak rule for constructing multidimensional grid points

$$d \leq i_1 + i_2 \leq d + \mu$$

That is, we select elements that belong to the cells in the above table for which the sum of indices of a column and a row, $i_1 + i_2$, is between d and $d + \mu$.

- This leads to the same Smolyak grids $\mathcal{H}^{2,0}$, $\mathcal{H}^{2,1}$ and $\mathcal{H}^{2,2}$ as under the construction built on nested sets. However, in our case, no grid points are repeated.

Step 2. Smolyak polynomials using disjoint sets

Disjoint sets of basis functions

The same construction as the one we used for constructing the grid points.

$$i = 1 : A_1 = \{1\};$$

$$i = 2 : A_2 = \{ _2(x), _3(x) \};$$

$$i = 3 : A_3 = \{ _4(x), _5(x) \}.$$

Step 2. Smolyak polynomials using disjoint sets

Tensor products of unidimensional disjoint sets of basis functions

		$i_2 = 1$	$i_2 = 2$	$i_2 = 3$
	$A_{i_1} \setminus A_{i_2}$	1	2 (y), 3 (y)	4 (y), 5 (y)
$i_1 = 1$	1	1	2 (y), 3 (y)	4 (y), 5 (y)
$i_1 = 2$	2 (x) 3 (x)	2 (x) 3 (x)	2 (x) 2 (y), 2 (x) 3 (y) 3 (x) 2 (y), 3 (x) 3 (y)	2 (x) 4 (y), 2 (x) 5 (y) 3 (x) 4 (y), 3 (x) 5 (y)
$i_1 = 3$	4 (x) 5 (x)	4 (x) 5 (x)	4 (x) 2 (y), 4 (x) 3 (y) 5 (x) 2 (y), 5 (x) 3 (y)	4 (x) 4 (y), 4 (x) 5 (y) 5 (x) 4 (y), 5 (x) 5 (y)

For example, for $\mu = 1$, we get

$$\mathcal{P}^{2,1}(x, y; b) = b_{11} + b_{21} \ 2(x) + b_{31} \ 3(x) + b_{12} \ 2(y) + b_{13} \ 3(y).$$

Step 3. Lagrange-style interpolation for finding coefficients

- Simply find the coefficients so that a polynomial with M basis functions passes through M given grid points.
- Let $f : [\square 1, 1]^d \rightarrow \mathbb{R}$ be a smooth function.
- Let $\mathcal{P}(\cdot; b)$ be a polynomial function, $\mathcal{P}(x; b) = \sum_{n=1}^M b_n \Psi_n(x)$, where $\Psi_n : [\square 1, 1]^d \rightarrow \mathbb{R}$ is a d -dimensional basis function; $b \equiv (b_1, \dots, b_M)$ is a coefficient vector.
- We construct a set of M grid points $\{x_1, \dots, x_M\}$ within $[\square 1, 1]^d$, and we compute b so that the true function, f , and its approximation, $\mathcal{P}(\cdot; b)$ coincide in all grid points:

$$\begin{bmatrix} f(x_1) \\ \vdots \\ f(x_M) \end{bmatrix} = \begin{bmatrix} \hat{f}(x_1; b) \\ \vdots \\ \hat{f}(x_M; b) \end{bmatrix} = \overbrace{\begin{bmatrix} \Psi_1(x_1) & \cdots & \Psi_M(x_1) \\ \vdots & \ddots & \vdots \\ \Psi_1(x_M) & \cdots & \Psi_M(x_M) \end{bmatrix}}^{\equiv \mathcal{B}} \cdot \begin{bmatrix} b_1 \\ \vdots \\ b_M \end{bmatrix}.$$

Lagrange-style interpolation

- Provided that the matrix of basis functions \mathcal{B} has full rank, we have a system of M linear equations with M unknowns that admits a unique solution for b

$$\begin{bmatrix} b_1 \\ \dots \\ b_M \end{bmatrix} = \begin{bmatrix} \Psi_1(x_1) & \dots & \Psi_M(x_1) \\ \dots & \ddots & \dots \\ \Psi_1(x_M) & \dots & \Psi_M(x_M) \end{bmatrix}^{-1} \begin{bmatrix} f(x_1) \\ \dots \\ f(x_M) \end{bmatrix}.$$

By construction, approximation $\mathcal{P}(\cdot; b)$ coincides with true function f in all grid points, i.e., $\hat{f}(x_n; b) = f(x_n)$ for all $x_n \in \{x_1, \dots, x_M\}$.

- For orthogonal basis functions, matrix \mathcal{B} is well-conditioned.

Our Smolyak polynomial using disjoint sets

- We hope you liked our construction without cumbersome formulas.
- However, for those of you who do like more the Smolyak formula, we present our version of the Smolyak formula using disjoint sets.
- Our formula produces the same Smolyak polynomial but avoids repetitions:

$$\mathcal{P}^{d,\mu}(x_1, \dots, x_d; b) = \sum_{d \leq |i| \leq d+\mu} q^{|i|}(x_1, \dots, x_d), \quad (2)$$

where $q^{|i|}(x_1, \dots, x_d)$ is the sum of $q^{i_1, \dots, i_d}(x_1, \dots, x_d)$ whose indices satisfy $i_1 + \dots + i_d = |i|$,

$$\begin{aligned} & q^{i_1, \dots, i_d}(x_1, \dots, x_d) \\ &= \sum_{\ell_1=m(i_1 \square 1)+1}^{m(i_1)} \dots \sum_{\ell_d=m(i_d \square 1)+1}^{m(i_d)} b_{\ell_1 \dots \ell_d} \ell_1(x_1) \cdots \ell_d(x_d), \quad (3) \end{aligned}$$

where $\ell_1(x_1), \dots, \ell_d(x_d)$ are unidimensional basis functions, in dimensions $1, \dots, d$; $\ell_1(x_1) \cdots \ell_d(x_d)$ is a d -dimensional basis function; by convention, $m(0) = 0$ and $m(1) = 1$.

Nested-set versus disjoint-set constructions: comparison of costs

- Recall that $m(i_j) \equiv 2^{i_j - 1} + 1$ for $i_j \geq 2$ & $m(1) \equiv 1$ & $m(0) = 0$.
- **Number of terms.**

Nested-set construction:

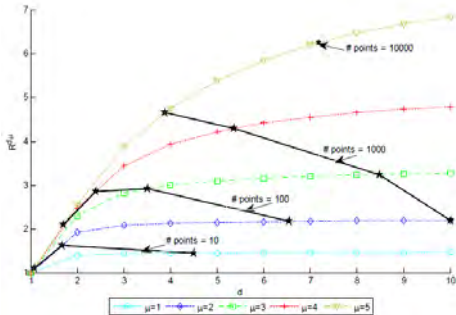
$$\sum_{\max(d, \mu+1) \leq |i| \leq d+\mu} \prod_{j=1}^d m(i_j)$$

Disjoint-set construction:

$$\sum_{d \leq |i| \leq d+\mu} \prod_{j=1}^d [m(i_j) - m(i_j - 1)]$$

Nested sets versus disjoint sets: comparison of costs

- A ratio of the number of terms under the two constructions, $\mathcal{R}^{d,\mu}$.



- For any $i_j \geq 2$, we reduce the number of terms by at least a factor of $\frac{2}{3}$ when we use disjoint sets instead of nested sets.

Adapting Smolyak methods to economic applications

- Our previous analysis was intended to make the Smolyak interpolation method more efficient (i.e., less costly).
- Now, we show how to better adapt the Smolyak method to economic applications.
- We will consider 3 ways to enhance the performance of the Smolyak method.
 - Anisotropic grid (different treatment of variables).
 - Adaptive domain (fitting the hypercube to a high probability set of a given economic model).
 - Efficient iterative procedure for finding fixed point coefficients.

Anisotropic grid

- The conventional Smolyak method treats all dimensions symmetrically: it uses the same number of grid points and basis functions for all variables.
- In economic applications, it may be of value to give different treatments to different variables.
- *Why?*
 - Decision functions may have more curvature in some variables than in others.
 - Some variables may have a larger range of values than the others.
 - Some variables may be more important than the others.
- **Literature.**
 - *Gerstner and Griebel (1998, 2003)*: dimension-adaptive tensor-product quadrature to integrate high-dimensional functions.
 - *Kouri (2012)*: anisotropic grids to solve PDE with uncertain coefficients.

Anisotropic grid

- Let μ_i be an approximation level in dimension i .
- Let $\mu = (\mu_1, \dots, \mu_d)$.
- Let $\mu^{\max} = \max\{\mu_1, \dots, \mu_d\}$
- Note that $\mu_j = i_j^{\max} \square 1$ where i_j^{\max} is the maximum index of the sets considered for dimension j .
- Smolyak grid is called *asymmetric (anisotropic)* if there is at least one dimension j such that $i_j \neq i_k$ for $\forall k \neq j$.
- $\mathcal{H}^{d,(\mu_1, \dots, \mu_d)} \equiv$ a d -dimensional anisotropic Smolyak grid of approximation levels $\mu = (\mu_1, \dots, \mu_d)$.
- $\mathcal{P}^{d,(\mu_1, \dots, \mu_d)} \equiv$ the corresponding Smolyak polynomial.

Tensor products of sets of unidimensional elements

		$i_2 = 1$	$i_2 = 2$
	$A_{i_1} \setminus A_{i_2}$	0	$\square 1, 1$
$i_1 = 1$	0	(0, 0)	(0, $\square 1$), (0, 1)
$i_1 = 2$	$\square 1$ 1	($\square 1, 0$) (1, 0)	($\square 1, \square 1$), ($\square 1, 1$) (1, $\square 1$), (1, 1)
$i_1 = 3$	$\frac{\square 1}{\sqrt{2}}$ $\frac{1}{\sqrt{2}}$	$\left(\frac{\square 1}{\sqrt{2}}, 0 \right)$ $\left(\frac{1}{\sqrt{2}}, 0 \right)$	$\left(\frac{\square 1}{\sqrt{2}}, \square 1 \right)$, $\left(\frac{\square 1}{\sqrt{2}}, 1 \right)$ $\left(\frac{1}{\sqrt{2}}, \square 1 \right)$, $\left(\frac{1}{\sqrt{2}}, 1 \right)$
$i_1 = 4$	$\frac{\square \sqrt{2+\sqrt{2}}}{2}$ $\frac{\square \sqrt{2\square \sqrt{2}}}{2}$ $\frac{\sqrt{2\square \sqrt{2}}}{2}$ $\frac{\sqrt{2+\sqrt{2}}}{2}$	$\left(\frac{\square \sqrt{2+\sqrt{2}}}{2}, 0 \right)$ $\left(\frac{\square \sqrt{2\square \sqrt{2}}}{2}, 0 \right)$ $\left(\frac{\sqrt{2\square \sqrt{2}}}{2}, 0 \right)$ $\left(\frac{\sqrt{2+\sqrt{2}}}{2}, 0 \right)$	$\left(\frac{\square \sqrt{2+\sqrt{2}}}{2}, \square 1 \right)$, $\left(\frac{\square \sqrt{2+\sqrt{2}}}{2}, 1 \right)$ $\left(\frac{\square \sqrt{2\square \sqrt{2}}}{2}, \square 1 \right)$, $\left(\frac{\square \sqrt{2\square \sqrt{2}}}{2}, 1 \right)$ $\left(\frac{\sqrt{2\square \sqrt{2}}}{2}, \square 1 \right)$, $\left(\frac{\sqrt{2\square \sqrt{2}}}{2}, 1 \right)$ $\left(\frac{\sqrt{2+\sqrt{2}}}{2}, \square 1 \right)$, $\left(\frac{\sqrt{2+\sqrt{2}}}{2}, 1 \right)$

Anysotropic Smolyak sets

- The Smolyak rule: select elements that satisfy

$$d \leq i_1 + i_2 \leq d + \mu^{\max}$$

- If $\mu = (1, 0)$, then $\mu^{\max} = 1$ and $2 \leq i_1 + i_2 \leq 3$. The 3 cells that satisfy this restriction are (a) $i_1 = 1, i_2 = 1$; (b) $i_1 = 1, i_2 = 2$; (c) $i_1 = 2, i_2 = 1$,

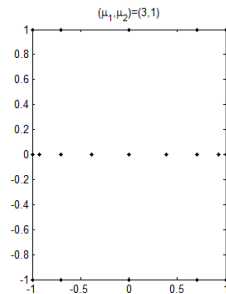
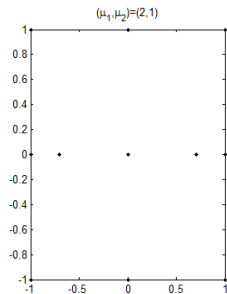
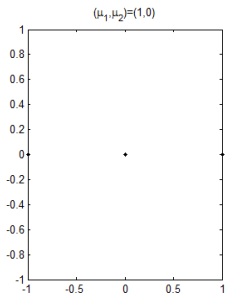
$$\mathcal{H}^{2,\{1,0\}} = \{(0, 0), (\square 1, 0), (1, 0)\}.$$

- If $\mu = (2, 1)$, then $\mu^{\max} = 2$ and $2 \leq i_1 + i_2 \leq 4$, there are 5 cells that satisfy this restriction (a) $i_1 = 1, i_2 = 1$; (b) $i_1 = 1, i_2 = 2$; (c) $i_1 = 2, i_2 = 1$; (d) $i_1 = 1, i_2 = 3$; (e) $i_1 = 2, i_2 = 2$; and 11 points:

$$\mathcal{H}^{2,\{2,1\}} = \{(\square 1, 1), (0, 1), (1, 1), (\square 1, 0), (0, 0), (1, 0), (\square 1, \square 1), (0, \square 1), (1, \square 1), (\frac{\square 1}{\sqrt{2}}, 0), (\frac{1}{\sqrt{2}}, 0)\}.$$

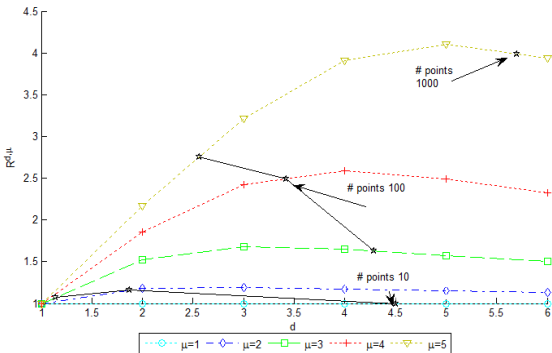
- If $\mu = (3, 1)$, then $\mu^{\max} = 3$ and $2 \leq i_1 + i_2 \leq 5$, there are 19 points.

Anisotropic grids: an illustration



Isotropic versus anisotropic grids: comparison of costs

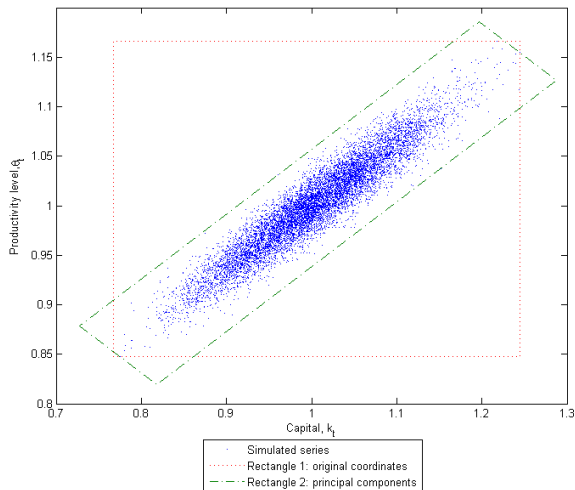
- We compare isotropic and anisotropic grids in which no elements are repeated. The number of elements depends on the specific restrictions $\{\mu_1, \dots, \mu_d\}$ imposed.
- We show a limiting case when we have $i_1^{\max} = \mu + 1$ in dimension 1 and we have $i_j^{\max} = 1$ for all other dimensions, $j = 2, \dots, d$.



Adaptive domain

- The Smolyak construction (built on extrema of Chebyshev polynomials) tells us how to represent and interpolate functions defined on a normalized d -dimensional hypercube $([0, 1]^d)$.
- The solution domain of a typical dynamic economic model does not have the shape of a hypercube but can be of any shape in a d -dimensional space.
- We describe how to effectively adapt a multidimensional hypercube to an unstructured solution domain of a given problem.
- By choosing an appropriate system of coordinates, we are able to reduce the Euler equation residuals significantly (without increasing the running time).

Conventional hypercube vs. a hypercube obtained after the change of variables



How to construct the conventional hypercube

- Obtain simulated data: $\{k_t, \theta_t\}_{t=1}^T$.
- Define $[\underline{k}, \bar{k}]$ and $[\underline{\theta}, \bar{\theta}]$ as intervals for the state variables that we get in the simulation.
- Where \bar{k} , and $\bar{\theta}$ represent the maxima in each coordinate and \underline{k} , and $\underline{\theta}$ represent the minima in each coordinate. Consider a linear transformation $(k, \theta) \in [\underline{k}, \bar{k}] \times [\underline{\theta}, \bar{\theta}]$

$$x = 2 \frac{k - \underline{k}}{\bar{k} - \underline{k}} - 1 \text{ and } y = 2 \frac{\theta - \underline{\theta}}{\bar{\theta} - \underline{\theta}} - 1. \quad (4)$$

- For instance, $[\underline{k}, \bar{k}] = [0.8k_s, 1.2k_s]$, with k_s being capital in the steady state, and $[\underline{\theta}, \bar{\theta}] = [\exp(\frac{0.8\sigma}{1-\rho}), \exp(\frac{0.8\sigma}{1-\rho})]$.

How to construct an adaptive hypercube

The procedure is similar to the one used in Judd, Maliar and Maliar (2011, 2013) for constructing cluster and EDS grids.

1. Obtain simulated data: $\{k_t, \theta_t\}_{t=1}^T$.
2. Normalize the data

$$\tilde{k}_t = \frac{k_t - \mu_k}{\sigma_k} \quad \text{and} \quad \tilde{\theta}_t = \frac{\theta_t - \mu_\theta}{\sigma_\theta}, \quad (5)$$

where μ_k and μ_θ are means, and σ_k and σ_θ are standard deviations of capital and productivity.

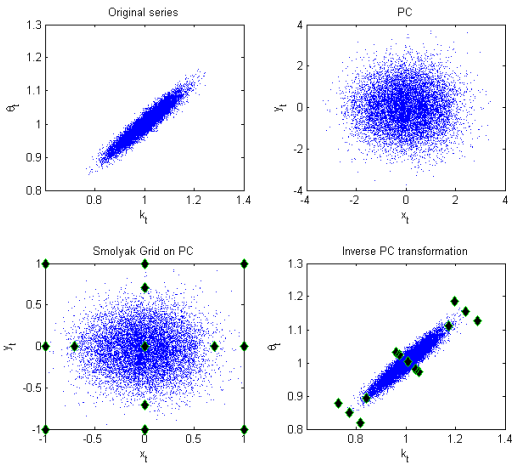
3. Consider the **singular value decomposition** of the matrix of the

normalized data $D \equiv \begin{bmatrix} \tilde{k}_1 & \tilde{\theta}_1 \\ \vdots & \vdots \\ \tilde{k}_T & \tilde{\theta}_T \end{bmatrix}$. $D = USV^\top$.

How to construct an adaptive hypercube

4. $Z \equiv DV$, the variables $\{z_t^1, z_t^2\}_{t=1}^T$ are called *principal components* of D and are orthogonal (uncorrelated).
5. In order to use Chebyshev polynomials, it is necessary to stay in the intervals between -1 and 1, therefore the same linear transformation as above is applied. Once we are in $[-1, 1]^2$ we can define interpolation nodes $\mathcal{H}^{2,\mu}$. In this step we collect $\underline{z}^1, \bar{z}^1, \underline{z}^2$ and \bar{z}^2 with the idea of inverting the procedure.
6. We can invert the process to obtain the interpolation nodes $\mathcal{H}^{2,\mu}$ in the same space that the real data $\{k_t, \theta_t\}_{t=1}^T$ are.

Smolyak grid on principal components



Standard neoclassical stochastic growth model

$$\max_{\{c_t, k_{t+1}\}_{t=0}^{\infty}} E_0 \sum_{t=0}^{\infty} \beta^t u(c_t)$$

subject to

$$c_t + k_{t+1} = (1 - \delta)k_t + \theta_t f(k_t)$$

$$\ln \theta_t = \rho \ln \theta_{t-1} + \sigma \varepsilon_t \text{ with } \varepsilon_t \sim N(0, 1)$$

where:

- $c_t, k_{t+1} \geq 0$, k_0 and θ_0 are given;
- $\beta \in (0, 1]$ is the discount factor;
- $U(c_t)$ is the utility function, increasing and concave;
- $\delta \in (0, 1]$ is the depreciation rate of capital;
- $f(k_t, \theta_t)$ is the production function; E_0 is the operator of conditional expectation;
- $\rho \in (-1, 1)$ and $\sigma > 0$.

The algorithm iterating on Euler equation

Smolyak algorithm

Initialization.

- Choose the approximation level, μ .
 - Construct the Smolyak grid $\mathcal{H}^{2,\mu} = \{(x_n, y_n)\}_{n=1,\dots,M}$ on $[\square 1, 1]^2$.
 - Compute the Smolyak basis functions in each grid point n .
The resulting $M \times M$ matrix is \mathcal{B} .
 - Fix $\Phi : (k, \theta) \rightarrow (x, y)$, where $(k, \theta) \in \mathbb{R}_+^2$ and $(x, y) \in [\square 1, 1]^2$.
Use $\Phi^{\square 1}$ to compute (k_n, θ_n) that corresponds to (x_n, y_n) in $\mathcal{H}^{2,\mu}$.
 - Choose integration nodes, ϵ_j , and weights, ω_j , $j = 1, \dots, J$.
 - Construct future productivities, $\theta'_{n,j} = \theta_n^\rho \exp(\epsilon_j)$ for all j ;
 - Choose an initial guess $b^{(1)}$.
-
-

The algorithm iterating on Euler equation

Step 1. Computation of a solution for K .

a. At iteration i , for $n = 1, \dots, M$, compute

- $k'_n = \mathcal{B}_n b^{(i)}$, where \mathcal{B}_n is the n th row of \mathcal{B} .
- $(x'_n, y'_{n,j})$ that correspond to $(k'_n, \theta'_{n,j})$ using Φ .
- Compute the Smolyak basis functions in each point $(x'_n, y'_{n,j})$.
- The resulting $M \times M \times J$ matrix is \mathcal{B}' .

- $k''_{n,j} = \mathcal{B}'_{n,j} b^{(i)}$, where $\mathcal{B}'_{n,j}$ is the n th row of \mathcal{B}' in state j .

- $c_n = (1 \square \delta) k_n + \theta_n f(k_n) \square k'_n$;

- $c'_{n,j} = (1 \square \delta) k'_n + \theta_n^\rho \exp(\epsilon_j) Af(k'_n) \square k''_{n,j}$ for all j ;

- $\widehat{k}'_n \equiv \beta \sum_{j=1}^J \omega_j \cdot \left[\frac{u_1(c'_{n,j})}{u_1(c_n)} [1 \square \delta + \theta_n^\rho \exp(\epsilon_j) f_1(k'_n)] k'_n \right]$

b. Find b that solves the system in Step 1a.

- Compute \widehat{b} that solves $\widehat{k}' = \mathcal{B}\widehat{b}$, i.e., $\widehat{b} = \mathcal{B}^{\square 1} \widehat{k}'_n$.

- Use damping to compute $b^{(i+1)} = (1 \square \xi) b^{(i)} + \xi \widehat{b}$, where $\xi \in (0, 1]$.

- Check for convergence: end Step 1 if $\frac{1}{M\xi} \sum_{n=1}^M \left| \frac{(k'_n)^{(i+1)} \square (k'_n)^{(i)}}{(k'_n)^{(i)}} \right| < 10^{\square \vartheta}$, $\vartheta > 0$.

Iterate on Step 1 until convergence.

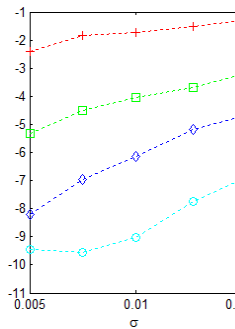
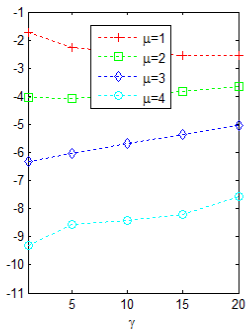
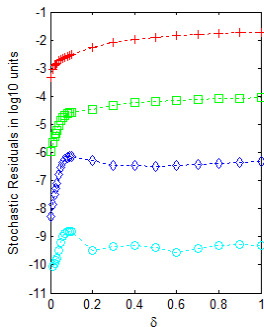
Results for the representative agent model

- CRRA utility function: $u(c) = \frac{c^{1-\alpha} - 1}{1-\alpha}$;
- Cobb-Douglas production function: $f(k) = k^\alpha$, with $\alpha = 1/3$;
- AR(1) process: $\ln \theta' = \rho \ln \theta + \sigma \varepsilon$, with $\rho = 0.95$
- Discount factor: $\beta = 0.99$.
- Benchmark values: $\delta = 0.025$, $\theta = 1$ and $\sigma = 0.01$.
- Then, we consider variations in δ , θ and σ one-by-one holding the remaining parameters at the benchmark values.

$$\begin{aligned}
 \delta &= \{0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1\}, \\
 &= \{1, 5, 10, 15, 20\}, \\
 \sigma &= \{0.001, 0.005, 0.01, 0.02, 0.03, 0.04, 0.05\}.
 \end{aligned}$$

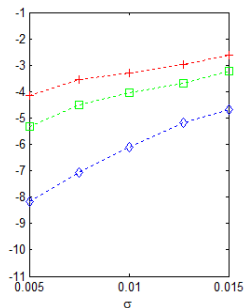
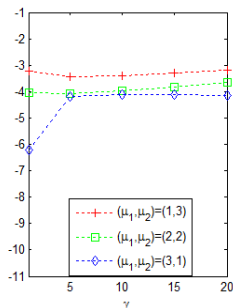
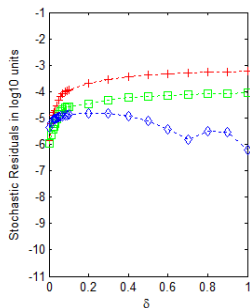
Conventional (isotropic) sparse grids under different approximation levels

- Consider approximation levels $\mu = 1, 2, 3, 4$.

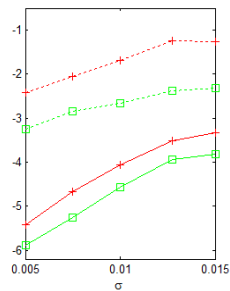
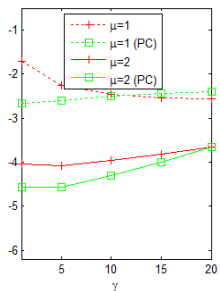
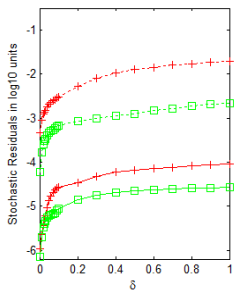


Anisotropic sparse grids

- Consider 2 anisotropic cases: $\mu = (3, 1)$ and $\mu = (1, 3)$.
- There are 9 elements in the first dimension and 3 elements in the second dimension \implies 15 grid points and 15 basis functions.



Adaptive domain



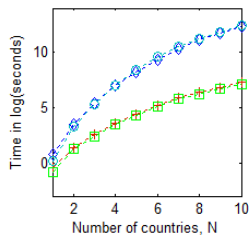
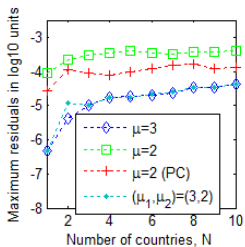
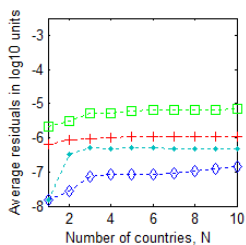
Multicountry model

$$\begin{aligned} & \max_{\{c_t^h, k_{t+1}^h\}_{t=0, \dots, \infty}^{h=1, \dots, N}} E_0 \left(\sum_{h=1}^N \tau^h \sum_{t=0}^{\infty} \beta^t u^h(c_t^h) \right) \\ \text{s.t. } & \sum_{h=1}^N c_t^h = \sum_{h=1}^N \left[\theta_t^h f^h(k_t^h) + k_t^h (1 - \delta) - k_{t+1}^h \right], \\ & \ln \theta_t^h = \rho \ln \theta_{t-1}^h + \epsilon_t^h, \end{aligned}$$

- $c_t^h, k_t^h, a_t^h, u^h, f^h$ and $\tau^h =$ consumption, capital, productivity level, utility function, production function, welfare weight of a country h ;
- $\epsilon_t^h \equiv \epsilon_t + \varpi_t^h$, $\epsilon_t \sim N(0, \sigma)$ is a common-for-all-countries shocks, $\varpi_t^h \sim N(0, \sigma)$ is a country-specific productivity shocks;
- Thus, $(\epsilon_t^1, \dots, \epsilon_t^N)^\top \sim \mathcal{N}(0_N, \Sigma)$, with $0_N \in \mathbb{R}^N$,

$$\Sigma = \begin{pmatrix} 2\sigma^2 & \dots & \sigma^2 \\ \dots & \dots & \dots \\ \sigma^2 & \dots & 2\sigma^2 \end{pmatrix} \in \mathbb{R}^{N \times N}.$$

Results for the multicountry model



Conclusion

- The Smolyak method is designed to deal with high-dimensional problems, but its cost still grows rapidly with dimensionality, especially if we target a high quality of approximation.
- We propose a variant of the Smolyak method that has a better performance (lower cost and higher accuracy).
 - We introduce formula for Smolyak polynomials that avoids repetitions and eliminates unnecessary function evaluations.
 - We propose a simple Lagrange-style technique for finding the polynomial coefficients.
 - We develop an anisotropic version of the Smolyak grid that takes into account an asymmetric structure of variables in economic model.
 - As a solution domain, we use a minimum hypercube that encloses the high-probability set of a given economic model.
- The above four improvements are related to Smolyak interpolation. Our last improvement is concerned with an iterative procedure for solving dynamic economic models. We propose to use fixed-point iteration instead of time iteration.

Anisotropic grids: accuracy bounds

We generalize Barthelmann et al. (2000) to anisotropic grids.

Consider the spaces $F_1^k = C^k([0, 1])$, with the norm

$\|f\| = \max \{ \|D^\alpha f\|_\infty \mid \alpha = 0, \dots, k \}$ and for $d > 1$, $F_d^k = C^k([0, 1]^d)$,

with the norm $\|f\| = \max \{ \|D^\alpha f\|_\infty \mid \alpha \in \mathbb{N}_0^d, \alpha_i \leq k \}$. Finite linear combinations of functions $f_1 \quad f_2 \quad \dots \quad f_d$ with $f_i \in F_d^k$ are dense in F_d^k

and $\|f_1 \quad f_2 \quad \dots \quad f_d\| = \|f_1\| \|f_2\| \dots \|f_d\|$. Let I_d denote the embedding $F_d^k \rightarrow C^k([0, 1]^d)$. Moreover, let

$\|S\| = \sup \{ \|S(f)\|_\infty \mid f \in F_d^k, \|f\| \leq 1 \}$ for $S : F_d^k \rightarrow C^k([0, 1]^d)$. We use $c_{d,k}$ to denote constants that only depend on d and k .

Theorem

For the space F_d^k we obtain

$$\|f_{1,\dots,d} - \widehat{f}_{1,\dots,d}\| \leq c \|f_{1,\dots,d}\| \mu^{2k} \sum_{j=1}^{d-1} \sum_{s=j}^{q-d+j} \prod_{v=1, \forall v: i_v \leq i_v^{\max}}^j \widehat{f}_v^{i_v} - \widehat{f}_v^{i_v-1} \quad (6)$$

Proof.

Maliar, Maliar and Valero (2013). This is a technical part, and we do in a separate paper.

Anisotropic grids: intuition for the theorem

The theorem provides a useful intuition.

- The only term affected by anisotropy is $\prod_{v=1, \forall v, i_v \leq i_v^{\max}}^j \widehat{f}_v^{i_v} \square \widehat{f}_v^{i_v} \square 1$.
- It shows how the quality of approximation improves when we increase the level of approximation.

Example: *Suppose we have a quadratic function. Then, the approximation is perfect along the dimension i_v , the terms in the product are zero starting from the third order and hence, the product is zero. A further increase in the order of approximation will not change the maximum norm. This is precisely the kind of case we want to show.*

Adaptive domain: accuracy bounds

Theorem

Assume that we interpolate function on hypercube domains H^1, H^2 such that $H^1 \subset H^2$. Then, the approximation errors satisfy

$$\|f_{1,\dots,d} \square A(q, d)(f)\|_{H^1} \geq \|f_{1,\dots,d} \square A(q, d)(f)\|_{H^2} .$$

The intuition is that when we reduce the domain, we always reduce the approximation errors according to the studied maximum norm because if the maximum error is in the domain H^2 the error does not change and if it is in H^1 but not in H^2 , the errors decrease.

A Tractable Framework for Analyzing a Class of Nonstationary Markov Models

Lilia Maliar, Serguei Maliar, John B. Taylor, Inna Tsener

Quantitative Economics (2020), 11/4, 1289-1323.

Minicourse
November 2022

Semi-Markov models

- **Markov model:** transition probabilities $P(s_{t+1} = s' | s_t = s)$ are time-invariant function of state s_t .
- **Semi-Markov model** is a generalization of Markov model that allow for time-varying transitions $P_t(s_{t+1} = s' | s_t = s)$, e.g.,

$$\begin{array}{ll} \text{Markov:} & z_{t+1} = \rho z_t + \sigma \varepsilon_{t+1} \\ \text{Semi-Markov:} & z_{t+1} = \rho_t z_t + \sigma_t \varepsilon_{t+1} \end{array} .$$

- Semi-Markov processes may result from DGEM models with:
 - deterministic trends (population growth, climate changes, etc.);
 - unbalanced types of technological progress;
 - changes in consumer's tastes and habits;
 - entry into or exit from a monetary union (Brexit);
 - nonrecurrent policy regime changes;
 - deterministic seasonals;
 - expectation-driven dynamics; etc.

⇒ **Markov models are known; we focus on semi-Markov models.**

Turnpike theorem

- **Turnpike (theorem)** referred to a class of convergence results of finite-horizon to infinite-horizon models.
- In infinite-horizon Markov model, we must find one time-invariant optimal value and/or decision functions.
- In infinite-horizon semi-Markov model, we must find an infinite sequence of time-varying optimal value and/or decision functions (because the optimal value and/or decision functions nontrivially change from one period to another).

⇒ *Under turnpike property, we can approximate the solution to **infinite-horizon semi-Markov model as a limit of finite-horizon semi-Markov model** when time horizon goes to infinity.*

Monetary policy

- We allow for monetary policies that are state contingent (rules) or time contingent (path) or both.
 - **Time-contingent**: implement policy on January 1, 2020.
 - **State-contingent**: implement policy when unemployment is 3%.
- Great Recession: Forward guidance is time contingent (path) policy.
- Policy normalization = switching back state-contingent policy (e.g., Taylor rule).

Questions:

- Should the Fed normalize policy **now** or **later**?
- Should the Fed normalize policy **gradually** or **all at once**?
- Should the regime shift be **announced in advance**?
- Should the policy normalization be **time** or **state contingent**?

⇒ **Both state- and time-contingent policies are of interest.**

The plan of the talk:

1. Explain and test the methodology of analyzing semi-Markov models using a simple growth model example.
2. Solve a collection of nonstationary growth models with:
 - capital augmenting technological progress;
 - anticipated regime switches;
 - parameter drifting;
 - time-varying volatility with a deterministic trend;
 - seasonal adjustments;
 - estimation and calibration of parameters in an unbalanced growth model using data on the U.S. economy.
3. Discuss an example that violates the turnpike theorem - new Keynesian model with forward guidance puzzle.
4. *Conduct time-contingent monetary policy experiments in a semi-Markov version of the large-scale ToTEM model of Bank of Canada.*

A growth model with time-varying fundamentals

Agent solves:

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

$$\text{s.t. } c_t + k_{t+1} = (1 - \delta) k_t + f_t(k_t, z_t),$$

$$z_{t+1} = \varphi_t(z_t, \varepsilon_{t+1}),$$

- $c_t \geq 0$ and $k_t \geq 0$ are consumption and capital, resp.;
- initial condition (k_0, z_0) is given;
- $u_t : \mathbb{R}_+ \rightarrow \mathbb{R}$ and $f_t : \mathbb{R}_+^2 \rightarrow \mathbb{R}_+$ and $\varphi_t : \mathbb{R}^2 \rightarrow \mathbb{R}$ are possibly time-varying utility function, production function and law of motion for exogenous state variable z_t , resp.;
- sequence of u_t , f_t and φ_t for $t \geq 0$ is known to the agent in period $t = 0$; ε_{t+1} is i.i.d.;
- $\beta \in (0, 1)$ = discount factor; $\delta \in [0, 1]$ = depreciation rate; $E_t[\cdot]$ = operator of expectation.

Why cannot we solve time-varying model with conventional solution methods?

A stationary Markov growth model:

$$\begin{aligned}
 V(k, z) &= \max_{c, k'} \{u(c) + \beta E[V(k', z')]\} \\
 \text{s.t. } k' &= (1 - \delta)k + zf(k) \leq c, \\
 \ln z' &= \rho \ln z + \varepsilon', \quad \varepsilon' \sim \mathcal{N}(0, \sigma^2).
 \end{aligned}$$

An interior solution satisfies the Euler equation:

$$u'(c) = \beta E[u'(c')(1 - \delta + z'f'(k'))].$$

- Conventional solution methods: *either iterate on Bellman equation until a fixed-point V is found or iterate on Euler equation until a fixed-point decision function $k' = K(k, z)$ is found.*
- However, if u , f , ρ and σ are time-dependent, then $V_t(\cdot) \neq V_{t+1}(\cdot)$ and $K_t(\cdot) \neq K_{t+1}(\cdot)$, i.e., no fixed-point functions V and K .
- We need to construct a sequence (path) of time-contingent value functions $(V_0(\cdot), V_1(\cdot), \dots)$, decision functions $(K_0(\cdot), K_1(\cdot), \dots)$.

Making a model stationary: balanced growth

Some semi-Markov models can be **reduced to Markov** models, for example, a class of balanced growth models.

However, the class of balanced growth models is limited:

- *King, Plosser and Rebelo (1988)* show that the standard neoclassical growth model is consistent with balanced growth only under the assumption of labor augmenting technological progress and under some additional restrictions on u and f .
- If one deviates from their assumptions, the property of balanced growth does not survive.

Making a model stationary: extending state space

Some semi-Markov models can be **reduced to Markov** models by extending the state space.

- Unanticipated Markov shocks can be treated as exogenous state variables by including them into the state space, for example
 - *Markov regime switching models* (Davig and Leeper, 2007, 2009, Farmer et al., 2011).
 - *stochastic volatility models* (Bloom, 2009, Fernández-Villaverde et al. 2010).
- Anticipated shocks of fixed horizon and periodicity are also consistent with stationarity, in particular,
 - *deterministic seasonals* (Barsky and Miron, 1989, Hansen and Sargent, 1993, 2013);
 - *news shocks* (Schmitt-Grohé and Uribe, 2012).

Irreducible semi-Markov models

Generally, semi-Markov models cannot be reduced to Markov models.

- Deterministic trends typically lead to unbalanced growth:
 - *investment-specific technical change (Krusell et al., 2000);*
 - *capital-augmenting technological progress, (Acemoglu, 2002);*
 - *decreasing labor-income shares (Karabarbounis and Brent, 2014).*
- Parameter changes also lead to time-dependent value and decision functions if they are anticipated; for example:
 - *anticipated accessions of new members to a monetary union or their exits (Brexit);*
 - *presidential elections with predictable outcomes;*
 - *credible policy announcements;*
 - *anticipated legislative changes; etc.*

⇒ **We are interested in studying irreducible semi-Markov models.**

Our approach: turnpike property!

- Let us show a motivating example of the turnpike property for a version of the model that admits a closed-form solution.
- We assume a full depreciation of capital $\delta = 1$, and Cobb-Douglas utility and production functions:

$$u_t(c) = \ln(c), \quad \text{and} \quad f_t(k, z) = zk^\alpha A_t^{1-\alpha},$$

where

- $A_t = A_0 g_A^t$ represents labor augmenting technological progress;
- the productivity level follows $\ln z_{t+1} = \rho \ln z_t + \sigma \epsilon_{t+1}$, where $\epsilon_{t+1} \sim \mathcal{N}(0, 1)$, $\rho \in (0, 1)$ and $\sigma \in (0, \infty)$.

Finite horizon model: backward induction

- The Bellman equation becomes for $t = 1, \dots, T$:

$$V_t(k_t) = \max_{k_{t+1}} \left\{ \ln z_t k_t^\alpha A_t^{1-\alpha} k_{t+1} + \beta E_t [V_{t+1}(k_{t+1})] \right\},$$

where we assume $V_{T+1}(k_{T+1}) = 0$ and hence, $k_{T+1} = 0$.

- It is well known that the Bellman equation admits a closed-form solution:

$$k_T = \frac{\alpha\beta}{1 + \alpha\beta} z_{T-1} k_{T-1}^\alpha A_{T-1}^{1-\alpha},$$

$$k_{T-1} = \frac{\alpha\beta(1 + \alpha\beta)}{1 + \alpha\beta(1 + \alpha\beta)} z_{T-2} k_{T-2}^\alpha A_{T-2}^{1-\alpha}, \quad \text{etc.}$$

The equilibrium functions are semi-Markov $k_T = K_T(z_{T-1}, k_{T-1})$, $k_{T-1} = K_{T-1}(z_{T-2}, k_{T-2})$, **i.e., we have a path of decision functions** K_T, K_{T-1}, \dots, K_0 .

Finite- versus infinite-horizon trajectories

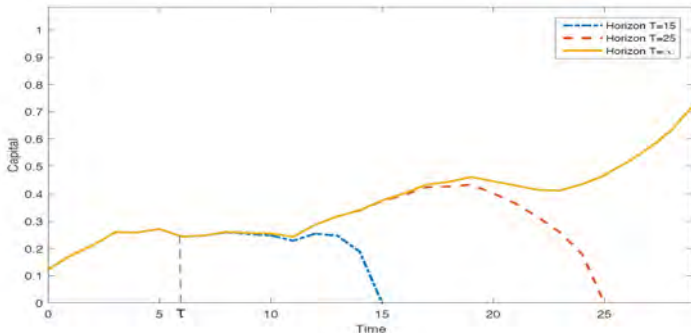


Figure 1. Finite- and infinite-horizon solutions in the growth model.

- Trajectories with $T = 15$, $T = 25$ and $T = \infty$ are identical up to τ .
- We can describe the trajectory of the economy with $T = \infty$ for the first τ periods by solving for $T = 15, 25$.

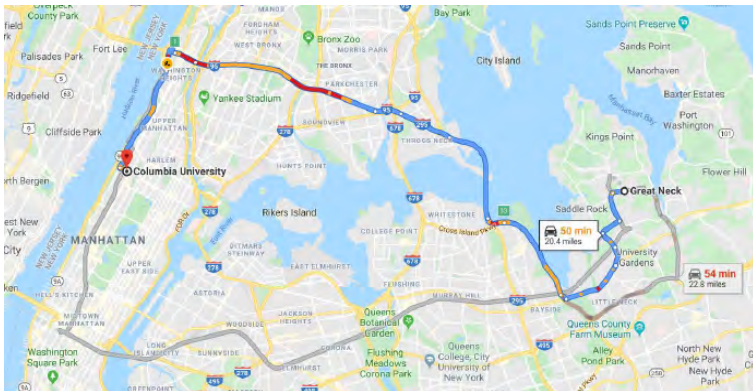
What is turnpike?

Turnpike \equiv highway.



Why is this property called "turnpike theorem"?

- **Turnpike theorems:** turnpike is often the fastest route between two points even if it is not a direct shortest route.
- Example: Driving from Great Neck to Columbia: *get on turnpike, stay on turnpike as much as possible and get off close to destination:*



Extended function path (EFP) method

We introduce a simple **extended function path (EFP)** framework for analyzing semi-Markov models:

- Step 1. Assume that, in some remote period, the economy becomes stationary and construct the usual Markov solution.
- Step 2. Given the terminal condition, solve backward the Bellman or Euler equations to construct a sequence (path) of value and decision functions.
- Step 3. Verify that the turnpike property holds (check that the finite-horizon solution is an accurate approximation to the infinite-horizon solution).

⇒ This framework is called "**EFP**" because it is related to **extended path (EP)** method of Fair and Taylor (1983).

Extended function path (EFP) framework

Step 0. Initialization. Choose some $T \gg \tau$ and construct a T -period stationary economy such that $u_t = u$, $f_t = f$ and $\varphi_t = \varphi$ for all $t \geq T$.

Step 1. Construct a stationary Markov capital function K satisfying:

$$u'(c) = \beta E [u'(c')(1 - \delta + f'(k', \varphi(z, \varepsilon')))]$$

$$c = (1 - \delta)k + f(k, z) - k'$$

$$c' = (1 - \delta)k' + f(k', \varphi(z, \varepsilon')) - k''$$

$$k' = K(k, z) \text{ and } k'' = K(k', \varphi(z, \varepsilon')).$$

Step 2. Construct a path for capital policy functions (K_0, \dots, K_T) that matches the terminal condition $K_T \equiv K$ and that satisfies for $t = 0, \dots, T - 1$:

$$u'_t(c_t) = \beta E_t [u'_{t+1}(c_{t+1})(1 - \delta + f'_{t+1}(k_{t+1}, \varphi_t(z_t, \varepsilon_{t+1})))]$$

$$c_t = (1 - \delta)k_t + f_t(k_t, z_t) - k_{t+1}$$

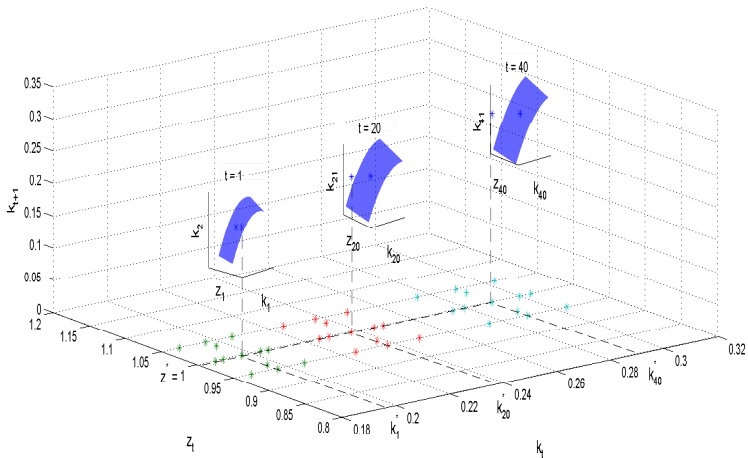
$$c_{t+1} = (1 - \delta)k_{t+1} + f_{t+1}(k_{t+1}, \varphi_t(z_t, \varepsilon_{t+1})) - k_{t+2}$$

$$k_{t+1} = K_t(k_t, z_t) \text{ and } k_{t+2} = K_{t+1}(k_{t+1}, \varphi_t(z_t, \varepsilon_{t+1})).$$

Step 3. Check the turnpike theorem.

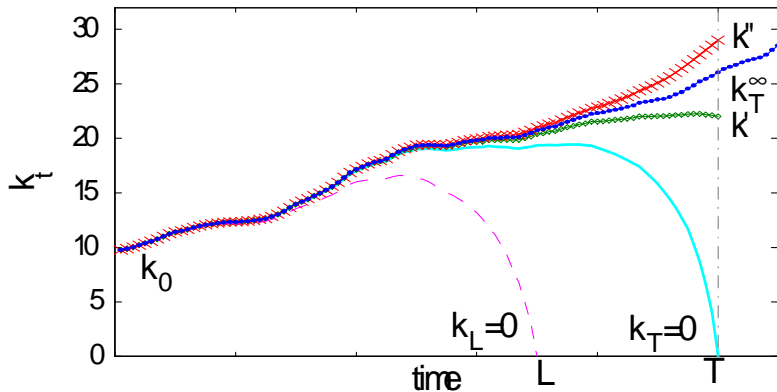
Output: the first τ functions (K_0, \dots, K_τ) constitute an approximate solution.

Example of function path constructed by EFP



Intuition behind turnpike theorem

When you are young, you behave as if you will live forever...



Theoretical foundations of EFP framework

We prove two formal results:

- **Theorem 1 (existence)**: EFP approximations exists, is unique and possess a semi-Markov structure.
- **Theorem 2 (turnpike)**: EFP can approximate a solution to infinite-horizon semi-Markov model with an arbitrary degree of precision as the time horizon T increases.

Theorem 1: Semi-Markov solution

Theorem 1 (*semi-Markov structure*). *In the T -period stationary economy, the solution exists, unique and semi-Markov.*

Proof. Under our assumptions on u_t , f_t and the objective function, FOCs are necessary for optimality. We will show that FOCs are also sufficient to identify the solution and to establish its semi-Markov structure. Our proof is constructive: it relies on backward induction.

Step 1. – At T , the economy becomes stationary and remains stationary forever, i.e., $u_t \equiv u$, $f_t \equiv f$ and $\varphi_t \equiv \varphi$ for all $t \geq T$.

– Thus, the model's equations and decision functions are time invariant for $t \geq T$.

– It is well known that under our assumptions on u_t , f_t and the objective function, there is a unique stationary Markov capital function K that satisfies the optimality conditions.

Theorem 1: Semi-Markov solution (cont.)

Step 2. Given the constructed T -period capital function $K_T \equiv K$, we define the capital functions K_{T-1}, \dots, K_0 in previous periods by using backward induction. The Euler equation for period $T-1$,

$$u'_{T-1}(c_{T-1}) = \beta E_{T-1} [u'_T(c_T)(1 - \delta + f'_T(k_T, z_T))],$$

where c_{T-1} and c_T are related to k_T and k_{T+1} in periods T and $T-1$ by

$$\begin{aligned} c_{T-1} &= (1 - \delta) k_{T-1} + f_{T-1}(k_{T-1}, z_{T-1}) - k_T, \\ c_T &= (1 - \delta) k_T + f_T(k_T, z_T) - k_{T+1}. \end{aligned}$$

- z_T follows a semi-Markov process, i.e., $z_T = \varphi_T(z_{T-1}, \varepsilon)$.
- By construction, we have that $k_{T+1} = K_T(k_T, z_T)$ is semi-Markov.
- Thus, we obtain a functional equation that defines k_T for each (k_{T-1}, z_{T-1}) , i.e., the capital decisions at period $T-1$ are given by a state-contingent function $k_T = K_{T-1}(k_{T-1}, z_{T-1})$.
- By proceeding iteratively backward, we construct $K_{T-1}(k_{T-1}, z_{T-1}), \dots, K_0(k_0, z_0)$. ■

Theorem 2: Turnpike theorem

Let us fix history $h_\infty = (\varepsilon_0, \varepsilon_1 \dots)$ and initial condition (k_0, z_0) and construct the productivity levels $\{z_t\}_{t=0}^T$ using the law of motion. Use the constructed functions $K_0(k_0, z_0), \dots, K_T(k_T, z_T)$ to generate the solution $\{c_t^T, k_t^T\}_{t=0}^\infty$ for the T -period stationary economy

$$k_{t+1}^T = K_t(k_t^T, z_t),$$

where $k_0^T = k_0$, and c_t^T satisfies the budget constraint for all $t \geq 0$.

Theorem 2 (Turnpike theorem): For any real number $\varepsilon > 0$ and any natural number τ , there exists a threshold terminal date $T(\varepsilon, \tau)$ such that for any $T \geq T(\varepsilon, \tau)$, we have

$$|k_t^\infty - k_t^T| < \varepsilon, \quad \text{for all } t \leq \tau,$$

$\{c_t^\infty, k_t^\infty\}_{t=0}^\infty \in \mathfrak{S}^\infty =$ solution in the infinite-horizon semi-Markov economy;

$\{c_t^T, k_t^T\}_{t=0}^T =$ solution in the T -period stationary semi-Markov economy.

Theorem 2: Turnpike theorem (cont.)

- **The convergence is uniform:** The turnpike theorem states that for all $T \geq T(\varepsilon, \tau)$, the constructed nonstationary Markov approximation $\{k_t^T\}$ is guaranteed to be within a given ε -accuracy range from the true solution $\{k_t^\infty\}$ during the initial τ periods.
- For periods $t > \tau$, our approximation may become insufficiently accurate and exit the ε -accuracy range.
- **Proof is technically involved:** *we follow the early turnpike literature, e.g., Brock and Gale (1969), Brock (1971), Brock and Mirman (1972, 1973), Mirman and Zilcha (1977), Brock and Majumdar (1978), Mitra and Zilcha (1981), Majumdar and Zilcha (1987), Mitra and Nyarko (1991), Joshi (1997).*
- But these papers are purely theoretical, while we offer a tractable quantitative EFP framework.

Assessing EFP accuracy in a model with balanced growth

- We parameterize the growth model by

$$u_t(c) = \frac{c^{1-\alpha}}{1-\alpha}, \quad \text{and} \quad f_t(k, z) = zk^\alpha A_t^{1-\alpha},$$

- $\alpha > 0$ and $\alpha \in (0, 1)$;
- $A_t = A_0 g_A^t$ = labor augmenting technological progress with an exogenous constant growth rate $g_A \geq 1$.
- Productivity is assumed to follow

$$\ln z_{t+1} = \rho \ln z_t + \sigma \varepsilon_{t+1}, \quad \varepsilon_{t+1} \sim \mathcal{N}(0, 1),$$

$$\rho \in (-1, 1), \quad \sigma \in (0, \infty).$$

- This version of the model is consistent with balanced growth; see King, Plosser and Rebelo (1988).
- We can solve the balanced growth model very accurately and use the accurate solution for comparison.

A comparison of four solution methods

We solve the nonstationary growth model using **four** alternative solution methods:

1. "*Exact solution*" is a very accurate solution to the model with a balanced growth path;
2. "*EFP solution*" is produced by the EFP method that solves the nonstationary model directly;
3. "*Naive solution*" is produced by replacing the nonstationary model with a sequence of stationary models, and it solves such models one by one.
 - it neglects a connection between the decision functions of different periods (unlike EFP);
4. "*Fair and Taylor solution*" is produced by using Fair and Taylor's (1983) method.

Graphical illustration of four solution methods

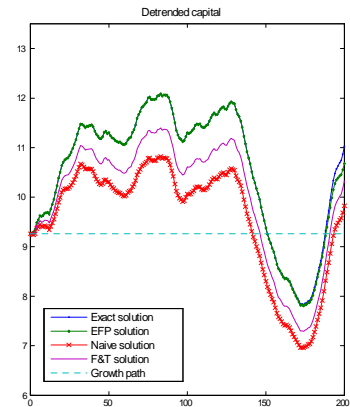
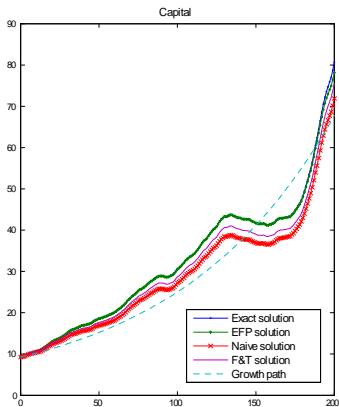


Table 1: Comparison of four solution methods

	Fair-Taylor (1983) method, $\tau = 1$		Naive method	EFP method $\tau = 200$		
Terminal condition	Steady state	Steady state	-	Balanced growth	T -period stationary	
T	200	400	200	200	200	400

Maximum errors across t periods in \log_{10} units

$t \in [0, 50]$	-1.29	-1.29	-1.04	-6.82	-6.01	-6.42
$t \in [0, 100]$	-1.18	-1.18	-0.92	-6.68	-4.39	-5.99
$t \in [0, 150]$	-1.14	-1.14	-0.89	-6.66	-2.89	-5.98
$t \in [0, 175]$	-1.14	-1.13	-0.89	-6.66	-2.10	-5.98
$t \in [0, 200]$	-1.14	-1.13	-0.89	-6.66	-1.45	-5.92

Running time, in seconds

Solution	1.2(+4)	6.1(+4)	28.9	104.9	99.1	225.9
Simulation	-	-	2.6	2.6	2.8	5.7
Total	1.2(+4)	6.1(+4)	31.5	107.6	101.9	231.6

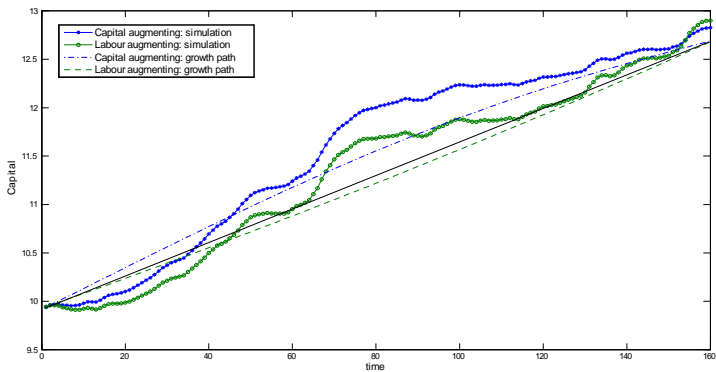
Application 1: Model with capital augmenting technological progress

- We assume a constant elasticity of substitution (CES) production function, and we allow for both labor and capital augmenting technological progresses,

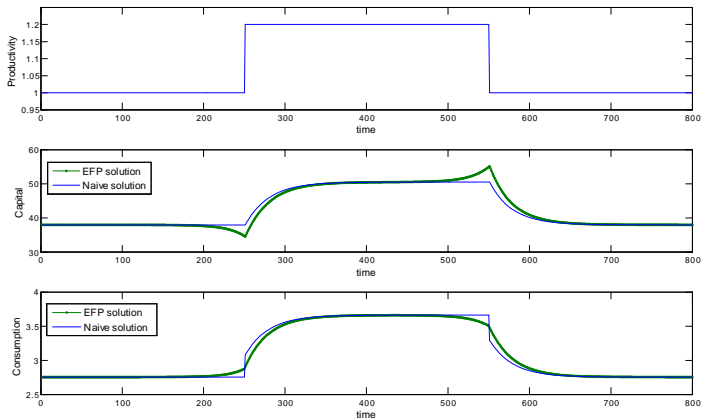
$$F(k_t, \ell_t) = [\alpha(A_{k,t}k_t)^v + (1 - \alpha)(A_{\ell,t}\ell_t)^v]^{1/v},$$

- $A_{k,t} = A_{k,0}g_{A_k}^t$; $A_{\ell,t} = A_{\ell,0}g_{A_\ell}^t$; $v \leq 1$; $\alpha \in (0, 1)$;
– g_{A_k} and g_{A_ℓ} = rates of capital and labour augmenting technological progresses, resp.
- Labor is supplied inelastically. Let $\ell_t = 1$ for all t . The corresponding production function by $f(k_t) \equiv F(k_t, 1)$.
- **The model with capital augmenting technological progress does not admit a balanced growth path.**

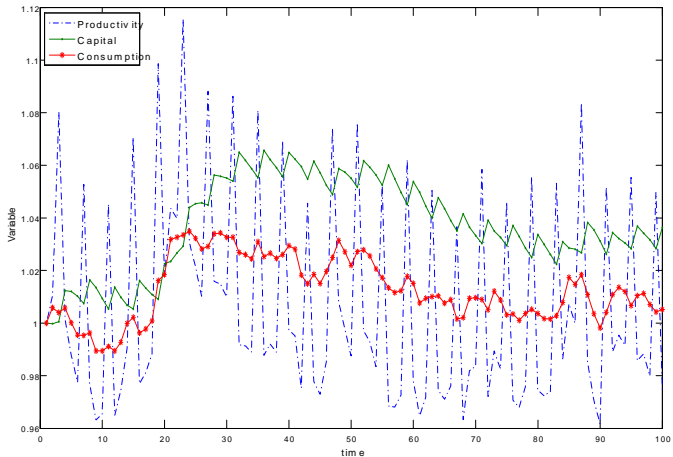
Application 1: Model with capital augmenting technological progress (cont.)



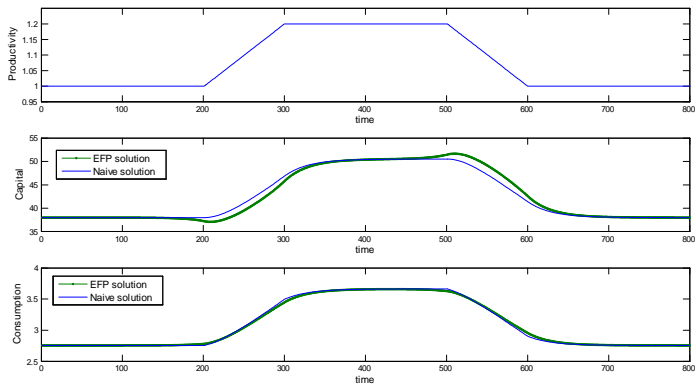
Application 2: Anticipated technology shocks



Application 3: A model with seasonal changes



Application 4: A nonstationary model with a parameter drift (cont.)



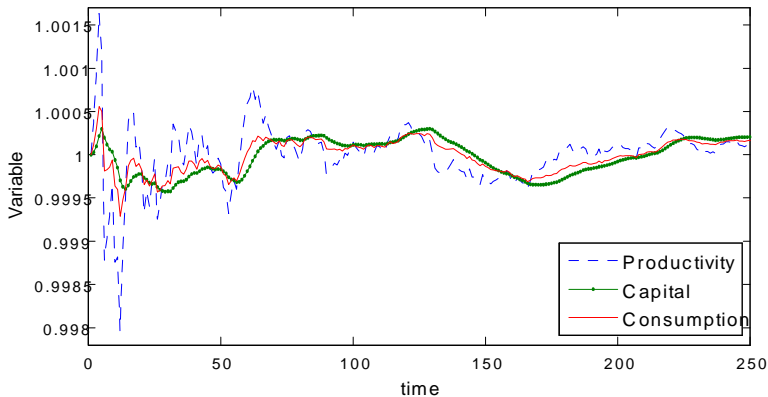
Application 5: Diminishing volatility

- We consider a model in which the volatility has both a stochastic and deterministic components.
- We modify the standard neoclassical stochastic growth model to include a diminishing volatility of the productivity shock:

$$\ln z_t = \rho \ln z_{t-1} + \sigma_t \varepsilon_t, \quad \sigma_t = \frac{B}{t^{\rho_\sigma}}, \quad \varepsilon_t \sim \mathcal{N}(0, 1),$$

- B = a scaling parameter;
- ρ_σ = a parameter that governs the volatility of z_t .
- The standard deviation of the productivity shock $B\sigma/t^{\rho_\sigma}$ decreases over time, reaching zero in the limit, $\lim_{t \rightarrow \infty} \frac{B\sigma}{t^{\rho_\sigma}} = 0$.

Application 3: Diminishing volatility (cont.)



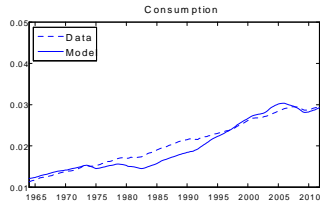
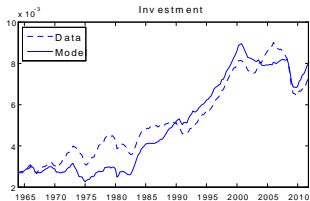
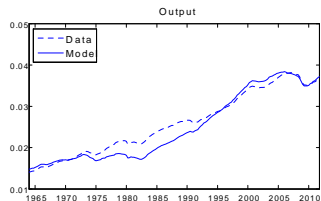
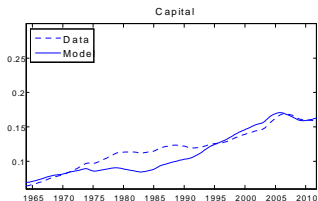
Application 6: Estimating unbalanced growth model with U.S. data

- In spite of Kaldor's (1961) facts, we still observe visible differences in growth rates across variables.
- We extend the benchmark model to include time-varying depreciation rate of capital,

$$\begin{aligned} & \max_{\{c_t, k_{t+1}\}_{t=0, \dots, \infty}} E_0 \sum_{t=0}^{\infty} \beta^t u(c_t) \\ & \text{s.t. } c_t + k_{t+1} = A_t z_t k_t^\alpha + (1 - d_t \delta_t) k_t, \\ & \ln \delta_t = \rho_\delta \ln \delta_{t-1} + \varepsilon_{\delta,t}, \quad \varepsilon_{\delta,t} \sim \mathcal{N}(0, \sigma_{\varepsilon_d}^2), \\ & \ln z_t = \rho_z \ln z_{t-1} + \varepsilon_{z,t}, \quad \varepsilon_{z,t} \sim \mathcal{N}(0, \sigma_{\varepsilon_z}^2), \end{aligned}$$

$d_t \delta_t$ = a time-varying depreciation rate; d_t = a trend component of depreciation, $d_t = d_0 g_d^t$; δ_t = a stochastic shock to depreciation.

Application 6: Estimating unbalanced growth model with U.S. data (cont.)



Application 7. New Keynesian model

- "*Forward guidance puzzle*": an observation that output and consumption are excessively sensitive to central bank announcements about future interest rates in new Keynesian models.
– Del Negro, Giannoni and Patterson (2015).
- McKay, Nakamura and Steinsson (2016):

$$\begin{aligned}x_t &= E_t [x_{t+1}] \square [\log R_t \square E_t \log \pi_{t+1} \square \log R_*], \\ \log \pi_t &= \beta E_t [\log \pi_{t+1}] + \kappa x_t, \\ \log R_{t+j} &= E_{t+j} [\log \pi_{t+j+1}] + \log R_* + \varepsilon_{t+j,t}.\end{aligned}$$

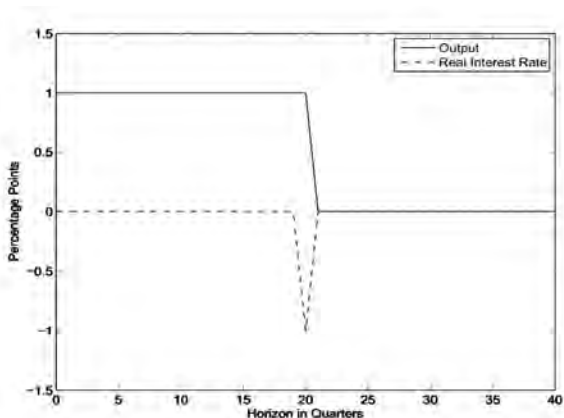
$$x_t \equiv \log Y_t \square \log Y_{N,t};$$

$\varepsilon_{t+j,t}$ = a $t + j$ -period shock to the interest rate that is announced in period t . This yields

$$x_t = \square \sum_{j=0}^{\infty} \varepsilon_{t+j,t}.$$

- *Today's shock to the interest rate has the same effect as a shock that happens a million years from now!*

Forward guidance puzzle - violation of turnpike theorem!



- *Maliar and Taylor (2018) argue that FG puzzle happens only under extreme and empirically implausible parameterizations.*

Conclusion

- Stationary Markov class of models is a dominant framework.
- A shortcoming of this framework is that it generally restricts the parameters of economic models to be constant, and it restricts the behavior patterns to be time invariant.
- We analyze a more flexible class of semi-Markov models that allows for time-varying fundamentals and decision functions.
- We propose and test EFP framework for solving, calibrating, simulating and estimating of parameters in such models.
- In another paper, we show semi-Markov policy experiments in large-scale bToTEM model of Bank of Canada.
- **THANK YOU!**

The Power of Open-Mouth Policies

Vadym B. Lepetyuk, Lilia Maliar, Serguei Maliar, John B. Taylor

CEPR Discussion Paper No. DP16262

https://cepr.org/active/publications/discussion_papers/dp.php?dpno=16262

Minicourse
November 2022

The views expressed in this presentation are solely those of the authors and may differ from official Bank of Canada views. No responsibility for them should be attributed to the Bank of Canada.

Motivation and Objective

- Some economic changes are recurrent
 - ▶ high and low productivity states
 - ▶ UK leaves the EU but might come back, etc.
- Some regime changes are nonrecurrent (hopefully) and can be welfare-ranked
 - ▶ abolishment of slavery
 - ▶ women's voting rights, etc.
- Many (recurrent or nonrecurrent) policies are pre-announced
 - ▶ taxes, tariffs, minimum wage, social security
 - ▶ custom unions, monetary unions;
 - ▶ inflation target, interest rate commitments
- What do we do?
 - ▶ Demonstrate the importance of anticipatory effects particularly in the context of new Keynesian models

Policy experiments in a large-scale central banking model

- 1 Gradual decline of neutral interest rate
- 2 Gradual change of inflation target (certain or probabilistic)
- 3 Normalization of monetary policy regarding future nominal interest rates, when economy is at ZLB
- 4 Switch to a more aggressive Taylor rule
- 5 Switch to price-level targeting
- 6 Switch to average inflation targeting

Findings

- 1 An anticipated gradual decline of neutral interest rate by 1 percent associated with a corresponding increase in agents' time discount factor leads (*ceteris paribus*) to an expansion of the economy by more than 1 percent.
- 2 Postponing an increase in the inflation target by one year brings a sizable increase in output over the transition. If this policy is implemented probabilistically, there are still substantial anticipation effects both before and after the resolution of uncertainty.
- 3 When an economy is at ZLB, forward guidance increases output growth over the transition; however an initial jump in output is invariant to the horizon of the forward guidance.
- 4 Twice as aggressive behavior of the central bank toward targeting inflation and output, switching to price-level targeting or to average-inflation targeting have only modest anticipation effects.

Existing solution methods

- Conventional perturbation: time-invariant (stationary) solutions
- Fair and Taylor (1983): a path under one specific realization of shocks using certainty equivalence
 - ▶ low accuracy in the presence of high volatilities or strong nonlinearities
- Maliar, Maliar, Taylor and Tsener (2020): a functional solution using accurate integration methods (Extended Function Path)
 - ▶ high implementation and computational burden
- Perturbation with news shocks by Schmitt-Grohé and Uribe (2012)
 - ▶ suitable for events that come with the same periodicity and duration, but not for one-time events

Methodological contribution

- We offer a novel perturbation-based framework for studying anticipated effects

Methodology of numerical analysis

- Equilibrium equations

$$E_t [G_t (y_{t-1}, y_t, y_{t+1}, z_{t+1}, z_t)] = 0$$

$$z_{t+1} = Z_t (z_t, \epsilon_{t+1})$$

where y_t = vector of endogenous variables;

z_t = vector of exogenous variables

- Solution

$$y_t = Y_t (z_t, y_{t-1})$$

a set of time-dependent equilibrium functions that satisfy the equilibrium equations in the relevant area of the state space

Methodology of numerical analysis

- Step I: Solving a T -period stationary economy
 - ▶ Assume that in a very remote period T , the economy becomes stationary, i.e., $G_t(\cdot) = G(\cdot)$ and $Z_t(\cdot) = Z(\cdot)$ for all $t \geq T$. The equilibrium system becomes

$$E_t [G(y_{t-1}, y_t, y_{t+1}, z_{t+1}, z_t)] = 0$$

$$z_{t+1} = Z(z_t, \epsilon_{t+1})$$

- ▶ Solve the system to find the solution

$$y_T = \hat{Y}_T(z_T, y_{T-1})$$

- Step II: Constructing a function path
 - ▶ Using a T -period solution $y_T = \hat{Y}_T(z_T, y_{T-1})$ as the terminal condition, Iterate backward for $T - 1, \dots, 1$ on the equilibrium conditions

$$E_t [G_t(y_{t-1}, y_t, \hat{Y}_{t+1}(z_{t+1}, y_t), z_{t+1}, z_t)] = 0$$

$$z_{t+1} = Z_t(z_t, \epsilon_{t+1})$$

to construct a sequence (path) of time-dependent value and decision functions $\{Y_{T-1}(\cdot), \dots, Y_1(\cdot)\}$.

Methodology of numerical analysis

- We use perturbation to find numerical approximations of decision functions.
- It is unclear around which point(s) the decision rules must be approximated, because nonstationary and unbalanced growth models have no deterministic steady state.
- Possible solution: time-varying growth rates that capture how much endogenous state variables grow due to the time trend or the parameter change.
 - ▶ particular growth rates could first be assumed (e.g., no growth)
 - ▶ then the growth rates could be updated iteratively

Portable Implementation

- Perturbation solutions from Dynare
- Standard second-order perturbation solution to a stationary model consists of

$$g(v; \sigma) \approx \underbrace{g(\bar{v}; 0) + g_x(\bar{v}; 0)(v - \bar{v})}_{\text{1st-order perturbation solution}} + \underbrace{\frac{1}{2}g_{xx}(\bar{v}; 0)(v - \bar{v})^2 + \frac{1}{2}g_{\sigma\sigma}(\bar{v}; 0)\sigma^2}_{\text{2nd-order terms}}$$

- ▶ a constant term of the policy function is given by $g(\bar{v}; 0) + \frac{1}{2}g_{\sigma\sigma}(\bar{v}; 0)\sigma^2$ and hence, is affected by variances of shocks.
- ▶ the first-order perturbation solution does not depend on the degree of volatility σ , i.e., $g_\sigma(\bar{v}; 0) = 0$
- ▶ the cross partial derivative $g_{\sigma x}(\bar{v}; 0)$ is zero; see Schmitt-Grohé and Uribe (2004).

Methods

Portable Implementation

- In Step I, a Taylor expansion of the policy functions in a stationary model is found around the deterministic steady state \bar{v} of the model.
- In Step II, we consider alternative options:
 - ① find solutions for v_{t+1} and v_t around v_t and v_{t-1} , respectively, such that $v_t = v_{t-1} \equiv \bar{v}_t$
 - ② consider v_{t+1} and v_t perturbed around \bar{v}_t and \bar{v}_{t-1} , respectively, such that $\bar{v}_t = \bar{v}_{t-1}\gamma_{v,t-1}$, where $\gamma_{v,t-1}$ is a time-dependent growth rate; in Dynare, it can be implemented by coding v_t and v_{t+1} with different variable names.

A simple illustration

Neoclassical stochastic growth model with labor augmenting technological progress

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

$$\text{s.t. } c_t + k_{t+1} = (1 - \delta) k_t + z_t f(k_t, A_t)$$

$$\ln z_{t+1} = \rho \ln z_t + \sigma \varepsilon_{t+1}, \quad \varepsilon_{t+1} \sim \mathcal{N}(0, 1)$$

where (k_0, z_0) is given; $A_t = A_0 \gamma_A^t$ is labor augmenting technological progress with $\gamma_A \geq 1$

Introducing turnpike theorem

Why cannot we solve a nonstationary model with conventional solution methods?

- if A_t grows over time, then the optimal decision functions change over time, i.e., $C_t(\cdot) \neq C_{t+1}(\cdot)$
 - ▶ there is no fixed-point solution $C(\cdot)$ so that the conventional methods are not applicable.
- Under additional restrictions on preferences and technology, the model has balanced growth and can be converted into stationary; see King et al. (1988).
- We will not focus on this special case but will approximate a sequence (path) of time-dependent functions $\{C_0(\cdot), C_1(\cdot), \dots\}$
- In doing this, we exploit a turnpike theorem



Turnpike theorem: an illustration

- For illustration of turnpike property, consider for a version of the growth model that admits a closed-form solution:
 - ▶ Full depreciation of capital $\delta = 1$
 - ▶ Cobb-Douglas utility and production functions

$$u_t(c) = \ln(c) \quad \text{and} \quad f_t(k, z) = zk^\alpha A_t^{1-\alpha}$$

- Bellman equation

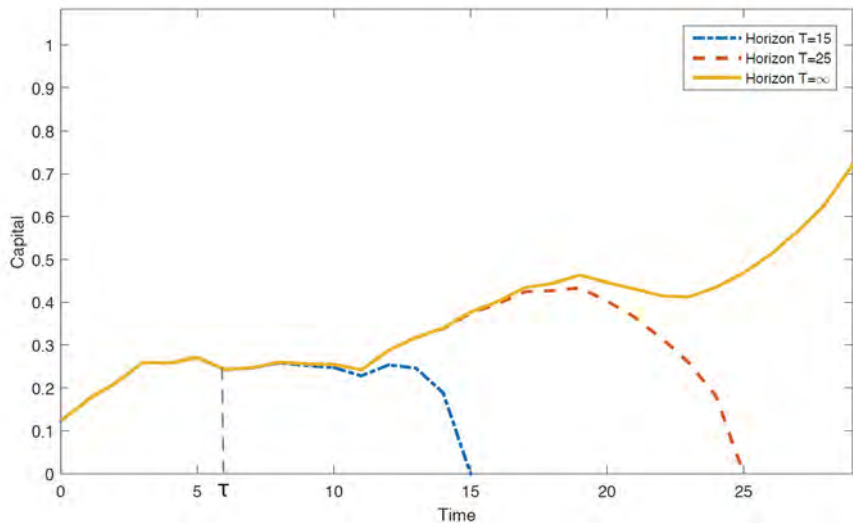
$$V_t(k_t) = \max_{k_{t+1}} \left\{ \ln(z_t k_t^\alpha A_t^{1-\alpha} - k_{t+1}) + \beta E_t[V_{t+1}(k_{t+1})] \right\}, \quad t = 1, \dots, T$$

- Assume $V_{T+1}(k_{T+1}) = 0$ and hence, $k_{T+1} = 0$
- The closed-form solution

$$k_T = \frac{\alpha\beta}{1 + \alpha\beta} z_{T-1} k_{T-1}^\alpha A_{T-1}^{1-\alpha}, \quad k_{T-1} = \frac{\alpha\beta(1 + \alpha\beta)}{1 + \alpha\beta(1 + \alpha\beta)} z_{T-2} k_{T-2}^\alpha A_{T-2}^{1-\alpha}, \quad \dots$$

- Parameterization: $\beta = 0.99$, $\alpha = 0.36$, $\rho = 0.95$, $\sigma = 0.01$ and $g_A = 1.01$

Turnpike theorem: an illustration



Turnpike theorem (informal)

If we are interested in the behavior of infinite-horizon non-stationary economy during some initial number of periods τ , we can accurately approximate the infinite-horizon solution by solving the finite-horizon model.

Turnpike theorem (more formal)

For any real number $\varepsilon > 0$ and any natural number τ , there exists a threshold terminal date $T(\varepsilon, \tau)$ such that for any $T \geq T(\varepsilon, \tau)$, we have

$$|k_t^\infty - k_t^T| < \varepsilon, \quad \text{for all } t \leq \tau$$

$\{c_t^\infty, k_t^\infty\}_{t=0}^\infty \in \mathfrak{S}^\infty =$ solution in the infinite-horizon economy;

$\{c_t^T, k_t^T\}_{t=0}^T =$ solution in the T -period stationary economy.

Illustration of perturbation procedure for the growth model

- In Step I, we construct a stationary (time-invariant) model of period T and construct the corresponding Markov decision rule for consumption $C_T(\cdot)$
- In Step II, we use $C_T(\cdot)$ to iterate backward on Euler equations in order to construct a sequence (path) of time-dependent value and decision functions $\{C_{T-1}(\cdot), C_{T-2}(\cdot), \dots, C_0(\cdot)\}$, respectively.
- As a final step, we should check the turnpike theorem by verifying that the constructed finite-horizon solution for initial τ periods converges periodwise to a limiting $\{C_0^*(\cdot), C_1^*(\cdot), \dots, C_\tau^*(\cdot)\}$ as time horizon T increases.

Details of Step II

- In Step II, we start from the constructed terminal condition for T and proceed backward to compute the path of the decision functions for $t = T - 1, T - 2, \dots, 0$ by iterating backward on

$$u'(c_t) = \beta E_t [u'(C_{t+1}(k_{t+1}, z_{t+1})) (1 - \delta + z_{t+1} f_k(k_{t+1}, A_{t+1}))]$$

$$k_{t+1} = (1 - \delta) k_t + z_t f(k_t, A_t) - c_t$$

- In particular, for period $T - 1$, given $c_T = C_T(k_T, z_T)$, we compute in Dynare the decision function for $c_{T-1} = C_{T-1}(k_{T-1}, z_{T-1})$
- In period $T - 2$, given $c_{T-1} = C_{T-1}(k_{T-1}, z_{T-1})$ we find $c_{T-2} = C_{T-2}(k_{T-2}, z_{T-2})$ and so on until the entire solution path is constructed.
- Perturbation solutions in Dynare are obtained around a deterministic growth path.

Details of Step II

- We consider five alternative methods for constructing the deterministic growth path:
 - ▶ We either assume some exogenous growth rates or precompute the growth rates endogenously by shutting down uncertainty in the model.
 - ▶ Also, for each deterministic growth-path specification, we have two versions of the algorithm: one in which a next-period policy function takes into account the volatility of uncertainty σ , and the other in which it does not setting $\sigma = 0$.
 - ▶ Motivation: Perturbation policy functions of second and higher orders of approximation are not passing in general through a deterministic steady state of the model. Even in the balanced growth model, if true policy functions for period $t + 1$ are combined with the model's equations written for period t , the deterministic steady state would not be a solution of the deterministic version of the combined system of equations. This feature can be overcome by recognizing explicitly that $C_{t+1}(\cdot)$ depends on σ and by setting σ to zero when computing the deterministic steady state.

Methods 1 and 2

- Methods 1 and 2 find local approximations of today's consumption policy function $C_t(k_t, z_t)$ given the next-period function $C_{t+1}(k_{t+1}, z_{t+1})$
- The difference between these two methods lies only in the point around which the local approximation is taken
- Method 1 finds local approximation in period t around a point $(k_t^*, 1)$ that solves the following system of two equations for c_t^* and k_t^*

$$\begin{aligned} u'(c_t^*) &= \beta u'(C_{t+1}(k_t^*, 1)) [1 - \delta + f_k(k_t^*, A_{t+1})] \\ k_t^* &= (1 - \delta) k_t^* + f(k_t^*, A_t) - c_t^* \end{aligned}$$

- In Method 2, the approximation is conducted around a point $(k_t^\diamond, 1)$ that solves the following system of two equations for c_t^\diamond and k_t^\diamond

$$\begin{aligned} u'(c_t^\diamond) &= \beta u'(C_{t+1}(k_t^\diamond, 1; 0)) [1 - \delta + f_k(k_t^\diamond, A_{t+1})] \\ k_t^\diamond &= (1 - \delta) k_t^\diamond + f(k_t^\diamond, A_t) - c_t^\diamond \end{aligned}$$

where $C_{t+1}(\cdot, \cdot; \sigma)$ records explicitly the dependence of the decision function on the uncertainty parameter σ . DecisionFunction

Methods 1 and 2

- In both methods, the point of approximation is computed assuming that today's and tomorrow's capital are the same because we assume no growth rate of capital.
- Evidently, the first-order perturbation solutions obtained by Method 1 and 2 are identical, as such solutions do not depend on uncertainty.

Methods 3 and 4

- Methods 3 and 4 are the same as Methods 1 and 2, except they explicitly account for time-varying growth rates.
- In both methods, the growth rates $\gamma_{kt} = \tilde{k}_{t+1}/\tilde{k}_t$ are computed from $\left\{ \tilde{k}_{t+1} \right\}_{t=1}^T$ that solves (recursively backwards)

$$u'(\tilde{c}_t) = \beta u'(\tilde{c}_{t+1}) \left(1 - \delta + f \left(\tilde{k}_{t+1}, A_{t+1} \right) \right)$$

$$\tilde{k}_{t+1} = (1 - \delta) \tilde{k}_t + f \left(\tilde{k}_t, A_t \right) - \tilde{c}_t$$

- Method 3 perturbs the solution in period t around a point $(k_t^*, 1)$ that solves for k_t^* and c_t^* the following system of two equations:

$$u'(c_t^*) = \beta u'(C_{t+1}(\gamma_{kt} k_t^*, 1)) (1 - \delta + f(\gamma_{kt} k_t^*, A_{t+1}))$$

$$\gamma_{kt} k_t^* = (1 - \delta) k_t^* + f(k_t^*, A_t) - c_t^*$$

- Method 4 finds a perturbation solution around a point $(k_t^\diamond, 1; 0)$ and finds c_t^\diamond and k_t^\diamond by solving

$$u'(c_t^\diamond) = \beta u'(C_{t+1}(\gamma_{kt} k_t^\diamond, 1; 0)) (1 - \delta + f(\gamma_{kt} k_t^\diamond, A_{t+1}))$$

$$\gamma_{kt} k_t^\diamond = (1 - \delta) k_t^\diamond + f(k_t^\diamond, A_t) - c_t^\diamond$$

Method 5

- Method 5 is close to Method 3, but the path for growth rates is computed iteratively.
- We begin by exogenously fixing the path $\{\gamma_{k,t}\}_{t=1}^T$ and obtaining the policy functions for a stochastic version of the model; this is similar to Method 3.
- We simulate the model with the realized values of shocks set to zero,
- ... compute the growth rates of capital over this simulated path,
- ... and obtain the policy functions for a stochastic version of the model.
- We can repeat the previous step as many times as necessary.

Numerical results

- Utility and production function

$$u(c) = \frac{c^{1-\gamma} - 1}{1-\gamma} \quad \text{and} \quad f(k, A) = A^{1-\alpha} k^\alpha$$

- Parameterization

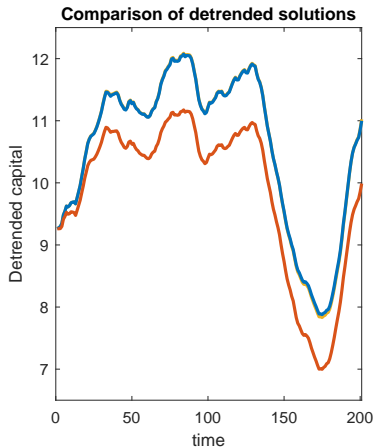
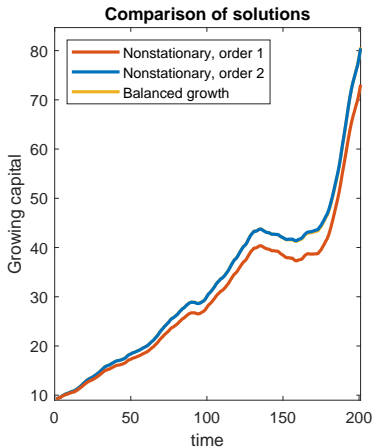
$$\alpha = 0.36, \quad \beta = 0.99, \quad \delta = 0.025, \quad \rho = 0.95$$

$$\gamma = 5, \quad \sigma_\varepsilon = 0.03, \quad \gamma_A = 1.01, \quad T = 200$$

- For all simulations, we use the same initial condition (k_0, z_0) and the same sequence of productivity shocks $\{z_t\}_{t=1}^T$.
- We report absolute unit-free mean and maximum differences between our approximate and balanced growth (“exact”) solutions (in \log_{10} units).

	Method 1	Method 2	Method 3	Method 4	Method 5					
First-order solution										
Errors, in \log_{10} units										
Horizon	Mean	Max	Mean	Max	Mean	Max				
[0, 50]	-1.41	-1.13	-1.50	-1.22	-1.53	-1.23				
[0, 100]	-1.24	-0.97	-1.33	-1.11	-1.35	-1.12				
[0, 150]	-1.14	-0.77	-1.25	-1.08	-1.27	-1.08				
[0, 175]	-1.07	-0.58	-1.22	-0.94	-1.23	-0.97				
[0, 200]	-1.04	-0.58	-1.19	-0.94	-1.20	-0.97				
Running time, in seconds										
Solution	161.57	157.60	294.28	288.03	317.38					
Second-order solution										
Errors, in \log_{10} units										
Horizon	Mean	Max	Mean	Max	Mean	Max	Mean	Max	Mean	Max
[0, 50]	-2.28	-2.03	-2.83	-2.53	-3.48	-2.95	-2.79	-2.24	-3.51	-2.92
[0, 100]	-2.12	-1.90	-2.77	-2.53	-3.30	-2.93	-2.46	-2.05	-3.30	-2.91
[0, 150]	-2.05	-1.80	-2.75	-2.53	-3.26	-2.88	-2.33	-2.03	-3.26	-2.88
[0, 175]	-2.00	-1.71	-2.71	-2.15	-3.14	-2.23	-2.38	-2.03	-3.14	-2.23
[0, 200]	-2.04	-1.71	-2.61	-1.79	-3.07	-2.23	-2.43	-2.03	-3.09	-2.23
Running time, in seconds										
Solution	167.99	167.18	308.87	296.46	337.60					

Approximate and exact solutions



Comparison to the other methods

	Fair-Taylor (1983) method	Naive method	Global EFP	P-EFP	
Type of approximation	path	path	3rd order	1st order	2nd order
Maximum errors, in \log_{10} units					
[0, 50]	-1.29	-1.04	-6.35	-1.23	-2.92
[0, 100]	-1.18	-0.92	-4.76	-1.12	-2.91
[0, 150]	-1.14	-0.89	-3.22	-1.08	-2.88
[0, 175]	-1.14	-0.89	-2.47	-0.97	-2.23
[0, 200]	-1.14	-0.89	-1.51	-0.97	-2.23
Running time, in seconds					
Solution	1.2(+4)	28.9	199.4	317.4	337.6
Simulation	-	2.6	0.0244	0.0271	0.0348
Total	1.2(+4)	31.5	199.4	317.4	337.6

Numbers for Fair-Taylor (1983), Naive, Global EFP are taken from MMTT (2020)

A scaled-down version of ToTEM

- We take a scaled-down version of ToTEM constructed in LMM (2020)
- DSGE model of Canadian economy that captures the main features of ToTEM, the main Bank of Canada policy model
- Households of the same type with differentiated labour services
- Production of final-goods and production of commodities
- Taylor-type interest rate rule
- Meaningful trade
 - ▶ some final goods and commodities are exported
 - ▶ foreign imports are used in domestic production

Two-stage production of final goods

- 1 In the first stage, intermediate goods are produced by identical perfectly competitive firms from labour, capital, commodities, and imports
- 2 In the second stage, a variety of final goods are produced by monopolistically competitive firms from the intermediate goods. The variety of final goods is then aggregated into the final consumption good

First stage of production

- Production of intermediate goods by perfectly competitive firms

$$Z_t^g = \left(\delta_l (A_t L_t)^{\frac{\sigma-1}{\sigma}} + \delta_k (u_t K_t)^{\frac{\sigma-1}{\sigma}} + \delta_{com} (COM_t^d)^{\frac{\sigma-1}{\sigma}} + \delta_m (M_t)^{\frac{\sigma-1}{\sigma}} \right)^{\frac{\sigma}{\sigma-1}}$$

$$\log(A_t) = \varphi_a \log(A_{t-1}) + (1 - \varphi_a) \log(\bar{A}) + \xi_t^a$$

- Capital depreciation with varying depreciation rate

$$K_{t+1} = (1 - d_t) K_t + I_t$$

$$d_t = d_0 + \bar{d} e^{\rho(u_t - 1)}$$

- Quadratic cost of adjusting investment

$$Z_t^n = Z_t^g - \frac{\chi_i}{2} \left(\frac{I_t}{I_{t-1}} - 1 \right)^2 I_t$$

- Firms maximize their profits

$$\max E_0 \sum_{t=0}^{\infty} \mathcal{R}_{0,t} (P_t^z Z_t^n - W_t L_t - P_t^{com} COM_t^d - P_t^i I_t - P_t^m M_t)$$

s.t. the technology constraints

$$\mathcal{R}_{t,t+j} = \beta^j (\lambda_{t+j}/\lambda_t) (P_t/P_{t+j}) = \text{stochastic discount factor}$$

Second stage of production

- Each monopolistically competitive firm produces a differentiated good

$$Z_{it} = \min \left(\frac{Z_{it}^n}{1 - s_m}, \frac{Z_{it}^{mi}}{s_m} \right)$$

- The differentiated goods Z_{it} are aggregated into the final good Z_t

$$Z_t = \left(\int_0^1 Z_{it}^{\frac{\varepsilon-1}{\varepsilon}} di \right)^{\frac{\varepsilon}{\varepsilon-1}}$$

- Cost minimization implies

$$Z_{it} = \left(\frac{P_{it}}{P_t} \right)^{-\varepsilon} Z_t$$

where $P_t = \left(\int_0^1 P_{it}^{1-\varepsilon} di \right)^{\frac{1}{1-\varepsilon}}$

Second stage of production

- Two types of monopolistically competitive firms: rule-of-thumb firms of measure ω and forward-looking firms of measure $1 - \omega$
- Within each type, with probability θ the firms index their price to the (time-varying) inflation target $\bar{\pi}_t$
- With probability $1 - \theta$ the rule-of-thumb firms partially index their price

$$P_{it} = (\pi_{t-1})^\gamma (\bar{\pi}_t)^{1-\gamma} P_{i,t-1}$$

- The optimizing forward-looking firms solve

$$\max_{P_t^*} E_t \left\{ \sum_{j=0}^{\infty} \theta^j \mathcal{R}_{t,t+j} \left(\prod_{k=1}^j \bar{\pi}_{t+k} P_t^* Z_{i,t+j} - (1 - s_m) P_{t+j}^z Z_{i,t+j} - s_m P_{t+j} Z_{i,t+j} \right) \right\}$$

subject to demand constraints

$$\text{s.t. } Z_{i,t+j} = \left(\frac{\prod_{k=1}^j \bar{\pi}_{t+k} P_t^*}{P_{t+j}} \right)^{-\varepsilon} Z_{t+j}$$

Production details

- Intermediate good demand and supply

$$Z_t^n = \int_0^1 Z_{it}^n di = (1 - s_m) \int_0^1 Z_{it} di = (1 - s_m) \Delta_t Z_t$$

$$\Delta_t = \int_0^1 \left(\frac{P_{it}}{P_t} \right)^{-\varepsilon} di = \text{price dispersion}$$

- Law of motion of price dispersion

$$\Delta_t = \theta \left(\frac{\bar{\pi}_t}{\pi_t} \right)^{-\varepsilon} \Delta_{t-1} + (1 - \theta) \omega \left(\frac{(\pi_{t-1})^\gamma (\bar{\pi}_t)^{1-\gamma}}{\pi_t} \right)^{-\varepsilon} \Delta_{t-1} \\ + (1 - \theta) (1 - \omega) \left(\frac{P_t^*}{P_t} \right)^{-\varepsilon}$$

- Investment goods and noncommodity exports are produced from the final goods according to linear technology, $P_t^i = \iota_i P_t$ and $P_t^{nc} = \iota_x P_t$

Imports

- Intermediate imported goods are brought from abroad by monopolistically competitive importers; abroad they are priced in foreign currency
- Final imported good M_t is aggregated from the intermediate imported goods M_{it}

$$M_t = \left(\int_0^1 M_{it}^{\frac{\varepsilon_m - 1}{\varepsilon_m}} di \right)^{\frac{\varepsilon_m}{\varepsilon_m - 1}}$$

- The demand for an intermediate imported good

$$M_{it} = \left(\frac{P_{it}^m}{P_t^m} \right)^{-\varepsilon_m} M_t$$

where $P_t^m = \left(\int_0^1 (P_{it}^m)^{1-\varepsilon_m} di \right)^{\frac{1}{1-\varepsilon_m}}$

Imports

- With probability θ_m the intermediate importers index their selling price (in domestic currency) to the inflation target $\bar{\pi}_t$
- With probability $1 - \theta_m$ the importers optimize profits when their price is effective

$$\max_{P_t^{m*}} E_t \left[\sum_{j=0}^{\infty} (\theta_m)^j \mathcal{R}_{t,t+j} \left(\prod_{k=1}^j \bar{\pi}_{t+k} P_t^{m*} M_{i,t+j} - e_{t+j} P_{t+j}^{mf} M_{i,t+j} \right) \right]$$

subject to demand constraints

$$M_{i,t+j} = \left(\frac{\prod_{k=1}^j \bar{\pi}_{t+k} P_t^{m*}}{P_{t+j}^m} \right)^{-\varepsilon_m} M_{t+j}$$

P_t^{mf} = foreign price of imports;

e_t = nominal exchange rate (domestic price of a unit of foreign currency)

Foreign demand and balance of payments

- The foreign demand for Canadian noncommodity exports

$$X_t^{nc} = \gamma^f \left(\frac{P_t^{nc}}{e_t P_t^f} \right)^{-\varphi} Z_t^f$$

P_t^{nc}/e_t = foreign price of noncommodity exports;

P_t^f = foreign general price level

- The balance of payments

$$\frac{e_t B_t^f}{R_t^f (1 + \kappa_t^f)} - e_t B_{t-1}^f = P_t^{nc} X_t^{nc} + P_t^{com} X_t^{com} - P_t^m M_t$$

B_t^f = domestic holdings of foreign-currency denominated bonds;

κ_t^f = risk premium

Foreign economy dynamics

- Three exogenous processes describe the evolution of the rest of the world
- Foreign demand for Canadian noncommodity exports

$$\log \left(Z_t^f \right) = \varphi_{Zf} \log \left(Z_{t-1}^f \right) + (1 - \varphi_{zf}) \log \left(\bar{Z}^f \right) + \xi_t^{zf}$$

- Foreign interest rate

$$\log \left(r_t^f \right) = \varphi_{rf} \log \left(r_{t-1}^f \right) + (1 - \varphi_{rf}) \log \left(\bar{r} \right) + \xi_t^{rf}$$

- Foreign commodity price

$$\log \left(p_t^{comf} \right) = \varphi_{comf} \log \left(p_{t-1}^{comf} \right) + (1 - \varphi_{comf}) \log \left(\bar{p}^{comf} \right) + \xi_t^{comf}$$

Commodity production

- Commodities are produced from the final goods by perfectly competitive domestic firms

$$COM_t = (Z_t^{com})^{s_z} (A_t F)^{1-s_z} - \frac{\chi_{com}}{2} \left(\frac{Z_t^{com}}{Z_{t-1}^{com}} - 1 \right)^2 Z_t^{com}$$

F = a fixed production factor (land)

- Commodities are sold domestically and exported to the rest of the world

$$COM_t = COM_t^d + X_t^{com}$$

- Commodities are sold at the world price

$$P_t^{com} = e_t P_t^{comf}$$

Households

- Representative household's period utility function

$$U_t = \frac{\mu}{\mu - 1} (C_t - \xi \bar{C}_{t-1})^{\frac{\mu-1}{\mu}} \exp \left(\frac{\eta(1-\mu)}{\mu(1+\eta)} \int_0^1 (L_{ht})^{\frac{\eta+1}{\eta}} dh \right) \eta_t^c$$

\bar{C}_t = aggregate consumption;

L_{ht} = labour service of type h

$$\log(\eta_t^c) = \varphi_c \log(\eta_{t-1}^c) + \xi_t^c$$

Households

- Representative household solves

$$E_t \left[\sum_{j=0}^{\infty} \beta^j U_{t+j} \right]$$

$$\text{s.t. } P_t C_t + \frac{B_t}{R_t} + \frac{e_t B_t^f}{R_t^f (1 + \kappa_t^f)} = B_{t-1} + e_t B_{t-1}^f + \int_0^1 W_{ht} L_{ht} dh + \Pi_t$$

B_t = holdings of domestic bonds;

B_t^f = holdings of foreign-currency denominated bonds

- Endogenous risk premium induces stationarity

$$\kappa_t^f = \varsigma (\bar{b}^f - b_t^f)$$

$$b_t^f = e_t B_t^f / (\pi_{t+1}^f P_t \bar{Y}) = \text{normalized bond holdings}$$

Labour unions

- Representative household supplies a variety of differentiated labour L_{ht}
- Differentiated labour service is aggregated

$$L_t = \left(\int_0^1 L_{ht}^{\frac{\varepsilon_w - 1}{\varepsilon_w}} dh \right)^{\frac{\varepsilon_w}{\varepsilon_w - 1}}$$

and used in the first stage of production

- Cost minimization implies the differentiated labour demand

$$L_{ht} = \left(\frac{W_{ht}}{W_t} \right)^{-\varepsilon_w} L_t$$

W_{ht} = wage for labour of type h ;

$$W_t \equiv \left(\int_0^1 W_{ht}^{1-\varepsilon_w} dh \right)^{\frac{1}{1-\varepsilon_w}}$$

Labour unions

- Two types of labour unions: rule-of-thumb unions of measure ω_w and forward-looking unions of measure $1 - \omega_w$
- Within each type, with probability θ_w a union indexes wage to the inflation target $\bar{\pi}_t$, $W_{it} = \bar{\pi}_t W_{i,t-1}$
- With probability $1 - \theta_w$ rule-of-thumb unions partially index their wage

$$W_{it} = (\pi_{t-1}^w)^{\gamma_w} (\bar{\pi}_t)^{1-\gamma_w} W_{i,t-1}$$

- Forward-looking unions that do not index their wage choose the wage by maximizing household utility when the wage would be effective

$$\begin{aligned} & \max_{W_t^*} E_t \left[\sum_{j=0}^{\infty} (\beta \theta_w)^j U_{t+j} \right] \\ \text{s.t. } & L_{h,t+j} = \left(\frac{\prod_{k=1}^j \bar{\pi}_{t+k} W_t^*}{W_{t+j}} \right)^{-\varepsilon_w} L_{t+j}, \\ & P_{t+j} C_{t+j} = \prod_{k=1}^j \bar{\pi}_{t+k} W_t^* L_{h,t+j} dh + \dots \end{aligned}$$

Monetary policy

- Taylor rule

$$R_t^n = \rho_r R_{t-1} + (1 - \rho_r) [\bar{R} + \rho_\pi (\pi_t - \bar{\pi}_t) + \rho_Y (\log Y_t - \log \bar{Y}_t)] + \eta_t^r$$

$$\eta_t^r = \varphi_r \eta_{t-1}^r + \xi_t^r$$

- Potential output changes with productivity

$$\log \bar{Y}_t = \varphi_{zf} \log \bar{Y}_{t-1} + (1 - \varphi_{zf}) \log \left(\frac{A_t \bar{Y}}{\bar{A}} \right)$$

Market clearing conditions

- Resource feasibility condition

$$Z_t = C_t + \iota_i I_t + \iota_x X_t^{nc} + Z_t^{com} + v_z Z_t$$

- GDP

$$Y_t = C_t + I_t + X_t^{nc} + X_t^{com} - M_t + v_y Y_t$$

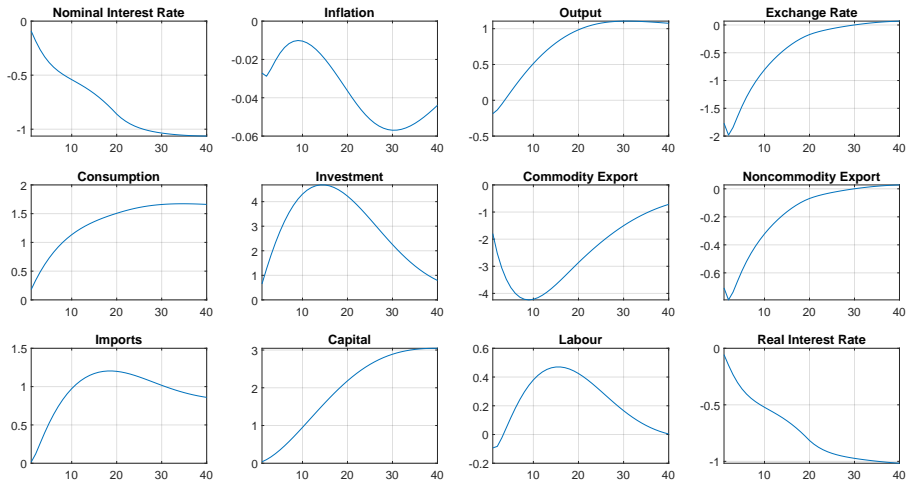
- GDP deflator

$$P_t^y Y_t = P_t C_t + P_t^i I_t + P_t^{nc} X_t^{nc} + P_t^{com} X_t^{com} - P_t^m M_t + v_y p_t^y Y_t$$

A decline in the real neutral interest rate

- Nakamura and Steinsson (2018): Fed's announcements contain information about the path of the neutral interest rate
- We consider an anticipated gradual decline in the neutral interest rate
 - ▶ initial value of the real neutral interest rate is 3 percent
 - ▶ it starts to go down to 2 percent gradually over 20 quarters
 - ▶ the assumed decrease in the neutral interest rate is translated into an increase in the discount factor
- Main findings
 - ▶ gradual increase in consumption, investment, labor, capital, and imports
 - ▶ output expands due to significantly higher capital

A gradual decline in the real neutral interest rate



A change in the inflation target

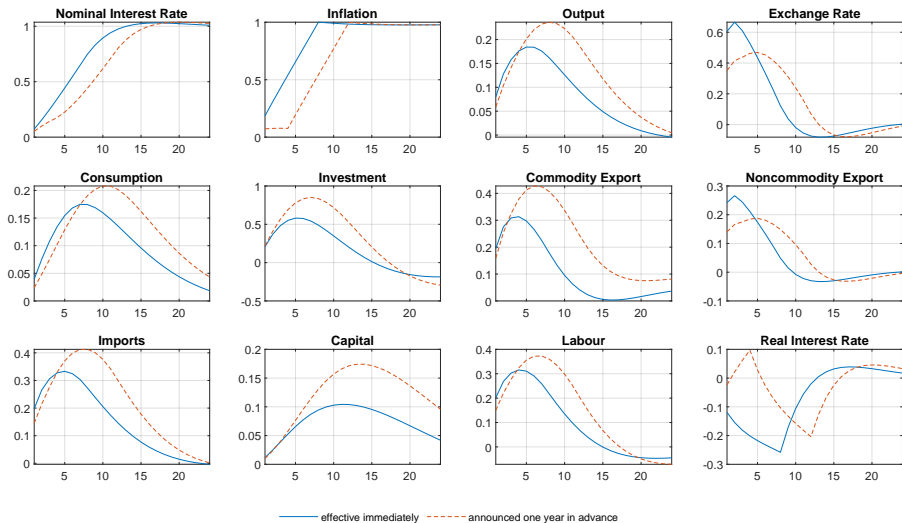
- Following the Great Recession of 2007-2009, there have been proposals to increase the inflation target in developed economies due to ZLB
 - ▶ Summers (1991) and Fischer (1996) suggest to keep an inflation target in the range of 1 to 3 percent mainly because of ZLB
 - ▶ Krugman (1998) proposes to use a 4 percent inflation target in the Japanese economy to deal with persisting deflation
 - ▶ Blanchard, Dell'Arriccia and Mauro (2010), Williams (2009) and Ball (2013) argue that a higher inflation target would have prevented the interest rate from falling to the ZLB
 - ▶ In Canada, the inflation target was maintained at the level of 2 percent since 1995; BoC considered an increase of the target in 2016 as the neutral rate has felt and the ZLB probability has increased; anticipation effects might have been present
- We consider a change in the inflation target $\bar{\pi}_t$ that appears in the Taylor rule
- A new inflation target is announced in advance and everyone considers the announcement to be fully credible
 - 1 at $t = 1$, the CB makes an announcement that starting from $t = 1$, it will gradually increase the inflation target from 2 to 3 percent during 8 quarters
 - 2 the same change takes place but starting from $t = 5$ (i.e., in one year); inflation target remains at the new (higher) level forever

A gradual increase in the inflation target

• Main findings

- ▶ Inflation follows the same pattern as the target
 - ★ assuming full credibility of the inflation-target policy, the agents who are not optimizers index their price by inflation target
- ▶ The economy experiences an investment- and export-driven growth
 - ★ output, investment and commodity exports jump up following the announcement
- ▶ The expansionary effect on the economy is temporary
 - ★ output begins to descend toward its original level after four year
- ▶ It pays for the central bank to announce the policy in advance as output increases more during the transition
 - ★ agents expect the real interest rate to be lower in the near future, and they accumulate more capital in advance of the more favorable environment

A gradual increase in the inflation target



Residuals in the model's equations on the simulated path, \log_{10} units

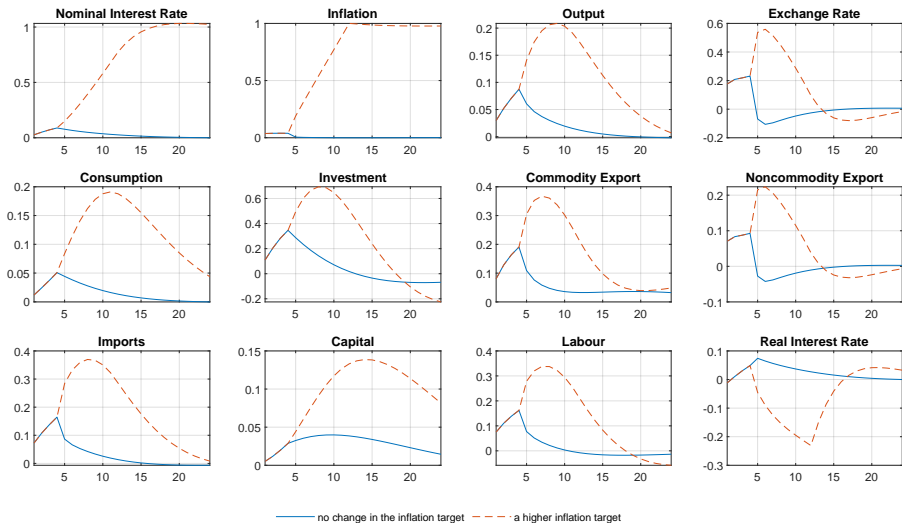
	R_t	π_t	Y_t	C_t	I_t	X_t^{nc}	X_t^{com}	M_t	L_t	K_t
Average	-4.66	-5.25	-4.09	-4.47	-5.11	-4.10	-3.17	-5.10	-4.42	-5.91
Maximum	-4.30	-5.17	-4.03	-4.38	-5.03	-4.08	-3.13	-4.98	-4.37	-5.84

- Residuals are very small, smaller than we report in LMM (2020) for the global solution when the economy was hit by a large foreign demand shock

A gradual increase in the inflation target (50% probability)

- We consider a probabilistic setting in which agents rationally expect that the inflation target might change in the future
 - ▶ a 50-percent chance that starting from $t = 5$ the inflation target $\bar{\pi}_t$ gradually increases from 2 to 3 percent during 8 quarters;
 - ▶ otherwise, the inflation target remains the same
- Our computational method is easy to adapt for modeling of this experiment:
 - ▶ When computing policies in period $t = 4$, we specify explicitly using the Dynare macro language that period-4-expectations are weighted sums of expectations over the two possible realizations in period $t = 5$
- Main findings
 - ▶ Inflation mimics the behavior of the inflation target
 - ▶ All the variables experience mild increases, which are due to anticipatory effects on the side of economic agents.

A gradual increase in the inflation target (50% probability)



Monetary policy normalization

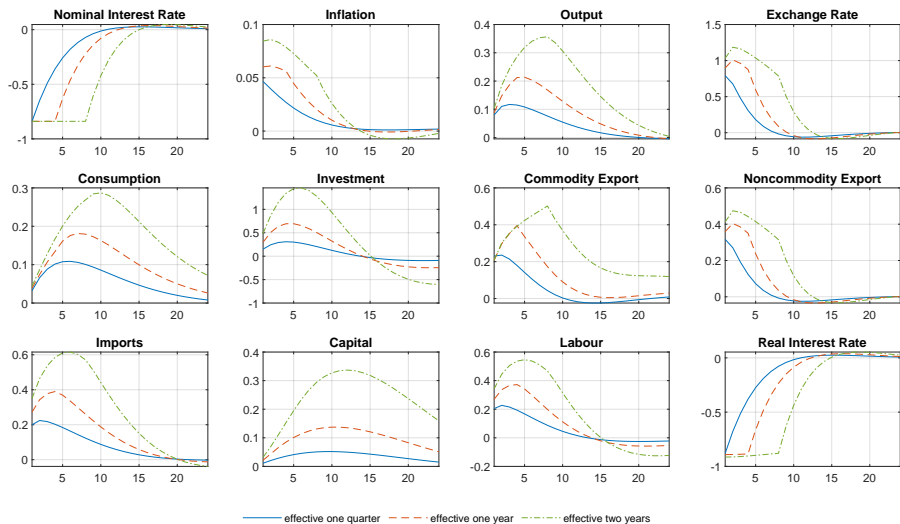
- At ZLB, central banks could not rely on Taylor rules
- Eggertsson and Woodford (2003), Forward Guidance: central bank's promises to keep low interest rates for longer periods helps to alleviate negative consequences of binding ZLB
- We assume that at $t = 1$ the CB announces that it will keep the interest rate at ELB for T periods

$$R_t = R^{elb}$$

and it will return to the standard Taylor rule afterwards

- We compare three cases, depending on whether the interest-rate policy returns to normal: in one quarter, in one year, in two years
- Interest rate is initially below its risky steady state in all the cases
- Main findings
 - ▶ After the policy announcement, the economy experiences an expansion due to strong exports and investment
 - ▶ The initial reaction of output is roughly of equal size in all three cases
 - ▶ The more prolonged is the forward guidance the larger is the expansion

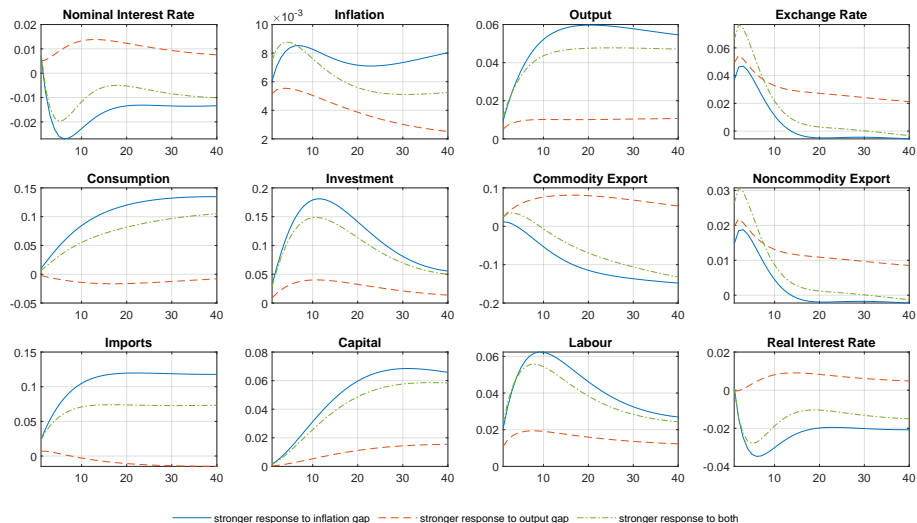
Forward guidance



Switching to a more aggressive Taylor rule

- We consider a one-time change in the sensitivity of the policy rate to inflation and output gap
 - ▶ values of ρ_π and ρ_Y in the Taylor rule are doubled
 - ▶ the change is announced at $t = 1$ and occurs at $t = 2$
- Main findings
 - ▶ The increase in the sensitivity to inflation ρ_π is more effective in expanding the economy, both over the transition and in the long-run
 - ▶ Switching effects are quantitatively small in the absence of shocks

A switch to more aggressive Taylor rules



Switching from inflation targeting to price-level targeting

- Under inflation targeting, past deviations of inflation from target are ignored (“lets bygones to be bygones”)
- Under price-level targeting (PLT), any movement in inflation above target is matched with an equal and opposite movement in inflation below target, so that the economy moves along a predetermined price path
- PLT rule

$$R_t = \rho_r R_{t-1} + (1 - \rho_r) [\bar{R} + \rho_\pi (\log P_t - \log \bar{P}_t) + \rho_Y (\log Y_t - \log \bar{Y}_t)]$$

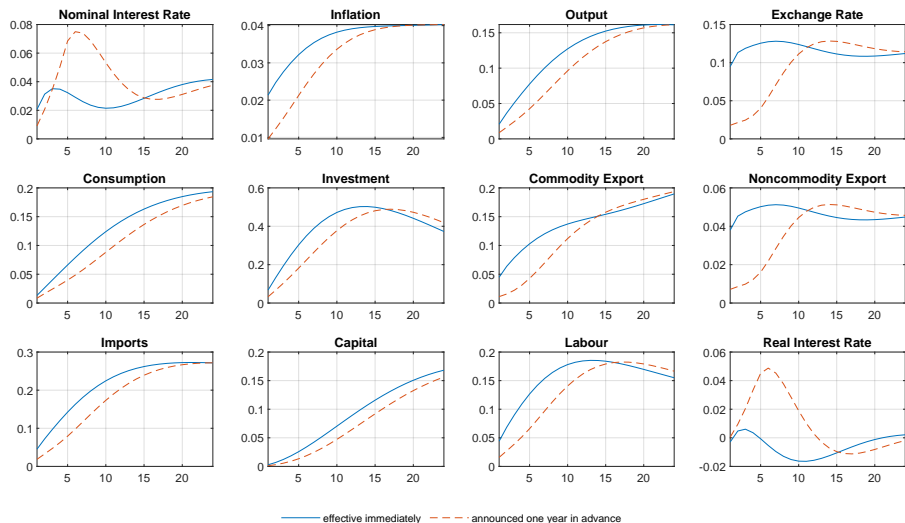
where $\bar{P}_t = \bar{P}_{t-1} \bar{\pi}_t$

- Svensson (1999): PLT reduces inflation variability without an increase in output variability
- Bank of Canada (2011): PLT is marginally beneficial
- Bernanke (2017): temporary PLT at ZLB

Switching from inflation targeting to price-level targeting

- Implementation
 - ▶ The policy change becomes effective either immediately (at $t = 1$) or in one year after being announced (at $t = 5$)
- Main finding
 - ▶ The immediately implemented policy gives larger benefits than the policy announced one year in advance

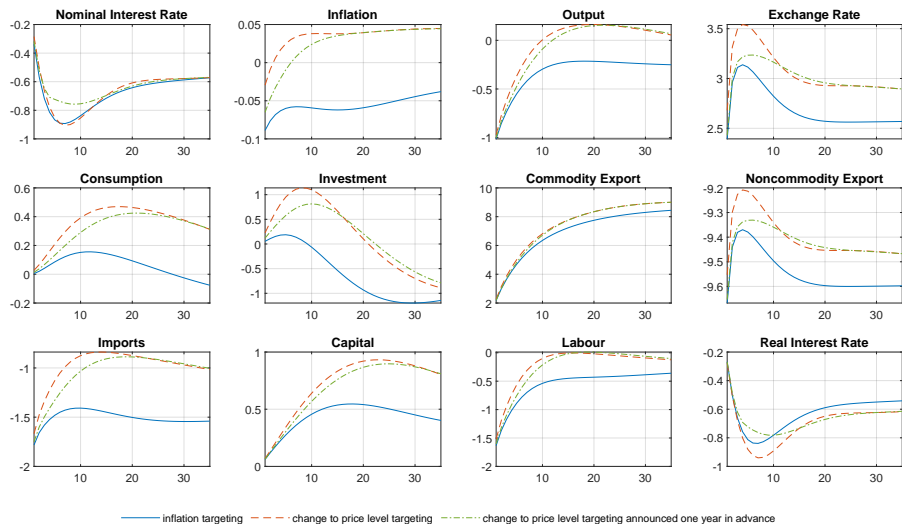
A switch to price-level targeting



Negative foreign demand shock and a switch to price-level targeting

- Another implementation
 - ▶ A negative innovation in the random-walk process for the foreign demand
- Main findings
 - ▶ With the policy switch, output recovers to nearly the same level as before the permanent foreign demand shock
 - ▶ The immediate policy change has larger impacts than the delayed policy change

A negative foreign demand shock and a switch to price-level targeting



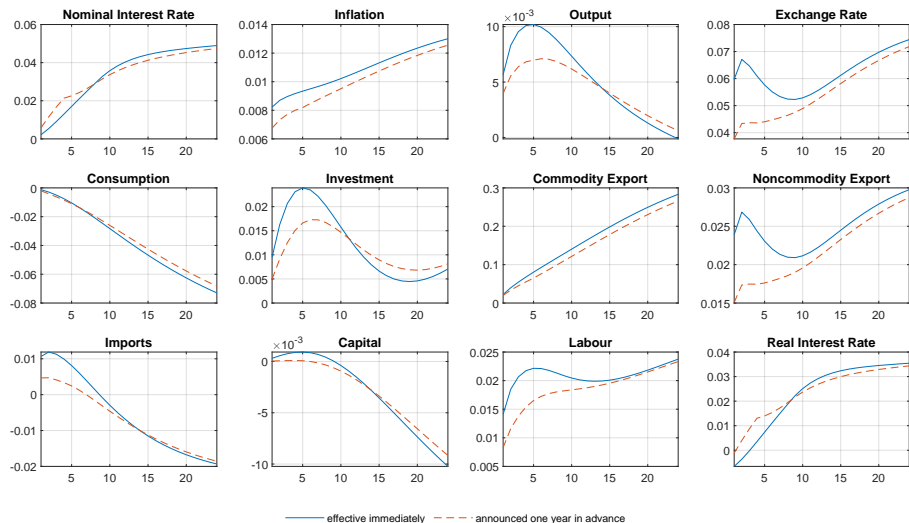
Switching from inflation targeting to average inflation targeting

- Average inflation targeting is a middle ground between price-level targeting and inflation targeting; see Nessén and Vestin (2005)
- Average inflation targeting rule

$$R_t = \rho_r R_{t-1} + (1 - \rho_r) \left[\bar{R} + \rho_\pi \left(\frac{1}{M+1} \sum_{j=0}^M \pi_{t-j} - \bar{\pi}_t \right) + \rho_Y (\log Y_t - \log \bar{Y}_t) \right]$$

- Amano et al. (2020) find that the optimal M ranges from 2 to 8
- We assume $M = 8$ to match the largest value found
- Main finding
 - ▶ Anticipation effects on the economy in the absence of any shocks are very modest

Switching from inflation targeting to average inflation targeting



Are news shocks the same as anticipation effects?

- News shocks could be an important driving force of business cycles; see Barro and King (1984), Beaudry and Portier (2006), Beaudry and Portier (2007), Jaimovich and Rebelo (2009).
- Schmitt-Grohé and Uribe (2012) propose a tractable perturbation framework for dealing with anticipated changes
 - ▶ the changes are modelled as periodic shocks that follow a Markov process
 - ▶ an anticipated shock happens with some specified periodicity

Are news shocks the same as anticipation effects?

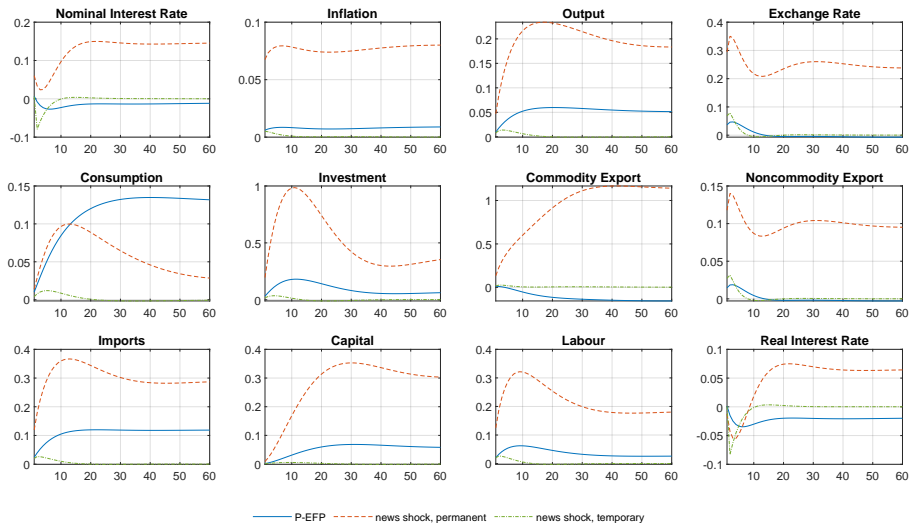
- We consider again a switch to a more aggressive Taylor rule, namely,
 - ▶ the sensitivity to inflation ρ_π in the Taylor rule is doubled
 - ▶ the permanent change in ρ_π is announced at $t = 1$ and implemented at $t = 2$
- The analysis of Schmitt-Grohé and Uribe (2012) does not specify how their method can be used for analyzing non-recurrent anticipated events
- We try two ways of adapting periodic news shocks to our experiment:
 - ▶ a unit-root process, $\rho_{\pi,t} = \rho_{\pi,t-1} + \varepsilon_{t-1}$
 - ★ $\rho_{\pi,0} = \rho_{\bar{\pi}}$, $\varepsilon_1 = \rho_{\bar{\pi}}$ which implies $\rho_{\pi,1} = \rho_{\bar{\pi}}$ and $\rho_{\pi,t} = 2\rho_{\bar{\pi}}$ for all $t \geq 2$
 - ▶ temporary news shock, $\rho_{\pi,t} = \rho_{\bar{\pi}} + \varepsilon_{t-1}$
 - ★ $\rho_{\pi,2} = 2\rho_{\bar{\pi}}$ in period $t = 2$ and $\rho_{\pi,t} = \rho_{\bar{\pi}}$ in all other periods
 - ▶ in both cases, the volatility of the news shock is assumed to be zero
- The initial risky steady state is the same for all three solutions

Are news shocks the same as anticipation effects?

- Main findings

- ▶ Qualitatively similar dynamics (in the initial periods), but numerically different second-order responses
- ▶ The news-shock approach with the permanent shock significantly overstates the importance of anticipated shocks
 - ★ the economy is unexpectedly hit by a shock and remains away from the previous steady state forever
- ▶ Under the first-order perturbation, the economy would remain at the deterministic steady state in the two news-shock solutions

Are news shocks the same as anticipation effects?



Conclusion

- We provide a correct evaluation on the effects of agents' expectations on policy outcomes within a DSGE framework
- We focus on anticipation effects of monetary policy in a context of a realistic DSGE model of central banking
- The anticipation effects are the strongest for such time dependent economic changes as
 - ▶ a gradual change in the natural rate of interest
 - ▶ policy-rate normalization in the aftermath of the ZLB crisis
 - ▶ a gradual change in the inflation target level
- The other time dependent policy changes lead to more modest anticipation effects
 - ▶ a switch to a more aggressive policy rate rule
 - ▶ a switch to price-level targeting
 - ▶ a switch to average inflation targeting
- We offer a simple and tractable perturbation-based framework for solving and simulating a macroeconomic model
- Our analysis makes it possible to construct a precise model-consistent path of real-world economies

Machine learning, artificial intelligence and deep learning methods for dynamic economic models:

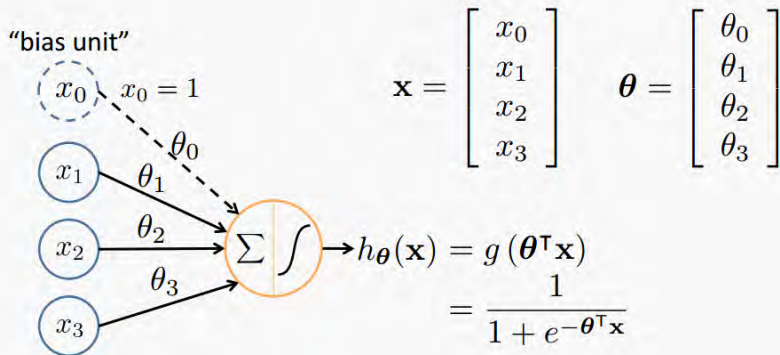
Supervised (deep) learning

Lilia Maliar, Serguei Maliar

November 2022

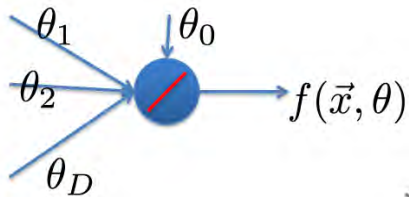
Minicourse

Neuron Model: Logistic Unit

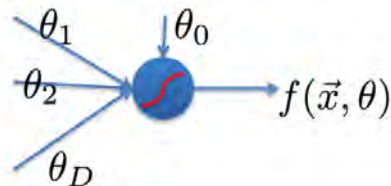


Sigmoid (logistic) activation function: $g(z) = \frac{1}{1 + e^{-z}}$

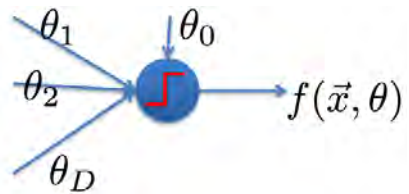
Types of Neurons



Linear Neuron



Logistic Neuron

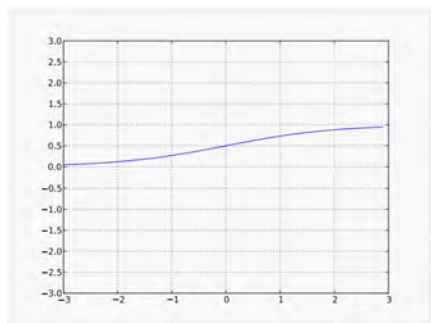


Perceptron

Potentially more. Require a convex loss function for gradient descent training.

Activation - sigmoid

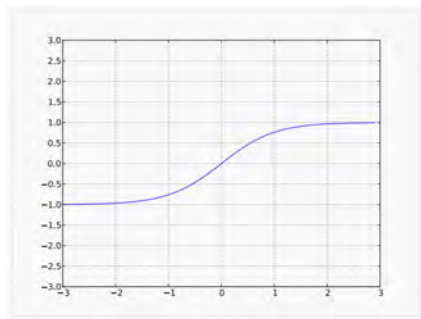
- Squashes the neuron's pre-activation between 0 and 1
- Always positive
- Bounded
- Strictly increasing



$$g(x) = \frac{1}{1 + e^{-x}}$$

Activation - hyperbolic tangent (tanh)

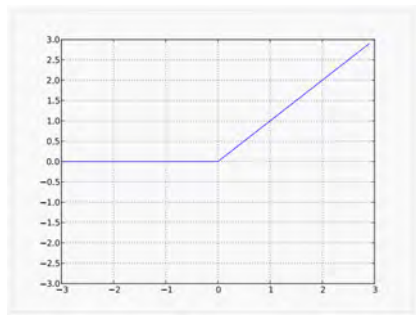
- Squashes the neuron's pre-activation between -1 and 1
- Can be positive or negative
- Bounded
- Strictly increasing



$$g(x) = \tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$

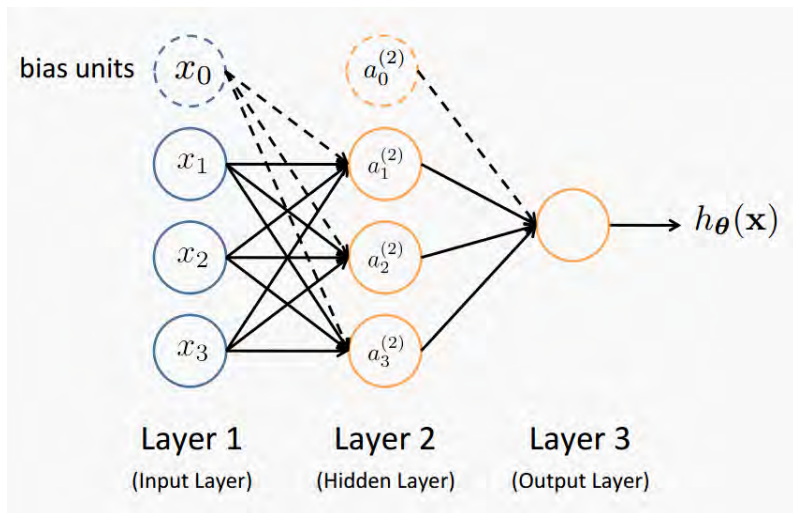
Activation - rectified linear (relu)

- Bounded below by 0, always non-negative
- Not upper bounded
- Tends to give neurons with sparse activities



$$g(x) = \text{relu}(x) = \max(0, x)$$

Deep Neural Network

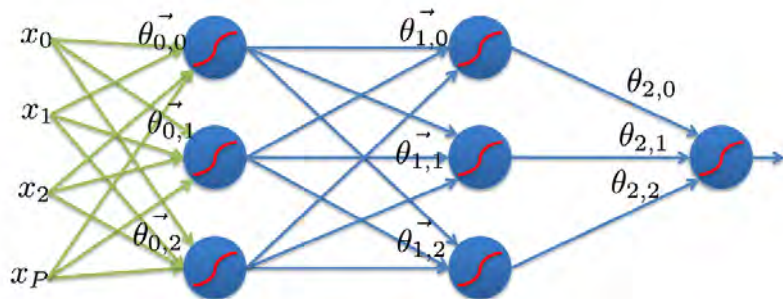


Feed-Forward Process

- Input layer units are set by some exterior function (think of these as sensors), which causes their output links to be activated at the specified level
- Working forward through the network, the input function of each unit is applied to compute the output value
 - Usually this is just the weighted sum of the activation on the links feeding into this node
- The activation function transforms this input function into a final value
 - Typically this is a nonlinear function, often a sigmoid function corresponding to the “threshold” of that node

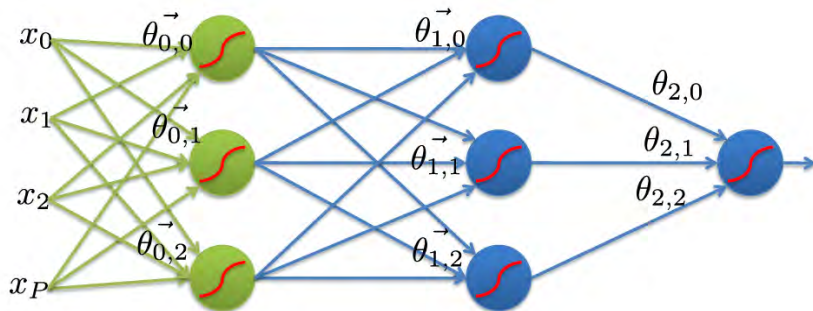
Feed-Forward Networks

Predictions are fed forward through the network to classify.



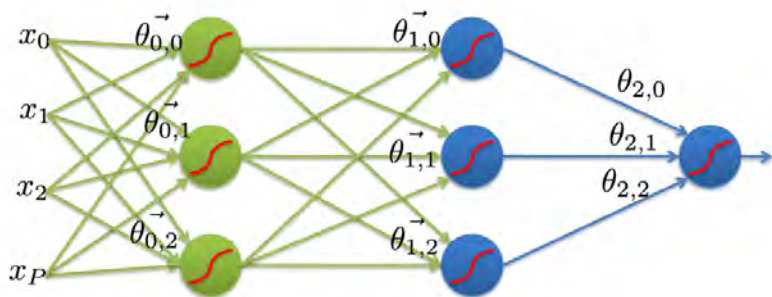
Feed-Forward Networks

Predictions are fed forward through the network to classify



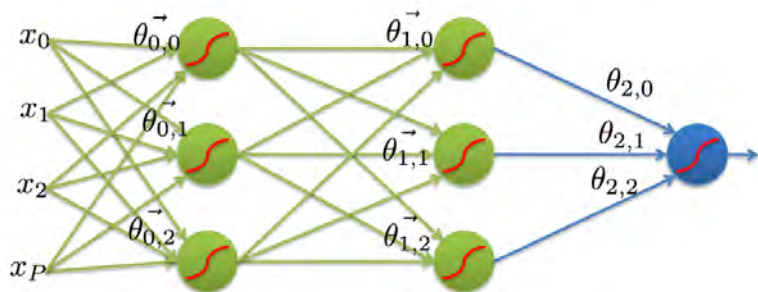
Feed-Forward Networks

Predictions are fed forward through the network to classify



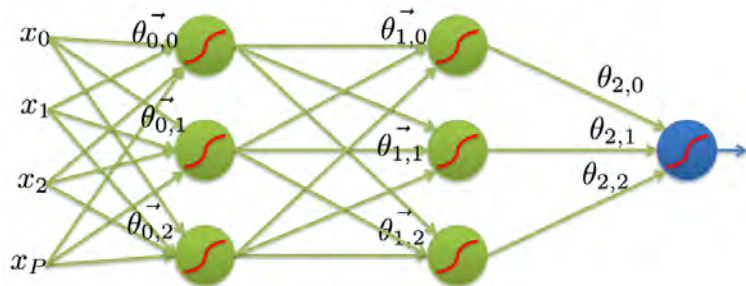
Feed-Forward Networks

Predictions are fed forward through the network to classify



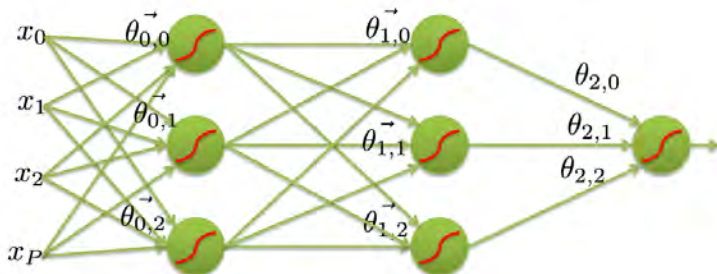
Feed-Forward Networks

Predictions are fed forward through the network to classify

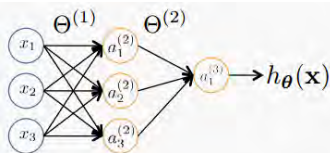


Feed-Forward Networks

Predictions are fed forward through the network to classify



Neural Network



$a_i^{(j)}$ = “activation” of unit i in layer j
 $\Theta^{(j)}$ = weight matrix controlling function mapping from layer j to layer $j + 1$

$$a_1^{(2)} = g(\Theta_{10}^{(1)} x_0 + \Theta_{11}^{(1)} x_1 + \Theta_{12}^{(1)} x_2 + \Theta_{13}^{(1)} x_3)$$

$$a_2^{(2)} = g(\Theta_{20}^{(1)} x_0 + \Theta_{21}^{(1)} x_1 + \Theta_{22}^{(1)} x_2 + \Theta_{23}^{(1)} x_3)$$

$$a_3^{(2)} = g(\Theta_{30}^{(1)} x_0 + \Theta_{31}^{(1)} x_1 + \Theta_{32}^{(1)} x_2 + \Theta_{33}^{(1)} x_3)$$

$$h_{\theta}(x) = a_1^{(3)} = g(\Theta_{10}^{(2)} a_0^{(2)} + \Theta_{11}^{(2)} a_1^{(2)} + \Theta_{12}^{(2)} a_2^{(2)} + \Theta_{13}^{(2)} a_3^{(2)})$$

If network has s_j units in layer j and s_{j+1} units in layer $j+1$, then $\Theta^{(j)}$ has dimension $s_{j+1} \times (s_j + 1)$.

$$\Theta^{(1)} \in \mathbb{R}^{3 \times 4} \quad \Theta^{(2)} \in \mathbb{R}^{1 \times 4}$$

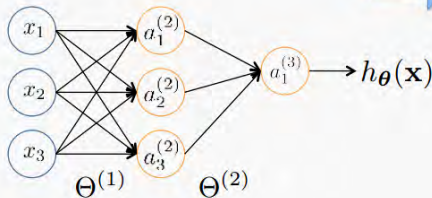
Vectorization

$$a_1^{(2)} = g \left(\Theta_{10}^{(1)} x_0 + \Theta_{11}^{(1)} x_1 + \Theta_{12}^{(1)} x_2 + \Theta_{13}^{(1)} x_3 \right) = g \left(z_1^{(2)} \right)$$

$$a_2^{(2)} = g \left(\Theta_{20}^{(1)} x_0 + \Theta_{21}^{(1)} x_1 + \Theta_{22}^{(1)} x_2 + \Theta_{23}^{(1)} x_3 \right) = g \left(z_2^{(2)} \right)$$

$$a_3^{(2)} = g \left(\Theta_{30}^{(1)} x_0 + \Theta_{31}^{(1)} x_1 + \Theta_{32}^{(1)} x_2 + \Theta_{33}^{(1)} x_3 \right) = g \left(z_3^{(2)} \right)$$

$$h_{\Theta}(\mathbf{x}) = g \left(\Theta_{10}^{(2)} a_0^{(2)} + \Theta_{11}^{(2)} a_1^{(2)} + \Theta_{12}^{(2)} a_2^{(2)} + \Theta_{13}^{(2)} a_3^{(2)} \right) = g \left(z_1^{(3)} \right)$$



Feed-Forward Steps:

$$\mathbf{z}^{(2)} = \Theta^{(1)} \mathbf{x}$$

$$\mathbf{a}^{(2)} = g(\mathbf{z}^{(2)})$$

Add $a_0^{(2)} = 1$

$$\mathbf{z}^{(3)} = \Theta^{(2)} \mathbf{a}^{(2)}$$

$$h_{\Theta}(\mathbf{x}) = \mathbf{a}^{(3)} = g(\mathbf{z}^{(3)})$$

TensorFlow

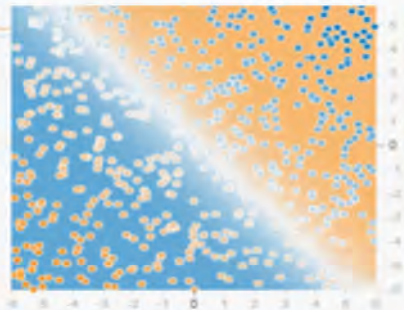
TensorFlow Demonstration

<https://playground.tensorflow.org>

TensorFlow Playground



www.educba.com



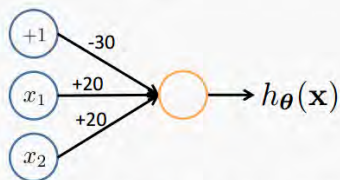
Understanding Representations

Representing Boolean Functions

Simple example: AND

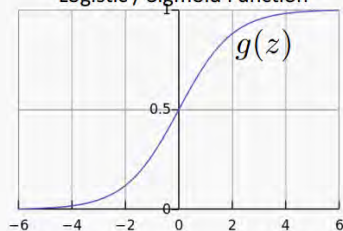
$$x_1, x_2 \in \{0, 1\}$$

$$y = x_1 \text{ AND } x_2$$



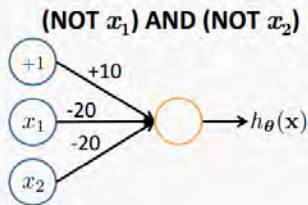
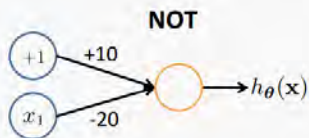
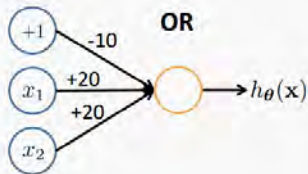
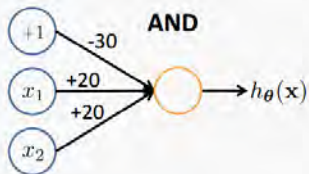
$$h_{\theta}(\mathbf{x}) = g(-30 + 20x_1 + 20x_2)$$

Logistic / Sigmoid Function

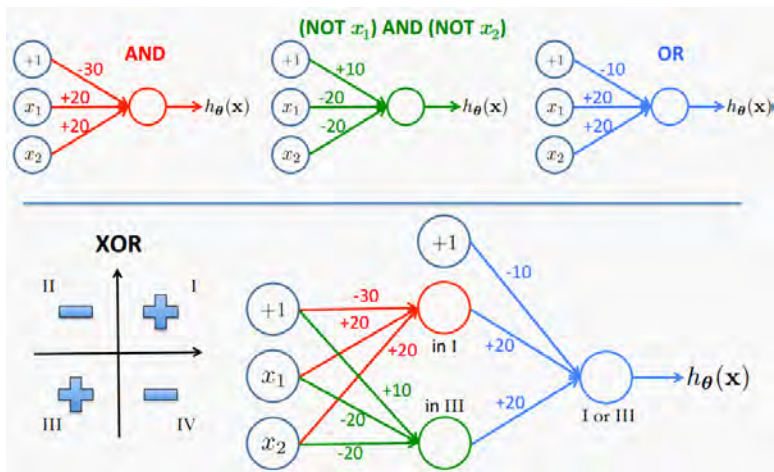


x_1	x_2	$h_{\theta}(\mathbf{x})$
0	0	$g(-30) \approx 0$
0	1	$g(-10) \approx 0$
1	0	$g(-10) \approx 0$
1	1	$g(10) \approx 1$

Representing Boolean Functions



Combining Representations to Create Non-Linear Functions



Multiclass Classification

Multiple Output Units: One-vs-Rest



Pedestrian



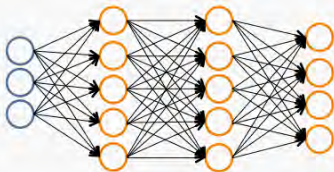
Car



Motorcycle



Truck



$$h_{\Theta}(\mathbf{x}) \in \mathbb{R}^K$$

We want:

$$h_{\Theta}(\mathbf{x}) \approx \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

when pedestrian

$$h_{\Theta}(\mathbf{x}) \approx \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}$$

when car

$$h_{\Theta}(\mathbf{x}) \approx \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}$$

when motorcycle

$$h_{\Theta}(\mathbf{x}) \approx \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$

when truck

Multiple Output Units: One-vs-Rest



We want:

$$h_{\Theta}(\mathbf{x}) \approx \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

when pedestrian

$$h_{\Theta}(\mathbf{x}) \approx \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}$$

when car

$$h_{\Theta}(\mathbf{x}) \approx \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}$$

when motorcycle

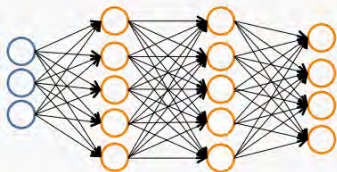
$$h_{\Theta}(\mathbf{x}) \approx \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$

when truck

- Given $\{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \dots, (\mathbf{x}_n, y_n)\}$
- Must convert labels to 1-of- K representation

– e.g., $y_i = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}$ when motorcycle, $y_i = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}$ when car, etc.

Neural Network Classification



Binary classification

$y = 0$ or 1

1 output unit ($s_{L-1} = 1$)

Given:

$\{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \dots, (\mathbf{x}_n, y_n)\}$

$\mathbf{s} \in \mathbb{N}^{+L}$ contains # nodes at each layer

– $s_0 = d$ (# features)

Multi-class classification (K classes)

$\mathbf{y} \in \mathbb{R}^K$ e.g. $\begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$, $\begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}$, $\begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$, $\begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$
pedestrian car motorcycle truck

K output units ($s_{L-1} = K$)

Activation - softmax

- For multi-class classification:
 - we need multiple outputs (1 output per class)
 - we would like to estimate the conditional probability $p(y = c | x)$
- We use the softmax activation function at the output:

$$g(x) = \text{softmax}(x) = \left[\frac{e^{x_1}}{\sum_c e^{x_c}} \dots \frac{e^{x_c}}{\sum_c e^{x_c}} \right]$$

Layering Representations



20×20 pixel images

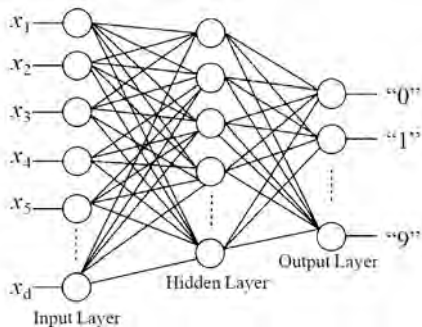
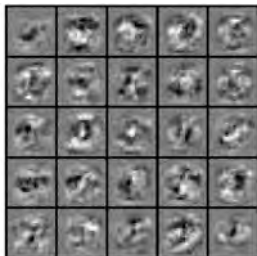
$d = 400$ 10 classes



Each image is “unrolled” into a vector x of pixel intensities.

$x_1 \dots x_{20}$
:
 $x_{381} \dots x_{400}$

Layering Representations

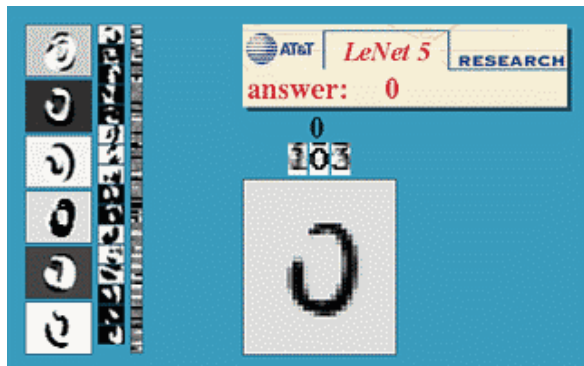


Visualization of Hidden Layer

LeNet 5

- LeNet 5 Demonstration

<http://yann.lecun.com/exdb/lenet>



Neural Network Learning

Universal approximation theorem

“a single hidden layer neural network with a linear output unit can approximate any continuous function arbitrarily well, given enough hidden units”

Hornik, 1991

Learning in NN: Backpropagation

- We cycle through our examples
 - If the output of the network is correct, no changes are made
 - If there is an error, weights are adjusted to reduce the error
- The trick is to assess the blame for the error and divide it among the contributing weights

Cost function for logistic regression

- Consider a classification problem with two labels $y \in \{0, 1\}$.
- To understand the cost function of the neural network, we first derive the cost function for logistic regression using the maximum likelihood estimator.
- Let us denote

$$\begin{aligned}p(y = 1 \mid x; \theta) &= h_{\theta}(x) \\p(y = 0 \mid x; \theta) &= 1 - h_{\theta}(x)\end{aligned}$$

- Note that this formula can be written compactly as

$$p(y \mid x; \theta) = (h_{\theta}(x))^y (1 - h_{\theta}(x))^{1-y}$$

- Indeed if $y = 1$, then $p(y \mid x; \theta) = (h_{\theta}(x))^1 (1 - h_{\theta}(x))^0$
- Indeed if $y = 0$, then $p(y \mid x; \theta) = (h_{\theta}(x))^0 (1 - h_{\theta}(x))^1$

Cost function for logistic regression (cont.)

- Assume that n samples are generated independently, we get

$$\begin{aligned}L(\theta) &= p(\bar{\mathbf{y}} \mid \bar{\mathbf{x}}; \theta) = \prod_{i=1}^n p\left(y^{(i)} \mid x^{(i)}; \theta\right) \\ &= \prod_{i=1}^n \left(h_{\theta}\left(x^{(i)}\right)\right)^{y^{(i)}} \left(1 - h_{\theta}\left(x^{(i)}\right)\right)^{1 - y^{(i)}}\end{aligned}$$

- Taking the log to simplify calculations, we get

$$\begin{aligned}\ell(\theta) &= \log L(\theta) \\ &= \sum_{i=1}^n y^{(i)} \log\left(h_{\theta}\left(x^{(i)}\right)\right) + \left(1 - y^{(i)}\right) \log\left(1 - h_{\theta}\left(x^{(i)}\right)\right)\end{aligned}$$

Cost Function: logistic regression versus neural network

Logistic Regression:

$$J(\theta) = -\frac{1}{n} \sum_{i=1}^n [y_i \log h_{\theta}(\mathbf{x}_i) + (1 - y_i) \log (1 - h_{\theta}(\mathbf{x}_i))] + \frac{\lambda}{2n} \sum_{j=1}^d \theta_j^2$$

Neural Network:

$$h_{\Theta} \in \mathbb{R}^K \quad (h_{\Theta}(\mathbf{x}))_i = i^{\text{th}} \text{ output}$$

$$J(\Theta) = -\frac{1}{n} \left[\sum_{i=1}^n \sum_{k=1}^K y_{ik} \log (h_{\Theta}(\mathbf{x}_i))_k + (1 - y_{ik}) \log \left(1 - (h_{\Theta}(\mathbf{x}_i))_k \right) \right]$$
$$+ \frac{\lambda}{2n} \sum_{l=1}^{L-1} \sum_{i=1}^{s_{l-1}} \sum_{j=1}^{s_l} \left(\Theta_{ji}^{(l)} \right)^2$$

k^{th} class:	true, predicted
not k^{th} class:	true, predicted

Optimizing the Neural Network

$$J(\Theta) = -\frac{1}{n} \left[\sum_{i=1}^n \sum_{k=1}^K y_{ik} \log(h_{\Theta}(\mathbf{x}_i))_k + (1 - y_{ik}) \log(1 - (h_{\Theta}(\mathbf{x}_i))_k) \right] \\ + \frac{\lambda}{2n} \sum_{l=1}^{L-1} \sum_{i=1}^{s_{l-1}} \sum_{j=1}^{s_l} (\Theta_{ji}^{(l)})^2$$

Solve via: $\min_{\Theta} J(\Theta)$

$J(\Theta)$ is not convex, so GD on a neural net yields a local optimum

- But, tends to work well in practice

Need code to compute:

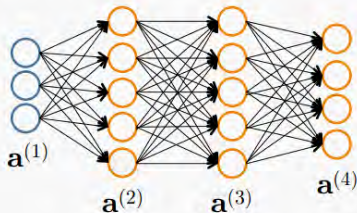
- $J(\Theta)$
- $\frac{\partial}{\partial \Theta_{ij}^{(l)}} J(\Theta)$

Forward Propagation

- Given one labeled training instance (\mathbf{x}, y) :

Forward Propagation

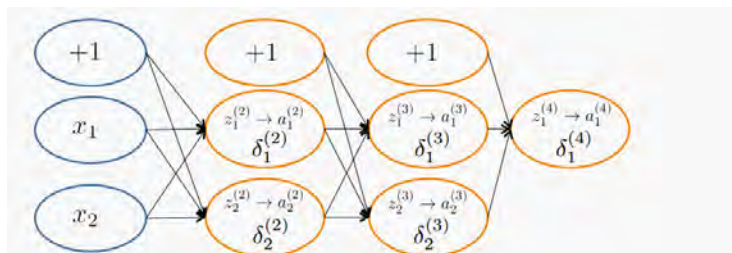
- $\mathbf{a}^{(1)} = \mathbf{x}$
- $\mathbf{z}^{(2)} = \Theta^{(1)}\mathbf{a}^{(1)}$
- $\mathbf{a}^{(2)} = g(\mathbf{z}^{(2)})$ [add $a_0^{(2)}$]
- $\mathbf{z}^{(3)} = \Theta^{(2)}\mathbf{a}^{(2)}$
- $\mathbf{a}^{(3)} = g(\mathbf{z}^{(3)})$ [add $a_0^{(3)}$]
- $\mathbf{z}^{(4)} = \Theta^{(3)}\mathbf{a}^{(3)}$
- $\mathbf{a}^{(4)} = h_{\Theta}(\mathbf{x}) = g(\mathbf{z}^{(4)})$



Backpropagation Intuition

- Each hidden node j is “responsible” for some fraction of the error $\delta_j^{(l)}$ in each of the output nodes j to which it connects
- $\delta_j^{(l)}$ is divided according to the strength of the j connection between hidden node and the output node
- Then, the “blame” is propagated back to provide the error values for the hidden layer.

Backpropagation Intuition

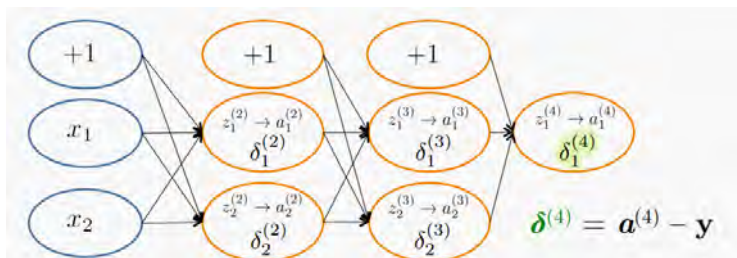


$\delta_j^{(l)}$ = “error” of node j in layer l

$$\text{Formally, } \delta_j^{(l)} = \frac{\partial}{\partial z_j^{(l)}} \text{cost}(\mathbf{x}_i)$$

where $\text{cost}(\mathbf{x}_i) = y_i \log h_{\Theta}(\mathbf{x}_i) + (1 - y_i) \log(1 - h_{\Theta}(\mathbf{x}_i))$

Backpropagation Intuition

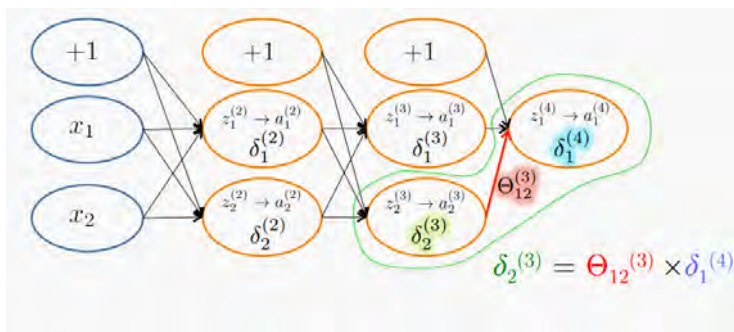


$\delta_j^{(l)}$ = "error" of node j in layer l

$$\text{Formally, } \delta_j^{(l)} = \frac{\partial}{\partial z_j^{(l)}} \text{cost}(\mathbf{x}_i)$$

where $\text{cost}(\mathbf{x}_i) = y_i \log h_{\Theta}(\mathbf{x}_i) + (1 - y_i) \log(1 - h_{\Theta}(\mathbf{x}_i))$

Backpropagation Intuition

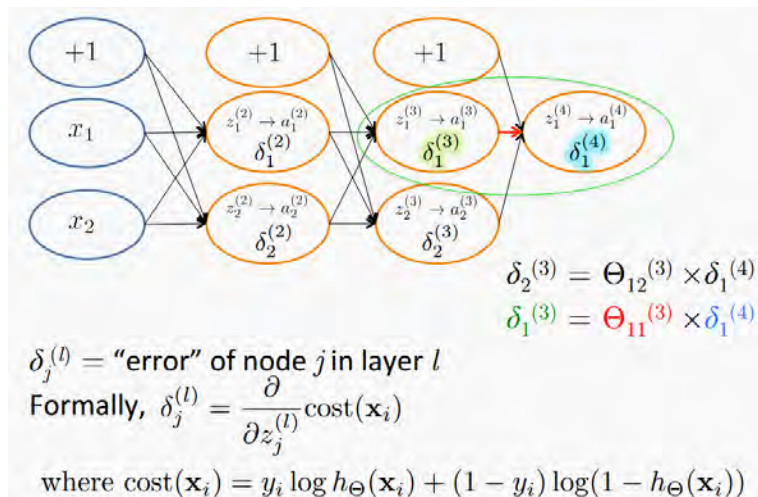


$\delta_j^{(l)}$ = “error” of node j in layer l

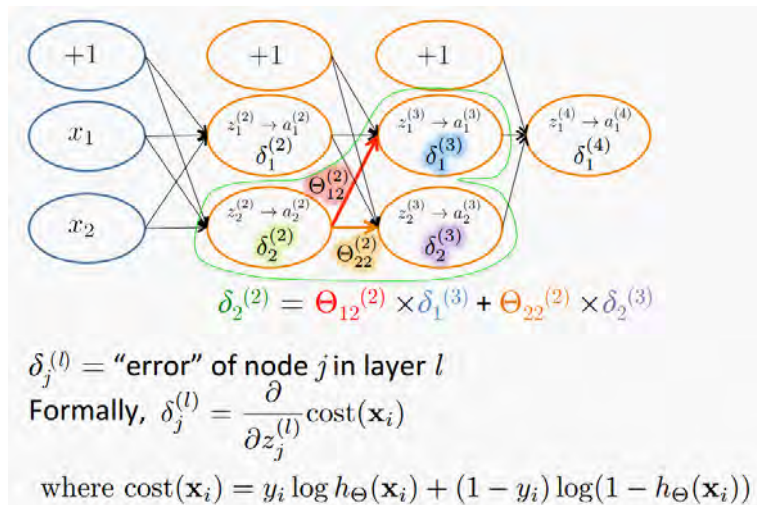
Formally, $\delta_j^{(l)} = \frac{\partial}{\partial z_j^{(l)}} \text{cost}(\mathbf{x}_i)$

where $\text{cost}(\mathbf{x}_i) = y_i \log h_{\Theta}(\mathbf{x}_i) + (1 - y_i) \log(1 - h_{\Theta}(\mathbf{x}_i))$

Backpropagation Intuition



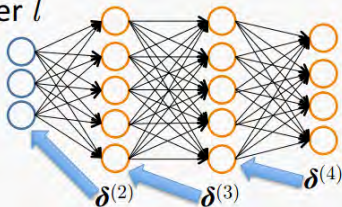
Backpropagation Intuition



Backpropagation: Gradient Computation

Let $\delta_j^{(l)}$ = “error” of node j in layer l

(#layers $L = 4$)



Backpropagation

- $\delta^{(4)} = \mathbf{a}^{(4)} - \mathbf{y}$
- $\delta^{(3)} = (\Theta^{(3)})^T \delta^{(4)} .* g'(\mathbf{z}^{(3)})$
- $\delta^{(2)} = (\Theta^{(2)})^T \delta^{(3)} .* g'(\mathbf{z}^{(2)})$
- (No $\delta^{(1)}$)

Element-wise product $.*$

$$g'(\mathbf{z}^{(3)}) = \mathbf{a}^{(3)} .* (1 - \mathbf{a}^{(3)})$$

$$g'(\mathbf{z}^{(2)}) = \mathbf{a}^{(2)} .* (1 - \mathbf{a}^{(2)})$$

$$\frac{\partial}{\partial \Theta_{ij}^{(l)}} J(\Theta) = a_j^{(l)} \delta_i^{(l+1)}$$

(ignoring λ ; if $\lambda = 0$)

Training a Neural Network via Gradient Descent with Backprop

Given: training set $\{(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_n, y_n)\}$

Initialize all $\Theta^{(l)}$ randomly (NOT to 0!)

Loop // each iteration is called an epoch

Set $\Delta_{ij}^{(l)} = 0 \quad \forall l, i, j$ (Used to accumulate gradient)

For each training instance (\mathbf{x}_i, y_i) :

Set $\mathbf{a}^{(1)} = \mathbf{x}_i$

Compute $\{\mathbf{a}^{(2)}, \dots, \mathbf{a}^{(L)}\}$ via forward propagation

Compute $\delta^{(L)} = \mathbf{a}^{(L)} - y_i$

Compute errors $\{\delta^{(L-1)}, \dots, \delta^{(2)}\}$

Compute gradients $\Delta_{ij}^{(l)} = \Delta_{ij}^{(l)} + a_j^{(l)} \delta_i^{(l+1)}$

Compute avg regularized gradient $D_{ij}^{(l)} = \begin{cases} \frac{1}{n} \Delta_{ij}^{(l)} + \lambda \Theta_{ij}^{(l)} & \text{if } j \neq 0 \\ \frac{1}{n} \Delta_{ij}^{(l)} & \text{otherwise} \end{cases}$

Update weights via gradient step $\Theta_{ij}^{(l)} = \Theta_{ij}^{(l)} - \alpha D_{ij}^{(l)}$

Until weights converge or max #epochs is reached

Backpropagation

Backprop Issues

“Backprop is the cockroach of machine learning. It’s ugly, and annoying, but you just can’t get rid of it.”

- Geoff Hinton

Problems:

- black box
- local minima

Machine learning, artificial intelligence and deep learning methods for dynamic economic models:

Unsupervised learning

Lilia Maliar, Serguei Maliar

November 2022

Minicourse

Hypercube versus ergodic set

- Conventional projection methods including the Smolyak method operate on exogenously given hypercube.
- However, many areas of the hypercube have low probability of occurrence - we might not need to know the solution in low probability areas.
- Stochastic simulation methods construct the solution on a set of simulated points where the solution "lives".

Operating on the ergodic set can save on cost a lot!

Illustrative example: a representative-agent model

The representative-agent neoclassical stochastic growth model:

$$\max_{\{k_{t+1}, c_t\}_{t=0}^{\infty}} E_0 \sum_{t=0}^{\infty} \beta^t u(c_t)$$

$$\text{s.t. } c_t + k_{t+1} = (1 - \delta) k_t + \theta_t f(k_t),$$

$$\ln \theta_{t+1} = \rho \ln \theta_t + \epsilon_{t+1}, \quad \epsilon_{t+1} \sim \mathcal{N}(0, \sigma^2),$$

where initial condition (k_0, θ_0) is given;

$u(\cdot)$ = utility function; $f(\cdot)$ = production function;

c_t = consumption; k_{t+1} = capital; θ_t = productivity;

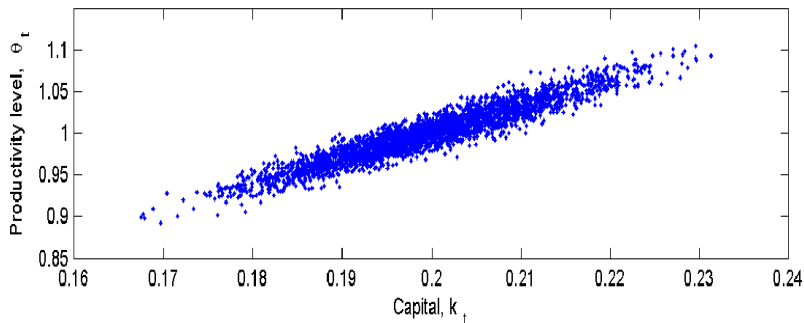
β = discount factor; δ = depreciation rate of capital;

ρ = autocorrelation coefficient of the productivity level;

σ = standard deviation of the productivity shock.

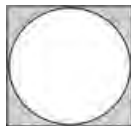
Advantage of stochastic simulation method

Advantage of stochastic simulation method: "Grid" is adaptive: we solve the model only in the area of the state space that is visited in simulation.



Reduction in cost in a 2-dimensional case

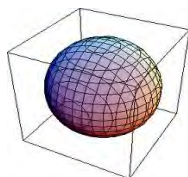
- How much can we save on cost using the ergodic-set domain comparatively to the hypercube domain?
- Suppose the ergodic set is a circle (it was an ellipse in the figure).
- In the 2-dimensional case, a circle inscribed within a square occupies about 79% of the area of the square.
- The reduction in cost is proportional to the shaded area in the figure.



- It does not seem to be a large saving in cost.

Reduction in cost in a d-dimensional case

- In a 3-dimensional case, the gain is larger (a volume of a sphere of diameter 1 is 52% of the volume of a cube of width 1)



- In a d -dimensional case, the ratio of a hypersphere's volume to a hypercube's volume

$$\mathcal{V}^d = \begin{cases} \frac{(\pi/2)^{\frac{d-1}{2}}}{1 \cdot 3 \cdot \dots \cdot d} & \text{for } d = 1, 3, 5, \dots \\ \frac{(\pi/2)^{\frac{d}{2}}}{2 \cdot 4 \cdot \dots \cdot d} & \text{for } d = 2, 4, 6, \dots \end{cases}$$

- \mathcal{V}^d declines very rapidly with dimensionality of state space. When $d = 10 \Rightarrow \mathcal{V}^d = 3 \cdot 10^{-3}$ (0.3%). When $d = 30 \Rightarrow \mathcal{V}^d = 2 \cdot 10^{-14}$.
- We face a tiny fraction of cost we would have faced on the hypercube.

Starting point: Parameterized expectations algorithm (PEA) of den Hann and Marcet (1990)

- PEA solves a model as follows:

Step 1. Guess policy functions / value function.

Step 2. Simulate time series solution.

Step 3. Use simulation results to recompute the guess.

Iterate on *Steps 2* \square *3* until convergence.

- Step 3 requires
 - to fit an approximating function to the simulated data (regression);
 - to evaluate conditional expectations (integration).

PEA of den Haan and Marcet (1990) more formally

Parameterize the marginal utility function,

$$u'(c_t) = E_t \{ \beta u'(c_{t+1}) [1 - \delta + z_{t+1} f'(k_{t+1})] \} \approx \Psi(k_t, z_t; b),$$

where $\Psi(k_t, z_t; b) = \exp(b_0 + b_1 \ln k_t + b_2 \ln z_t + \dots + b_n (\ln z_t)^L)$ is an exponentiated polynomial. Write the constraint as

$$k_{t+1} = (1 - \delta) k_t + z_t f(k_t) - u'^{\square 1}[\Psi(k_t, z_t; b)].$$

- Fix $b = (b_0, \dots, b_n)$. Given $\{z_t\}_{t=0}^T$, simulate $\{c_t, k_{t+1}\}_{t=0}^T$ and construct

$$y_t \equiv \beta u'(c_{t+1}) [1 - \delta + z_{t+1} f'(k_{t+1})],$$

- Run a non-linear LS (NLLS) regression $y_t = \Psi(k_t, z_t; b) + \varepsilon \Rightarrow$ get \hat{b} .
- Compute the next-iteration input $b^{(j+1)}$ using fixed-point iteration

$$b^{(j+1)} = (1 - \xi) b^{(j)} + \xi \hat{b},$$

where $\xi \in (0, 1]$ = damping parameter.

A promising stochastic integration method

PEA uses simulated series for two objectives:

- A solution domain.
- A grid of nodes for approximating conditional expectation.

A specific one-node Monte Carlo integration method is used:

$$E_t [u' (c_{t+1}) [1 - \delta + z_{t+1} f' (k_{t+1})]] \\ \approx u' (c_{t+1}) [1 - \delta + z_{t+1} f' (k_{t+1})] \equiv y_t.$$

Here, integral is approximated by a next-period realization of the integrand:

$$integral(t + 1) \approx one\ random\ draw\ of\ integrand(t + 1)$$

1. *This is an inaccurate integration method for each given period but the inaccuracies offset one another when we run a regression to approximate the expectation function (although at a slow squareroot rate of convergence).*
2. *But this is a cheap integration method. Just one integration node (simulated point) is used in each period.*

Numerical instability problem

- PEA works well for 1st-degree polynomials but is numerically unstable under higher (even 2-nd) degree polynomials.
- For example, Den Haan and Marcet (1990) removed a cross term $\ln k_t \ln z_t$ in the 2-nd degree polynomial,

$$\exp \left(b_0 + b_1 \ln k_t + b_2 \ln z_t + b_3 \ln k_t^2 + b_4 \ln z_t^2 + \underbrace{b_5 \ln k_t \ln z_t}_{\text{was removed}} \right).$$

- What is the reason for numerical stability?
- Under *the linear regression model*, $y = Xb + \varepsilon$, we have the OLS estimator

$$\hat{b} = (X^T X)^{-1} X^T y,$$

where X in the above PEA example is

$$X = \begin{pmatrix} 1 & k_0 & z_0 & k_0^2 & z_0^2 & k_0 z_0 \\ \dots & & & \dots & & \dots \\ 1 & k_T & z_T & k_T^2 & z_T^2 & k_T z_T \end{pmatrix}$$

- It turned out that monomial terms constructed on simulated series are highly collinear so that the LS problem is ill-conditioned.

Ill-conditioned LS problem

- The degree of ill-conditioning of $X^T X$ is measured by the condition number

$$\kappa(X^T X) \equiv \frac{\lambda_1}{\lambda_n}$$

λ_1 = the largest eigenvalue of $X^T X$; λ_n = its smallest eigenvalue.

- Ill-conditioning:** $\kappa(X^T X)$ is large $\implies X^T X$ is close to being singular (not invertible).

Multicollinearity of variables produces ill-conditioning

Under ordinary polynomials, monomials forming X are highly correlated, OLS coefficients "jump" and the iterative process fails to converge.

Example

Consider an approximation problem $y = Xb + \varepsilon$ such that $y = (0, 0)^T$ and

$$\begin{bmatrix} 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 + \phi & 1 \\ 1 & 1 + \phi \end{bmatrix} \begin{bmatrix} b_1 \\ b_2 \end{bmatrix} + \begin{bmatrix} \varepsilon_1 \\ \varepsilon_2 \end{bmatrix}.$$

The OLS solution is

$$\hat{b}_1 = \frac{1}{\phi} \left[\frac{\varepsilon_2 - \varepsilon_1 (1 + \phi)}{2 + \phi} \right] \quad \text{and} \quad \hat{b}_2 = \frac{1}{\phi} \left[\frac{\varepsilon_1 - \varepsilon_2 (1 + \phi)}{2 + \phi} \right].$$

Sensitivity of \hat{b}_1 and \hat{b}_2 to perturbation in $(\varepsilon_1, \varepsilon_2)^T$ is proportional to $1/\phi$. If $\phi \approx 0$ (multicollinearity), then a small perturbation $(\varepsilon_1, \varepsilon_2)^T$ produces large changes in \hat{b}_1 and \hat{b}_2 .

Poor scaling of variables also produces ill-conditioning

Polynomial terms forming X have very different means and variances (due to different scales among either the state variables, k_t and z_t , or the polynomial terms of different orders, like k_t and k_t^5).

Example

Consider an approximation problem $y = Xb + \varepsilon$ such that $y = (0, 0)^T$ and

$$\begin{bmatrix} 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & \phi \end{bmatrix} \begin{bmatrix} b_1 \\ b_2 \end{bmatrix} + \begin{bmatrix} \varepsilon_1 \\ \varepsilon_2 \end{bmatrix}.$$

The OLS solution is

$$\hat{b}_1 = \varepsilon_1 \text{ and } \hat{b}_2 = \frac{\varepsilon_2}{\phi}.$$

Sensitivity of \hat{b}_2 to perturbation in ε_2 is proportional to $1/\phi$. If $\phi \approx 0$ (poor scaling), then a small perturbation ε_2 produces a large change in \hat{b}_2 .

Generalized Stochastic Simulation Algorithm (GSSA)

Kenneth L. Judd, Lilia Maliar and Serguei Maliar, (2011). Numerically stable and accurate stochastic simulation approaches for solving dynamic models. *Quantitative Economics* 2, 173-210.

GSSA augments PEA with **machine learning methods**, in particular, regularization and principal component analysis which allows to

- correct numerical instability of PEA;
- attain high accuracy of solutions.

Preprocessing

- To use the machine learning methods, you need to preprocess X by normalizing the data.
- *Center* - subtract the sample mean from each observation.
- *Scale* - divide each observation by the sample standard deviation.
- Scaling helps to have comparable range of values.
- By construction, a centered variable has a zero mean, and a scaled variable has a unit standard deviation.
- After a regression model is estimated, the coefficients in the original (unnormalized) regression model are restored.

LS approaches to the linear regression model

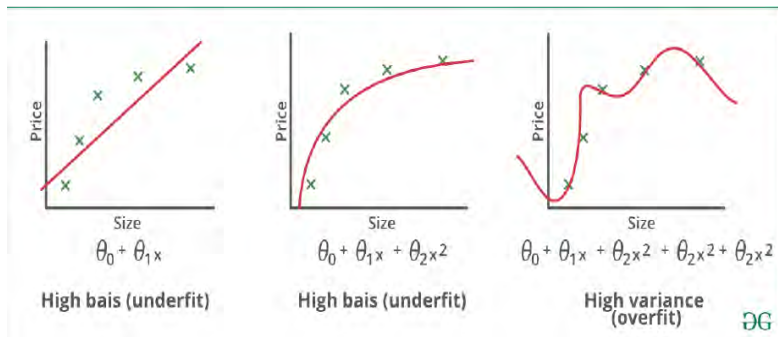
Two LS approaches that are more numerically stable and more suitable for dealing with ill-conditioning than the standard OLS approach.

1. *Regularized LS using Tikhonov regularization (RLS-Tikhonov)*: relies on a specific (Tikhonov) regularization of the ill-conditioned LS problem that imposes penalties based on the size of the regression coefficients.
2. *LS using SVD or truncated SVD (principle component analysis (PCA))*: uses a singular-value decomposition of X .

The LS-SVD approach finds a solution to the original ill-conditioned LS problem, while the RLS-Tikhonov and PCA approach modify (regularizes) the original ill-conditioned LS problem into a less ill-conditioned problem.

Regularization

- Regularization - process of re-formulating an ill-conditioned problem by imposing additional restrictions on the solution.
- In machine learning, it is used for the problem of overfitting.



In the lasy case, impose a penalty: $1000 \times \theta_3^2$ and $1000 \times \theta_4^2$.

RLS-Tikhonov

- Tikhonov regularization - the most commonly used regularization method in approximation theory.
- Impose an L_2 penalty on the size of the regression coefficients:

$$\min_b \|y - Xb\|_2^2 + \eta \|b\|_2^2 = \min_b (y - Xb)^\top (y - Xb) + \eta b^\top b$$

where $\eta \geq 0$ = regularization parameter.

- Find the FOC with respect to b

$$\hat{b}(\eta) = (X^\top X + \eta I_n)^{-1} X^\top y$$

where I_n = an identity matrix of order n .

- *Note:* add a positive constant to $X^\top X$ prior to inverting this matrix. \implies Even if $X^\top X$ is singular, the matrix $X^\top X + \eta I_n$ is non-singular. \implies Can compute its inverse.
- **Another regularization scheme is Lasso:** $\dots + \eta \|b\|_1$ (or $|b_1| + \dots + |b_n|$).

Singular Value Decomposition (SVD)

- Handy mathematical technique that has application to many problems
- Given any $m \times n$ matrix A , compute "eigenvectors" and "eigenvalues" of matrix A

$$[U, S, V] = \text{svd}(A)$$

The diagram shows the equation $A = UDV^T$. The matrix A is in grey. U is in pink, D is in blue, and V is in orange. A pink arrow points from the text "Left singular vectors" to U . A blue arrow points from the text "Singular values" to D . An orange arrow points from the text "Right singular vectors" to V . The superscript T is in grey.

U is $m \times n$ and orthonormal

S is $n \times n$ and diagonal

V is $n \times n$ and orthonormal.

Singular Value Decomposition (not full ranked)

We can also compute SVD of X which is not full ranked:


$$[U, S, V] = \text{svd}(X)$$

$$\begin{pmatrix} X \end{pmatrix} = \begin{pmatrix} U \end{pmatrix} \begin{pmatrix} s_1 & 0 & 0 \\ 0 & \dots & 0 \\ 0 & 0 & s_n \end{pmatrix} \begin{pmatrix} V \end{pmatrix}^T$$

left singular values singular values right singular values

- If X is singular, some of the s_i will be 0.
- In general $\text{rank}(X) = \text{number of nonzero } s_i$.
- SVD is mostly unique (up to permutation of singular values, or if some s_i are equal).
- Elements of S , s_1, \dots, s_n , are $\sqrt{\lambda_1}, \dots, \sqrt{\lambda_n}$.
- Columns of V are eigenvectors of $X'X$.

Singular Value Decomposition (example)



SVD - Example

A = U Λ V^T - example:

			retrieval						
			inf.	brain	lung				
		data	↓						
↑		$\begin{bmatrix} \text{CS} \\ \downarrow \\ \text{MD} \\ \downarrow \end{bmatrix}$	$\begin{bmatrix} 1 & 1 & 1 & 0 & 0 \\ 2 & 2 & 2 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 \\ 5 & 5 & 5 & 0 & 0 \\ 0 & 0 & 0 & 2 & 2 \\ 0 & 0 & 0 & 3 & 3 \\ 0 & 0 & 0 & 1 & 1 \end{bmatrix}$	$=$	$\begin{bmatrix} 0.18 & 0 \\ 0.36 & 0 \\ 0.18 & 0 \\ 0.90 & 0 \\ 0 & 0.53 \\ 0 & 0.80 \\ 0 & 0.27 \end{bmatrix}$	\times	$\begin{bmatrix} 9.64 & 0 \\ 0 & 5.29 \end{bmatrix}$	\times	$\begin{bmatrix} 0.58 & 0.58 & 0.58 & 0 & 0 \\ 0 & 0 & 0 & 0.71 & 0.71 \end{bmatrix}$

LS-SVD

- Back to the GSSA algorithm.
- SVD of the matrix $X \in \mathbb{R}^{T \times n}$

$$X = USV^T$$

where $U \in \mathbb{R}^{T \times n}$ and $V \in \mathbb{R}^{n \times n}$ = orthogonal matrices ($U^T U = U U^T = I$ and $U^T = U^{-1}$); $S \in \mathbb{R}^{n \times n}$ = diagonal matrix with diagonal entries $s_1 \geq s_2 \geq \dots \geq s_n \geq 0$, *singular values* of X .

- The OLS estimator $\hat{b} = (X^T X)^{-1} X^T y$ in terms of the SVD:

$$\hat{b} = (V S^T S V^T)^{-1} V S^T U^T y = V S^{-1} U^T y$$

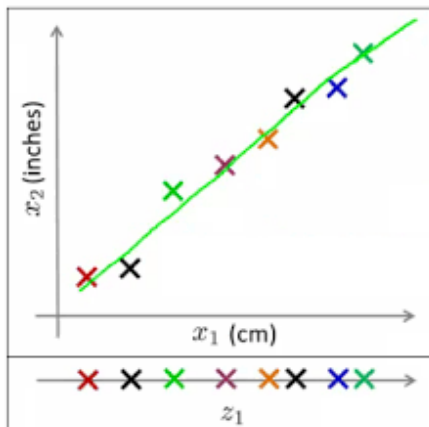
- If $X^T X$ is well-conditioned \implies the OLS formula and the LS-SVD formula give identical estimates of b .
- However, if $X^T X$ is ill-conditioned and the standard OLS estimator cannot be computed \implies it is still possible that matrices X and S are sufficiently well-conditioned, $\mathcal{K}(S) = \sqrt{\mathcal{K}(X^T X)}$ \implies can compute the LS-SVD estimator.

Principal component method: Idea

- Here, we use principal component (PC) method to reduce the degree of multicollinearity.
- PC analysis is also used in machine learning for dimensionality reduction.
- Suppose you have two variables, x_1 and x_2 that are highly collinear.
- To reduce from two dimensions to one dimension, we find a direction (a vector u_1) onto which to project the data so as to minimize the projection error.

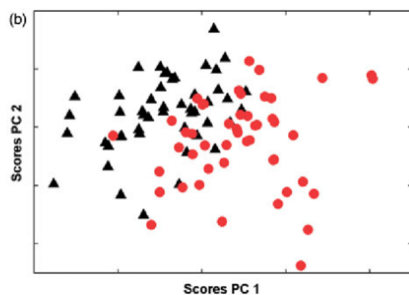
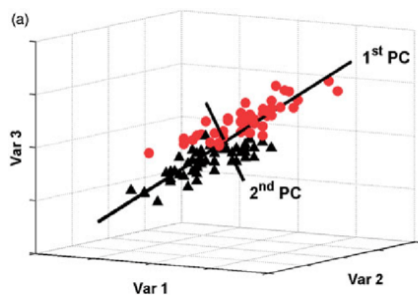
Dimensionality reduction in machine learning

Motivation I: data compression, e.g., reduce dimensionality: from 2 to 1 features



Dimensionality reduction in machine learning

Motivation II: Data Visualization, e.g., visualize 3d figure in 2d



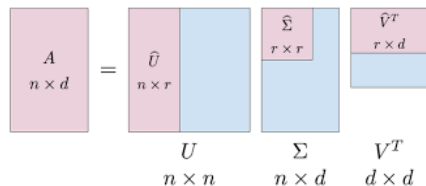
Dimensionality reduction with SVD

Dimensionality reduction with SVD

```
[U,S,V]=svd(A);
```

```
Ureduce = U(:,1:r);
```

```
z = Ureduce'*x;
```



Reconstruction from Compressed Representation

SVD – Reconstruction

$$A=USV^T \quad (\text{full decomposition})$$

$$A_k=U_k S_k V_k^T \quad (\text{reconstruction})$$

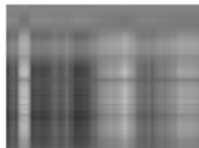
- S_k : the first k singular values.
- U_k : the first k columns of U ,
- V_k : the first k columns of U

$$\|A - A_k\|_{\text{Frobenius}} = \min_B \|A - B\|_{\text{Frobenius}}$$

where B is any rank k matrix

$$\begin{bmatrix} 1 & 1 & 1 & 0 & 0 \\ 2 & 2 & 2 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 \\ 5 & 5 & 5 & 0 & 0 \\ 0 & 0 & 0 & 2 & 2 \\ 0 & 0 & 0 & 3 & 3 \\ 0 & 0 & 0 & 1 & 1 \end{bmatrix} \sim \begin{bmatrix} 1 & 1 & 1 & 0 & 0 \\ 2 & 2 & 2 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 \\ 5 & 5 & 5 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Image reconstruction example



(a) 1 principal component



(b) 5 principal component



(c) 9 principal component



(d) 13 principal component



(e) 17 principal component



(f) 21 principal component



(g) 25 principal component



(h) 29 principal component

Algorithm for choosing the number of principal components

The singular values are ordered by the number of importance for explaining the variation.

$$[U, S, V] = \text{svd}(X), \quad S = \begin{bmatrix} s_1 & 0 & 0 \\ 0 & \dots & 0 \\ 0 & 0 & s_n \end{bmatrix}$$

s_1 is the first most important value,
 s_2 is the second most important value, etc.

- Pick smallest value of r for which

$$\frac{\sum_{i=1}^r S_{ii}}{\sum_{i=1} S_{ii}} \geq 0.99$$

(99% of variance retained)

Principal component method in GSSA

- $Z \equiv XV$, where $X \in \mathbb{R}^{T \times n}$, $Z \in \mathbb{R}^{T \times n}$ and $V \in \mathbb{R}^{n \times n}$.
- z_1, \dots, z_n are *principal components* of X .
- They are orthogonal, $z_i^\top z_i = s_i^2$ and $z_j^\top z_i = 0$ for any $j \neq i$, where $s_i = i$ th singular value of X .
- *Idea*: reduce ill-conditioning of X to a "desired" level by excluding low-variance principal components corresponding to small singular values.
- Let $\kappa =$ largest condition number of X that we are willing to accept.
- Compute $\frac{s_1}{s_2}, \dots, \frac{s_1}{s_n}$, where $s_1 =$ largest singular value.
- $\mathcal{K}(X) = \frac{s_1}{s_n} =$ actual condition number of the matrix X .

Principal component method

- Let $Z^r \equiv (z_1, \dots, z_r) \in \mathbb{R}^{T \times r}$ be the first r principal components for which $\frac{s_1}{s_i} \leq \kappa$.
- Remove the last $n - r$ principal components for which $\frac{s_1}{s_i} > \kappa$.
- By construction, $\mathcal{K}(Z^r) \leq \kappa$.
- Re-write the linear regression model in terms of Z^r ,

$$y = Z^r \vartheta^r + \varepsilon,$$

where $\vartheta^r \in \mathbb{R}^r$ = vector of coefficients.

- Estimate ϑ^r using any of the LS methods described.
- Find $\hat{b} = V^r \hat{\vartheta}^r \in \mathbb{R}^n$, where $V^r = (v_1, \dots, v_r) \in \mathbb{R}^{n \times r}$ contains the first r right singular vectors of X .

Epsilon Distinguishable Set and Cluster Grid Algorithms

Merging projection and stochastic simulation

Lilia Maliar and Serguei Maliar, (2015). “Merging simulation and projection approaches to solve high-dimensional problems with an application to a new Keynesian model”, *Quantitative Economics* 6, 1-47.

What do we do?

- **Similar to stochastic simulation approach:** use simulation to identify and approximate the ergodic set.
- **Similar to projection approach:** construct a fixed grid and use quadrature integration to accurately solve the model on that grid.
- We use integration and optimization methods that are tractable in high-dimensional problems: *non-product monomial integration formulas and derivative-free solvers.*

Epsilon-distinguishable set (EDS) algorithm

The clustering-style EDS technique is the key novel piece of our analysis

The EDS grid construction:

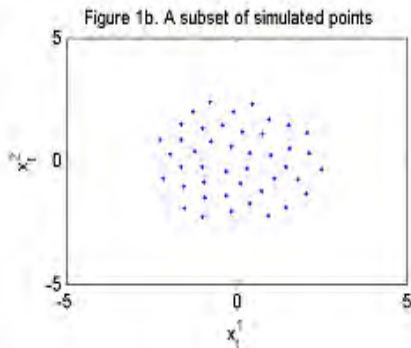
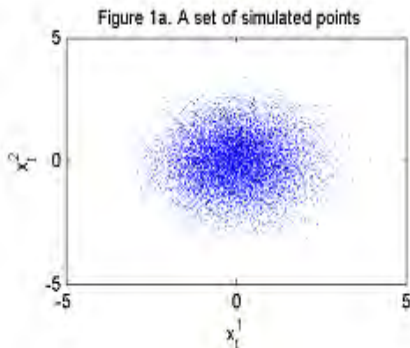
- we select an ε -distinguishable subset of simulated points that covers the support of the ergodic measure roughly uniformly.
- " ε -distinguishable set (EDS)" = a set of points situated at the distance at least ε from one another, where $\varepsilon > 0$ is a parameter.

We also use conventional clustering analysis:

- hierarchical agglomerative,
- k-mean.

A grid of points covering support of the ergodic measure

An illustration of an ε -distinguishable set.



A class of stochastic processes

Let us guess a solution to the model (for example, take perturbation solution).

A class of discrete-time stochastic processes:

$$x_{t+1} = \varphi(x_t, \epsilon_{t+1}), \quad t = 0, 1, \dots,$$

$\epsilon \in E \subseteq \mathbb{R}^p$ = vector of p independent and identically distributed shocks;
 $x \in X \subseteq \mathbb{R}^d$ = vector of d (exogenous and endogenous) state variables;
 x is endowed with its relative Borel σ -algebra denoted by \mathbb{X} .

- *Example, $k_{t+1} = K(k_t, \theta_t)$ and $\theta_{t+1} = \theta_t^\rho \exp(\epsilon_{t+1})$.*

Assumption 1. *There exists a unique ergodic set \mathcal{A}^* and the associated ergodic measure μ .*

Assumption 2. *The ergodic measure μ admits a representation in the form of a density function $g : X \rightarrow \mathbb{R}^+$ such that $\int_{\mathcal{A}} g(x) dx = \mu(\mathcal{A})$ for every $\mathcal{A} \subseteq \mathbb{X}$.*

A two-step clustering technique

A two-step procedure for forming a discrete approximation to the ergodic set.

1. We identify an area of the state space that contains nearly all the probability mass.
2. We cover this area with a finite set of points that are roughly evenly spaced.

An essentially ergodic set

We define a high-probability area of the state space using the level set of the density function g .

Def. A set $\mathcal{A}^\eta \subseteq \mathcal{A}^*$ is called a η -level ergodic set if $\eta > 0$ and

$$\mathcal{A}^\eta \equiv \{x \in X : g(x) \geq \eta\}.$$

- The mass of \mathcal{A}^η under the density $g(x)$ is equal to $p(\eta) \equiv \int_{g(x) \geq \eta} g(x) dx$.
- If $p(\eta) \approx 1$, then \mathcal{A}^η contains all X except for points where the density is lowest.
- In this case, \mathcal{A}^η is called an *essentially ergodic set*.

Law of iterated logarithm

LIL: The ergodic measure can be approximated by simulation.

$P =$ random draws $x_1, \dots, x_n \subseteq \mathbb{R}^d$ generated with $\mu : \mathbb{R}^d \rightarrow \mathbb{R}^+$.

$C(P; J) =$ counts the number of points from P in a given $J \subseteq \mathbb{R}^d$.

$\mathcal{J} =$ intersection of all subintervals $\prod_{i=1}^d [\square\infty, v_i)$, where $v_i > 0$.

Proposition: (Law of iterated logarithm). For every dimensionality d and every continuous function μ , we have

$$\lim_{n \rightarrow \infty} \left\{ \sup_{J \in \mathcal{J}} \left| \frac{C(P; J)}{n} - \mu(J) \right| \cdot \left(\frac{2n}{\log \log n} \right)^{1/2} \right\} = 1, \quad \text{a.e.}$$

Proof: See Kiefer (1961, Theorem 2).

That is, the empirical distribution function $\hat{\mu}(J) \equiv \frac{C(P; J)}{n}$ converges asymptotically to the true distribution function $\mu(J)$ for every $J \in \mathcal{J}$ at the rate given by $\left(\frac{2n}{\log \log n} \right)^{1/2}$.

Multivariate kernel density estimation

(Algorithm \mathcal{A}^η): Selection of points within an essentially ergodic set.

Step 1 Simulate $x_{t+1} = \varphi(x_t, \epsilon_{t+1})$ for T periods.

Step 2 Select each κ th point to get a set P of n points $x_1, \dots, x_n \in X \subseteq \mathbb{R}^d$.

Step 3 Estimate the density function $\hat{g}(x_i) \approx g(x_i)$ for all $x_i \in P$.

Step 4 Remove all points for which the density is below η .

To estimate the density function \hat{g} from the simulated data, we use a multivariate kernel algorithm

$$\hat{g}(x) = \frac{1}{n(2\pi)^{d/2} \bar{h}^d} \sum_{i=1}^n \exp \left[-\frac{D(x, x_i)^2}{2\bar{h}^2} \right],$$

where \bar{h} is the bandwidth parameter, and $D(x, x_i)$ is the distance between x and x_i .

- The complexity of Algorithm \mathcal{A}^η is $O(n^2)$ because it requires to compute pairwise distances between all the sample points.
- We remove 5% of the sample which has the lowest density.

Constructing EDS

Def. Let (X, D) be a bounded metric space. A set P^ε consisting of points $x_1^\varepsilon, \dots, x_M^\varepsilon \in X \subseteq \mathbb{R}^d$ is called ε -distinguishable if $D(x_i^\varepsilon, x_j^\varepsilon) > \varepsilon$ for all $1 \leq i, j \leq M : i \neq j$, where $\varepsilon > 0$ is a parameter.

(Algorithm P^ε): Construction of an EDS.

Let P be a set of n point $x_1, \dots, x_n \in X \subseteq \mathbb{R}^d$.

Let P^ε begin as an empty set, $P^\varepsilon = \{\emptyset\}$.

Step 1 Select $x_i \in P$. Compute $D(x_i, x_j)$ to all x_j in P .

Step 2 Eliminate from P all x_j for which $D(x_i, x_j) < \varepsilon$.

Step 3 Add x_i to P^ε and eliminate it from P .

Iterate on Steps 1-3 until all points are eliminated from P .

Proposition: The complexity of Algorithm P^ε is of order $O(nM)$.

Measuring distance between points

- Both estimating the density and constructing an EDS requires us to measure the distance between simulated points.
- Generally, variables in economic models have different measurement units and are correlated.
- This affects the distance between the simulated points and hence, affects the resulting EDS.
- Therefore, prior to using Algorithm \mathcal{A}^η and Algorithm P^ε , we normalize and orthogonalize the simulated data using *Principal Component* transformation.

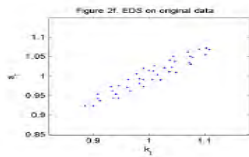
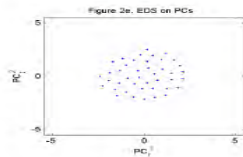
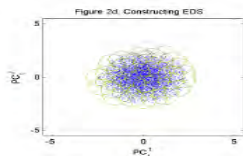
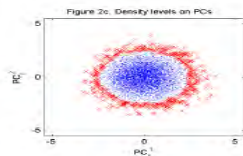
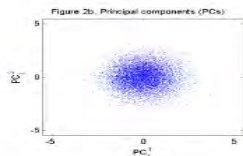
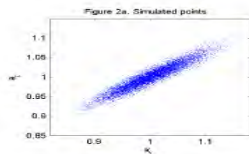
Principal component transformation

- Let $X \in \mathbb{R}^{n \times d}$ be simulated data normalized to zero mean and unit variance.
- Perform the singular value decomposition of X , i.e., $X = UQV^T$, where $U \in \mathbb{R}^{n \times d}$ and $V \in \mathbb{R}^{d \times d}$ are orthogonal matrices, and $Q \in \mathbb{R}^{d \times d}$ is a diagonal matrix.
- Perform a linear transformation of X using $PC \equiv XV$.
- $PC = (PC^1, \dots, PC^d) \in \mathbb{R}^{n \times d}$ are *principal components* (PCs) of X , and are orthogonal (uncorrelated), i.e., $(PC^{\ell'})^T PC^\ell = 0$ for any $\ell' \neq \ell$.
- Distance between two observations x_i and x_j is the Euclidean distance between their PCs

$$D(x_i, x_j) = \left[\sum_{\ell=1}^d (PC_i^\ell - PC_j^\ell)^2 \right]^{1/2},$$

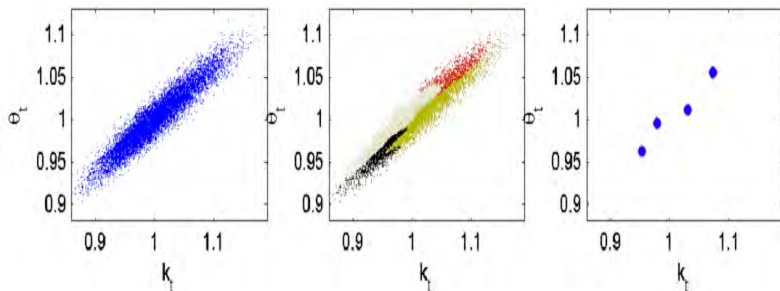
where PC^1, \dots, PC^d are normalized to unit variance.

Illustrating the EDS technique



Hierarchical clustering – another procedure for approximating the ergodic set

- Instead of constructing an EDS, we can use methods from cluster analysis to select a set of representative points from a given set of simulated points.
- We partition the simulated data into clusters (groups of closely-located points) and replace each cluster with one representative point.



Steps of the agglomerative hierarchical clustering algorithm

Clustering algorithm

(Algorithm P^c): Agglomerative hierarchical clustering algorithm.

Initialization Choose M , the number of clusters to be created.

In a zero-order partition $\mathcal{P}^{(0)}$, each simulated point represents a cluster.

Step 1 Compute all pairwise distances between the clusters in a partition $\mathcal{P}^{(i)}$.

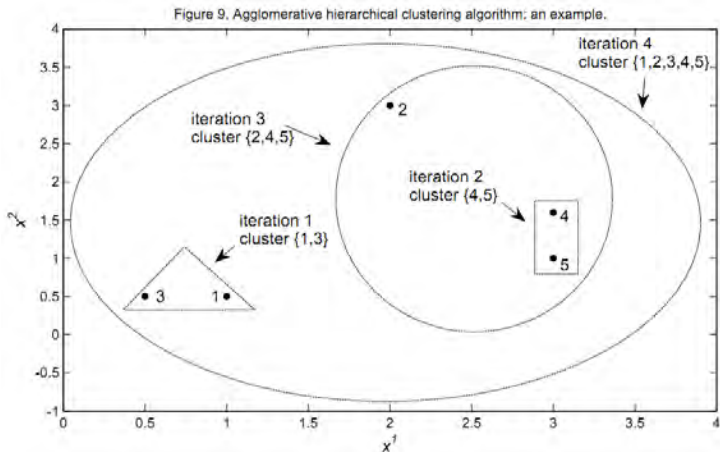
Step 2 Merge a pair of clusters with the smallest distance to obtain $\mathcal{P}^{(i+1)}$.

Iterate on Steps 1 and 2. Stop when the number of clusters in the partition is M .

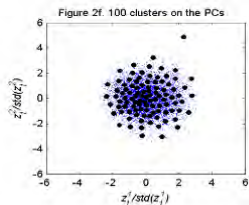
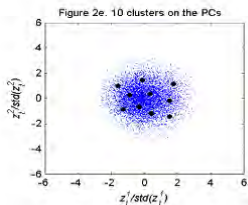
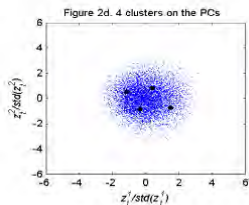
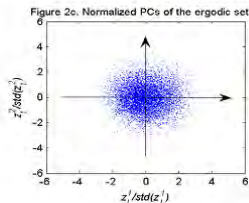
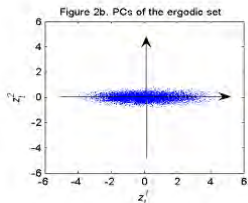
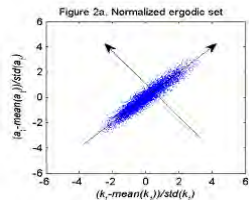
Represent each cluster with the closest to the cluster's center simulated point.

As a measure of distance between two clusters, we use Ward's measure of distance.

Agglomerative hierarchical clustering algorithm: an example



Clusters on principal components of the ergodic set



K-means clustering

Inputs:

- K (number of clusters)
- Training set $\{x^{(1)}, x^{(2)}, \dots, x^{(m)}\} \in \mathbb{R}^n$

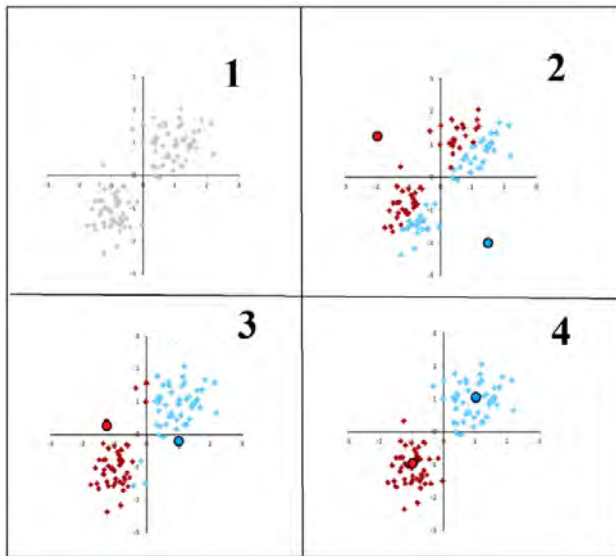
Randomly initialize K cluster centroids $\mu_1, \mu_2, \dots, \mu_K \in \mathbb{R}^n$

Repeat

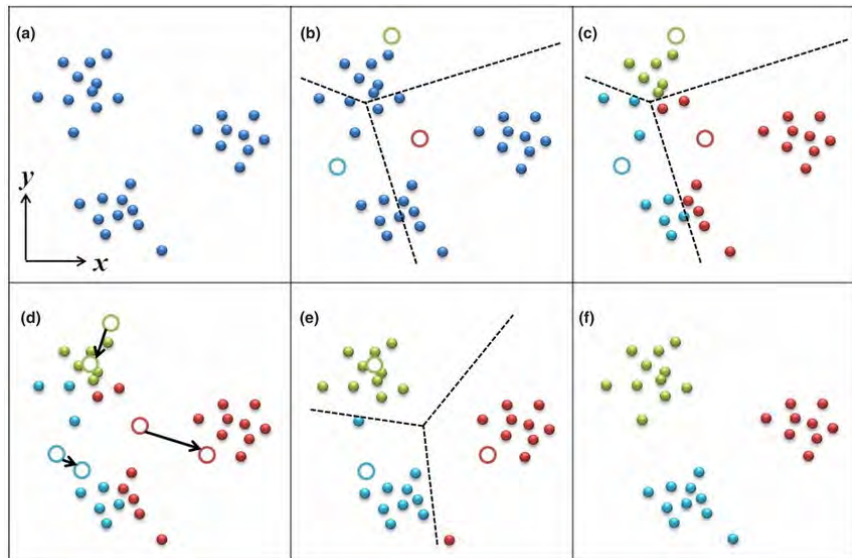
1. Assign each data point to the closest cluster (based on the cluster centroids).
2. Update the cluster centroids (by calculating the mean of all object in the cluster).

Iteration on steps 1 and 2 until convergence.

K-means algorithm (illustration)



K means for more than two clusters



Mathematical foundations of EDS clustering

We provide mathematical foundations for the EDS grid

- We establish computational complexity, dispersion, cardinality and degree of uniformity of the EDS grid constructed on simulated series.
- We perform the typical and the worst-case analysis for the discrepancy of the EDS grid.
- We relate our results to recent mathematical literature on
 - covering problems (e.g., measuring entropy); see, Temlyakov (2011).
 - random sequential packing problems, (e.g., germ contagion); see, Baryshnikov et al. (2008).

Dispersion of points in the EDS

Def. Let P be a set consisting of points $x_1, \dots, x_n \in X \subseteq \mathbb{R}^d$, and let (X, D) be a bounded metric space. The dispersion of P in X is given by

$$d_n(P; X) = \sup_{x \in X} \inf_{1 \leq i \leq n} D(x, x_i), \quad (1)$$

where D is a (Euclidean) metric on X .

Def. Let S be a sequence of elements on X , and let $x_1, \dots, x_n \in X \subseteq \mathbb{R}^d$ be the first n terms of S . The sequence S is called *low-dispersion* if $\lim_{n \rightarrow \infty} d_n(S; X) = 0$.

Proposition. Let P be any set of n points $x_1, \dots, x_n \in X \subseteq \mathbb{R}^d$ with a dispersion $d_n(P; X) < \varepsilon$. Let (X, D) be a bounded metric space, and let P^ε be an EDS $x_1^\varepsilon, \dots, x_M^\varepsilon$ constructed by Algorithm P^ε . Then, the dispersion of P^ε is bounded by $\varepsilon < d_M(P^\varepsilon; X) < 2\varepsilon$.

Number of points in the EDS

Proposition. Let P be any set of n points $x_1, \dots, x_n \in B(0, r) \subseteq \mathbb{R}^d$ with a dispersion $d_n(P; X) < \varepsilon$. Then, the number of points in P^ε constructed by Algorithm P^ε is bounded by $\left(\frac{r}{2\varepsilon}\right)^d \leq M \leq \left(1 + \frac{r}{\varepsilon}\right)^d$.

To construct an EDS with a given target number of points \overline{M} , we use a simple *bisection method*:

(Algorithm \overline{M}): Construction of an EDS with a target number of points \overline{M} .

For iteration $i = 1$, fix $\varepsilon_{\min}^{(1)}$ and $\varepsilon_{\max}^{(1)}$ such that $M(\varepsilon_{\max}^{(1)}) \leq \overline{M} \leq M(\varepsilon_{\min}^{(1)})$.

Step 1 On iteration i , take $\varepsilon = \frac{\varepsilon_{\min}^{(i)} + \varepsilon_{\max}^{(i)}}{2}$, construct an EDS and compute $M(\varepsilon)$.

Step 2 If $M(\varepsilon) > \overline{M}$, then set $\varepsilon_{\min}^{(i+1)} = \varepsilon$; and otherwise, set $\varepsilon_{\max}^{(i+1)} = \varepsilon$.

Iterate on Steps 1 and 2 until $M(\varepsilon)$ converges.

- To find the initial values of ε_{\min} and ε_{\max} , we use the bounds established in the above proposition.

Discrepancy

The degree of uniformity of EDSs. Standard notion of uniformity in the literature – discrepancy from the uniform distribution.

Def. Let P be a set consisting of points $x_1, \dots, x_n \in X \subseteq \mathbb{R}^d$, and let \mathcal{J} be a family of Lebesgue-measurable subsets of X . The discrepancy of P under \mathcal{J} is given by $\mathcal{D}_n(P; \mathcal{J}) = \sup_{J \in \mathcal{J}} \left| \frac{C(P; J)}{n} - \lambda(J) \right|$, where $C(P; J)$ counts the number of points from P in J , and $\lambda(J)$ is a Lebesgue measure of J .

Proposition. Let P be any set of n points $x_1, \dots, x_n \in B(0; 1) \subseteq \mathbb{R}^d$ with a dispersion $d_n(P; X) < \varepsilon$. Then, the discrepancy of an EDS constructed by Algorithm P^ε under B is bounded by

$$D_M(P^\varepsilon; \mathcal{B}) \leq \frac{\sqrt{2^d} \square 1}{\sqrt{2^d + 1}}.$$

Existence results for a covering-number problem

- Temlyakov (2011) studies the problem of finding a covering number – a minimum number of balls of radius ε which cover a given compact set (such as a d -dimensional hypercube or hypersphere).
- He shows that there exists an EDS P^ε on a unit hypercube $[0, 1]^d$ whose discrepancy converges to 0 as $M \rightarrow \infty$ (i.e., $\varepsilon \rightarrow 0$).
- However, constructing such an EDS is operationally difficult and costly.
- Also, Temlyakov (2011) selects points from a compact subset of \mathbb{R}^d , and his analysis cannot be directly applied to our problem of finding an ε -distinguishable subset of a given finite set of points.

Probabilistic results: random sequential packing problems

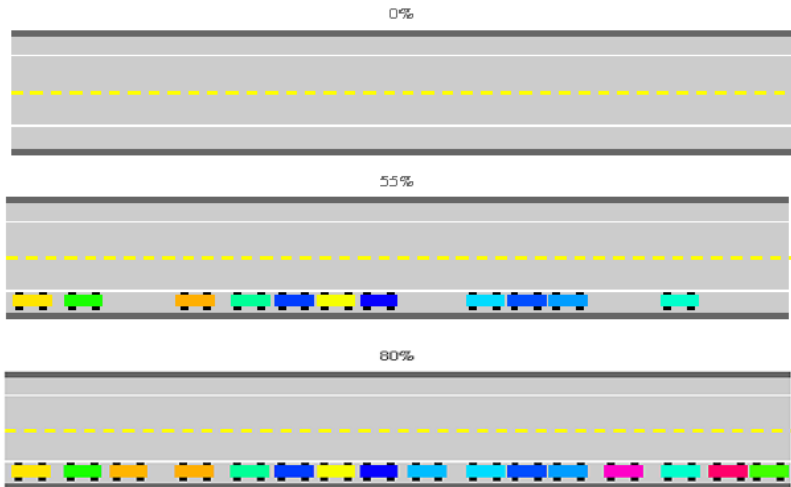
Probabilistic analysis of an EDS is non-trivial as points are spatially dependent: once we place a point in an EDS, it affects the placement of all subsequent points.

A random sequential packing problem:

- consider a bounded set $X \subseteq \mathbb{R}^d$ and a sequence of d -dimensional balls whose centers are i.i.d. random vectors $x_1, \dots, x_n \in X$ with a given density function g .
- A ball is packed if and only if it does not overlap with any ball which has already been packed. If not packed, the ball is discarded. At saturation, the centers of accepted balls constitute an EDS.

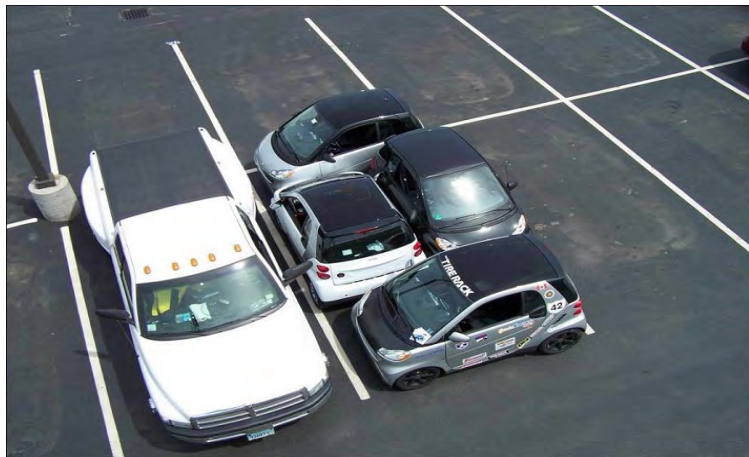
Probabilistic results: random sequential packing problems

Rényi's (1958) car parking model



Probabilistic results: random sequential packing problems

Rényi's (1958) car parking model



Probabilistic results: random sequential packing problems

Rényi's (1958) car parking model



Probabilistic results: random sequential packing problems

Rényi's (1958) car parking model



Probabilistic results for random sequential packing problems

- For a multidimensional case, Baryshnikov et al. (2008) show that the sequential packing measure, induced by the accepted balls centers, satisfies the LIL.
- Thus, the discrepancy of EDS converges to 0 asymptotically if the density of points in an EDS is uniform in the limit $\varepsilon \rightarrow 0$. However, the density of points in an EDS depends on the density function g of the stochastic process used to produce the data.
- Hence, an EDS needs not be uniform in the limit even in the probabilistic sense (unless the density function is uniform).

Our best- and worst-case scenarios

Implications of our analysis for Rényi's (1958) car parking model.

The best- and worst-case scenarios: cars occupy between 50% and 100% of the roadside ($\frac{1}{2} \leq \lim_{\varepsilon \rightarrow 0} M\varepsilon \leq 1$).

1. Distance $1/2$ between cars: evil drivers park their cars to leave as little parking space to other drivers as possible
2. Distance 0 between cars: a police officer directs the cars to park in a socially efficient way).

Our best- and worst-case scenarios

The worst-case scenario for discrepancy in Rényi's (1958) model,
 $\mathcal{D}_M^s(P^\varepsilon; \mathcal{B}) \leq \frac{\sqrt{2} \square 1}{\sqrt{2+1}} \approx 0.17$, which is obtained under $\lambda^* = \frac{\sqrt{2}}{\sqrt{2+1}}$.

1. To attain this bound, consider an EDS on $[0, 1]$ such that on the interval $[0, \lambda^*]$, all points are situated on a distance 2ε , and on $[\lambda^*, 1]$, all points are situated on the distance ε .
2. In the first interval, we have $\frac{\lambda^*}{2\varepsilon} \leq M \leq \frac{\lambda^*}{2\varepsilon} + 1$ points and in the second interval, we have $\frac{1 \square \lambda^*}{\varepsilon} \leq M \leq \frac{1 \square \lambda^*}{\varepsilon} + 1$ points.
3. On the first interval, the limiting discrepancy is

$\lim_{\varepsilon \rightarrow 0} \left[\lambda^* \square \frac{\frac{\lambda^*}{2\varepsilon}}{\frac{\lambda^*}{2\varepsilon} + \frac{1 \square \lambda^*}{\varepsilon}} \right] = \frac{\sqrt{2} \square 1}{\sqrt{2+1}} \approx 0.17$, the same value as implied by our propositions.

Comparison of EDS grid with other grids in the literature

Figure 3a. Tensor-product Chebyshev grid

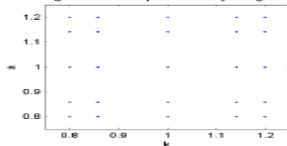


Figure 3b. Sobol sequence

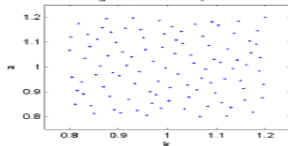


Figure 3c. Smolyak grid

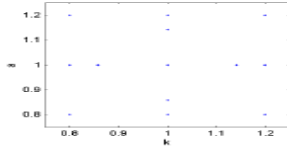


Figure 3d. Monomial grid

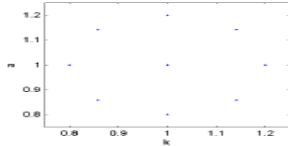


Figure 3e. Simulated points

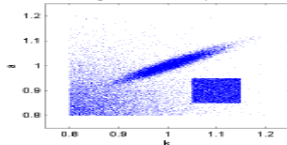
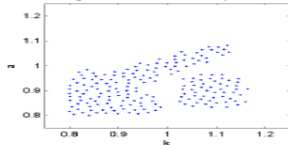


Figure 3f. EDS on simulated points



Description of the EDS algorithm iterating on Euler equation

Parameterize the RHS of the Euler equation by a polynomial $\widehat{K}(k, \theta; b)$,

$$E \left\{ \beta \frac{u'(c')}{u'(c)} [1 \square \delta + \theta' f'(k')] k' \right\} \\ \approx \widehat{K}(k, \theta; b) = b_0 + b_1 k + b_2 \theta + \dots + b_n \theta^L$$

Step 1. Simulate $\{k_t, \theta_t\}_{t=1}^{T+1}$. Construct an EDS grid, $\{k_m, \theta_m\}_{m=1}^M$.

Step 2. Fix $b \equiv (b_0, b_1, b_2, \dots, b_n)$. Given $\{k_m, \theta_m\}_{m=1}^M$ solve for $\{c_m\}_{m=1}^M$.

Step 3. Compute the expectation using numerical integration (quadrature integration or monomial rules)

$$\widehat{k}'_m \equiv E \left\{ \beta \frac{u'(c'_m)}{u'(c_m)} [1 \square \delta + \theta'_m f'(k'_m)] k'_m \right\}.$$

Regress \widehat{k}'_m on $(1, k_m, \theta_m, k_m^2, \theta_m^2, \dots, \theta_m^L) \implies$ get \widehat{b} .

Step 4. Solve for the coefficients using damping,

$$b^{(j+1)} = (1 \square \xi) b^{(j)} + \xi \widehat{b}, \quad \xi \in (0, 1).$$

Representative-agent model: parameters choice

Production function: $f(k_t) = k_t^\alpha$ with $\alpha = 0.36$.

Utility function: $u(c_t) = \frac{c_t^{1-\frac{1}{\sigma}}}{1-\frac{1}{\sigma}}$ with $\sigma \in \{\frac{1}{5}, 1, 5\}$.

Process for shocks: $\ln \theta_{t+1} = \rho \ln \theta_t + \epsilon_{t+1}$ with $\rho = 0.95$ and $\sigma = 0.01$.

Discount factor: $\beta = 0.99$.

Depreciation rate: $\delta = 0.025$.

Accuracy is measured by an Euler-equation residual,

$$\mathcal{R}(k_i, \theta_i) \equiv E_i \left[\beta \frac{c_{i+1}^{-\frac{1}{\sigma}}}{c_i^{-\frac{1}{\sigma}}} (1 - \delta + \alpha \theta_{i+1} k_{i+1}^{\alpha-1}) \right] - 1.$$

Table 1. Accuracy and speed of the Euler equation EDS algorithm in the representative-agent model

Polynomial degree	Mean error	Max error	CPU (sec)
1st degree	$\square 4.29$	$\square 3.31$	24.7
2nd degree	$\square 5.94$	$\square 4.87$	0.8
3rd degree	$\square 7.26$	$\square 6.04$	0.9
4th degree	$\square 8.65$	$\square 7.32$	0.9
5th degree	$\square 9.47$	$\square 8.24$	5.5

Target number of grid points is $\overline{M} = 25$.

Realized number of grid points is $M(\varepsilon) = 27$.

Mean and Max are unit-free Euler equation errors in log10 units, e.g.,

- $\square 4$ means $10^{\square 4} = 0.0001$ (0.01%);
- $\square 4.5$ means $10^{\square 4.5} = 0.0000316$ (0.00316%).

Benchmark parameters: $\beta = 1$, $\delta = 0.025$, $\rho = 0.95$, $\sigma = 0.01$.

In the paper, also consider $\gamma = 1/5$ (low risk aversion) and $\gamma = 5$ (high risk aversion). Accuracy and speed are similar.

Autocorrection of the EDS grid

Figure 4. Convergence of the EDS grid starting from capital series normalized to 10 steady state levels

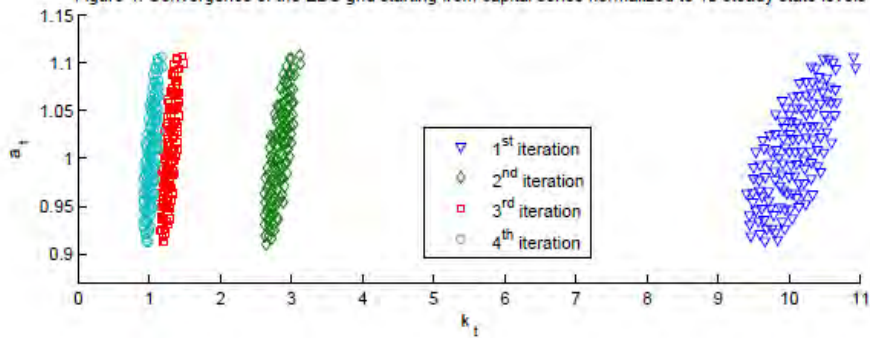


Table 2: Accuracy and speed in the one-agent model:
Smolyak grid versus EDS grid

	Test on a simulation				Test on a hypercube			
Polyn.	Smolyak grid		EDS grid		Smolyak grid		EDS grid	
deg.	Mean	Max	Mean	Max	Mean	Max	Mean	Max
1st	-3.31	-2.94	-4.23	-3.31	-3.25	-2.54	-3.26	-2.38
2nd	-4.74	-4.17	-5.89	-4.87	-4.32	-3.80	-4.41	-3.25
3rd	-5.27	-5.13	-7.19	-6.16	-5.39	-4.78	-5.44	-4.11

Description of the EDS algorithm iterating on Bellman equation

Parameterize the value function by a polynomial $V(\cdot) \approx \widehat{V}(\cdot; b)$:

$$\begin{aligned} & \max_{k', c} \left\{ u(c) + \beta E \left[\widehat{V} \left(k', \theta'; b \right) \right] \right\} \\ & \approx \widehat{V}(k, \theta; b) = b_0 + b_1 k + b_2 \theta + \dots + b_n \theta^L. \end{aligned}$$

Step 1. Find \widehat{K} corresponding to $\widehat{V}(\cdot; b)$. Simulate $\{k_t, \theta_t\}_{t=1}^{T+1}$.

Construct an EDS grid, $\{k_m, \theta_m\}_{m=1}^M$.

Step 2. Fix $b \equiv (b_0, b_1, b_2, \dots, b_n)$. Given $\{k_m, \theta_m\}_{m=1}^M$ solve for $\{c_m\}_{m=1}^M$.

Step 3. Compute the expectation using numerical integration (quadrature integration or monomial rules)

$$V_m \equiv u(c_m) + \beta E \widehat{V} \left(k'_m, \theta'_m; b \right).$$

Regress V_m on $\left(1, k_m, \theta_m, k_m^2, \theta_m^2, \dots, \theta_m^L \right) \implies \widehat{b}$.

Step 4. Solve for the coefficients using damping,

$$b^{(j+1)} = (1 - \xi) b^{(j)} + \xi \widehat{b}, \quad \xi \in (0, 1).$$

Table 3. Accuracy and speed of the Bellman equation EDS algorithm in the representative-agent model

Polynomial degree	Mean error	Max error	CPU (sec)
1st degree	□	□	□
2nd degree	□3.98	□3.11	0.5
3rd degree	□5.15	□4.17	0.4
4th degree	□6.26	□5.12	0.4
5th degree	□7.42	□5.93	0.4

Target number of grid points is $\overline{M} = 25$.

Realized number of grid points is $M(\varepsilon) = 27$.

Multi-country model

The planner maximizes a weighted sum of N countries' utility functions:

$$\max_{\{c_t^h, k_{t+1}^h\}_{h=1}^N}_{t=0}^{\infty} E_0 \left(\sum_{h=1}^N v^h \sum_{t=0}^{\infty} \beta^t u^h(c_t^h) \right)$$

subject to

$$\sum_{h=1}^N c_t^h + \sum_{h=1}^N k_{t+1}^h = \sum_{h=1}^N k_t^h (1 - \delta) + \sum_{h=1}^N \theta_t^h f^h(k_t^h),$$

where v^h is country h 's welfare weight.

Productivity of country h follows the process

$$\ln \theta_{t+1}^h = \rho \ln \theta_t^h + \epsilon_{t+1}^h,$$

where $\epsilon_{t+1}^h \equiv \varsigma_{t+1} + \varsigma_{t+1}^h$ with $\varsigma_{t+1} \sim \mathcal{N}(0, \sigma^2)$ is identical for all countries and $\varsigma_{t+1}^h \sim \mathcal{N}(0, \sigma^2)$ is country-specific.

Table 3. Accuracy and speed in the multi-country model

	Polyn. degree	M1			Q(1)		
		Mean	Max	CPU	Mean	Max	CPU
N=2	1st	□4.09	□3.19	44 sec	□4.07	□3.19	45 sec
	2nd	□5.45	□4.51	2 min	□5.06	□4.41	1 min
	3rd	□6.51	□5.29	4 min	□5.17	□4.92	2 min
N=20	1st	□4.21	□3.29	20 min	□4.17	□3.28	3 min
	2nd	□5.08	□4.17	5 hours	□4.83	□4.10	32 min
N=40	1st	□4.23	□3.31	5 hours	□4.19	□3.29	2 hours
	2nd	□	□	-	□4.86	□4.48	24 hours
N=100	1st	□4.09	□3.24	10 hours	□4.06	□3.23	36 min
N=200	1st	□	□	-	□3.97	□3.20	2 hours

M1 means monomial integration with $2N$ nodes; Q(1) means quadrature integration with one node in each dimension; Mean and Max are mean and maximum unit-free Euler equation errors in log10 units, respectively; CPU is running time.

A new Keynesian (NK) model

A stylized new Keynesian model with Calvo-type price frictions and a Taylor (1993) rule with the ZLB

- *Literature that estimates the models:*
 - Christiano, Eichenbaum and Evans (2005), Smets and Wouters (2003, 2007), Del Negro, Schorfheide, Smets and Wouters (2007).
- *Literature that finds numerical solutions:* mostly relies on local (perturbation) solution methods. Few papers apply global solution methods to low-dimensional problems.
- *Perturbation:*
 - most use linear approximations (Christiano, Eichenbaum&Rebelo, 2009);
 - some use quadratic approx. (Kollmann, 2002, Schmitt-Grohé&Uribe, 2007);
 - very few use cubic approximations (Rudebusch and Swanson, 2008).
- *Global solution methods:*
 - Adjemian and Juillard (2011): extended path method of Fair&Taylor;
 - Aruoba and Schorfheide (2012): cluster grid algorithm;
 - Fernández-Villaverde, Gordon, Guerrón-Quintana, Rubio-Ramírez (2012).

A new Keynesian (NK) model

Assumptions:

- *Households* choose consumption and labor.
- Perfectly competitive *final-good firms* produce goods using intermediate goods.
- Monopolistic *intermediate-good firms* produce goods using labor and are subject to sticky price (à la Calvo, 1983).
- *Monetary authority* obeys a Taylor rule with zero lower bound (ZLB).
- *Government* finances a stochastic stream of public consumption by levying lump-sum taxes and by issuing nominal debt.
- *6 exogenous shocks and 8 state variables* \implies The model is large scale (it is expensive to solve or even intractable under conventional global solution methods that rely on product rules).

The representative household

The utility-maximization problem:

$$\max_{\{C_t, L_t, B_t\}_{t=0, \dots, \infty}} E_0 \sum_{t=0}^{\infty} \beta^t \exp(-\eta_{u,t}) \left[\frac{C_t^{1-\vartheta}}{1-\vartheta} \exp(-\eta_{L,t}) \frac{L_t^{1+\vartheta}}{1+\vartheta} \right]$$
$$\text{s.t. } P_t C_t + \frac{B_t}{\exp(\eta_{B,t}) R_t} + T_t = B_{t-1} + W_t L_t + \Pi_t$$

where $(B_0, \eta_{u,0}, \eta_{L,0}, \eta_{B,0})$ is given.

- C_t , L_t , and B_t = consumption, labor and nominal bond holdings, resp.;
- P_t , W_t and R_t = the commodity price, nominal wage and (gross) nominal interest rate, respectively;
- T_t = lump-sum taxes;
- Π_t = the profit of intermediate-good firms;
- β = discount factor; $\beta > 0$ and $\vartheta > 0$.

The representative household

Stochastic processes for shocks

- $\eta_{u,t}$ and $\eta_{L,t}$ = exogenous preference shocks;
- $\eta_{B,t}$ = exogenous premium in the return to bonds;

$$\begin{aligned}\eta_{u,t+1} &= \rho_u \eta_{u,t} + \epsilon_{u,t+1}, & \epsilon_{u,t+1} &\sim \mathcal{N} \left(\overset{\square}{0}, \sigma_u^2 \right) \\ \eta_{L,t+1} &= \rho_L \eta_{L,t} + \epsilon_{L,t+1}, & \epsilon_{L,t+1} &\sim \mathcal{N} \left(\overset{\square}{0}, \sigma_L^2 \right) \\ \eta_{B,t+1} &= \rho_B \eta_{B,t} + \epsilon_{B,t+1}, & \epsilon_{B,t+1} &\sim \mathcal{N} \left(\overset{\square}{0}, \sigma_B^2 \right)\end{aligned}$$

Final-good producers

The profit-maximization problem:

- Perfectly competitive producers
- Use intermediate goods $i \in [0, 1]$ as inputs

$$\begin{aligned} \max_{Y_t(i)} \quad & P_t Y_t \propto \int_0^1 P_t(i) Y_t(i) di \\ \text{s.t.} \quad & Y_t = \left(\int_0^1 Y_t(i)^{\frac{\epsilon-1}{\epsilon}} di \right)^{\frac{\epsilon}{\epsilon-1}}, \quad \epsilon \geq 1 \end{aligned} \quad (2)$$

- $Y_t(i)$ and $P_t(i)$ = quantity and price of an intermediate good i , resp.;
- Y_t and P_t = quantity and price of the final good, resp.;
- Eq (2) = production function (Dixit-Stiglitz aggregator function).

Result 1: Demand for the intermediate good i : $Y_t(i) = Y_t \left(\frac{P_t(i)}{P_t} \right)^{-\epsilon}$.

Result 2: Aggregate price index $P_t = \left(\int_0^1 P_t(i)^{1-\epsilon} di \right)^{\frac{1}{1-\epsilon}}$.

Intermediate-good producers

The cost-minimization problem:

- Monopolistically competitive
- Use labor as an input
- Are hit by a productivity shock
- Are subject to sticky prices

$$\begin{aligned} \min_{L_t(i)} \quad & \text{TC}(Y_t(i)) = (1 - v) W_t L_t(i) \\ \text{s.t.} \quad & Y_t(i) = \exp(\eta_{z,t}) L_t(i) \\ & \eta_{z,t+1} = \rho_z \eta_{z,t} + \epsilon_{z,t+1}, \quad \epsilon_{z,t+1} \sim \mathcal{N}(0, \sigma_z^2) \end{aligned}$$

- TC = nominal total cost (net of government subsidy v);
- $L_t(i)$ = labor input;
- $\exp(\eta_{z,t})$ is the productivity level.

Intermediate-good producers (price decisions)

Calvo-type price setting:

1 \square θ of the firms sets prices optimally, $P_t(i) = \tilde{P}_t$, for $i \in [0, 1]$;
 θ is not allowed to change the price, $P_t(i) = P_{t-1}(i)$, for $i \in [0, 1]$.

The profit-maximization problem of a reoptimizing firm i :

$$\begin{aligned} \max_{\tilde{P}_t} \quad & \sum_{j=0}^{\infty} \beta^j \theta^j E_t \left\{ \Lambda_{t+j} \left[\tilde{P}_t Y_{t+j}(i) - P_{t+j} mc_{t+j} Y_{t+j}(i) \right] \right\} \\ \text{s.t.} \quad & Y_t(i) = Y_t \left(\frac{P_t(i)}{P_t} \right)^{-\varepsilon} \end{aligned} \quad (3)$$

- Eq (3) is the demand for an intermediate good i ;
- Λ_{t+j} is the Lagrange multiplier on the household's budget constraint;
- mc_{t+j} is the real marginal cost of output at time $t + j$.

Government

The government budget constraint:

$$T_t + \frac{B_t}{\exp(\eta_{B,t}) R_t} = P_t \frac{\bar{G}Y_t}{\exp(\eta_{G,t})} + B_{t+1} + vW_tL_t$$

- $\frac{\bar{G}Y_t}{\exp(\eta_{G,t})} = G_t$ is government spending;
- vW_tL_t is the subsidy to the intermediate-good firms;
- $\eta_{G,t}$ is a government-spending shock,

$$\eta_{G,t+1} = \rho_G \eta_{G,t} + \epsilon_{G,t+1}, \quad \epsilon_{G,t+1} \sim \mathcal{N}(0, \sigma_G^2)$$

Monetary authority

Taylor rule with ZLB on the net nominal interest rate:

$$R_t = \max \left\{ 1, R_* \left(\frac{R_{t-1}}{R_*} \right)^\mu \left[\left(\frac{\pi_t}{\pi_*} \right)^{\phi_\pi} \left(\frac{Y_t}{Y_{N,t}} \right)^{\phi_y} \right]^{1-\mu} \exp(\eta_{R,t}) \right\}$$

- R_* = long-run gross nominal interest rate;
- π_t = gross inflation rate between $t-1$ and t ;
- π_* = inflation target;
- $Y_{N,t}$ = natural level of output;
- $\eta_{R,t}$ = monetary shock

$$\eta_{R,t+1} = \rho_R \eta_{R,t} + \epsilon_{R,t+1}, \quad \epsilon_{R,t+1} \sim \mathcal{N}(0, \sigma_R^2)$$

Natural equilibrium

"Natural equilibrium" - the model in which the potential inefficiencies have been eliminated:

- Natural level of output $Y_{N,t}$ in the Taylor rule is a solution to a **planner's problem**

$$\begin{aligned} \max_{\{C_t, L_t\}_{t=0, \dots, \infty}} E_0 \sum_{t=0}^{\infty} \beta^t \exp(\eta_{u,t}) & \left[\frac{C_t^{1-\vartheta}}{1-\vartheta} \exp(\eta_{L,t}) \frac{L_t^{1+\vartheta}}{1+\vartheta} \right] \\ \text{s.t. } C_t = \exp(\eta_{z,t}) L_t & G_t \end{aligned}$$

where G_t is given.

- This implies

$$Y_{N,t} = \left[\frac{\exp(\eta_{z,t})^{1+\vartheta}}{[\exp(\eta_{G,t})] \exp(\eta_{L,t})} \right]^{\frac{1}{\vartheta+1}}$$

Summary of equilibrium conditions

- **Aggregate production**

$$Y_t = \exp(\eta_{z,t}) L_t \Delta_t$$

- **Aggregate resource constraint**

$$C_t + G_t = Y_t$$

- **Taylor rule with ZLB on the net nominal interest rate**

$$R_t = \max \left\{ 1, R_* \left(\frac{R_{t-1}}{R_*} \right)^\mu \left[\left(\frac{\pi_t}{\pi_*} \right)^{\phi_\pi} \left(\frac{Y_t}{Y_{N,t}} \right)^{\phi_y} \right]^{1-\mu} \exp(\eta_{R,t}) \right\}$$

- **Natural level of output**

$$Y_{N,t} = \left[\frac{\exp(\eta_{z,t})^{1+\vartheta}}{[\exp(\eta_{G,t})] \exp(\eta_{L,t})} \right]^{\frac{1}{\vartheta+1}}.$$

Summary of equilibrium conditions

We have

- Stochastic processes for 6 exogenous shocks $\{\eta_{u,t}, \eta_{L,t}, \eta_{B,t}, \eta_{z,t}, \eta_{G,t}, \eta_{R,t}\}$.
- 8 endogenous equilibrium equations & 8 unknowns $\{C_t, Y_t, R_t, L_t, \Delta_t, \pi_t, F_t, S_t\}$.
- 2 endogenous state variables $\{\Delta_{t-1}, R_{t-1}\}$.
- Thus, there are 8 (endogenous plus exogenous) state variables.

Computational papers on ZLB

How to impose the ZLB on interest rate?

- Perturbation methods do not allow us to impose the ZLB in the solution procedure.
- The conventional approach in the literature is to disregard the ZLB when computing perturbation solutions and to impose the ZLB in simulations when running accuracy checks (that is, whenever R_t happens to be smaller than 1 in simulation, we set it at 1).
 - Christiano, Eichenbaum&Rebelo (2009)
- **In contrast**, our global EDS method does allow to impose the ZLB both in the solution and simulation procedures.

Parameter values

We calibrate the model using the results in Smets and Wouters (2003, 2007), and Del Negro, Smets and Wouters (2007).

- Preferences: $\eta = 1$; $\vartheta = 2.09$; $\beta = 0.99$
- Intermediate-good production: $\varepsilon = 4.45$
- Fraction of firms that cannot change price: $\theta = 0.83$
- Taylor rule: $\phi_y = 0.07$; $\phi_\pi = 2.21$; $\mu = 0.82$
- Inflation target: $\pi_* \in \{1, 1.0598\}$
- Government to output ratio: $\bar{G} = 0.23$
- Stochastic processes for shocks:
 $\rho_u = 0.92$; $\rho_L = 0.25$; $\rho_B = 0.22$; $\rho_z = 0.95$; $\rho_R = 0.15$; $\rho_G = 0.95$
 $\sigma_u = 0.54\%$; $\sigma_L \in \{18.21\%, 40.54\%\}$; $\sigma_B = 0.23\%$; $\sigma_z = 0.45\%$;
 $\sigma_R = 0.28\%$; $\sigma_G = 0.38\%$

We compute 1st and 2nd perturbation solutions using Dynare, and we compute 2nd and 3rd degree EDS solutions.

Time-series solution and EDS grid

Figure 5. Simulated points and the grid for a new Keynesian model: ZLB is imposed

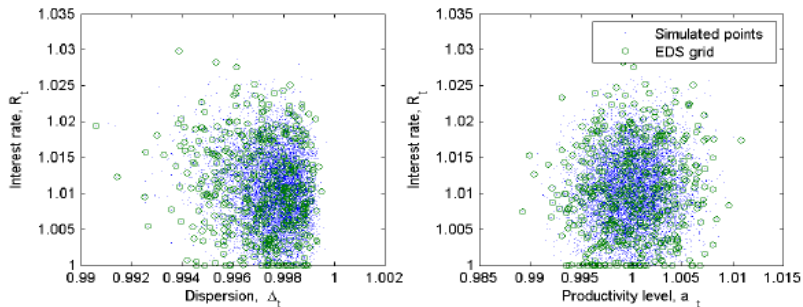


Table 4. Accuracy and speed in the NK model with 0% inflation target and 18.21% volatility of labor shock

	PER1	PER2	EDS2	EDS3
CPU	0.15		24.3	4.4
Mean	□3.03	□3.77	□3.99	□4.86
Max	□1.21	□1.64	□2.02	□2.73
R_{min}	0.9916	0.9929	0.9931	0.9927
R_{max}	1.0340	1.0364	1.0356	1.0358
$Fr_{(R \leq 1)}, \%$	2.07	1.43	1.69	1.68
$\Delta R, \%$	0.17	0.09	0.05	0
$\Delta C, \%$	1.00	0.19	0.12	0
$\Delta Y, \%$	1.00	0.19	0.12	0
$\Delta L, \%$	0.65	0.33	0.16	0
$\Delta \pi, \%$	0.30	0.16	0.11	0

PER 1 and PER 2 = 1st and 2nd order Dynare solutions; EDS2 and EDS3 = 2nd and 3rd degree EDS algorithm; Mean and Max = average and max absolute errors (in log10 units); R_{min} and R_{max} = min and max R; Fr = frequency of $R \leq 1$; ΔX = max difference from EDS3.

Table 5. Accuracy and speed in the NK model with 5.98% inflation target and 40.54% volatility of labor shock

	PER1	PER2	EDS2	EDS3
CPU	0.15		22.1	12.0
Mean	□2.52	□2.90	□3.43	□4.00
Max	□0.59	□0.42	□1.31	□1.91
R_{min}	1.0014	1.0065	1.0060	1.0060
R_{max}	1.0615	1.0694	1.0653	1.0660
$Fr(R \leq 1), \%$	0	0	0	0
$\Delta R, \%$	0.63	0.39	0.25	0
$\Delta C, \%$	6.57	1.49	0.72	0
$\Delta Y, \%$	6.57	1.48	0.72	0
$\Delta L, \%$	3.16	1.30	0.54	0
$\Delta \pi, \%$	1.05	0.79	0.60	0

PER 1 and PER 2 = 1st and 2nd order Dynare solutions; EDS2 and EDS3 = 2nd and 3rd degree EDS; Mean and Max = average and max absolute errors (in log10 units); R_{min} and R_{max} = min and max R; Fr = frequency of $R \leq 1$; ΔX = max difference from EDS3.

Table 6. Accuracy and speed in the NK model with 0% inflation target, 18.21% volatility of labor shock and ZLB

	PER1	PER2	EDS2	EDS3
CPU	0.15		21.4	3.58
Mean	□3.02	□3.72	□3.57	□3.65
Max	□1.21	□1.34	□1.58	□1.81
R_{min}	1.0000	1.0000	1.0000	1.0000
R_{max}	1.0340	1.0364	1.0348	1.0374
$Fr(R_{\leq 1}), \%$	1.76	1.19	2.46	2.23
$\Delta R, \%$	0.33	0.34	0.34	0
$\Delta C, \%$	4.31	3.65	2.26	0
$\Delta Y, \%$	4.33	3.65	2.26	0
$\Delta L, \%$	3.37	3.17	2.45	0
$\Delta \pi, \%$	1.17	1.39	0.79	0

PER 1 and PER 2 = 1st and 2nd order Dynare solutions; EDS2 and EDS3 = 2nd and 3rd degree EDS; Mean and Max = average and max absolute errors (in log10 units); R_{min} and R_{max} = min and max R; Fr = frequency of $R_{\leq 1}$; ΔX = max difference from EDS3.

Simulated series: ZLB is not imposed versus ZLB is imposed

Figure 6a. A time-series solution to a new Keynesian model: ZLB is not imposed

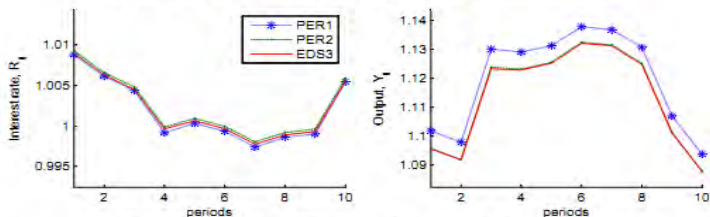
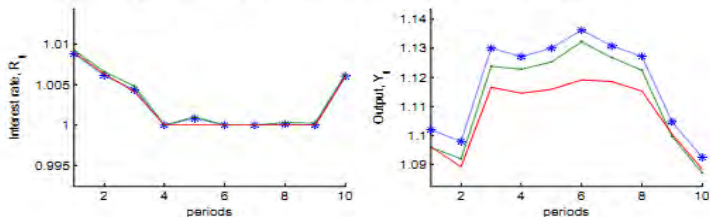


Figure 6b. A time-series solution to a new Keynesian model: ZLB is imposed



Main lessons:

A mix of machine learning techniques taken together allows us to address the challenges of high-dimensional problems:

- Clustering analysis for constructing domain - a tiny fraction of the standard hypercube domain;
- monomial and one-node integration rules;
- fixed-point iteration for finding policy functions;
- iteration-on-allocation and precomputation approaches for solving for intratemporal choice.

A proper coordination of the above techniques is crucial for accuracy and speed.

Machine learning, artificial intelligence and deep learning methods for dynamic economic models:

Reinforcement learning

Lilia Maliar, Serguei Maliar

November 2022

Minicourse

Reinforcement learning

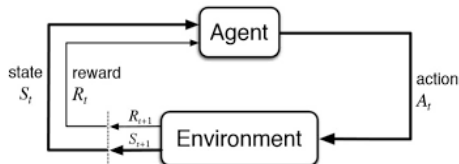
- **Reinforcement learning** (RL) is an area of machine learning concerned with how software agents ought to take actions in an environment in order to maximize some notion of cumulative reward.
- Reinforcement learning is one of three basic machine learning paradigms, alongside supervised learning and unsupervised learning.
- It differs from supervised learning in that it needs not label input/output pairs and correct sub-optimal actions explicitly. Instead the focus is on finding a balance between exploration (of uncharted territory) and exploitation (of current knowledge).
- The environment is typically stated in the form of a Markov decision process (MDP), because many reinforcement learning algorithms for this context utilize dynamic programming techniques.
- The main difference between the classical dynamic programming methods and reinforcement learning algorithms is that the latter do not assume knowledge of an exact mathematical model of the MDP and they target large MDPs where exact methods become infeasible.

Defining reinforcement learning problem

- Markov property: Current state completely characterized the state of the world
- Define by tuple of objects (S, A, R, P, β)
 - S : set of possible states (*capital, productivity*)
 - A : set of possible actions (*consumption choices*)
 - R : distribution of reward given (state, action) pair (*utility level*)
 - P : transition probability, i.e., distribution over next state given (state, action) pair (*next period capital and shock*)
 - β : discount factor

Markov Decision Process

- Markov decision process is mathematical formulation of RL problem



- At time $t = 0$, environment samples initial state $s_0 \sim p(s_0)$
- Then, for $t=0, T$
 - Agent selects action a_t
 - Environment samples reward $r_t \sim R(\cdot | s_t, a_t)$
 - Environment samples next state $s_{t+1} \sim P(\cdot | s_t, a_t)$
 - Agent receives reward r_t and next state s_{t+1}
- Recall Google DeepMind learns to play Atari.
<https://www.youtube.com/watch?v=V1eYniJ0Rnk>

Policy function

- A policy π is a function from S to A that specify what action to take in each state
- Objective find policy π^* that maximizes cumulative discounted reward $\sum_{t>0} \beta^t r_t$.
- Formally,

$$\pi^* = \arg \max_{\pi} E \left[\sum_{t>0} \beta^t r_t \mid \pi \right],$$

where $s_0 \sim p(s_0)$, $a_t \sim \pi(\cdot \mid s_t)$, $s_{t+1} \sim p(\cdot \mid s_t)$.

- Following a policy produces sample trajectories (or paths) $s_0, a_0, r_0, s_1, a_1, r_1, \dots$

Value function and Q-learning

- How good is a state?
- The value function at state s is the expected cumulative reward from following the policy from state s :

$$V^\pi(s) = E \left[\sum_{t>0} \beta^t r_t \mid s_0 = s, \pi \right]$$

- How good is a state-action pair?
- The Q-value function at state s and action a is the expected cumulative reward from taking action a in state s and then following the policy

$$Q^\pi(s, a) = E \left[\sum_{t>0} \beta^t r_t \mid s_0 = s, a_0 = a, \pi \right]$$

Bellman equation

- The optimal Q -value function Q^* is the maximum expected cumulative reward achievable from a given (state, action) pair:

$$Q^*(s, a) = \max_{\pi} E \left[\sum_{t>0} \beta^t r_t \mid s_0 = s, a_0 = a, \pi \right]$$

- Q^* satisfies the Bellman equation

$$Q^*(s, a) = E_{s' \sim \mathcal{E}} \left[r + \beta \max_{a'} [Q^*(s', a') \mid s, a] \right]$$

- Intuition: if the optimal state-action values for the next time step $Q^*(s', a')$ are known, then the optimal strategy is to take the action that maximizes the expected value of $r + \beta \max_{a'} Q^*(s', a')$.
- The optimal policy π^* corresponds to taking the best action in any state as specified by Q^* .

Solving for the optimal policy

- Value iteration algorithm: Use Bellman equation as an iterative update

$$Q_{i+1}(s, a) = E_{s' \sim \mathcal{E}} \left[r + \beta \max_{a'} [Q_i(s', a') \mid s, a] \right]$$

Q_i will converge to Q^* as $i \rightarrow$ infinity.

- What is the problem with this?
- Not scalable. Must compute $Q(s, a)$ for every state-action pair. If state is current game state pixels, computationally infeasible to compute for entire state space!

Solving for optimal policy: Q-learning

- Q-learning. Use a function approximator to estimate the action-value function

$$Q(s, a; \theta) \approx Q^*(s, a)$$

where θ function parameters, weights.

- If the function approximator is a deep neural network \Rightarrow deep q-learning!
- Remember: want to find a Q -function that satisfies the Bellman equation.

Forward and backward paths

- Forward pass:

$$\text{Loss function: } L_i(\theta_i) = E_{s,a \sim \rho(\cdot)} \left[(y_i - Q(s, a; \theta_i))^2 \right]$$

$$\text{where } y_i = E_{s' \sim \mathcal{E}} \left[r + \beta \max_{a'} [Q(s', a'; \theta_{i-1}) \mid s, a] \right]$$

- Backward pass:

Gradient update (with respect to Q-function parameters θ):

$$\nabla_{\theta_i} L_i(\theta_i) = E_{s,a \sim \rho(\cdot), s' \sim \mathcal{E}} \left[\left[r + \beta \max_{a'} Q(s', a'; \theta_{i-1}) - Q(s', a'; \theta_i) \right] \nabla_{\theta_i} Q(s', a'; \theta_i) \right]$$

- Iteratively try to make the Q-value close to the target value y_i it should have, if Q-function corresponds to optimal Q^* (and optimal policy π^*).

Training the Q-network: experience replay

- Learning from batches of consecutive samples is problematic
 - samples are correlated \Rightarrow inefficient learning
 - current Q network parameters determine next training samples (e.g., if maximizing action is to move left, training samples will be dominated by samples from left-handed size) \Rightarrow can lead to bad feedback loops
- Addresses these problems using experience replay
 - Continually update a replay memory table of transitions (s_t, a_t, r_t, s_{t+1}) as game (experience) episodes are played.
 - Train Q-network on random minibatches of transitions from the replay memory instead of consecutive samples.
- Each transition can also contribute to multiple weight updates \Rightarrow greater data efficiency.

Algorithm 1: Deep Q-learning with Experience Replay

Initialize replay memory D and function Q with random weights

for $episodes = 1, M$ do

 Initialize sequence s_1

 for $t = 1, T$ do

 With probability ϵ select a random action a_t

 Otherwise, select $a_t = \max_a Q^*(s_t, a; \theta)$

 Execute action a_t in emulator and observe reward r_t

 Set s_{t+1}

 Store transition (s_t, a_t, r_t, s_{t+1}) in D

 Sample random minibatch of transition (s_j, a_j, r_j, s_{j+1}) from D

 Set

$$y_j = \begin{cases} r_j & \text{for terminal } s_{j+1} \\ r_j + \beta \max_{a'} Q^*(s_{j+1}, a'; \theta) & \text{for non-terminal } s_{j+1} \end{cases}$$

 Perform a gradient descent step on $y_j - Q(\phi_j, a_j; \theta)$

 end for

end for

The problem with Q-learning

- What is the problem with Q-learning?
- The function Q can be very complicated
- *Examples. A robot grasping an object has a very high dimensional state \Rightarrow hard to learn exact value of every (state, action) pair.*
- But the policy can be much simpler: just close your hand.
- Can we learn a policy directly, e.g., finding the best policy from a collection of policies? (without costly estimation of Q values).

Policy gradient (REINFORCE)

- Formally, let's define a class of parameterized policies:
 $\Pi = \{\pi_\theta, \theta \in R^m\}$.
- For each policy, define its value $J(\theta) = E[\sum_{t>0} \beta^t r_t \mid \pi_\theta]$
- We want to find the optimal policy $\theta^* = \arg \max_{\pi} J(\theta)$
- Reinforce algorithm $J(\theta) = E_{\tau \sim p(\tau, \theta)} [r(\tau)] = \int_{\tau} r(\tau) p(\tau, \theta) d\tau$
where $r(\tau)$ is the reward of a trajectory $\tau = (s_0, a_0, r_0, s_1, a_1, r_1, \dots)$
- Let's differentiate this:
- $\nabla_{\theta} J(\theta) = \int_{\tau} r(\tau) \nabla_{\theta} p(\tau, \theta) d\tau$
- Gradient of an expectation is problematic when p depends on θ .

Dealing with expectation function

- We can use a nice trick

$$\nabla_{\theta} p(\tau, \theta) = p(\tau, \theta) \frac{\nabla_{\theta} p(\tau, \theta)}{p(\tau, \theta)} = p(\tau, \theta) \nabla_{\theta} \log p(\tau, \theta)$$

$$\nabla_{\theta} J(\theta) = \int_{\tau} r(\tau) p(\tau, \theta) \nabla_{\theta} \log p(\tau, \theta) d\tau =$$
$$E_{\tau \sim p(\tau, \theta)} [r(\tau) \nabla_{\theta} \log p(\tau, \theta)]$$

- Can estimate with Monte Carlo sampling.
- Can we compute these quantities without knowing these transition probabilities?

We have $p(\tau, \theta) = \prod_{t \geq 0} p(s_{t+1} | s_t, a_t) \pi_{\theta}(a_t | s_t)$

$$\log p(\tau, \theta) = \sum_{t \geq 0} \log p(s_{t+1} | s_t, a_t) + \log \pi_{\theta}(a_t | s_t)$$

$\nabla_{\theta} \log p(\tau, \theta) = \sum_{t > 0} \nabla_{\theta} \log \pi_{\theta}(a_t | s_t)$ does not depend on transition probabilities.

- Therefore, when sampling a trajectory we can estimate $\nabla_{\theta} J(\theta)$ with

$$\nabla_{\theta} J(\theta) \approx \sum_{t > 0} r(\tau) \nabla_{\theta} \log \pi_{\theta}(a_t | s_t)$$

Intuition

$$\nabla_{\theta} J(\theta) \approx \sum_{t>0} r(\tau) \nabla_{\theta} \log \pi_{\theta}(a_t | s_t)$$

- if $r(\tau)$ is high, push up the probabilities of actions seen
- if $r(\tau)$ is low, push down the probabilities of actions seen
- Might seem simplistic to say that if a trajectory is good then all its actions were good but in expectation it averages out!
- However, this suffers from high variance because credit assignment is really hard. can we help the estimator?

Variance reduction: baseline

- Problem: the raw trajectory is not necessarily meaningful. For example, if rewards are all positive, you keep pushing up probability of actions.
- What is important then? Whether a reward is better or worse than what you expect to get.
- Idea: Introduce a baseline function dependent on the state

$$\nabla_{\theta} J(\theta) \approx \sum_{t>0} \left(\sum_{t>t'} \beta^{t'-t} r_{t'} - b(s_t) \right) \nabla_{\theta} \log \pi_{\theta}(a_t | s_t)$$

- Baseline: want to push the probability of an action from a state if this action was better than the expected value of what we should get from that state.
- Intuitively, we are happy with an action a_t in a state s_t if $Q^{\pi}(s_t, a_t) - V^{\pi}(s_t)$ is large. On the contrary, we are unhappy with the action if it is small.

$$\nabla_{\theta} J(\theta) \approx \sum_{t>0} (Q^{\pi_{\theta}}(s_t, a_t) - V^{\pi_{\theta}}(s_t)) \nabla_{\theta} \log \pi_{\theta}(a_t | s_t)$$

Actor critic algorithm

- Problem; We do not know Q and V . Can we learn them?
- Yes, using Q learning. We combine Policy Gradients and Q -learning by training both an actor (the policy) and a critic (the Q -function).
 - The actor decides which action to take, and the critic tells the actor how good its action was and how it should adjust.
 - Also, alleviates the task of the critic as it only has to learn the value of (state, action) pair generated by the policy.
 - Can also incorporate Q -learning tricks, e.g., experience replay.
- Remark: we can define by the advantage function how much an action was better than expected

$$A^\pi(s, a) = Q^\pi(s, a) - V^\pi(s)$$

- It only has to learn the value function along the path where it matters

Algorithm 2. Actor-critic algorithm

Initialize policy parameters θ , critic parameters ϕ
for iteration = 1,2,... do
 Sample the trajectories under the current policy
 For $i=1,\dots,m$ do
 for $t=1,T$ do
 $A_t = \sum_{t>t'} \beta^{t'-t} r_t^i - V_\phi(s_t^i)$
 $\Delta\theta = \Delta\theta + A_t \nabla_\theta \log \pi_\theta(a_t^i | s_t^i)$
 $\Delta\phi = \sum_i \sum_t \nabla_\phi A_t^2$
 end for
 end for

Materials

For preparing this content, we used multiple sources including:

- Fei-Fei Li, Justin Johnson, Serena Yeung. Reinforcement Learning. Stanford University. Lecture 14.
<https://www.youtube.com/watch?v=lvoHnicueoE>
- Sutton R. and A. Barto (2018). Reinforcement Learning. Second edition. The MIT Press.
- Silvero, D. Reinforcement learning. University College of London, Lecture slides.
- Ng, A., Reinforcement Learning and Control. Lecture notes. Stanford University.
- Ng, A., Policy Gradient (Reinforce). Lecture notes. Stanford University.

When the U.S. catches a cold, Canada sneezes: a lower-bound tale told by deep learning

Vadym Lepetyuk Lilia Maliar Serguei Maliar

The views expressed in this presentation are solely those of the authors and may differ from official Bank of Canada views. No responsibility for them should be attributed to the Bank of Canada.

November, 2022

Large-scale DSGE models for projection and policy analysis

Mostly used by central banks and government agencies:

- International Monetary Fund's Global Economy Model, GEM (Bayoumi et al., 2001);
- US Federal Reserve Board's SIGMA model (Erceg et al., 2006);
- Bank of Canada Terms of Trade Economic Model, ToTEM II (Dorich et al. 2013);
- European Central Bank's New Area-Wide Model, NAWM (Coenen et al. 2008);
- Bank of England COMPASS model (Burgess et al., 2013);
- Swedish Riksbank's Ramses II model (Adolfson et al., 2013).

What are central bank models aimed to do?

1. Central bank models must mimic as close as possible the actual economies in every possible dimension.
 - *Then, the policymakers can produce realistic simulation of alternative policies and to choose the best one*
2. Central bank models must be rich and flexible enough to describe interactions between many variables of interest, including different types of foreign and domestic consumption, investment, capital, labor, prices, exchange rate, etc.
 - *Central bank models may contain hundreds of equations*
 - *Their estimation, calibration, solution and simulation are highly nontrivial tasks*
3. Central banks need DSGE models for policy analysis.
 - *Econometric models have limitations for policy analysis (Lucas critique)*

Linearized solutions to central bank models

- The central banks use linear (first-order) perturbation methods
 - dynare, IRIS.
- *Advantages:*
 - computationally inexpensive;
 - simple to use;
 - can be applied to very large problems.
- *Potential drawbacks:*
 - insufficiently accurate in the presence of strong nonlinearities;
 - does not allow to study occasionally binding constraints (with an exception of the linear case; see OccBin of Guerrieri and Iacoviello, 2015, and IRIS).
- *Nonlinear effects can be economically significant:*
 - Approximation errors in the new Keynesian model can reach hundreds percent near ZLB; see Judd, Maliar and Maliar (Econometrica, 2017).

The view of nonlinearity before Great Recession

Policymakers were not concerned about nonlinearities before Great Recession.

- Leahy (2013): *“Prior to the crisis, it was easier to defend the proposition that non-linearities were unimportant than it was to defend the proposition that non-linearities were essential for understanding macroeconomic dynamics.”*
- Bullard (2013): *“... the idea that U.S. policymakers should worry about the nonlinearity of the Taylor-type rule and its implications is sometimes viewed as an amusing bit of theory without real ramifications. Linear models tell you everything you need to know. And so, from the denial point of view, we can stick with our linear models...”*

The view of nonlinearity after Great Recession

Policymakers were confronted with questions after Great Recession.

- *Macroeconomists overlooked Great Recession. Why?*
 - Is it because they had bad models?
 - Or is it because they used bad linear methods that distorted policy implications of good models?
- In the light of these questions, Bank of Canada created a working group whose objective was to construct a global solution to its large-scale DSDE model which led to the present paper.

Methodological contribution

- Apply deep supervised and unsupervised learning techniques to break the curse of dimensionality and to construct accurate global nonlinear solutions to large-scale central bank models.
- Quantify the difference between local linear and global nonlinear solutions in realistically calibrated central bank models.
- Explore whether inaccuracies of the linearization method can significantly distort policy implications of central bank models.
- Establish the key determinants of the recent ZLB episode in the Canadian economy.
- Analyze policies that would allow to avoid the ZLB crisis in the future.

Empirical findings

1. We demonstrate that the international transmission of ELB is empirically plausible mechanism for explaining the Canadian ELB experience.
2. We show that it is relatively easy to generate realistic ELB (or ZLB) episodes in new Keynesian models via the foreign shocks unlike the domestic shocks.
3. We find that the Canadian economy would entirely avoid the ELB episode if the target inflation rate were 3 instead of 2 percent.
4. Contrary to what we expected, we find that the ELB constraint plays a relatively minor role in the bToTEM's performance.
5. We discover that other nonlinearities – those not associated with ELB – can play an important role in the model's predictions such as the uncertainty effect.
6. Most strikingly, we find that the closing condition, used to ensure stationarity in open-economy models, plays an important role in the bToTEM dynamics. This is at odds with well known result of Schmitt-Grohé and Uribe (2003) because we consider nonlinear model while they focus on linearization.

Outline of the talk

1. Great Recession in Canada.
2. Bank of Canada ToTEM model for projections and policy analysis.
3. Breaking the curse of dimensionality using machine learning:
 - *clustering analysis*;
 - *neural networks and deep learning*.
4. Policy experiments:
understanding the Canadian ZLB crisis.
5. Conclusion.

Great Recession 2007-2009 and ZLB crisis in Canada

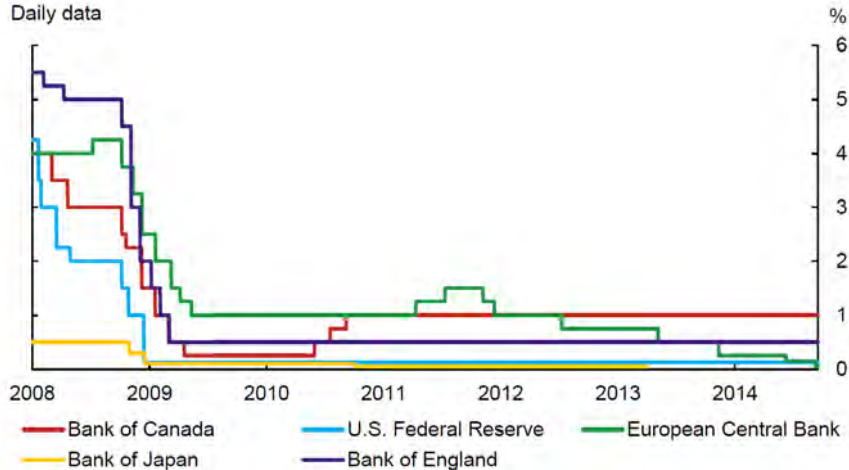
- Canada was not at the epicenter of Great Recession.
- Unlike the U.S. and Europe, the Canadian economy did not experience a subprime crisis in 2007-2008.
- However, the contagion spread through a number of transmission channels.
- For Canada, an important transmission channel was through a direct impact on foreign trade.

There is a popular saying “*When the U.S. sneezes, Canada catches a cold*”. But this time it went the other way around: it was the U.S. that caught the (subprime crisis) cold, and it was Canada that sneezed.

Great Recession in Canada

A sharp decline in target interest rates

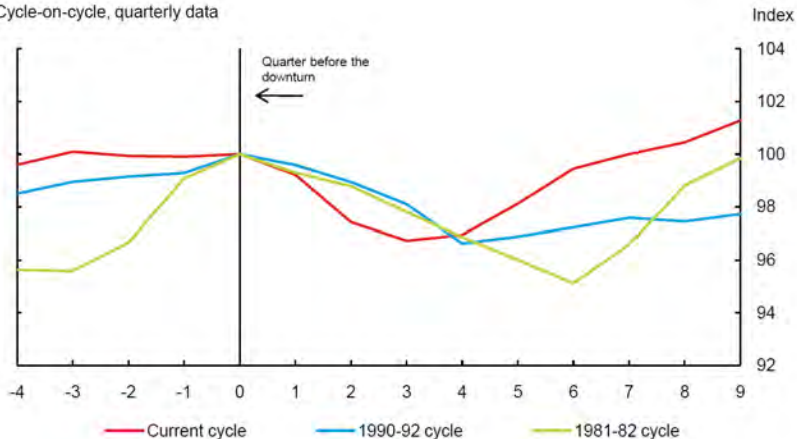
Daily data



Great Recession in Canada

A faster drop and faster recovery in real GDP

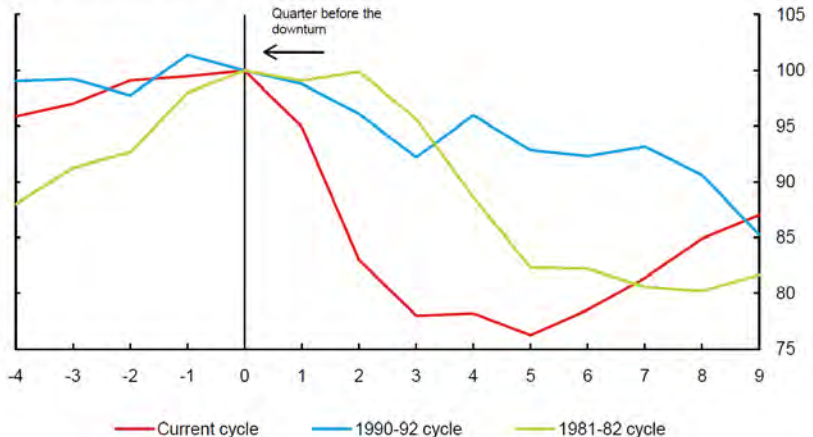
Cycle-on-cycle, quarterly data



Great Recession in Canada

A rapid drop in real fixed investment

Cycle-on-cycle, quarterly data

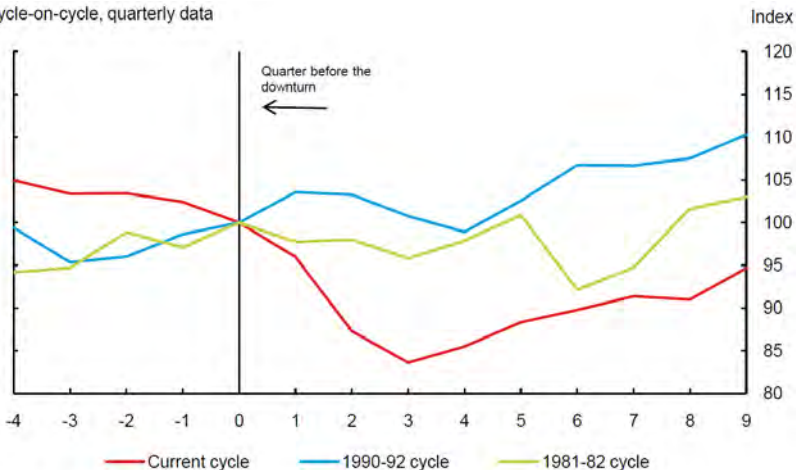


Sources: Statistics Canada and Bank of Canada calculations

Great Recession in Canada

A historic drop in real exports

Cycle-on-cycle, quarterly data



Sources: Statistics Canada and Bank of Canada calculations

Full-scaled ToTEM model of Bank of Canada

- The Terms of Trade Economic Model (ToTEM) – the main projection and policy analysis model of the Bank of Canada.
- Small-open economy model.
- ToTEM includes 356 equations and unknowns
⇒ It is too large for the existing global solution methods.

A scaled-down version of ToTEM

- We construct a scaled-down version of ToTEM, which we call a “baby ToTEM” (bToTEM) model.
- bToTEM includes 49 equations and unknowns
⇒ It is still a large-scale model.
- Two production sectors: final-good production and commodity production.
- Meaningful trade: final goods, commodities, imports.
- One representative household, with differentiated labour services.
- Taylor-type interest rate rule
- Six shocks, including exogenous rest-of-the-world (ROW) shocks.

Differences between ToTEM and bToTEM

- *ToTEM*
 - 5 distinct production sectors (consumption goods and services, investment goods, government goods, noncommodity export goods, and commodities);
 - 4 sectors are identical except of parameters, while the commodity sector is different;
 - A separate economic model of the rest of the world (ROW);
 - 3 types of households (they differ in their saving opportunities);
 - 8 Phillips curves.
- *bToTEM*
 - The final-good production sector is identical in structure to the consumption goods and services sector of ToTEM;
 - Linear technologies for transforming the output of this sector into other types of output that correspond to the remaining ToTEM's sectors;
 - The ROW sector is modeled using exogenous processes for foreign variables;
 - All households are of the same type;
 - 3 Phillips curves.

Production of final goods

Two stages:

1. In the first stage, intermediate goods are produced by identical perfectly competitive firms from labour, capital, commodities, and imports.
2. In the second stage, a variety of final goods are produced by monopolistically competitive firms from the intermediate goods. The variety of final goods is then aggregated into the final consumption good.

First stage of production

- Perfectly competitive firms produce an intermediate good:

$$Z_t^g = \left(\delta_l (A_t L_t)^{\frac{\sigma-1}{\sigma}} + \delta_k (u_t K_t)^{\frac{\sigma-1}{\sigma}} + \delta_{com} (COM_t^d)^{\frac{\sigma-1}{\sigma}} + \delta_m (M_t)^{\frac{\sigma-1}{\sigma}} \right)^{\frac{\sigma}{\sigma-1}}, \quad (1)$$

L_t , K_t , COM_t^d = labour, capital and commodity inputs, resp.,
 M_t = imports, u_t = capital utilization, A_t = the level of labour-augmenting technology,

$$\log(A_t) = \varphi_a \log(A_{t-1}) + (1 - \varphi_a) \log(\bar{A}) + \xi_t^a. \quad (2)$$

- Capital depreciates according to the following law of motion

$$K_{t+1} = (1 - d_t) K_t + I_t, \quad (3)$$

where d_t is the depreciation rate, and I_t is investment.

- The depreciation rate increases with capital utilization:

$$d_t = d_0 + \bar{d} e^{\rho(u_t - 1)}. \quad (4)$$

First stage of production (cont.)

- The firms incur a quadratic adjustment cost when adjusting the level of investment. The net output is given by

$$Z_t^n = Z_t^g \square \frac{\chi_i}{2} \left(\frac{I_t}{I_{t-1}} \square 1 \right)^2 I_t. \quad (5)$$

- The objective of the firms is to choose $L_t, K_{t+1}, I_t, COM_t, M_t, u_t$ in order to maximize profits

$$E_0 \sum_{t=0}^{\infty} \mathcal{R}_{0,t} \square (P_t^z Z_t^n \square W_t L_t \square P_t^{com} COM_t^d \square P_t^i I_t \square P_t^m M_t)$$

s.t. (1) \square (5),

$\mathcal{R}_{t,t+j} = \beta^j (\lambda_{t+j}/\lambda_t) (P_t/P_{t+j}) =$ stochastic discount factor.

Second stage of production

- A monopolistically competitive firm $i \in [0, 1]$ produces a differentiated good

$$Z_{it} = \min \left(\frac{Z_{it}^n}{1 \square s_m}, \frac{Z_{it}^{mi}}{s_m} \right),$$

Z_{it}^n = intermediate good; Z_{it}^{mi} = manufactured input;
 s_m = a Leontief parameter.

- The differentiated goods Z_{it} are aggregated into the final good Z_t according to a CES function:

$$Z_t = \left(\int_0^1 Z_{it}^{\frac{\varepsilon \square 1}{\varepsilon}} di \right)^{\frac{\varepsilon}{\varepsilon \square 1}}.$$

- Cost minimization implies

$$Z_{it} = \left(\frac{P_{it}}{P_t} \right)^{\square \varepsilon} Z_t,$$

where $P_t = \left(\int_0^1 P_{it}^{1 \square \varepsilon} di \right)^{\frac{1}{1 \square \varepsilon}}$.

Second stage of production (cont.)

- The final good is used as the manufactured inputs by each of the monopolistically competitive firms.
- Two types of monopolistically competitive firms: rule-of-thumb firms of measure ω and forward-looking firms of measure $1 - \omega$.
- Within each type, with probability θ the firms index their price to the (time-varying) inflation target $\bar{\pi}_t$.
- The rule-of-thumb firms, which do not index their price in the current period, partially index their price:

$$P_{it} = (\pi_{t-1})^{1-\theta} (\bar{\pi}_t)^{\theta} P_{i,t-1}.$$

Second stage of production (cont.)

- The optimizing forward-looking firms solve:

$$\max_{P_t^*} E_t \left\{ \sum_{j=0}^{\infty} \theta^j \mathcal{R}_{t,t+j} \prod_{k=1}^j \bar{\pi}_{t+k} P_t^* Z_{i,t+j} \right. \\ \left. \square (1 \square s_m) P_{t+j}^z Z_{i,t+j} \square s_m P_{t+j} Z_{i,t+j} \right\}$$

subject to demand constraints

$$\text{s.t. } Z_{i,t+j} = \left(\frac{\prod_{k=1}^j \bar{\pi}_{t+k} P_t^*}{P_{t+j}} \right)^{\square \varepsilon} Z_{t+j}.$$

Relation between the first and second stages of production

- Production in the first and second stages are related as

$$Z_t^n = \int_0^1 Z_{it}^n di = (1 - s_m) \int_0^1 Z_{it} di = (1 - s_m) \Delta_t Z_t,$$

$$\Delta_t = \int_0^1 \left(\frac{P_{it}}{P_t} \right)^{-\varepsilon} di = \text{price dispersion}.$$

- Law of motion of the price dispersion:

$$\Delta_t = \theta \left(\frac{\bar{\pi}_t}{\pi_t} \right)^{-\varepsilon} \Delta_{t-1} + (1 - \theta) \omega \frac{(\pi_{t-1}) (\bar{\pi}_t)^{1-\varepsilon}}{\pi_t} \Delta_{t-1} + (1 - \theta) (1 - \omega) \left(\frac{P_t^*}{P_t} \right)^{-\varepsilon}.$$

- Investment goods and noncommodity exports are produced from the final goods according to linear technology, $P_t^i = \iota_i P_t$ and $P_t^{nc} = \iota_x P_t$.

Imports

- Intermediate imported goods M_{it} are bounded into the final imported good M_t according to

$$M_t = \left(\int_0^1 M_{it}^{\frac{\varepsilon_m - 1}{\varepsilon_m}} di \right)^{\frac{\varepsilon_m}{\varepsilon_m - 1}}.$$

- The demand for an intermediate imported good i :

$$M_{it} = \left(\frac{P_{it}^m}{P_t^m} \right)^{\varepsilon_m} M_t,$$

where $P_t^m = \left(\int_0^1 (P_{it}^m)^{1 - \varepsilon_m} di \right)^{\frac{1}{1 - \varepsilon_m}}$.

Import (cont.)

- The optimizing forward-looking firms solves

$$\max_{P_t^{m*}} E_t \left[\sum_{j=0}^{\infty} (\theta_m)^j \mathcal{R}_{t,t+j} \prod_{k=1}^j \bar{\pi}_{t+k} P_t^{m*} M_{i,t+j} \square e_{t+j} P_{t+j}^{mf} M_{i,t+j} \right]$$

subject to demand constraints

$$M_{i,t+j} = \left(\frac{\prod_{k=1}^j \bar{\pi}_{t+k} P_t^{m*}}{P_{t+j}^m} \right)^{\square \varepsilon_m} M_{t+j},$$

P_t^{mf} = foreign price of imports;

e_t = nominal exchange rate (domestic price of a unit of foreign currency).

Foreign demand and foreign economy

- The foreign demand for Canadian noncommodity exports X_t^{nc} is given by the demand function

$$X_t^{nc} = f \left(\frac{P_t^{nc}}{e_t P_t^f} \right)^{\varphi} Z_t^f,$$

P_t^{nc}/e_t = foreign price of noncommodity exports;

P_t^f = foreign general price level;

Z_t^f = foreign output.

- The balance of payments

$$\frac{e_t B_t^f}{R_t^f (1 + \kappa_t^f)} - e_t B_{t-1}^f = P_t^{nc} X_t^{nc} + P_t^{com} X_t^{com} - P_t^m M_t,$$

B_t^f = domestic holdings of foreign-currency denominated bonds.;

κ_t^f = risk premium.

Foreign demand and foreign economy (cont.)

- The rest of the world is specified by three exogenous processes that describe the evolution of foreign variables.
- The foreign output Z_t^f

$$\log(Z_t^f) = \varphi_{Zf} \log(Z_{t-1}^f) + (1 - \varphi_{Zf}) \log(\bar{Z}^f) + \xi_t^{zf}.$$

- A foreign interest rate shock r_t^f

$$\log(r_t^f) = \varphi_{rf} \log(r_{t-1}^f) + (1 - \varphi_{rf}) \log(\bar{r}) + \xi_t^{rf}.$$

- A foreign commodity price p_t^{comf} is

$$\log(p_t^{comf}) = \varphi_{comf} \log(p_{t-1}^{comf}) + (1 - \varphi_{comf}) \log(\bar{p}^{comf}) + \xi_t^{comf},$$

$\xi_t^{zf}, \xi_t^{rf}, \xi_t^{comf}$ = normally distributed random variables;

$\varphi_{Zf}, \varphi_{rf}, \varphi_{comf}$ = autocorrelation coefficients.

Commodity production

- The commodities are produced from the final goods by representative, perfectly competitive domestic firms:

$$COM_t = (Z_t^{com})^{s_z} (A_t F)^{1-s_z} \square \frac{\chi_{com}}{2} \left(\frac{Z_t^{com}}{Z_{t-1}^{com}} \square 1 \right)^2 Z_t^{com},$$

F = a fixed production factor (land).

Here, the second term is quadratic adjustment costs.

- The commodities are sold at the rest-of-the-world price adjusted by the nominal exchange rate

$$P_t^{com} = e_t P_t^{comf}.$$

- The commodities are sold domestically (COM_t^d) or exported to the rest of the world (X_t^{com})

$$COM_t = COM_t^d + X_t^{com}.$$

Households

- The representative household's period utility function:

$$U_t = \frac{\mu}{\mu - 1} C_t \xi \bar{C}_{t-1}^{\frac{\mu-1}{\mu}} \exp\left(\frac{\eta(1-\mu)}{\mu(1+\eta)} \int_0^1 (L_{ht})^{\frac{\eta+1}{\eta}} dh\right) \eta_t^c,$$

C_t = consumption of finished goods;

\bar{C}_t = aggregate consumption;

L_{ht} = labour service of type h ;

η_t^c = a consumption demand shock.

- We assume

$$\log(\eta_t^c) = \varphi_c \log(\eta_{t-1}^c) + \xi_t^c,$$

ξ_t^c = a normally distributed variable; φ_c = an autocorrelation coefficient.

Households (cont.)

- The representative household solves

$$E_t \left[\sum_{j=0}^{\infty} \beta^j U_{t+j} \right]$$

$$\text{s.t. } P_t C_t + \frac{B_t}{R_t} + \frac{e_t B_t^f}{R_t^f (1 + \kappa_t^f)} = B_{t+1} + e_t B_{t+1}^f + \int_0^1 W_{ht} L_{ht} dh + \Pi_t,$$

B_t = holdings of domestic bonds;

B_t^f = holdings of foreign-currency denominated bonds;

Π_t are profits paid by the firms.

- To induce the stationarity of the model, we assume that the risk premium κ_t^f is

$$\kappa_t^f = \varsigma \left(\bar{b}^f - b_t^f \right),$$

$b_t^f = e_t B_t^f / \left(\pi_{t+1}^f P_t \bar{Y} \right)$ = normalized bond holdings; see Schmitt-Grohé and Uribe (2003).

Wage setting

- Households;
- Aggregating firms (labor packers) – in perfect competition;
- Labour unions – in monopolistic competition.

Labour packers

- The representative household supplies a variety of differentiated labour service L_{ht} , $h \in [0, 1]$ to the labour market.
- The differentiated labour service is aggregated by labor packers according to

$$L_t = \left(\int_0^1 L_{ht}^{\frac{\varepsilon_w - 1}{\varepsilon_w}} dh \right)^{\frac{\varepsilon_w}{\varepsilon_w - 1}}.$$

- L_t is used in the first stage of production.
- Cost minimization of the labor packer implies the demand

$$L_{ht} = \left(\frac{W_{ht}}{W_t} \right)^{-\varepsilon_w} L_t,$$

$$W_{ht} = \text{wage for labour of type } h; W_t \equiv \left(\int_0^1 W_{ht}^{1 - \varepsilon_w} dh \right)^{\frac{1}{1 - \varepsilon_w}}.$$

Labour unions (cont.)

- Two types: rule-of-thumb (measure ω_w) and forward-looking (measure $1 - \omega_w$) unions.
- Within each type, with probability θ_w a union indexes wage to the inflation target $\bar{\pi}_t$, $W_{it} = \bar{\pi}_t W_{i,t-1}$.
- The rule-of-thumb unions which do not index their wage set

$$W_{it} = \pi_{t-1}^w (\bar{\pi}_t)^{1-\omega_w} W_{i,t-1}.$$

- The forward-looking unions that do not index their wage solve:

$$\begin{aligned} & \max_{W_t^*} E_t \left[\sum_{j=0}^{\infty} (\beta \theta_w)^j U_{t+j} \right] \\ \text{s.t. } & L_{h,t+j} = \left(\frac{\prod_{k=1}^j \bar{\pi}_{t+k} W_t^*}{W_{t+j}} \right)^{\varepsilon_w} L_{t+j}, \end{aligned}$$

$$P_{t+j} C_{t+j} = \prod_{k=1}^j \bar{\pi}_{t+k} W_t^* L_{h,t+j} dh + \dots$$

Monetary policy

- Taylor rule:

$$\begin{aligned} R_t &= \rho_r R_{t-1} + (1 - \rho_r) [\bar{R} + \rho_\pi (\pi_t - \bar{\pi}_t) + \rho_Y (\log Y_t - \log \bar{Y}_t)] + \eta_t^r \\ &\equiv \Phi_t, \end{aligned}$$

\bar{Y}_t = potential output; η_t^r = interest rate shock,

$$\eta_t^r = \varphi_r \eta_{t-1}^r + \xi_t^r,$$

ξ_t^r = a normally distributed variable; φ_r = autocorrelation coefficient.

- Potential output changes with productivity as

$$\log \bar{Y}_t = \varphi_{zf} \log \bar{Y}_{t-1} + (1 - \varphi_{zf}) \log \left(\frac{A_t \bar{Y}}{\bar{A}} \right).$$

- If an effective lower bound (ELB) is imposed on the nominal interest rate, R_t^{ELB} , then

$$R_t = \max \{ \Phi_t, R_t^{ELB} \}.$$

Market clearing conditions

- Resource feasibility condition

$$Z_t = C_t + \iota_i I_t + \iota_x X_t^{nc} + Z_t^{com} + v_z Z_t.$$

- GDP

$$Y_t = C_t + I_t + X_t^{nc} + X_t^{com} \square M_t + v_y Y_t.$$

- GDP de ator

$$P_t^y Y_t = P_t C_t + P_t^i I_t + P_t^{nc} X_t^{nc} + P_t^{com} X_t^{com} \square P_t^m M_t + v_y p_t^y Y_t.$$

A list of model's variables

#	variable	notation
1	productivity	A_t
2	labour input	L_t
3	capital input	K_t
4	investment	I_t
5	commodities used domestically	COM_t^d
6	import	M_t
7	capital utilization	u_t
8	capital depreciation	d_t
9	gross production of intermediate good	Z_t^g
10	net production of intermediate good	Z_t^n
11	total production	Z_t
12	consumption	C_t
13	GDP	Y_t
14	marginal utility of consumption	λ_t
15	inflation	π_t
16	real marginal cost	rmc_t
17	consumption Phillips curve term	F_{1t}
18	consumption Phillips curve term	F_{2t}
19	price dispersion	Δ_t

A list of model's variables (cont.)

#	variable	notation
20	imported good inflation	π_t^m
21	imports Phillips curve term	F_{1t}^m
22	imports Phillips curve term	F_{2t}^m
23	inflation target	$\bar{\pi}_t$
24	real price of import	p_t^m
25	real exchange rate	s_t
26	nominal interest rate	R_t
27	interest rate shock process	η_t^r
28	real price of intermediate good	p_t^z
29	real wage	w_t
30	real price of commodities	p_t^{com}
31	marginal product of capital	MPK_t
32	interest rate on capital	R_t^k
33	Tobin's Q	q_t
34	real price of investment	p_t^i
35	foreign-currency price of commodities	p_t^{comf}
36	foreign real interest rate	r_t^f

A list of model's variables (cont.)

#	variable	notation
37	interest premium on foreign bonds	κ_t^f
38	holdings of foreign bonds in real terms	b_t^f
39	non-commodity export	X_t^{nc}
40	export of commodities	X_t^{com}
41	foreign demand	Z_t^f
42	total commodities produced	COM_t
43	final goods used in commodity production	Z_t^{com}
44	wage inflation	π_t^w
45	wage Phillips curve term	F_{1t}^w
46	wage Phillips curve term	F_{2t}^w
47	optimal wage	w_t^*
48	wage dispersion	Δ_t^w
49	foreign price of import	p_t^{mf}
50	GDP deflator	p_t^y
51	price of non-commodity export	p_t^{xz}
52	consumption demand shock process	η_t^c
53	potential GDP	\bar{Y}_t

Calibration

- 61 parameters to calibrate.
- Whenever possible, we use the same parameters as in ToTEM.
- We choose the remaining parameters to reproduce observations on the Canadian economy.
- We target the ratios of the following variables to nominal GDP:
 - consumption,
 - investment,
 - noncommodity export,
 - commodity export,
 - import,
 - total commodities,
 - labor input.

Parameters in endogenous model's equations

Parameter	Symbol	Value	Source
Rates			
– real interest rate	\bar{r}	1.0076	ToTEM
– discount factor	β	0.9925	ToTEM
– inflation target	$\bar{\pi}$	1.005	ToTEM
– nominal interest rate	\bar{R}	1.0126	ToTEM
– ELB on the nominal interest rate	R^{ELB}	1.01	fixed
Output production			
– CES elasticity of substitution	σ	0.5	ToTEM
– CES labor share parameter	δ_L	0.249	calibrated
– CES capital share parameter	δ_K	0.575	calibrated
– CES commodity share parameter	δ_{COM}	0.0015	calibrated
– CES import share parameter	δ_M	0.0287	calibrated
– investment adjustment cost	χ_I	20	calibrated
– fixed depreciation rate	d_0	0.0054	ToTEM
– variable depreciation rate	\bar{d}	0.0261	ToTEM
– depreciation semielasticity	ρ	4.0931	calibrated
– real investment price	ι_I	1.2698	ToTEM
– real noncommodity export price	ι_X	1.143	ToTEM
– labour productivity	\bar{A}	100	normalization

Parameters in endogenous model's equations

Parameter	Symbol	Value	Source
Price setting parameters for consumption			
– probability of indexation	θ	0.75	ToTEM
– RT indexation to past inflation		0.0576	ToTEM
– RT share	ω	0.4819	ToTEM
– elasticity of substitution of consumption goods	ε	11	ToTEM
– Leontieff technology parameter	s_m	0.6	ToTEM
Price setting parameters for imports			
– probability of indexation	θ^m	0.8635	ToTEM
– RT indexation to past inflation	m	0.7358	ToTEM
– RT share	ω^m	0.3	ToTEM
– elasticity of substitution of imports	ε^m	4.4	ToTEM
Price setting parameters for wages			
– probability of indexation	θ^w	0.5901	ToTEM
– RT indexation to past inflation	w	0.1087	ToTEM
– RT share	ω^w	0.6896	ToTEM
– elasticity of substitution of labour service	ε^w	1.5	ToTEM

Parameters in endogenous model's equations

Parameter	Symbol	Value	Source
Household utility			
– consumption habit	ξ	0.9396	ToTEM
– consumption elasticity of substitution	μ	0.8775	ToTEM
– wage elasticity of labor supply	η	0.0704	ToTEM
Taylor rule			
– interest rate persistence parameter	ρ_r	0.83	ToTEM
– interest rate response to inflation gap	ρ_π	4.12	ToTEM
– interest rate response to output gap	ρ_y	0.4	ToTEM
Other			
– capital premium	κ^k	0.0674	calibrated
– exchange rate persistence parameter	\varkappa	0.1585	ToTEM
– foreign commodity price	\bar{p}^{COMf}	1.6591	ToTEM
– foreign import price	\bar{p}^{Mf}	1.294	ToTEM
– risk premium response to debt	ς	0.0083	calibrated
– foreign demand elasticity	ϕ	0.4	calibrated
– elasticity in commodity production	s_z	0.8	calibrated
– land	F	0.1559	calibrated
– share of other components of output	v_Z	0.7651	calibrated
– share of other components of GDP	v_Y	0.311	calibrated

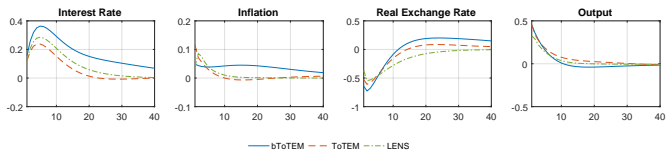
Parameters in exogenous model's equations

Parameter	Symbol	Value	Source
Shock persistence			
– persistence of interest rate shock	φ_R	0.25	ToTEM
– persistence of productivity shock	φ_A	0.9	fixed
– persistence of consumption demand shock	φ_C	0	fixed
– persistence of foreign output shock	φ_{Zf}	0.9	fixed
– persistence of foreign commodity price shock	φ_{pCOMf}	0.87	calibrated
– persistence of foreign interest rate shock	φ_{rf}	0.88	calibrated
Shock volatilities			
– volatility of interest rate shock	σ_R	0.0006	calibrated
– volatility of productivity shock	σ_A	0.0067	calibrated
– volatility of consumption demand shock	σ_C	0.0001	fixed
– volatility of foreign output shock	σ_{Zf}	0.0085	calibrated
– volatility of foreign commodity price shock	σ_{pCOMf}	0.0796	calibrated
– volatility of foreign interest rate shock	σ_{rf}	0.002	calibrated

A comparison of bToTEM to ToTEM and LENS

- The Bank of Canada uses a first-order perturbation method to solve ToTEM.
- For ToTEM, we use IRIS – open-source software used by the Bank of Canada for macroeconomic modeling.
- For bToTEM, we use IRIS Toolbox, as well as Dynare.
- We checked that the IRIS and Dynare packages produce indistinguishable numerical solutions for bToTEM.
- Also, we include for comparison the LENS model – another model of the Bank of Canada.
- LENS is a semistructural model.
- Both, the ToTEM and LENS models, include more shocks than bToTEM:
 - 52 shocks in ToTEM and 98 shocks in LENS.

Impulse response to a consumption demand shock



Understanding the difference between the bToTEM and ToTEM

- Consider a linearized version of the Phillips curve in the two models

$$\hat{\pi}_t = (1 - \theta) \omega \tilde{\phi}^{\square 1} \hat{\pi}_{t \square 1} + \beta \theta \tilde{\phi}^{\square 1} E[\hat{\pi}_{t+1}] + \tilde{\lambda} r \hat{m}c_t + \varepsilon_t^P,$$

$r \hat{m}c_t$ = real marginal cost;

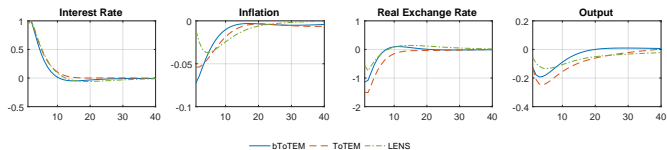
ε_t^P = weighted average of the inflation target and the deviation of markup from the steady state;

θ , ω are the price stickiness parameters;

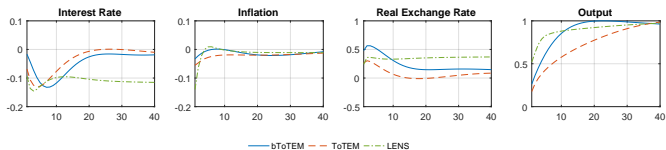
$\tilde{\phi}$ and $\tilde{\lambda}$ = parameters that are identical in ToTEM and bToTEM (defined in Dorich et al., 2013).

- We conclude that inflation dynamics are driven by differing $r \hat{m}c_t$.
- In ToTEM, a consumption shock triggers a reallocation of inputs into the consumption good sector from the other four sectors. With adjustment costs, this raises $r \hat{m}c_t$.
- In contrast, in bToTEM, there is one production sector and there are no input adjustment costs. The responses and decays of the $r \hat{m}c_t$ are far less pronounced.

Impulse response to an interest rate shock



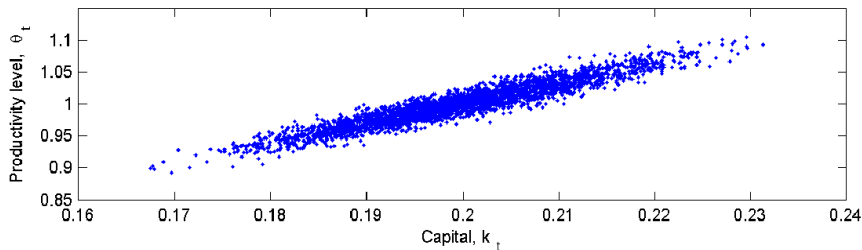
Impulse response to a permanent productivity shock



bToTEM: a serious challenge for global methods

- The models like bToTEM has not been yet studied in the literature using global methods.
- bToTEM contains 21 state variables (6 exogenous and 15 endogenous ones) \implies *curse of dimensionality*.
- Moreover, the bToTEM's equations are complex and require the use of numerical solvers.
- To solve the bToTEM model, we ameliorate the curse of dimensionality by focusing on the ergodic set of the model.

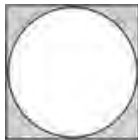
Advantages of ergodic set methods



- Ergodic set is constructed by stochastic simulation.
- "Grid" is adaptive: we solve the model only in the high-probability area of the state space—this is where the solution "lives"!

Reduction in cost in a 2-dimensional case

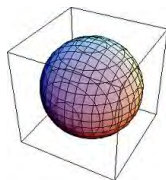
- How much can we save on cost using the ergodic-set domain comparatively to the hypercube domain?
- Suppose the ergodic set is a circle (it was an ellipse in the figure).
- In the 2-dimensional case, a circle inscribed within a square occupies about 79% of the area of the square.
- The reduction in cost is proportional to the shaded area in the figure.



- It does not seem to be a large gain.

Reduction in cost in a d-dimensional case

- In a 3-dimensional case, the gain is larger (a volume of a sphere of diameter 1 is 52% of the volume of a cube of width 1)



- In a d -dimensional case, the ratio of a hypersphere's volume to a hypercube's volume

$$\mathcal{V}^d = \begin{cases} \frac{(\pi/2)^{\frac{d-1}{2}}}{1 \cdot 3 \cdot \dots \cdot d} & \text{for } d = 1, 3, 5, \dots \\ \frac{(\pi/2)^{\frac{d}{2}}}{2 \cdot 4 \cdot \dots \cdot d} & \text{for } d = 2, 4, 6, \dots \end{cases}$$

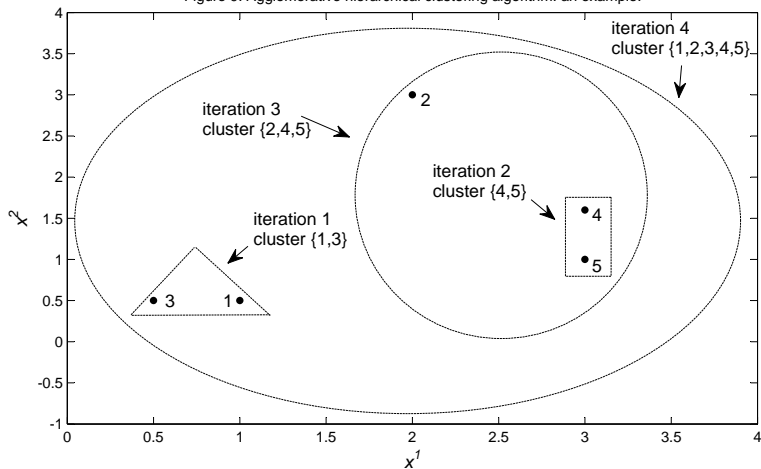
- \mathcal{V}^d declines very rapidly with dimensionality of state space. When $d = 10 \Rightarrow \mathcal{V}^d = 3 \cdot 10^{-3}$ (0.3%). When $d = 30 \Rightarrow \mathcal{V}^d = 2 \cdot 10^{-14}$.
- We face a tiny fraction of cost we would have faced on the hypercube.

Cluster grid algorithm (CGA)

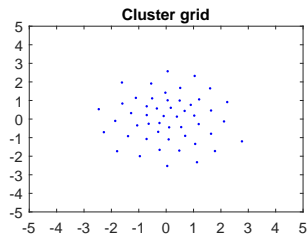
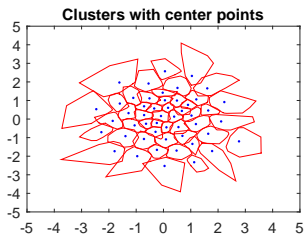
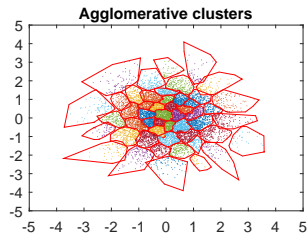
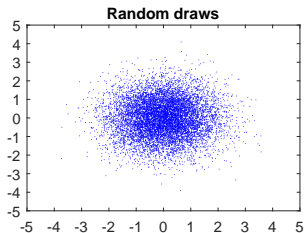
- CGA is an ergodic set algorithm of Maliar and Maliar (QE, 2015).
- It is a projection-style global solution method that uses adaptive grid:
 - the model is solved only in the area of the state space visited in simulation
- The essence of CGA is clustering methods from machine learning analysis.
 - Crude simulation produces points that are located close to one another; many points are redundant.
 - Clustering methods reduce the number of redundant points.
- CGA can accurately solve models with dozens of state variables.

Example: agglomerative hierarchical clustering algorithm

Figure 9. Agglomerative hierarchical clustering algorithm: an example.



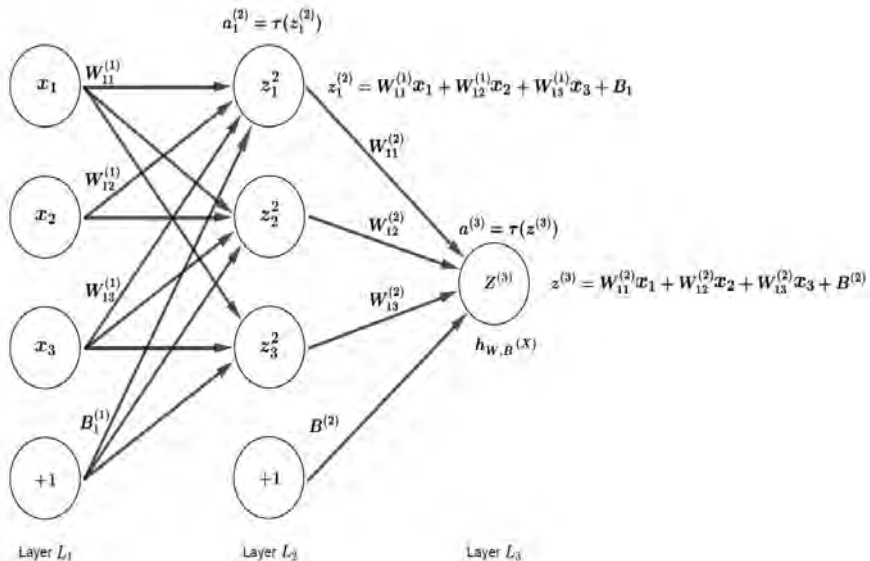
Example: Construction of a cluster grid



Deep learning

- In the 2017 version of the paper, we constructed nonlinear solutions using second-order polynomial approximation.
- But second-degree polynomial is not flexible enough to accurately approximate highly non-linear models like bToTEM.
- The difference between perturbation and global solutions cannot be too large as both of them are built using the same quadratic approximation function.
- To approximate nonlinearities more accurately, one needs more flexible approximating functions.
- In the new 2019 version, we introduce neural networks and deep learning techniques to produce more flexible approximations.

Artificial neural networks



Artificial neural networks

- $x = (x_0, x_1, \dots, x_n)'$ and
- $z_1^{(2)} = W_{11}^{(1)}x_1 + \dots + W_{1n}^{(1)}x_n + B^{(1)}$.
- $W_{jk}^{(l)}$ = weights that control mapping from layer l to layer $l + 1$.
- τ = activation function (computation by a neuron).
- $a^{(l)}$ = activated output in layer l .
- $h_{W,B} \square x^{(i)}$ = hypothesis about our relationship between x 's and y 's.
- Cost function $J(W, B)$ allows us to evaluate how different $h_{W,B} \square x^{(i)}$ is from y .
- To prevent overfitting, $J(W, B)$ may include penalty on large values of $W_{j,k}^{(l)}$.
- $\min_{W,B} J(W, B)$. Need to compute $\frac{\partial}{\partial W_{ij}^{(l)}} J(W, B)$. Use backpropagation to compute gradient of each layer.
- Use a mean-squared error (MSE):

$$J(W, B) = \frac{1}{m} \sum_{i=1}^m \left(h_{W,B} \square x^{(i)} \square y_k^{(i)} \right)^2 + \dots$$

Understanding the role of nonlinearities

- i). **(Uncertainty effect)**. *Linearized solutions do not depend on the degree of volatility σ , as do nonlinear solutions.*
 - ii). **(High-order effect)**. *Linearization method neglects high-order polynomial terms, unlike more flexible nonlinear solutions.*
 - iii). **(Solution-domain effect)**. *Perturbation (local) solutions are constructed to be accurate in a deterministic steady state, and their accuracy can deteriorate dramatically when deviating from the steady state, in particular, in the area of ELB.*
- Let's look at these three effects for perturbation solutions,

$$g(x, \sigma) \approx \underbrace{g(\bar{x}, 0) + g_x(\bar{x}, 0)(x - \bar{x})}_{\text{1st-order perturbation solution}} + \underbrace{\frac{g_{\sigma\sigma}(\bar{x}, 0)\sigma^2}{2}}_{\text{Uncertainty effect}} + \underbrace{\frac{g_{xx}(\bar{x}, 0)(x - \bar{x})^2}{2}}_{\text{High-order effect}}$$

- The second-order perturbation method addresses i) and ii) by the terms $\frac{1}{2}g_{\sigma\sigma}(\bar{x}, 0)\sigma^2$ and $\frac{1}{2}g_{xx}(\bar{x}, 0)(x - \bar{x})^2$, respectively. Plain perturbation methods do not address iii) but IRIS and OccBin do.
- Our DL solution method addresses i) - iii) more conclusively than the perturbation methods by using more flexible neural network approximation and global solution domain.

Experiment 1: Foreign-driven recession

- The U.S. is the main Canadian trade partner (around 75% of Canadian exports goes to the US).
- In 2008, the Canadian economy experienced a huge 16% drop in exports.
- In 2009–2010, the Bank of Canada targeted the overnight interest rate at 0.25% (the lower bound).

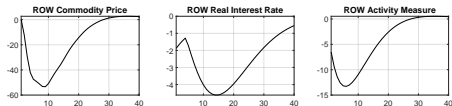
Experiment 1: Exogenous ROW shocks

An important question is how to calibrate the rest of the world (ROW) sector in the bToTEM model since foreign financial crisis affects not just foreign demand but also foreign prices and foreign interest rates.

- we use ToTEM to produce impulse responses for 3 ROW variables (interest rate, commodity price, output).
- we use them as exogenous shocks in the bToTEM model.

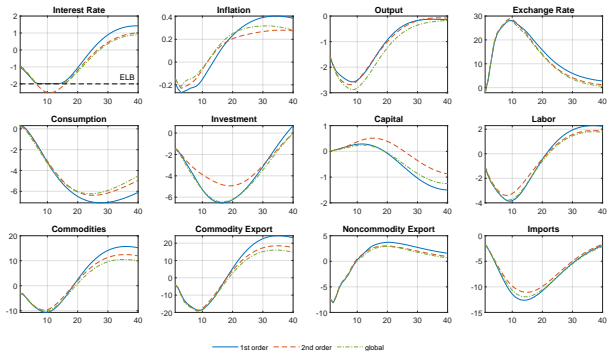
The ROW shocks generated by ToTEM is in line with the data.

- The ROW commodity price shock matches a 50 percent decline in the global commodity price index documented by the IMF.
- The ROW real interest rate shock matches a 5 percent peak-to-trough decline in the effective U.S. federal funds rate.
- The ROW activity measure shock is about 18 percent decline in the foreign activity measure estimated by the Bank of Canada.



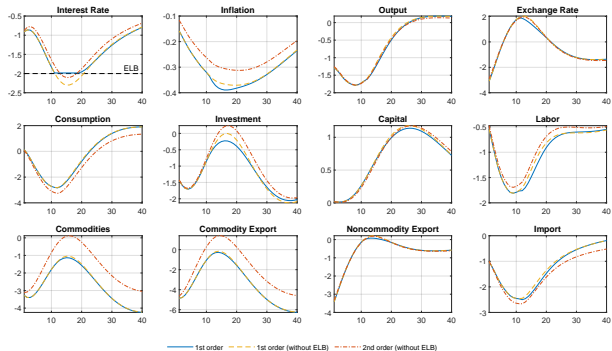
Experiment 1: Foreign-driven recession

Linear local, quadratic local and global solutions



Experiment 1: Domestic-driven recession

Linear local, quadratic local and global solutions



Approximation errors

Residuals in the model's equations on the simulated path, log 10 units

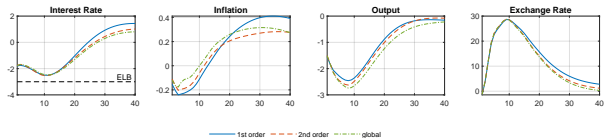
	Maximum error			Average error		
	Local 1st order	Local 2nd order	Global 2nd order	Local 1st order	Local 2nd order	Global 2nd order
R_t	-3.83	-3.84	-5.07	-4.13	-4.53	-5.74
π_t	-4.40	-3.81	-4.38	-4.65	-4.49	-5.08
s_t	-2.38	-1.96	-2.51	-3.16	-2.62	-3.24
Y_t	-2.59	-3.27	-3.42	-2.78	-3.90	-4.16
C_t	-3.19	-3.13	-3.94	-3.44	-3.79	-4.74
I_t	-3.01	-3.39	-3.51	-3.79	-4.01	-4.09
X_t^{com}	-1.75	-2.32	-2.73	-1.97	-2.88	-3.43
X_t^{nc}	-2.78	-2.36	-2.91	-3.56	-3.01	-3.64
M_t	-2.17	-2.90	-3.17	-2.41	-3.53	-3.88
Average	-2.76	-2.81	-3.41	-3.20	-3.45	-4.11
Max	-1.43	-1.44	-2.09	-1.92	-2.08	-2.75

Experiment 2: Higher inflation target

- For the last 25 years, the Bank of Canada adhered to the inflation targeting, however, every three to five years it revises their inflation-control framework.
- Current inflation target is 2%.
- A higher inflation target could be beneficial by reducing frequency and severity of ELB episodes.
- Kryvtsov and Mendes (2015), Dorich et al. (2017).
- We use the bToTEM model to assess the effects of an increase in the inflation target from 2% to 3%.

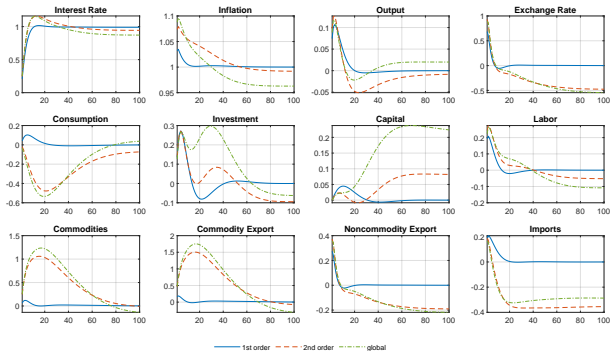
Experiment 2: Higher inflation target

A 3% inflation target could prevent ELB episodes similar to the 2009-10 episode



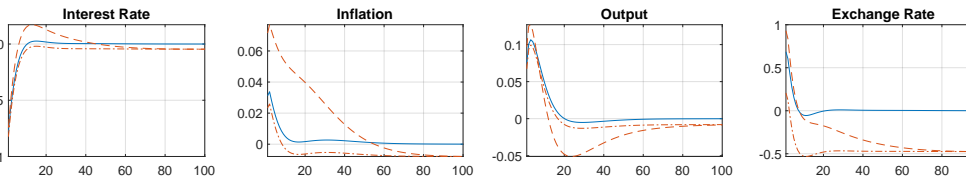
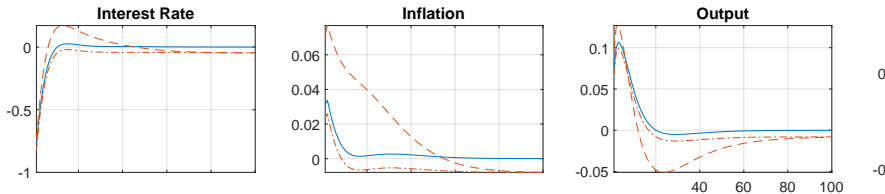
Experiment 2: Higher inflation target

Transition from the deterministic steady state with the inflation target of 2%



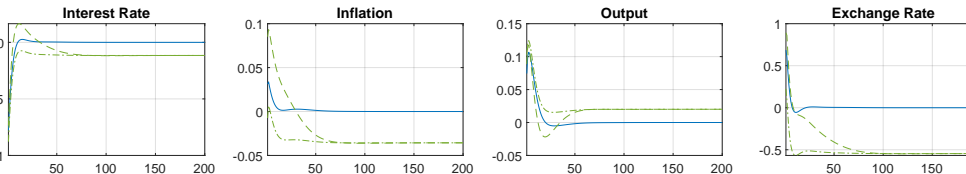
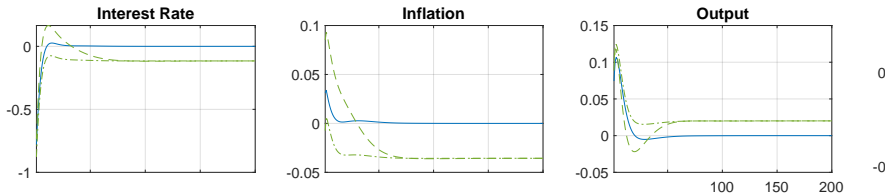
Experiment 2: Higher inflation target

Transition from the different states with the target of 2%. 2nd order



— 1st order - - - 2nd order (from deterministic steady state) - · - 2nd order (from stochastic steady state)

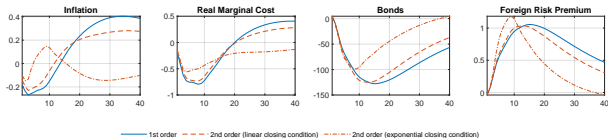
Experiment 2: Higher inflation target



— 1st order - - - global (from deterministic steady state) - · - global (from stochastic steady state)

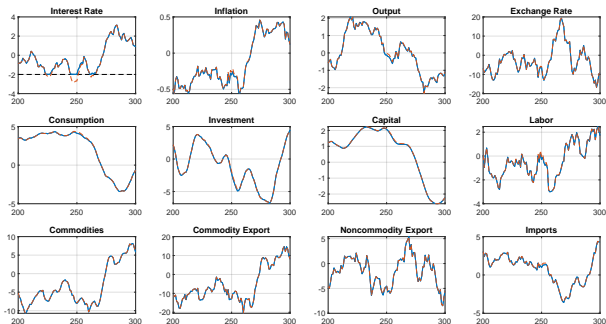
Closing condition

- We compare the closing condition in an exponential form, used in Schmitt-Grohé and Uribe (2003) $\kappa_t^f = \varsigma \left[\exp \left(\bar{b}^f \square b_t^f \right) \square 1 \right]$ and the linear form $\kappa_t^f = \varsigma \left(\bar{b}^f \square b_t^f \right)$.
- The importance of stationarity condition in the bToTEM model is surprising, given a well-known finding of Schmitt-Grohé and Uribe (2003) that the closing condition used does not significantly affect the implications of open-economy models.
- However, their analysis is obtained for linearized solutions in which the exponential and linear conditions coincide exactly but such conditions are different in our nonlinear analysis.



ELB irrelevance

- In bToTEM, the probability of reaching ELB in bToTEM is large (about 8 percent) and the average duration of ELB episodes is 5 quarters but its effect is very small.
- In Debartoli et al. (2019), the ELB constraint is irrelevant because of the availability of unconventional monetary policies (forward guidance, quantitative easing, etc.) while bToTEM, it is irrelevant due to the presence of the rule-of-thumb agents introduced to deal with the forward guidance puzzle.

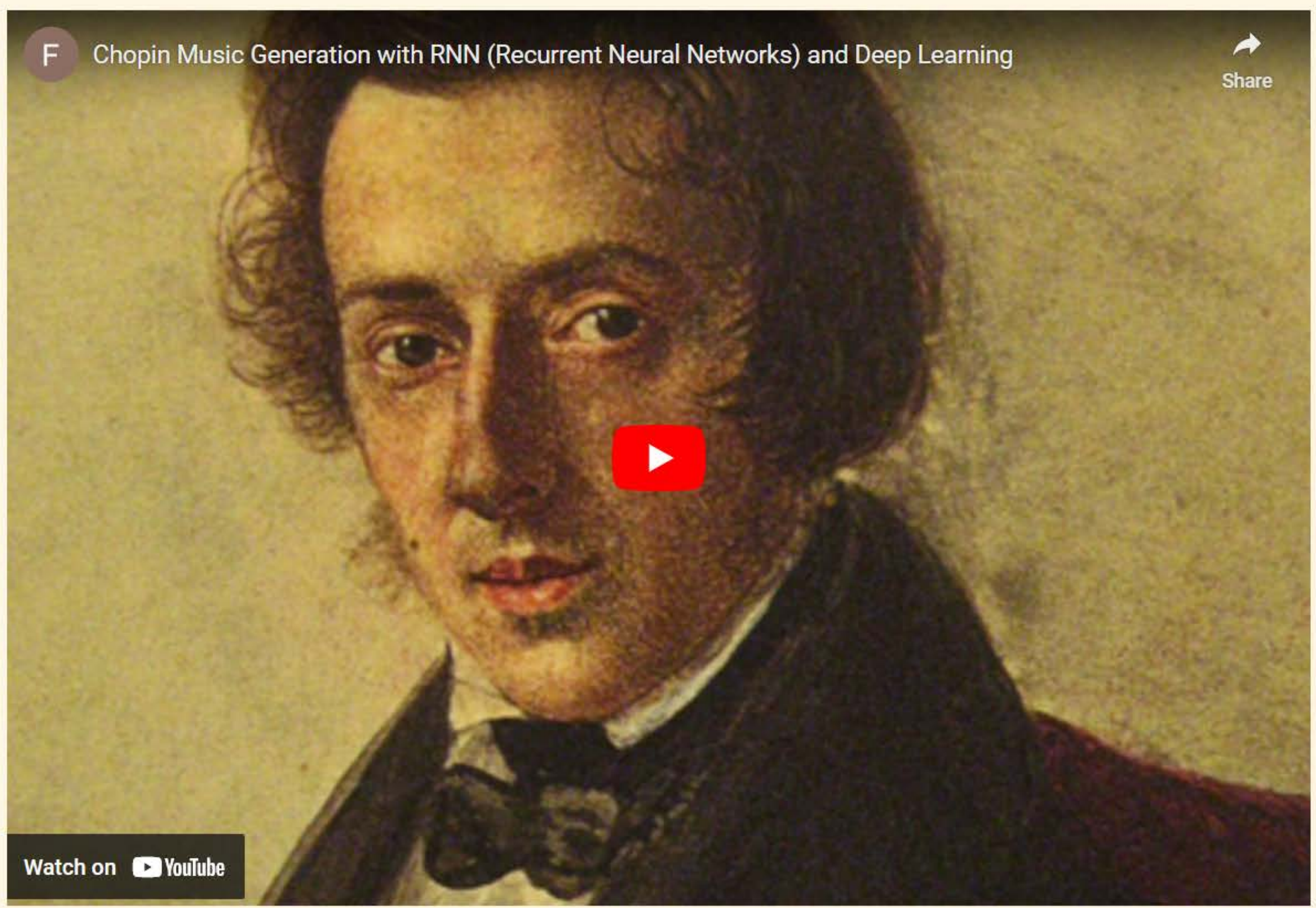


Conclusion

- Large-scale central banking models like ToTEM and bToTEM are analyzed by linearization methods.
- Our novel DL algorithm combines supervised and unsupervised learning to construct an accurate global fully nonlinear solution.
- What is the value added of deep learning for telling the Canadian ELB tale?
- It is fair to say that we could have discovered the ELB contagion mechanism by using linearization-based methods.
- But we would not know how reliable our linear solution is, and we would miss some dramatic effects of nonlinearities on the solution.
- In particular, we would overlook the uncertainty effect.
- Furthermore, we would miss the nonlinearity effects associated with the closing condition.
- On the other hand, the nonlinearity associated with the ELB turned out to be of a lesser importance than we expected.

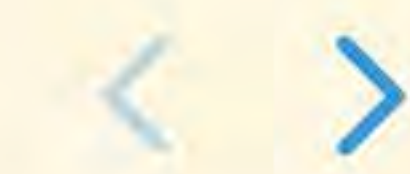
Conclusion (cont.)

- The bToTEM model constructed in the paper is a useful alternative model to the Bank of Canada.
- While the full-scale ToTEM is not yet feasible for global nonlinear methods, bToTEM can be solved nonlinearly and its accuracy can be assessed.
- In addition to the Bank of Canada, our deep learning analysis can be useful to all users of large-scale models, including researchers, central banks and government agencies who can benefit from our methodology of calibrating, solving, and simulating large-scale macroeconomic models, as well as designing nontrivial policy experiments within such models.



WILL ARTIFICIAL INTELLIGENCE REPLACE COMPUTATIONAL ECONOMISTS ANY TIME SOON?

Lilia Maliar, Serguei Maliar and Pablo Winant





INTRODUCTION



ARTIFICIAL INTELLIGENCE VERSUS ECONOMIC AGENT

- AI is a device that:
 - has a goal
 - perceives its environment
 - takes actions to achieve its goal
- This looks exactly like an economic agent!
 - goal = utility maximization, profit maximization, etc.
 - environment = state variables
 - action = controls
- So, maybe, general-purpose AI technology can solve the economic models for us



APPLICATIONS: AI VERSUS ECONOMICS

- AI has many impressive applications:
 - recognition of handwriting and speech
 - composing Chopin music, playing Go
 - facilitating computer vision
- At the same time, economists cannot solve:
 - heterogeneous-agent and life-cycle models
 - dynamic games and IO models
 - expensive nonlinear estimation
- So, maybe, machines can solve for us those difficult problems that we want but cannot currently solve



APPROACH IN THIS PAPER

- We do not invent a new computational method for economic models but reformulate the models themselves to fit into the existing state-of-the-art computational methods
- Specifically, we adapt deep learning (DL) framework built on multilayer neural networks with stochastic optimization to solving dynamic economic models
- Our code is written using Google TensorFlow platform - the same software that leads to break-ground applications in data science



THREE MAIN RESULTS

1. We offer a unified approach for casting three fundamental objects of economic dynamics - **lifetime reward, Bellman equation, Euler equation**- into DL objective functions
2. We show how to solve economic models on **random grids** with few grid points (instead of conventional fixed grids with a large set of points)
3. We introduce "**all-in-one expectation operator**" that merges all randomness into a single operator and allows to simultaneously implement both approximation and integration steps



THE APPLICATIONS

1. Consumption-saving problem with borrowing constraints via Fisher-Burmeister function (code is available at QuantEcon.org)
2. Krusell and Smith (1998) model with 1,000 agents: we do not use moments of distributions but construct decision functions of 2,001 state variables, literally feeding the entire distributions into neural networks
3. Large-scale new Keynesian model of the Bank of Canada (in another paper)



Type here to search



4:33 AM

11/18/2022



4

MODEL REDUCTION AND ILL-CONDITIONING

- Neural network automatically performs *model reduction!*
 - It learns to extract information from many inputs and condenses it into a smaller set of hidden layers
- Moreover, neural network can deal with *ill conditioning*
 - It learns to ignore redundant and collinear variables
- Taken together, these features allow us to deal with a huge state space of 2,001 variables



RELATION TO COMPUTATIONAL ECONOMICS

- The lifetime-reward maximization method is related to an indirect inference procedure of Smith (1987)
- The Euler-equation method is related to seminal contributions to numerical solution methods in economics: the projection method of Judd (1992) and PEA of Den Haan and Marcet (1990)
- The Bellman-equation method is related to conventional value and policy function iteration (see e.g., Rust, 1996, Judd, 1998, Santos, 1999, Aruoba et al., 2006, Stachurski, 2009)



RELATION TO THE LEARNING LITERATURE

- *Supervised Learning*: Duffy and McNelis (2001), Duarte (2018), Fernandez-Villaverde et al. (2018), Villa and Valaitis (2019), Lepetyuk et al. (2019), Azinovic et al. (2019)
- *Unsupervised Learning*: Judd et al. (2011), Maliar and Maliar (2015), Coleman et al. (2018)
- *Reinforcement Learning*: Sutton and Barto (2018), Powell (2010), Lepetyuk and Jirniy (2012)

**We differ in that we cast the entire economic model
into DL objective function**



PLAN OF THE TALK

- Canonical supervised learning
- Deep learning and neural networks
- Stochastic optimization
- Deep learning algorithm for optimization
- Casting economic models into DL objective functions
- Consumption-saving model
- Krusell and Smith (2019) model
- Conclusion



CANONICAL SUPERVISED LEARNING

CANONICAL SUPERVISED LEARNING = JUST A REGRESSION

- Output data: $y_i \in \mathbb{R}^{d_y}$
- Input data (features): $x_i \in \mathbb{R}^{d_x}$
- Approximation (prediction) function:
$$\varphi : \mathbb{R}^{d_x} \rightarrow \mathbb{R}^{d_y}$$
- *Goal:* to learn φ s.t. given $x_i \in \mathbb{R}^{d_x}$, the value $\varphi(x_i)$ provides an accurate prediction about true y_i
- The function φ is selected within a given family of parametric functions $\{\varphi(\cdot; \theta) : \theta \in \mathbb{R}^{d_\theta}\}$, $\theta =$ parameters vector.

EXPECTED LOSS

- *Loss function* $\ell : \mathbb{R}^{d_y} \longrightarrow \mathbb{R}$ = difference between the true output y and the predicted output $\varphi(x; \theta)$.
- Minimize the *expected loss (expected risk)* $\Xi(\theta)$:

$$\begin{aligned}\Xi(\theta) &\equiv \int_{\mathbb{R}^{d_x} \rightarrow \mathbb{R}^{d_y}} \ell(\varphi(x; \theta), y) dP(x, y) = \\ &= E_x[\ell(\varphi(x; \theta), y)];\end{aligned}$$

$P : \mathbb{R}^{d_x} \times \mathbb{R}^{d_y} \longrightarrow [0, 1]$ = a joint probability distribution function



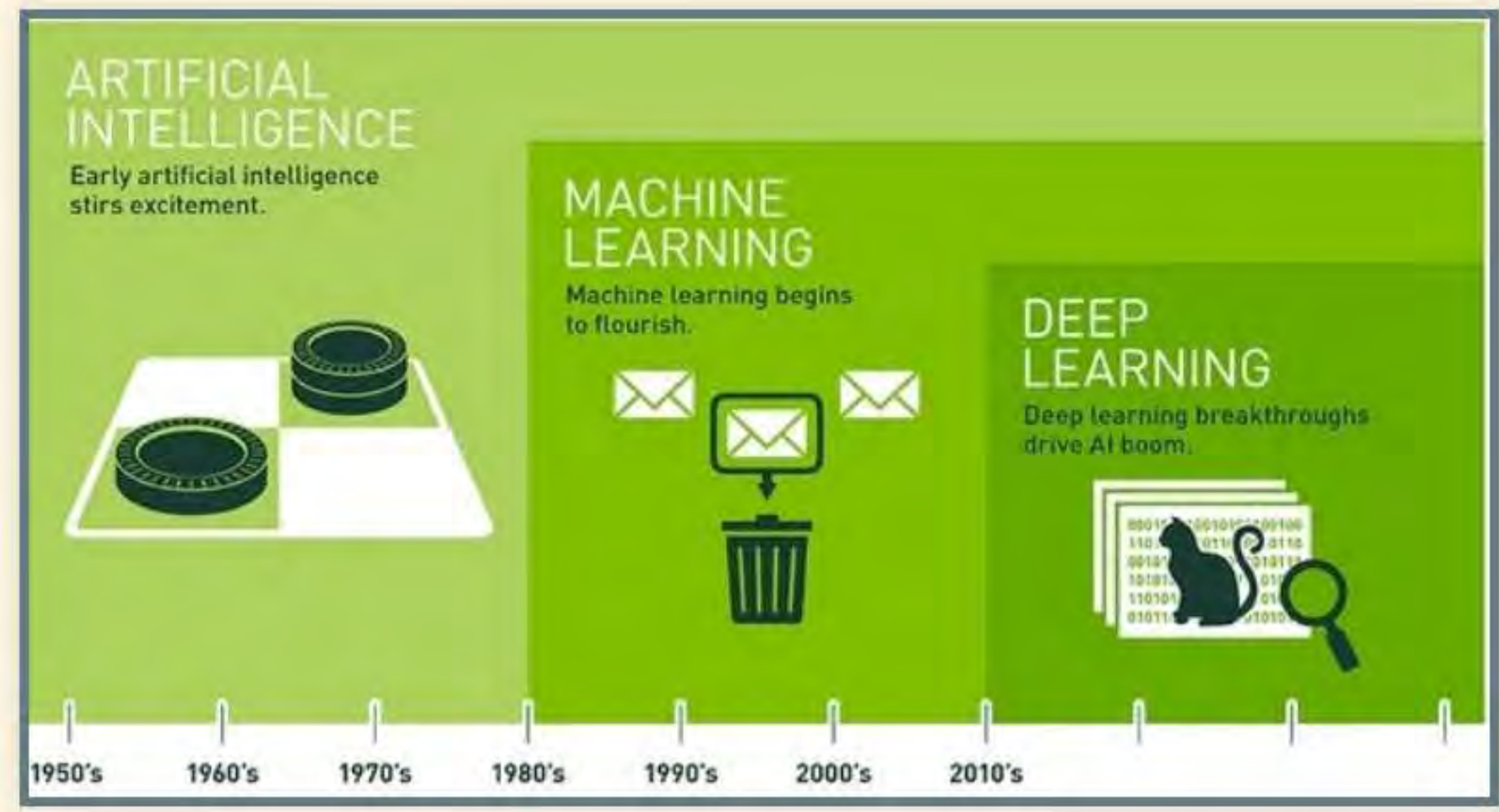
EMPIRICAL LOSS

- In practice, learn θ that minimizes *empirical loss*:
$$\Xi_n(\theta) \equiv \frac{1}{n} \sum_{i=1}^n \ell(\varphi(x_i; \theta), y_i),$$
 where $\{(x_i, y_i)\}_{i=1}^n$ is a given set of input-output pairs
- *Example: linear regression* $y = \theta x$, where
 - $\varphi(x; \theta) = \theta x$ is the prediction function and
 - $\ell(\theta x, y) = (y - \theta x)^2$ is the loss function



BACKGROUND ON DEEP LEARNING

AI, ML AND DL



- Artificial neural networks
 - invented in 50s
 - widely used in 80s and early 90s; less so in 90s
 - recent resurgence: state-of-the-art technique

DEEP NEURAL NETWORKS

- DL models are based on an artificial neural network
- in DL, each level learns to transform its input data into a slightly more abstract and composite representation
- the "deep" in "deep learning" indicates that there are *multiple* layers through which the data are transformed
- endowing neural networks with multiple layers is what precisely led to the break-through in the field of DL



GOOGLE TRENDS

- the word "deep learning" on Google Trends

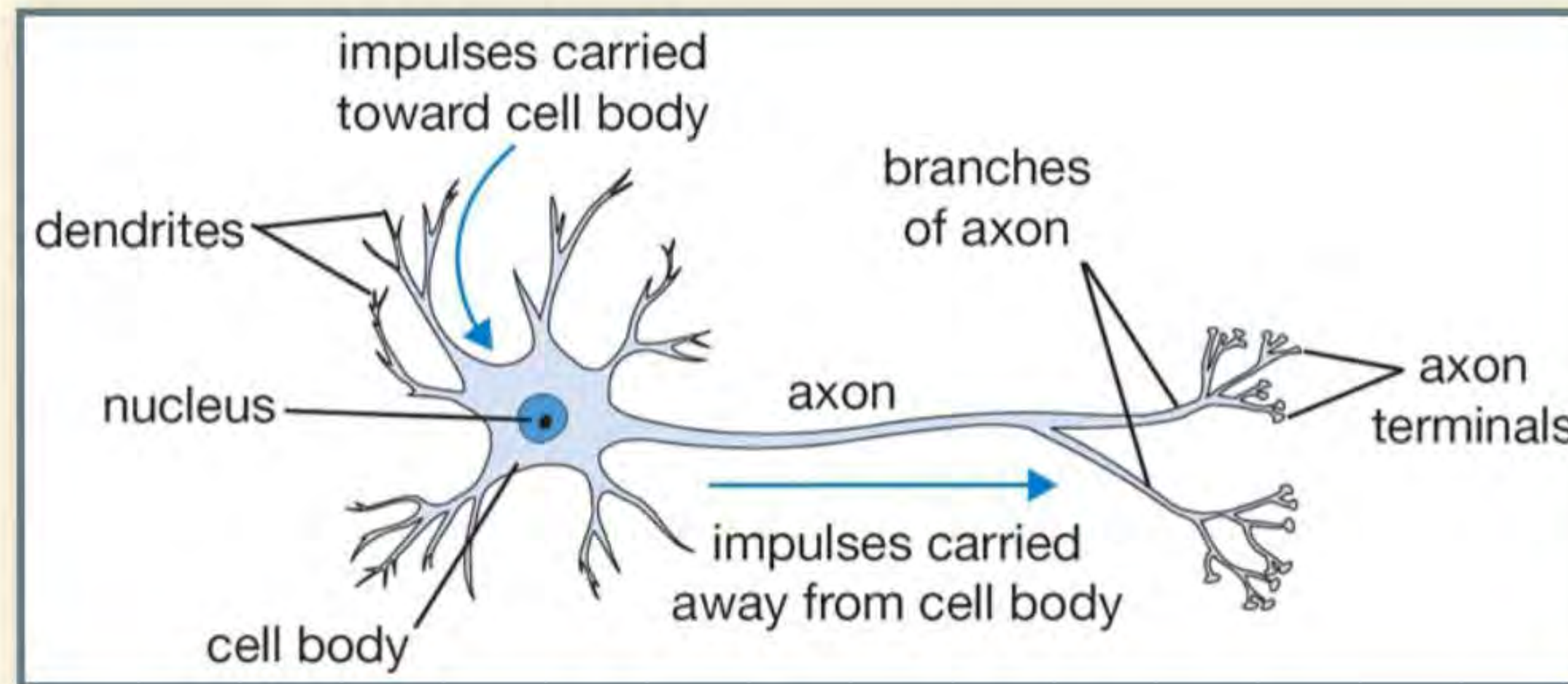


- software "Alpha-Go" defeated the South Korean champion Lee Se-dol in the game Alpha-Go in March 2016



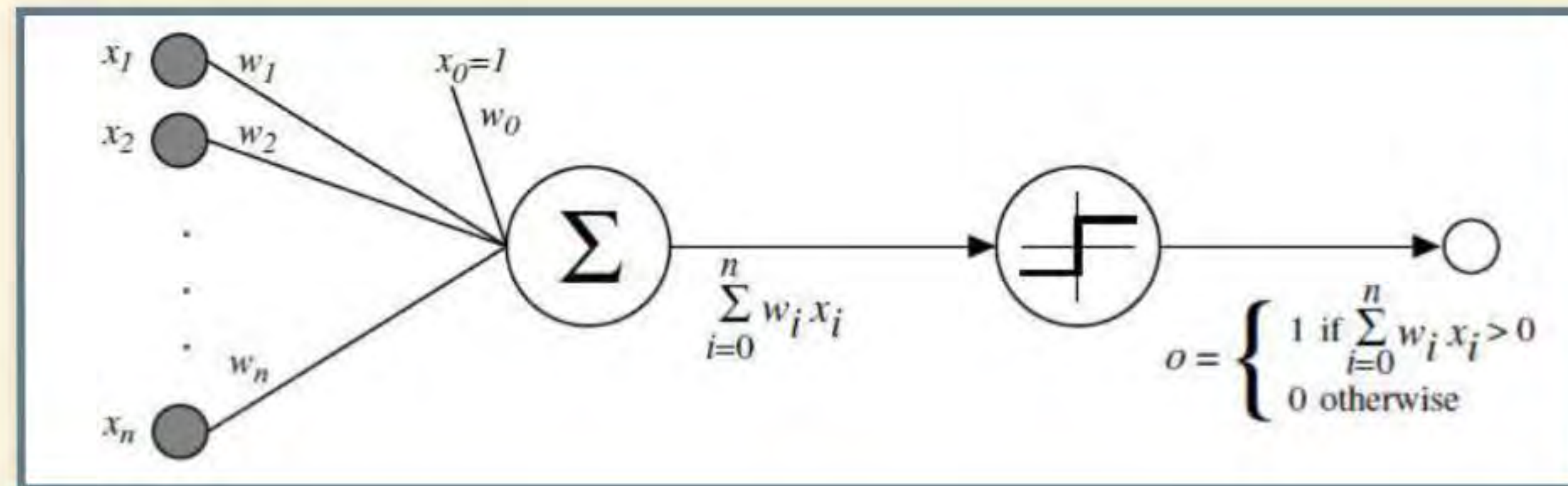
BIOLOGICAL NEURON

- algorithms trying to mimic neurons in the brain



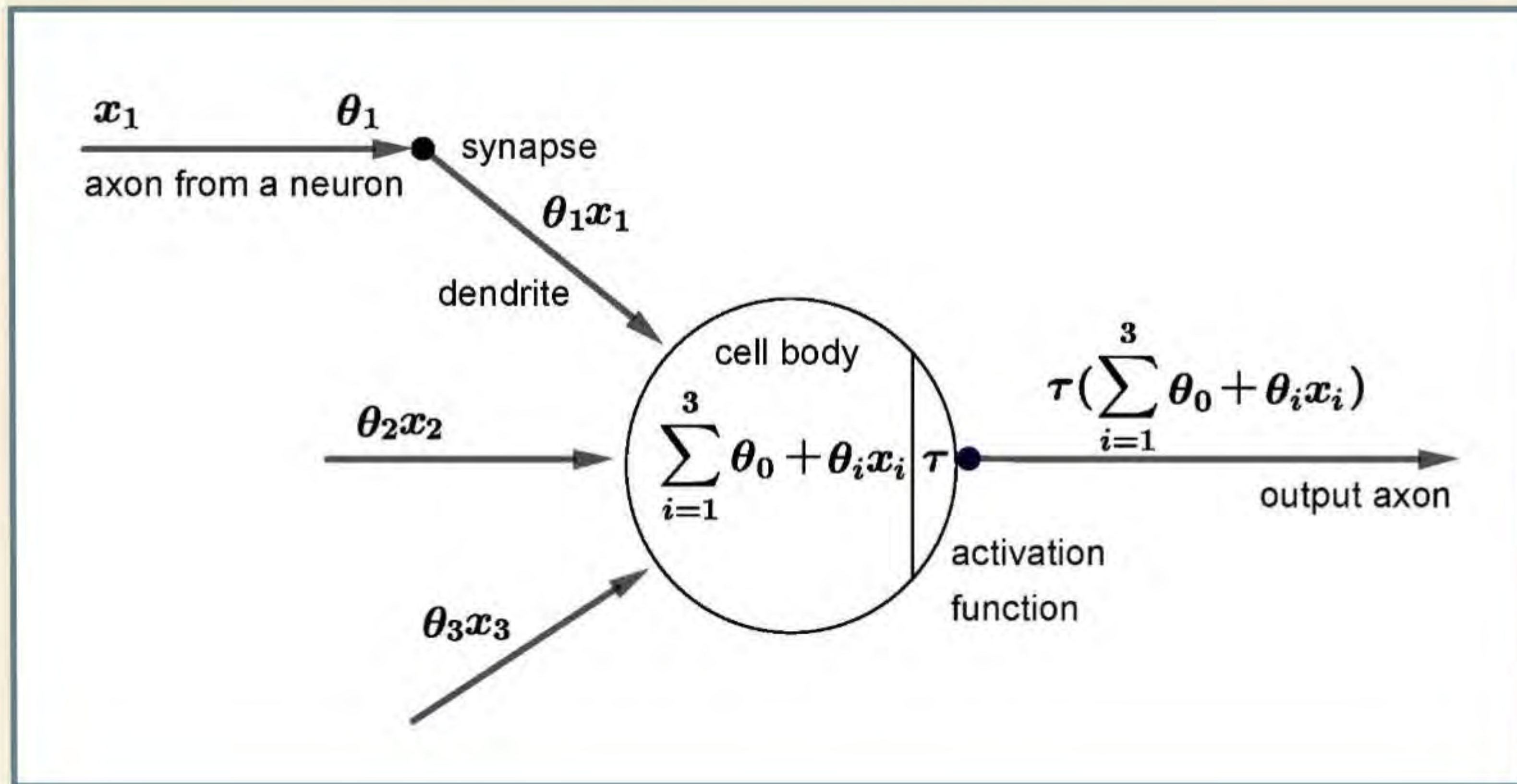
EARLY ARTIFICIAL NEURON

- a perceptron - the first simplified model of a biological neuron
- invented by a psychologist Frank Rosenblatt (1957)



MODERN ARTIFICIAL NEURON

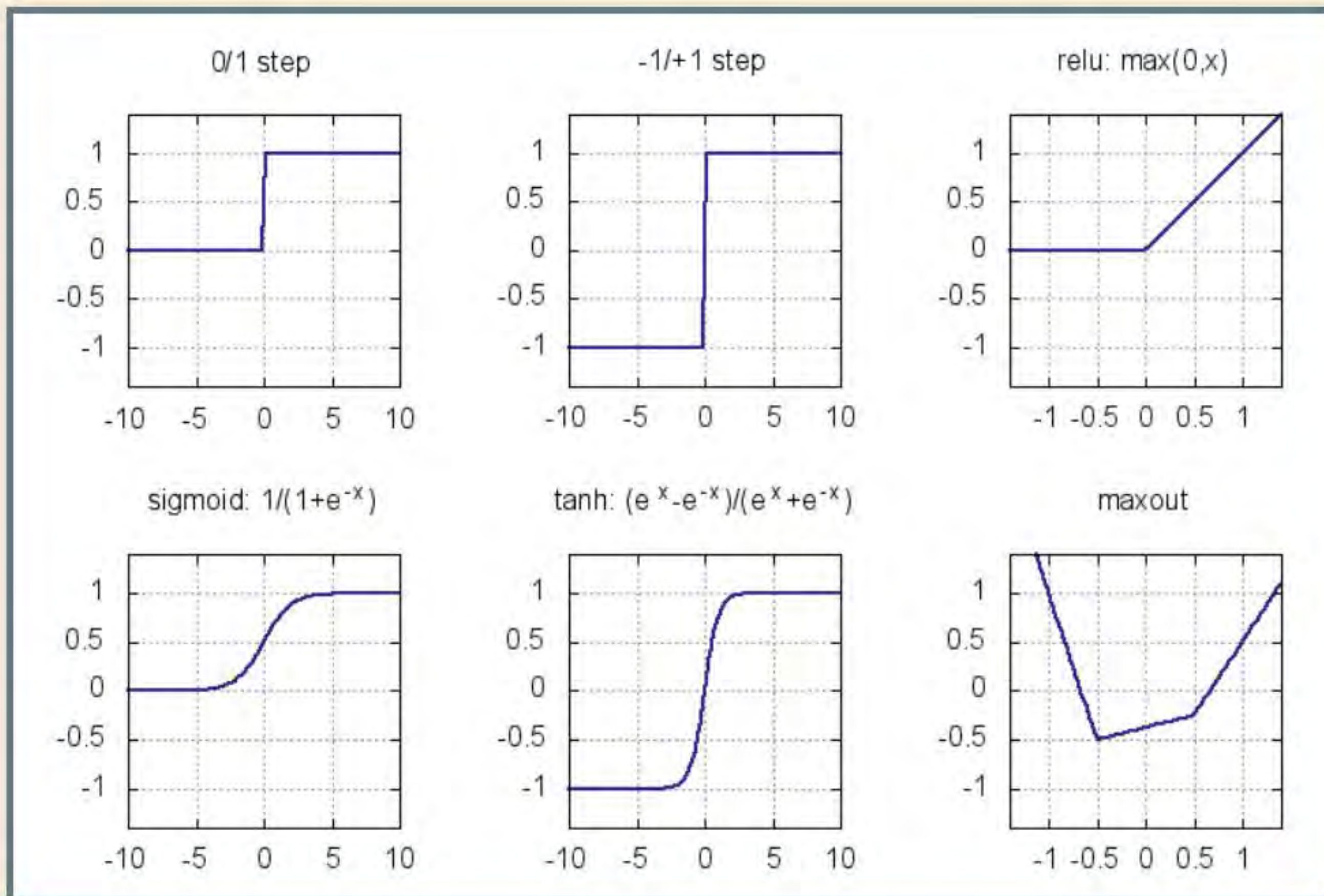
- arbitrary activation function τ



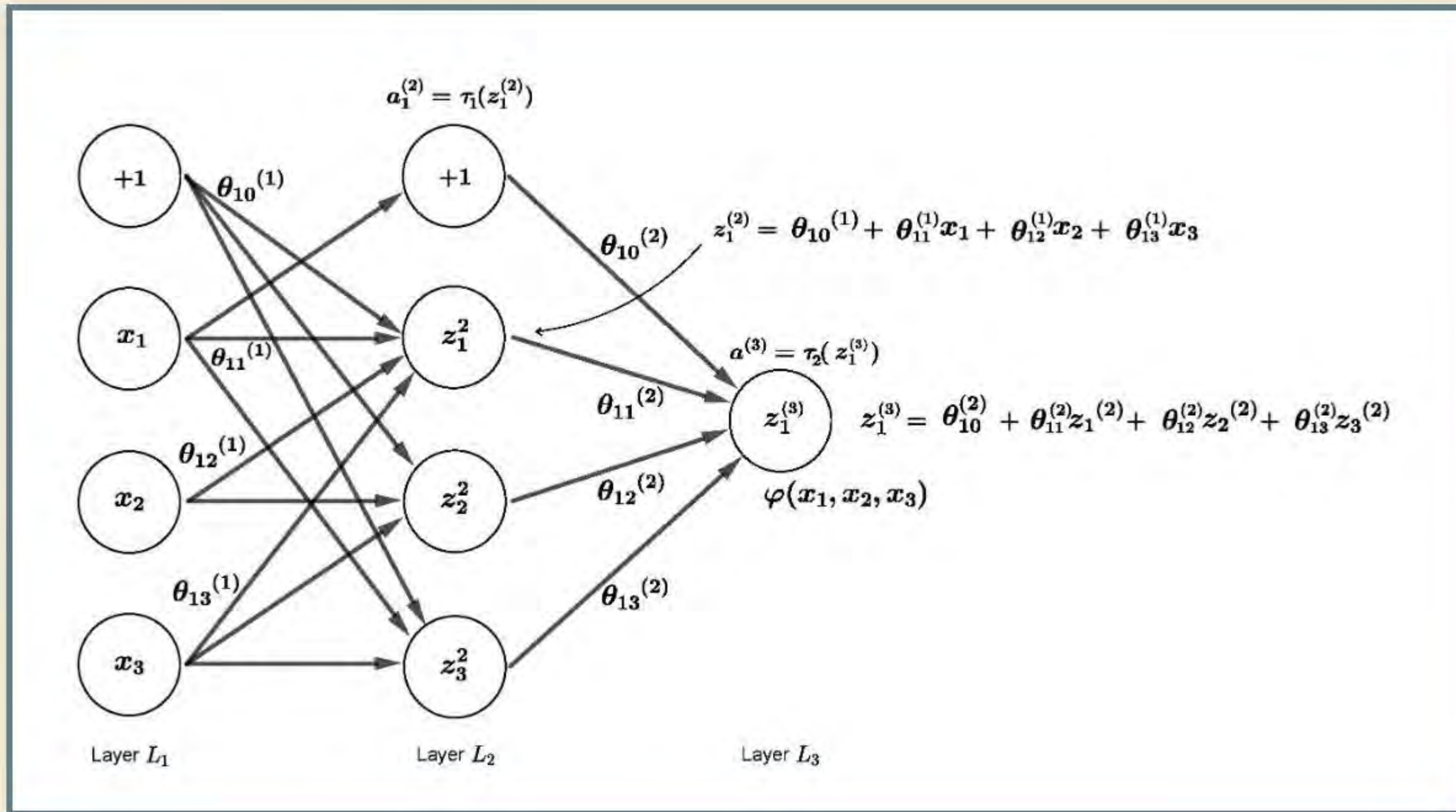
ACTIVATION FUNCTIONS

- heaviside step function: $\tau(x) = 1_{x \geq 0}$
- sigmoid (logistic): $\sigma(x) = \frac{1}{1+e^{-x}}$
 - fact: $\sigma'(x) = \sigma(x)(1 - \sigma(x))$
- tanh (hyperbolic tangent): $\tau(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$
- relu (rectified linear units): $r(x) = \max(0, x)$
- leaky relu: $r(x) = \max(\kappa x, x), \kappa \geq 0$
- maxout:
$$\tau(x) = \max(a_1 x + b_1, a_2 x + b_2, a_3 x + b_3)$$

ACTIVATION FUNCTIONS



NEURAL NETWORK: EXAMPLE



NEURAL NETWORK: NOTATION

- parameters: $(\theta^{(1)}, \theta^{(2)}) \equiv \theta$;
- $\theta_{ij}^{(l)}$: weight associated with the connection between neuron j in layer l and the neuron i in layer $l + 1$, for $i, j = 0, 1, 2, 3$, and $l = 1, 2$;
- $\theta^{(l)}$: matrix made by the weights between the neurons of layer l and the next layer $l + 1$;
- $z_i^{(l)}$: nonactivated output of neuron i in layer l ;
- π_l : activation function of layer l ;
- $a_i^{(l)}$: activated output of neuron i in layer l .



NEURAL NETWORK: FORWARD PROPAGATION

- input $x = (1, x_1, x_2, x_3)$, $a^{(1)} = x$
- $z_1^{(2)}$: nonactivated output of neuron 1 in layer 2
 - $z_1^{(2)} = \theta_{10}^{(1)} + \theta_{11}^{(1)} x_1 + \theta_{12}^{(1)} x_2 + \theta_{13}^{(1)} x_3$
- $a_1^{(2)}$: activated output of neuron 1 in layer 2
 - $a_1^{(2)} = \tau_1(z_1^{(2)})$
 - or $a_1^{(2)} = \tau_1(\theta^{(1)} x)$
- $a^{(3)}$: activated output in layer 3
 - $a^{(3)} = \tau_2(z_1^{(3)})$
 - or $a^{(3)} = \tau_2(\theta^{(2)} a^{(2)})$
- $a^{(3)} = \varphi(x; \theta)$: predicted output



BACKGROUND ON TRAINING OF MACHINES

TRAINING NEURAL NETWORKS

- Training of the machine = find the parameters of the approximating function
- $\Xi(\theta) = E_{\omega}[\xi(\omega; \theta)]$, and $\Xi_n(\theta) = \frac{1}{n} \sum_{i=1}^n \xi(\omega_i; \theta)$,
 $\{\omega_i\}_{i=1}^n$ = the given set of input-output pairs
 $\{(x_i, y_i)\}_{i=1}^n$
- define empirical risk $\Xi(\theta)$ by MSE:
$$\Xi_n(\theta) = \frac{1}{n} \sum_{i=1}^n (\varphi(x; \theta) - y)^2$$
- $\Xi_n(\theta)$ allows us to evaluate how different $\varphi(x; \theta)$ is from true y
- need to compute the gradient $\nabla_{\theta} \Xi_n$



BACKPROPAGATION: IDEA

- To compute the gradient, use backpropagation:
- perform forward propagation to calculate all the parameters θ
- calculate an error term backward for each neuron, starting from the output layer towards the input layer
- assess how much each neuron can be held responsible for a potential error in the network's output



STOCHASTIC GRADIENT

- *Gradient Descent:* (learning rate λ)

$$\theta_{k+1} \leftarrow \theta_k - \lambda_k \nabla_{\theta} \mathbf{E}_{\omega} [\xi(\omega; \theta_k)]$$

- *Stochastic Gradient Descent:*

$$\theta_{k+1} \leftarrow \theta_k - \lambda_k \nabla_{\theta} \xi(\omega; \theta_k)$$

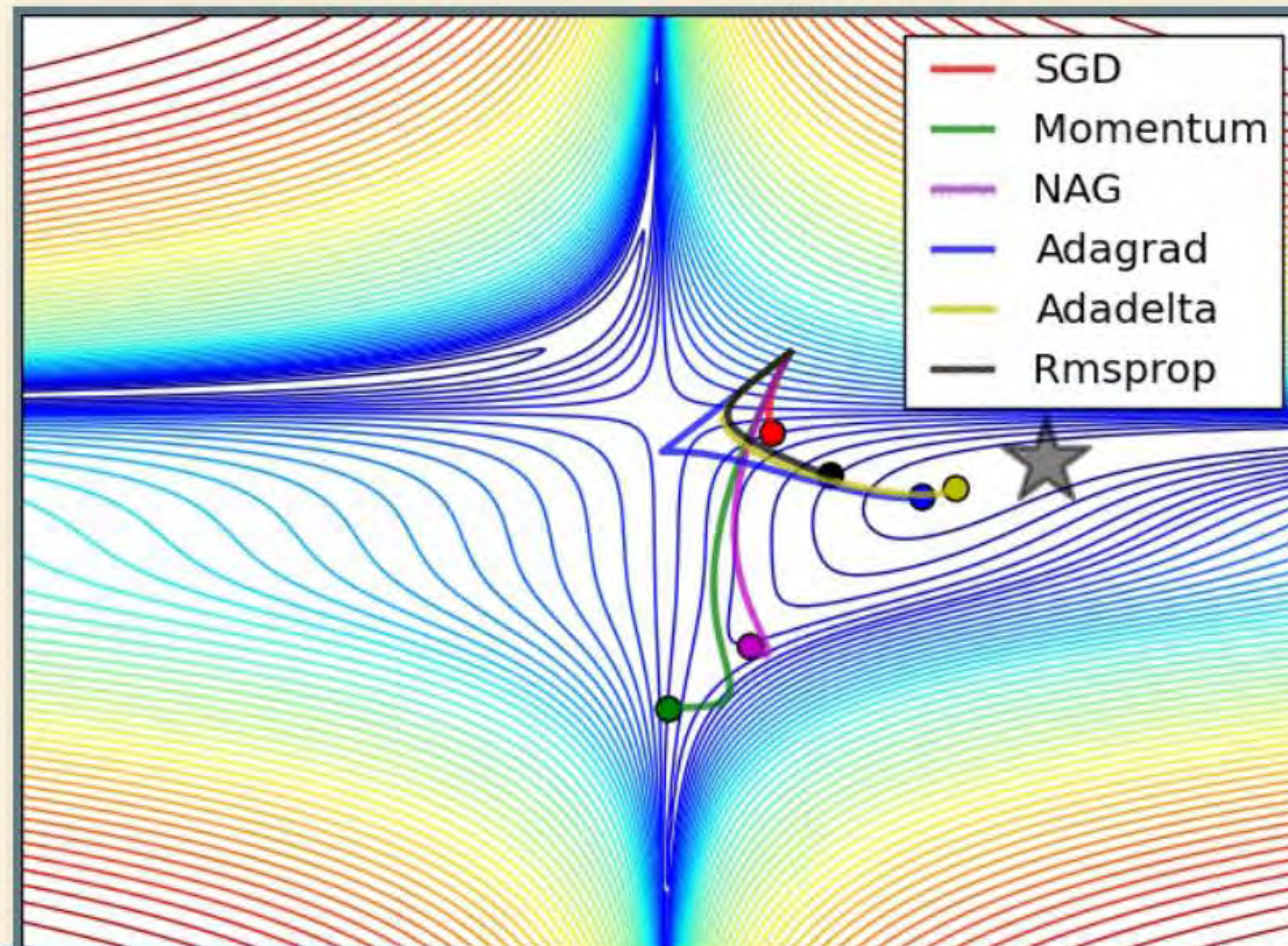
- *Batch Stochastic Gradient Descent:*

$$\theta_{k+1} \leftarrow \theta_k - \frac{\lambda_k}{n} \sum_{i=1}^n \nabla_{\theta} \xi(\omega_i; \theta_k)$$



GRADIENT DESCENT

- Goal: minimize theoretical risk $\Xi(\theta) = E_{\omega} [\xi(\omega; \theta)]$
- *Gradient Descent*: $\theta_{k+1} \leftarrow \theta_k - \lambda_k \nabla_{\theta} E_{\omega} [\xi(\omega; \theta_k)]$



A DEEP LEARNING ALGORITHM FOR OPTIMIZATION

DECISION FUNCTIONS ARE IMPLICIT IN ECONOMICS

- Canonical supervised learning is designed to estimate a regression $y = \varphi(x; \theta)$
- It is not directly suited for solving economic models where we need to find value/policy functions $\varphi(x; \theta)$ because the value of y is unknown
- For example, if we approximate the capital policy function $k' = \varphi(k; \theta)$, we don't know what k' is
- Therefore, we generalize canonical supervised learning to the case when y is defined implicitly



GENERALIZATION OF CANONICAL SUPERVISED LEARNING

- Introduce new variable $\omega \equiv (x, y)$
- Then, loss function is $\xi(\omega; \theta) \equiv \ell(\varphi(x; \theta), y)$
- This change of variables converts supervised learning into a generic optimization of expectation functions
- It also provides a link to unsupervised learning!



AI ALGORITHM FOR OPTIMIZATION

1. Define theoretical and empirical risks

$$\Xi(\theta) = E_{\omega}[\xi(\omega; \theta)], \text{ and } \Xi_n(\theta) = \frac{1}{n} \sum_{i=1}^n \xi(\omega_i; \theta),$$

$\{\omega_i\}_{i=1}^n$ = the given set of features (data points).

2. Define a family of parametric functions

$$\{\varphi(\cdot; \theta) : \theta \in \mathbb{R}^{d_{\theta}}\}, \theta = \text{parameters vector.}$$

3. Train the machine, i.e., find θ that minimizes the empirical risk, typically using stochastic gradient or batch stochastic gradient methods.
4. Check accuracy of the constructed approximation.

We are just left to frame the economic model as an objective for the empirical risk!



CONSUMPTION-SAVING PROBLEM

OBJECTIVE FUNCTIONS FOR DL

Three fundamental objects of economic dynamics:

- lifetime reward of a consumer (to be maximized)
- residuals in Bellman equation (to be minimized)
- residuals in Euler equation (to be minimized)

We show how to convert these three objects into the objective functions of DL algorithm



CONSUMPTION-SAVING PROBLEM WITH MULTIPLE SHOCKS

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

$$\text{s.t. } w_{t+1} = r e^{\varrho_t} (w_t - c_t) + e^{y_t} e^{p_t},$$

$$c_t \leq w_t,$$

(z_0, w_0) given.

- c_t = consumption; w_t = cash-on-hand; $r \in (0, \frac{1}{\beta})$.
- each $z_t \in \{y_t, p_t, \varrho_t, \chi_t\}$ follows an AR(1) process:
$$z_{t+1} = \rho z_t + \sigma \epsilon_{t+1}.$$

OPTIMALITY CONDITIONS

- *Kuhn-Tucker conditions:* $c - w \leq 0, h \geq 0$ and $(c - w)h = 0, h \equiv u'(c)e^{x-\varrho} - \beta r E[u'(c')e^{x'}] =$
Lagrange multiplier.

- *Bellman equation:*

$$V(z, w) = \max_{c, w'} \{u(c) + \beta E_{\epsilon}[V(z', w')]\}$$

CALIBRATION

- $u(c) = \frac{c^{1-\gamma}-1}{1-\gamma}, \gamma = 2$
- $\beta = 0.9, r = 1.04, \sigma_y = 0.1$
- Baseline unishock model: a temporary i.i.d. income shock y
- Multi-shock model: $\rho_y = 0.9$ and $\sigma_y = 0.1$;
 $\rho_p = 0.999$ and $\sigma_p = 0.001$; $\rho_\varrho = 0.2$ and
 $\sigma_\varrho = 0.001$; $\rho_\chi = 0.9$ and $\sigma_\chi = 0.01$

COMPUTATIONAL DETAILS

- Our code is in Python and Google's TensorFlow.
- We use Dolo interface written by Pablo Winant.
- *Decision function:*
 - parameterize $\frac{c_t}{w_t} \equiv \zeta_t = \sigma(\zeta_0 + \varphi(z_t, w_t; \theta))$,
 - $\varphi(\cdot)$ = neural network with 2 hidden layers,
 - $\sigma(x) = \frac{1}{1+e^{-x}}$ = sigmoid function,
 - hidden layers include 32 (leaky) relu neurons.
- *Value function:*
 - Parameterize $V_t = V_0 + \varphi(z_t, w_t; \theta)$,
 - $\varphi(\cdot)$ = neural network.



COMPUTATIONAL DETAILS (CONT.)

- *Training*: 16 random draws & $K = 50,000$ iterations using
 - a mini-batch gradient descent (GD) method;
 - ADAM - a version of the stochastic GD algorithm.
- *Accuracy*:
 - 1,024 random draws;
 - 10-node Gauss-Hermite quadrature rule for the unishock baseline case;
 - 100-node Monte Carlo integration method for the multishock case.



OBJECTIVE 1: LIFETIME REWARD MAXIMIZATION

- Parameterize consumption function $c(\cdot; \theta)$
- For a random vector $\omega \equiv (z_0, w_0, \epsilon_1, \dots, \epsilon_T)$ construct the expectation operator:

$$\Xi(\theta) \equiv E_{\omega} \left[\sum_{t=0}^T \beta^t u(c(z_t, w_t; \theta)) \right]$$

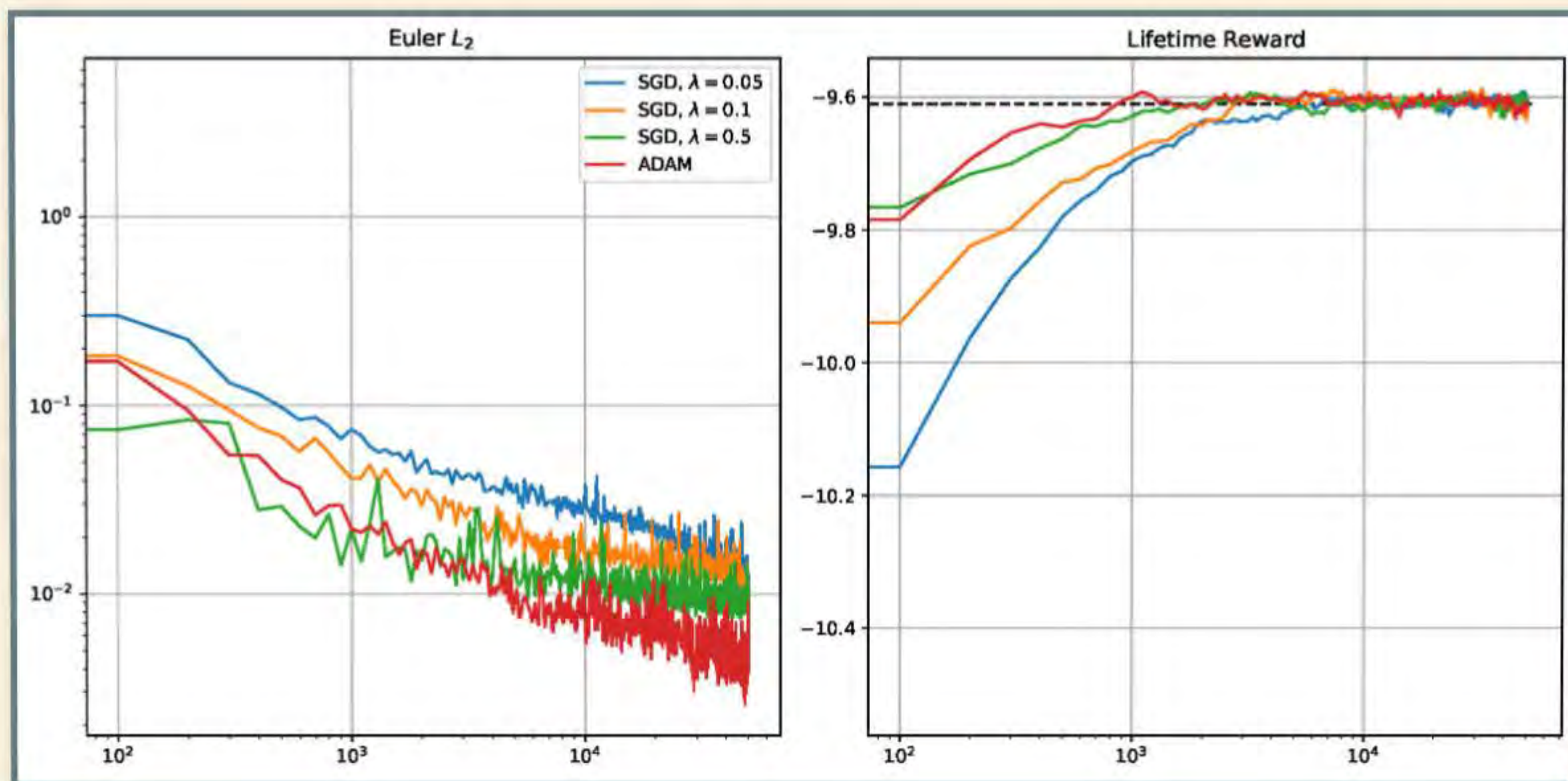
- E_{ω} has two types of randomness: a random state (m_0, s_0) and random future shocks $(\epsilon_1, \dots, \epsilon_T)$
- We call E_{ω} *all-in-one expectation operator*: it summarizes all random variables in one place

ALL-IN-ONE EXPECTATION OPERATOR

- A remarkable *distributive property* of
$$\Xi(\theta) = E_{(m_0, s_0, \epsilon_1, \dots, \epsilon_T)} \left[\sum_{t=0}^T \beta^t u(c(z_t, w_t; \theta)) \right]$$
- The same n random draws are used to evaluate all the expectation functions at once
- In contrast, an integration method that constructs 2 expectation functions $E_{(m_0, s_0)} [\cdot]$ and $E_{(\epsilon_1, \dots, \epsilon_T)} [\cdot]$ separately would require n draws for evaluating 1st expectation and n' draws for evaluating 2nd expectation -- $n \times n'$ draws in total



OBJECTIVE 1: LIFETIME REWARD MAXIMIZATION (CONT.)

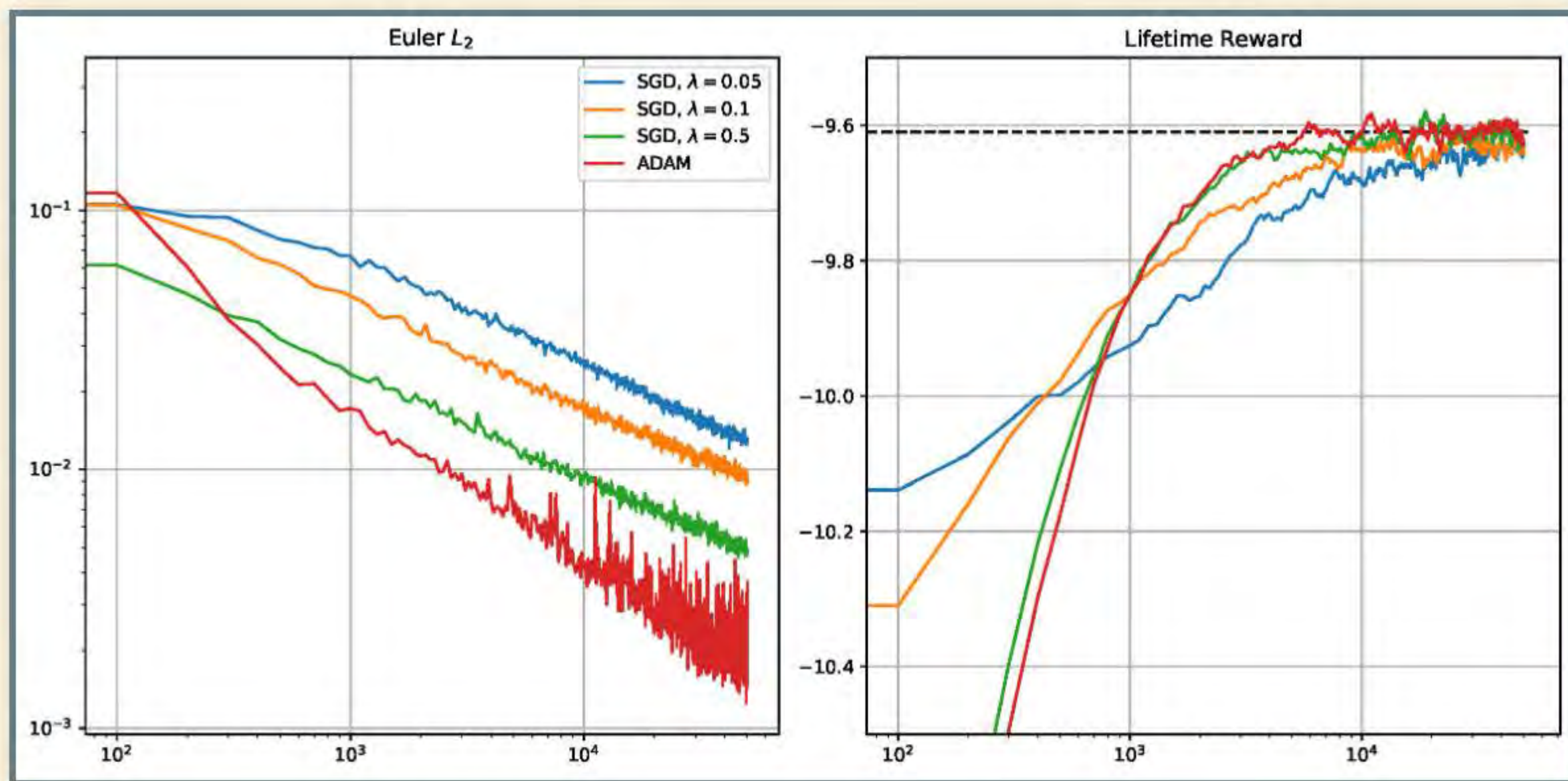


OBJECTIVE 2: EULER-EQUATION METHOD WITH KUHN-TUCKER CONDITIONS

- *Fischer-Burmeister (FB) function:*
 - $\Psi^{FB}(a, b) = a + b - \sqrt{a^2 + b^2} = 0$
 - with $a = c - w$ and $b = u'(c) - \beta r e^{\rho t} E_{\epsilon}[u'(c')]$
 - represents Kuhn-Tucker conditions $a \geq 0, b \geq 0$ and $ab = 0$ but differentiable
- *Objective function:* For a random vector (z, w) construct the expectation operator
$$\Xi(\theta) \equiv E_{(z, w)} [\Psi^{FB}(c - w, u'(c) - \beta r e^{\rho t} E_{\epsilon}[u'(c')])]^2$$
- Not all-in-one-expectation operator yet!



OBJECTIVE 2: EULER-RESIDUAL MINIMIZATION



OBJECTIVE 3: BELLMAN-RESIDUAL MINIMIZATION

- Bellman equation contains the maximum operator to be constructed within the main iterative cycle
- We propose to deal with this operator by combining Bellman residuals minimization with the max operator in a single objective function:
- Select $V(\cdot; \theta_1)$ and $c(\cdot; \theta_2)$, draw (z, w) and define
$$\Xi(\theta) = E_{(z,w)} [V(z, w; \theta_1) - u(c) - \beta E_\varepsilon V(z', w'; \theta_1)]^2 - v E_{(z,w)} [u(c) + \beta E_\varepsilon V(z', w'; \theta_1)]$$
- This is also not all-in-one-expectation operator!



Type here to search

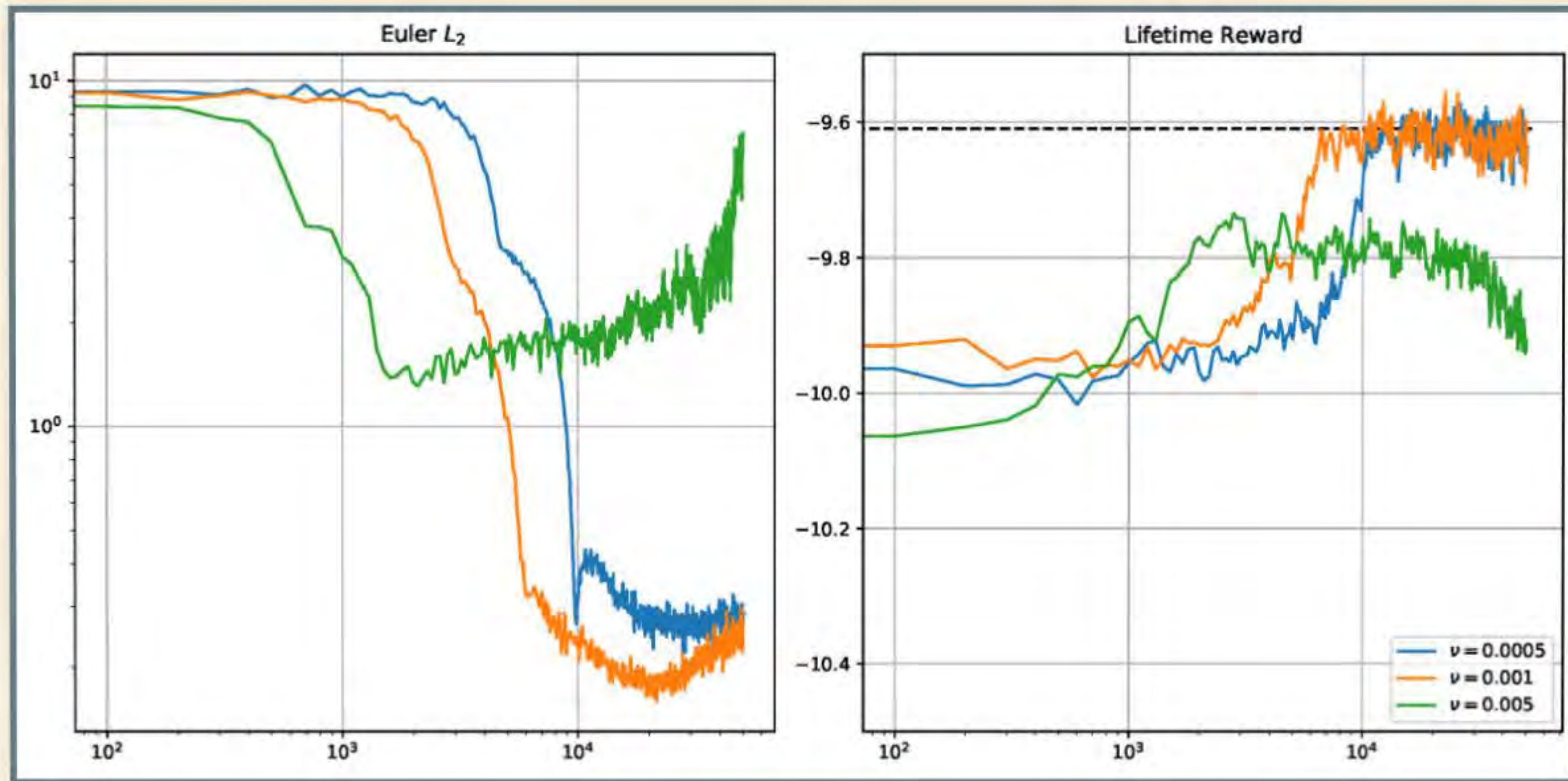


4:34 AM

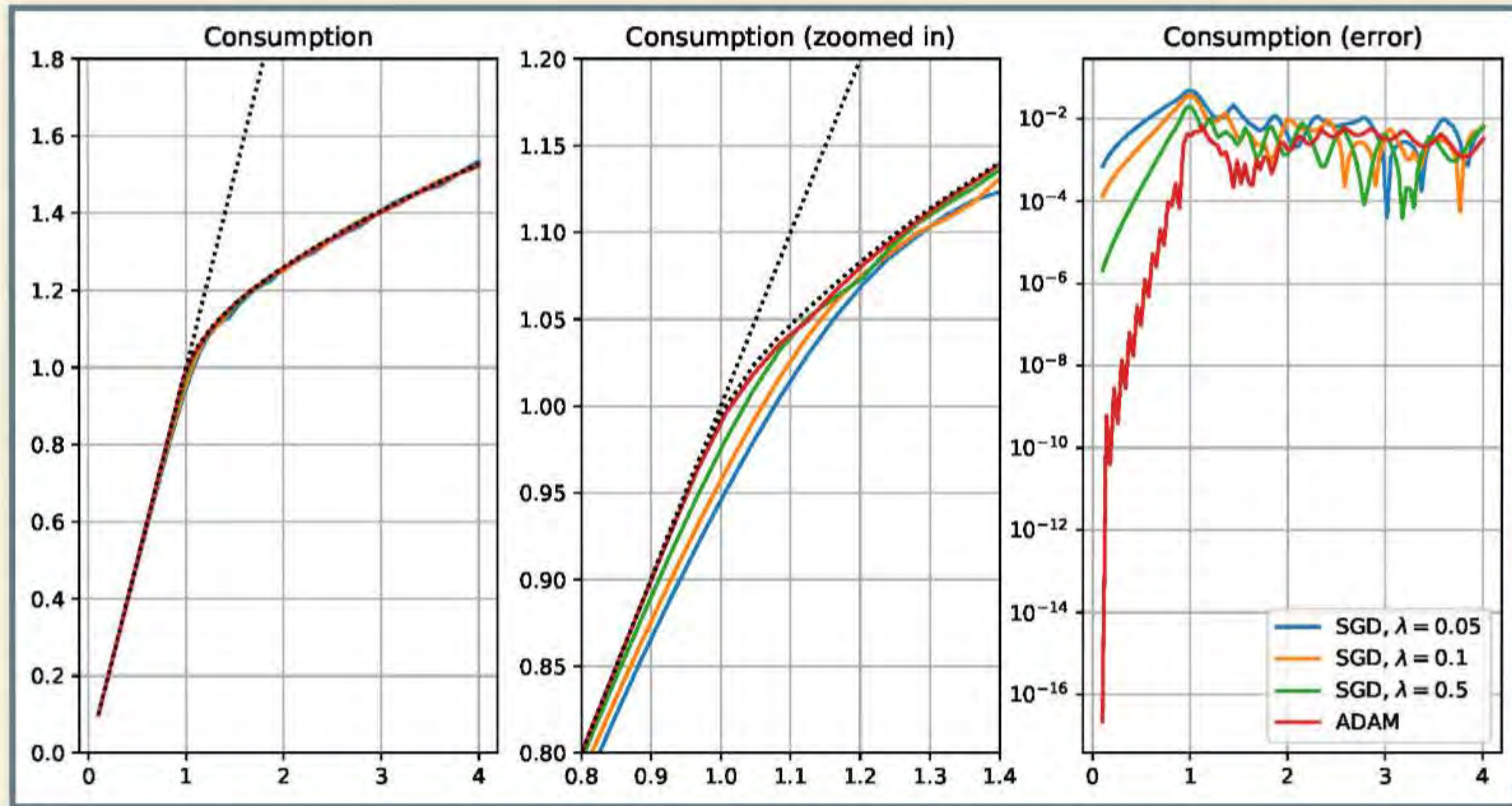
11/18/2022



OBJECTIVE 3: BELLMAN-RESIDUAL MINIMIZATION



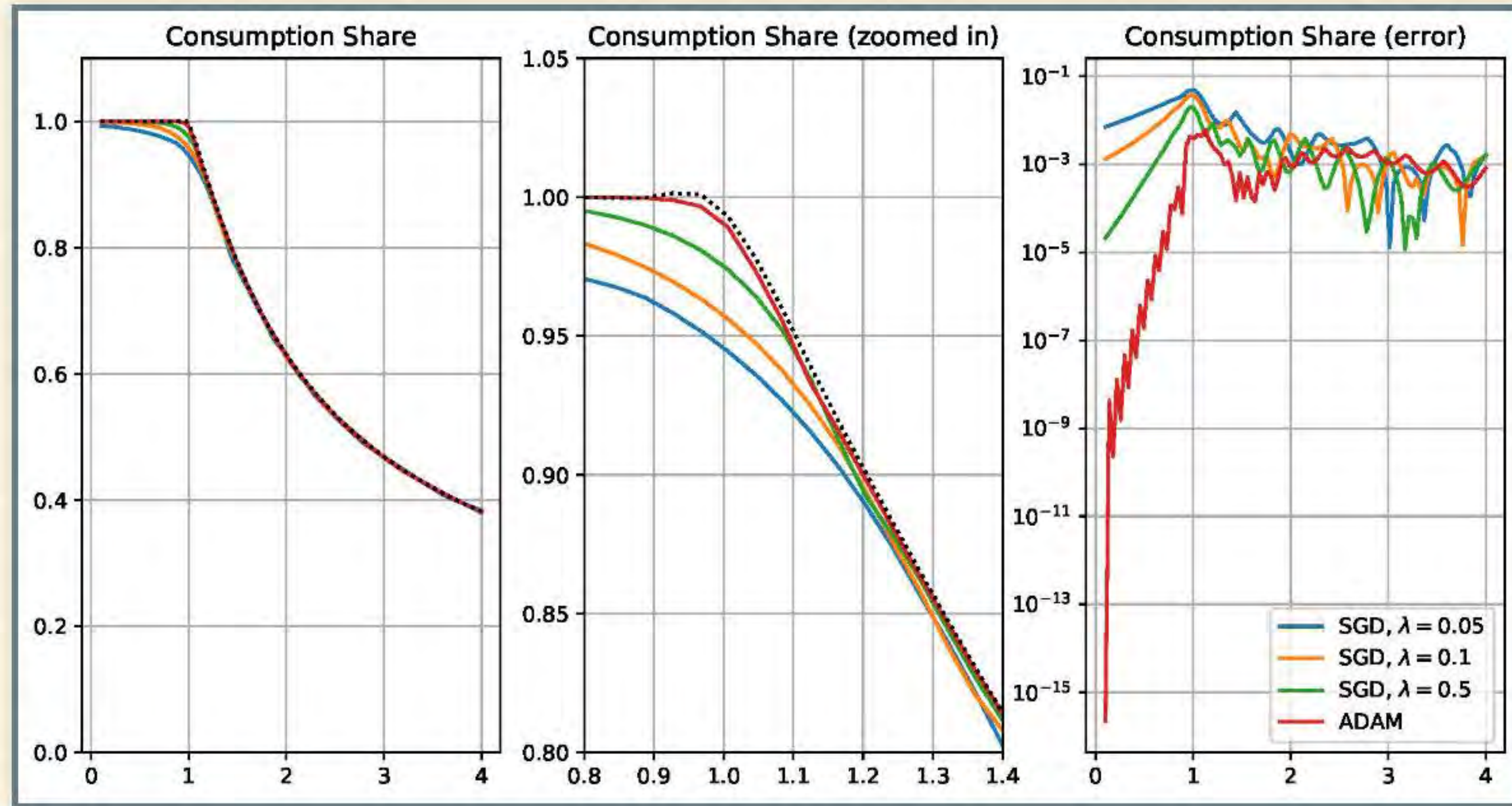
CONSUMPTION DECISION RULE



Training with the Euler-residual method

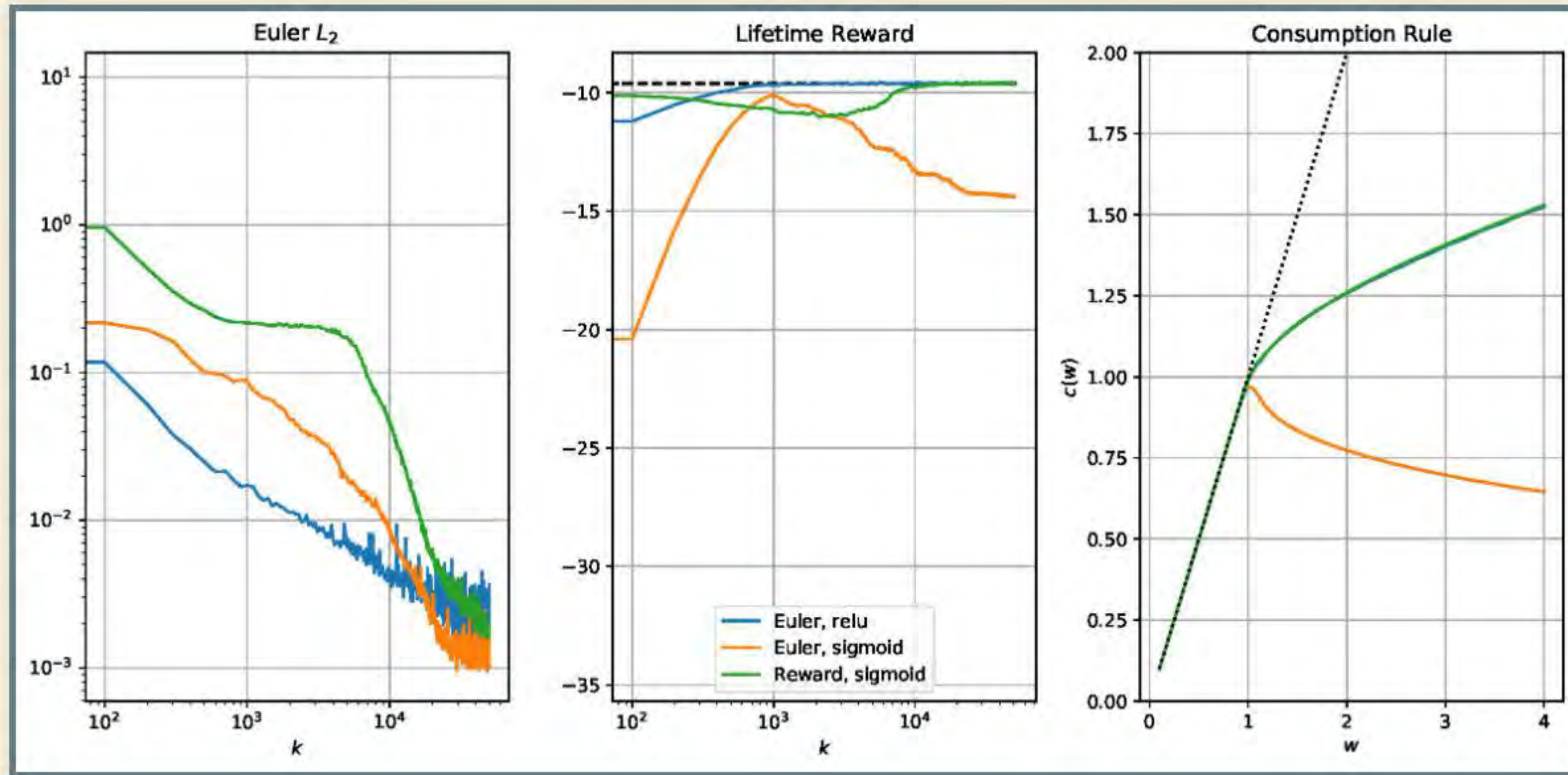


CONSUMPTION-SHARE DECISION RULE



Training with the Euler-residual method

ROLE OF ACTIVATION FUNCTIONS



Using a sigmoid activation function (instead of relu)

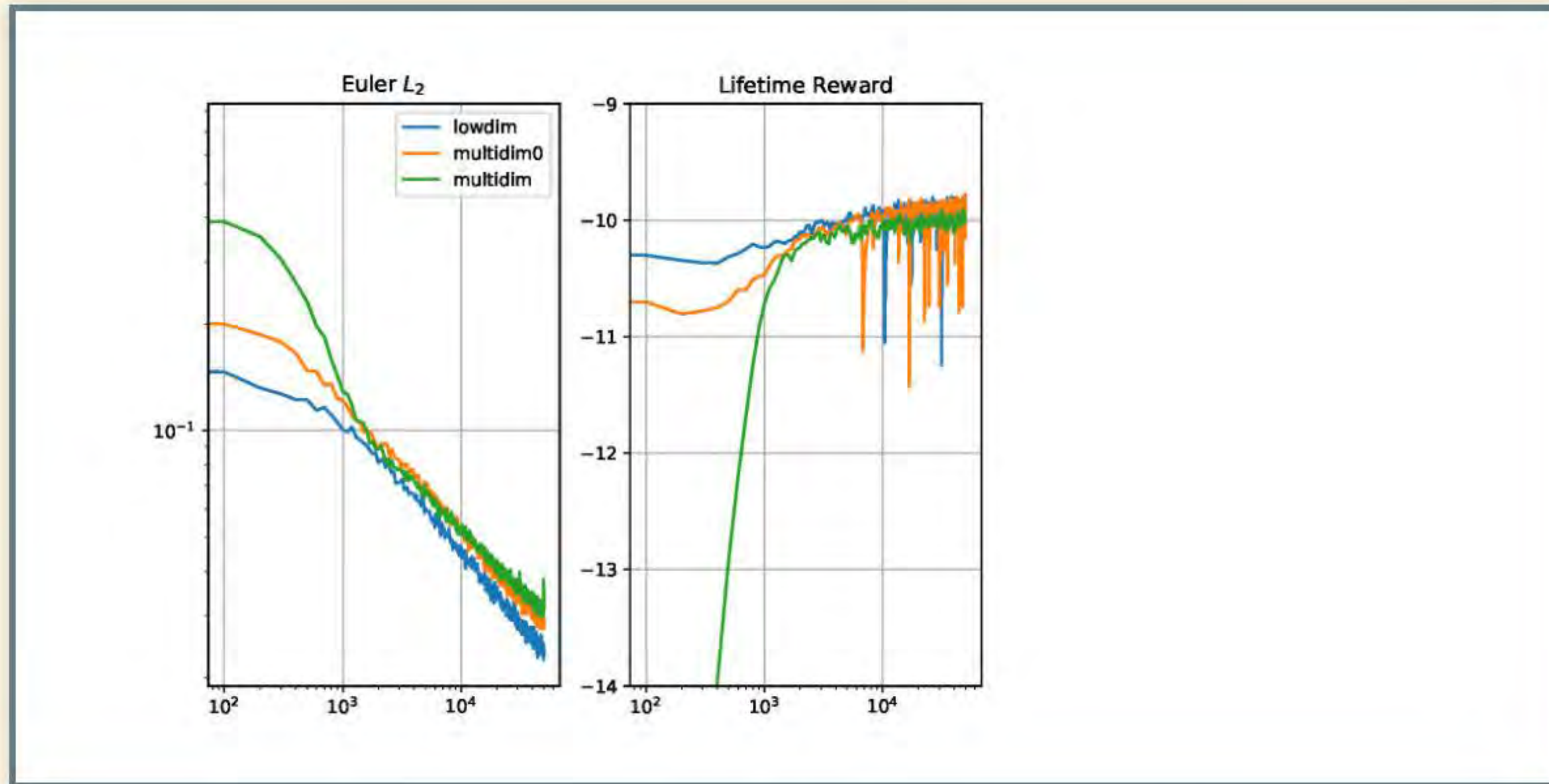


DIMENSIONALITY REDUCTION AND ROBUSTNESS TO ILL CONDITIONING

- We compare 3 models:
 - Model 1: "multidim": 4 shocks $\{y_t, p_t, \varrho_t, \chi_t\}$
 - Model 2: "lowdim": 1 shock $\{y_t\}$
 - Model 3: "multidim0": 4 shocks $\{y_t, p_t, \varrho_t, \chi_t\}$
but set $\{p_t, \varrho_t, \chi_t\} = 0 \Rightarrow$ multicollinearity
- We find that our DL method:
 - **Scales great (linearly)!**
 - **Eliminates useless states (model reduction)!**
 - **Is not affected by ill conditioning!**



THE MULTISHOCK MODEL



Training with the reward-maximization method



ALL-IN-ONE EXPECTATION OPERATORS

- In lifetime-reward maximization,
$$E_{(z_0, w_0)} [E_{(\epsilon_1, \dots, \epsilon_T)} u(\cdot)] = E_{(z_0, w_0, \epsilon_1, \dots, \epsilon_T)} [u(\cdot)]$$
 because expectation operators are related linearly
- In Euler- and Bellman-residual minimization,
$$E_{(z, w)} (E_{\epsilon} [f_j(z, w, \epsilon)])^2 \neq E_{(z, w)} E_{\epsilon} [f_j(z, w, \epsilon)^2],$$
 i.e., expectation operators cannot be naturally combined.
- As a result, stochastic gradient is biased
$$E_{(z, w)} (E_{\epsilon} [\nabla f_j(z, w, \epsilon)])^2 \neq \nabla E_{(z, w)} (E_{\epsilon} [f_j(z, w, \epsilon)])^2.$$



OUR CONTRIBUTION

- We came up with a simple technique that allows us to combine the expectation functions $E_{(z,w)}[\cdot]$ and $E_{\epsilon}[\cdot]$ in a single expectation operator in the presence of squares.
- Specifically, we use two independent random draws or two batches ϵ_1 and ϵ_2 for the two squared terms:
$$E_{\epsilon_1}[f(\epsilon_1)]E_{\epsilon_2}[f(\epsilon_2)] = E_{(\epsilon_1,\epsilon_2)}[f(\epsilon_1)f(\epsilon_2)]$$
- With this method, the stochastic gradient descent method is unbiased.



ALL-IN-ONE EXPECTATION WITH FISHER-BURMEISTER

- First, introduce μ = supplementary variable representing the expectation function

- Then, define the Euler-residual operator

$$E_{(z,w)} \{ [\Psi(c - w, u'(c) - \mu)]^2 + v[\beta r e^{\rho t} E_{\epsilon} [u'(c'(\epsilon))] - \mu]^2 \}$$

where v =exogenous weight

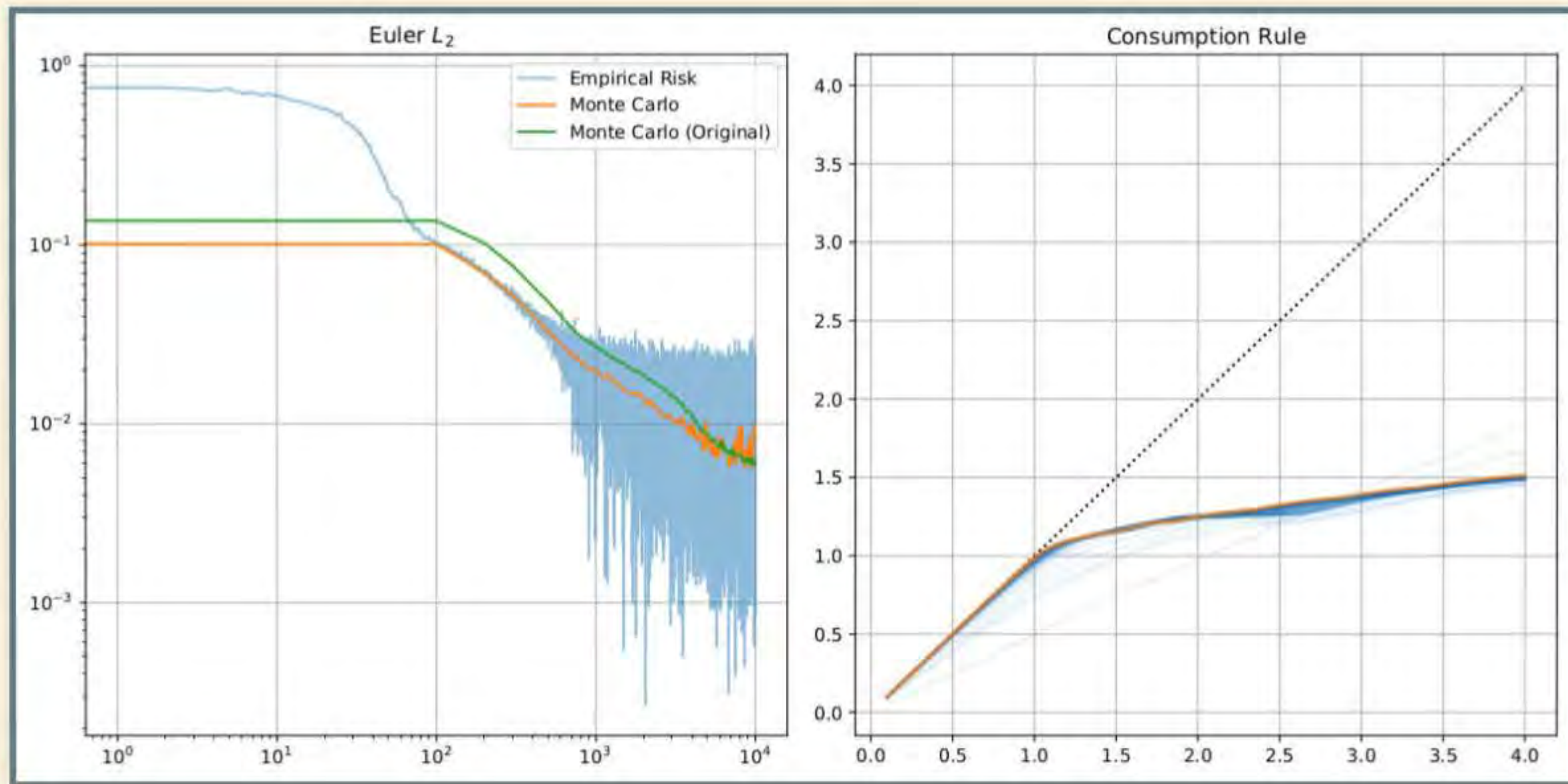
- Finally, construct all-in-one-expectation operator

$$\Xi(\theta) = E_{(z,w,\epsilon_1,\epsilon_2)} \{ [\Psi(c - w, u'(c) - \mu)]^2 + v[\beta r e^{\rho t} [u'(c'(\epsilon_1))] - \mu][\beta r e^{\rho t} [u'(c'(\epsilon_2))] - \mu] \}$$

- *Bellman-residual minimization* is similar



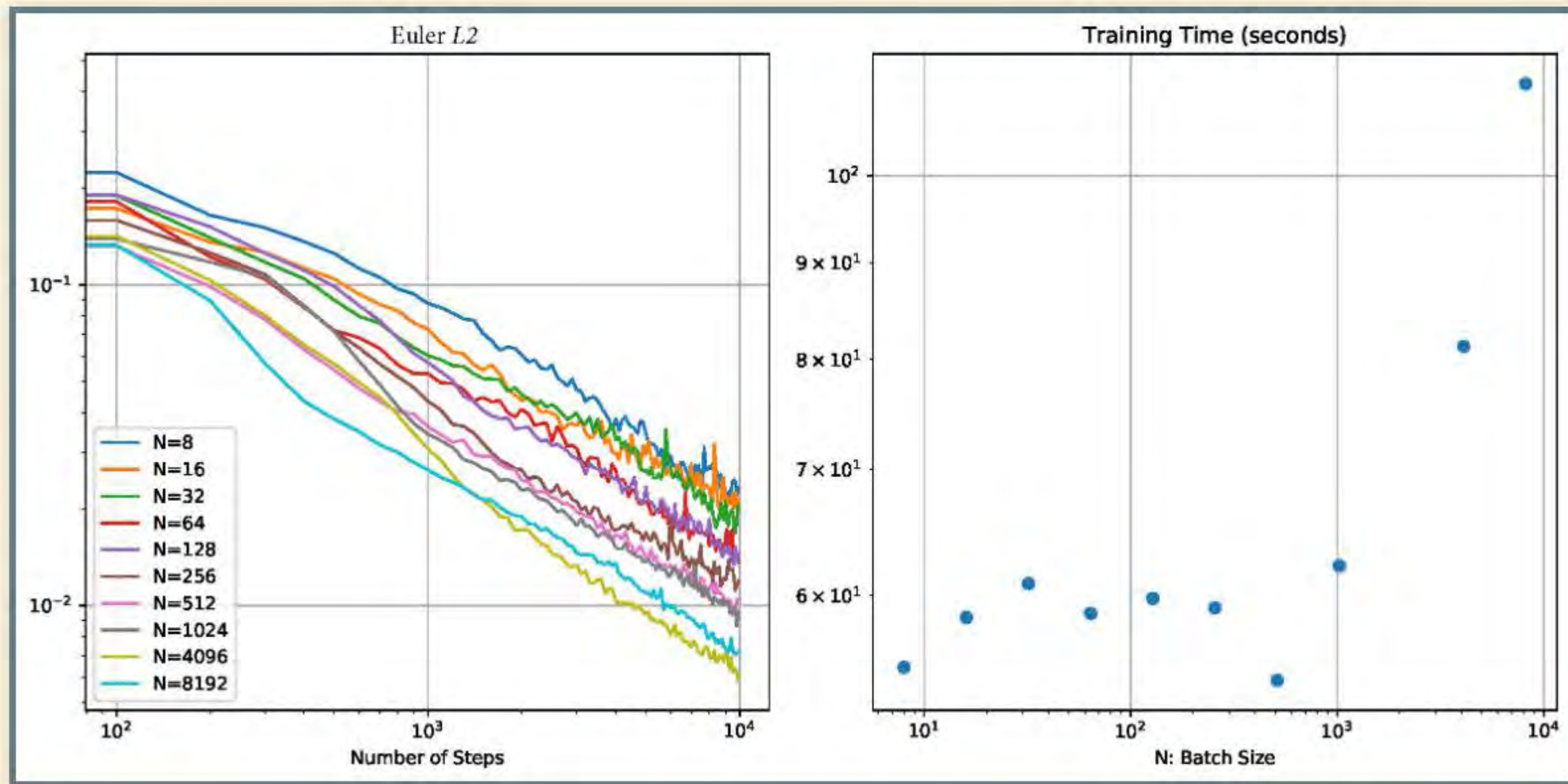
ALL-IN-ONE EXPECTATION OPERATOR (CONT.)



Training with Euler-residual method in the 4-shock model



SCALABILITY OF ALL-IN-ONE-EXPECTATION METHOD



Batch size varies from $N = 8$ to $N = 8192$ draws



KRUSELL AND SMITH (1998) MODEL

HETEROGENEOUS-AGENT SET UP

Agents $i = 1, \dots, \ell$. Identical in fundamentals but differ in productivity and capital.

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

$$\text{s.t. } c_t^i + k_{t+1}^i = R_t k_t^i + W_t z_t^i$$

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

$$k_{t+1}^i \geq 0, \text{ and } (k_0^i, z_0^i) \text{ given}$$

c_t^i, k_t^i are $z_t^i =$ consumption, capital, labor productivity

HETEROGENEOUS-AGENT SET UP (CONT.)

- Cobb-Douglas production function $z_t k_t^\alpha$, with $k_t = \sum k_t^i = \text{agg. capital}$, $z_t = \text{agg. productivity}$.

$$R_t = 1 - d + z_t \alpha k_t^{\alpha-1} \quad \& \quad W_t = z_t (1 - \alpha) k_t^{\alpha-1},$$

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

- State space: z_t & $\{k_t^i, z_t^i\}_{i=1}^l$.
- Krusell and Smith's (1998) method: replace distributions with a finite set of moments $m_t \Rightarrow$ approximate state space by $\{k_t^i, z_t^i, z_t, m_t\}$.



Type here to search



4:35 AM
11/18/2022



ALL-IN-ONE EXPECTATION

- We work directly with the actual state space.
- **Objective function:**

$$\Xi(\theta) = E_{(K, Z, z, \Sigma_1, \Sigma_2, \epsilon_1, \epsilon_2)} \{ [\Psi(c^i - w^i, u'(c^i) - \mu^i)]^2 + v[\beta[u'(c^{i'}(\Sigma_1, \epsilon_1))] - \mu^i] [\beta[u'(c^{i'}(\Sigma_2, \epsilon_2))] - \mu^i] \},$$

$$K = (k_1^1, \dots, k_1^\ell), Z = (z_1^1, \dots, z_1^\ell), \Sigma_1 = (\epsilon_1^1, \dots, \epsilon_1^\ell), \\ \Sigma_2 = (\epsilon_2^1, \dots, \epsilon_2^\ell); \epsilon_1, \epsilon_2 = \text{agg. productivity shocks.}$$

DL SOLUTION ALGORITHM

Procedure

- i) draw initial state z_0 & $\{K_0, Z_0\} = \{k_0^i, z_0^i\}_{i=1}^{\ell}$;
- ii) compute aggregate capital k_0 and prices R_0, W_0 ;
- iii) train neural network for ℓ agents;
- iv) compute next period distribution

$$z_1 \text{ and } \{K_1, Z_1\} = \{k_1^i, z_1^i\}_{i=1}^{\ell}.$$

Proceed iteratively until convergence.



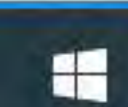
DL SOLUTION ALGORITHM (CONT.)

- **Ergodic-set domain**
 - deals with curse of dimensionality by simulating the state variables forward
- **Neural network handles multicollinearity**
 - include the state variables of agent i twice, $\varphi^i = \varphi(k_t^i, z_t^i, \{K_t, Z_t\}, z_t; \theta)$ but it is fine
- **Model reduction**
 - condenses $2\ell + 1 = 2,001$ state variables into 64 neurons in each hidden layer



CALIBRATION AND COMPUTATIONAL DETAILS

- $u(c) = \frac{c^{1-\gamma}-1}{1-\gamma}$ with $\gamma = 1$
- $\beta = 0.96; \rho = 0.9; \sigma = 0.07; \rho_z = 0.9$
 $\sigma_z = 0.4(1 - \rho_z^2)^{1/2}$
- ADAM with batch size of 10 and learning rate of 0.001
- $K = 100,000$ iterations (equal to simulation length)



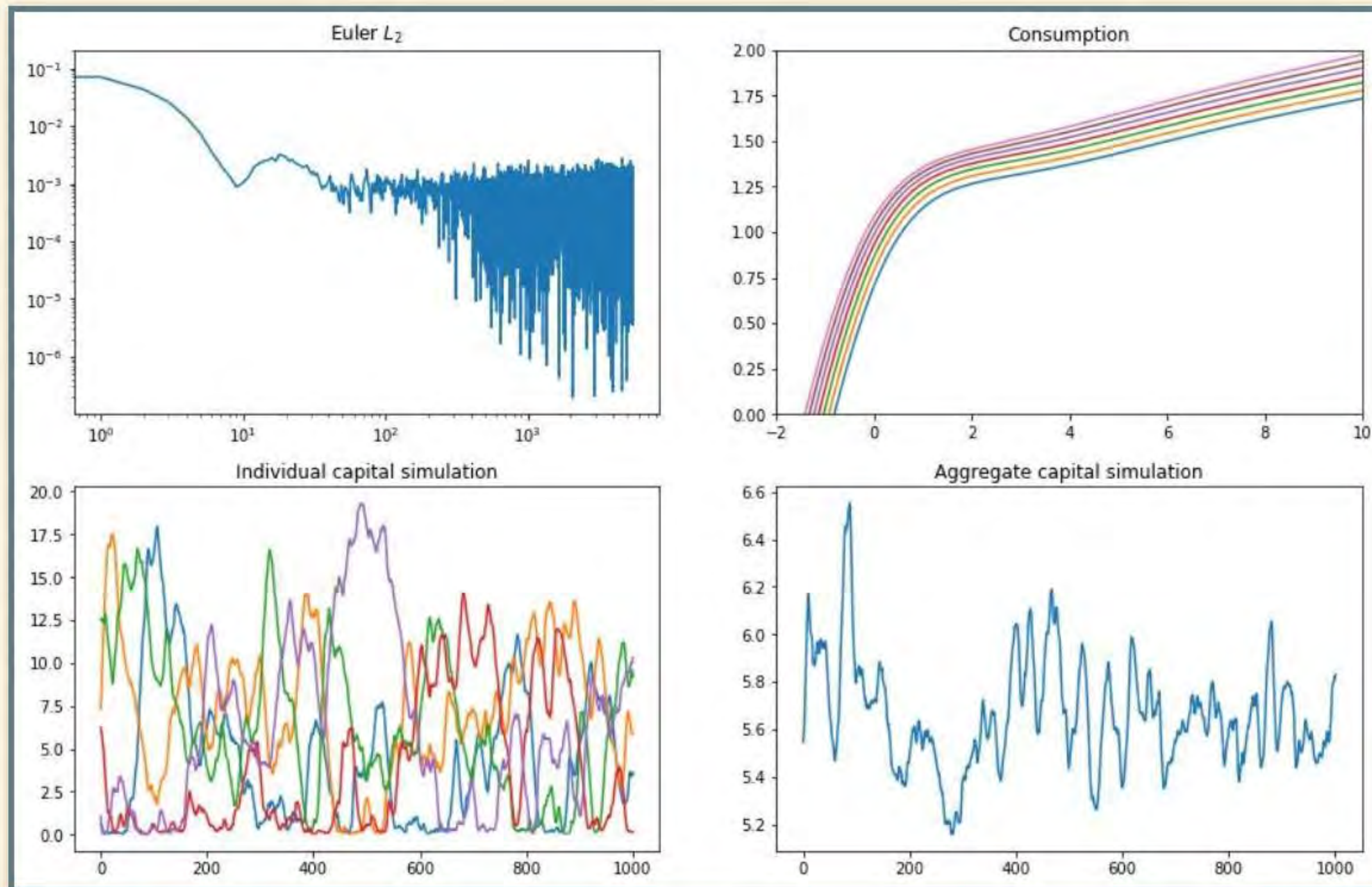
Type here to search



4:35 AM
11/18/2022



TRAINING WITH EULER-RESIDUAL METHOD, 5 AGENTS



SELECTED STATISTICS

ℓ	std(y)	corr(y,c)	Gini(k)	Bottom 40%	Top 20%	Top 1%	Time, sec.	R ²
1	1.51	0.858	-	-	-	-	235	0.999999
5	1.51	0.772	0.335	0.176	0.385	0.031	234	0.993473
10	1.51	0.595	0.391	0.144	0.428	0.036	254	0.995091
50	1.51	0.635	0.497	0.099	0.530	0.050	467	0.995284
100	1.51	0.658	0.450	0.121	0.486	0.047	1020	0.997600
500	1.51	0.462	0.484	0.096	0.506	0.052	7552	0.996554
1000	1.51	0.268	0.501	0.092	0.528	0.047	16435	0.995415





CONCLUSION

ENTHUSIASTIC VIEW

- Data scientists developed
 - powerful AI techniques for data analysis (DL)
 - general-purpose software for their efficient implementation (e.g., TensorFlow)
- Economists mostly develop own method & codes
 - some of which are surprisingly smart but typically, model-specific, not portable
- We show how to cast economic models into the state-of-the-art DL methods and software
- We hope this approach augments vastly the set of models tractable in economics



Machine learning, artificial intelligence and deep learning methods for dynamic economic models:
Deep learning for heterogenous agent models

Lilia Maliar, Serguei Maliar

November 2022

Minicourse

Discrete- versus continuous-set choices

- Macroeconomic models are generally built on 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.

The progress in modeling discrete choices is still limited!

The results in the present paper

- We introduce a deep learning classification (DLC) method that solves models with both continuous-set and discrete-set choices.
- To solve for continuous-set choices:
 - 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 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.

An illustration from data science: image recognition

- Consider the image recognition problem—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, Bengio and Courville (2016) for a survey of classification methods in data science.

Classification method for discrete choices in economics

- Our classification method in macroeconomics is analogous to the above image-recognition analysis.
- For example, we use a deep neural network to parameterize the probabilities of being full-time employed, part-time employed and unemployed.
- 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.

Remark:

- The earlier literature on indivisible labor (e.g., Rogerson (1996) and Hansen (1994)) construct discrete choice by introducing lotteries.
- Our probabilities have totally different meaning: they indicate which discrete choices is most likely to be optimal and hence, is selected.

Problems with high dimensionality

- The DLC 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 DLC method, we solve Krusell and Smith's (1998) model in which the agents face indivisible labor choices.

The literature on heterogeneous agent models

- Krusell and Smith's (1998) model is computationally challenging even in the absence of discrete choices.
- The state space may include thousands of state variables of heterogeneous agents and is prohibitively large.
- To make the model tractable, Krusell and Smith (1998) replace distributions with few aggregate moments but that approach does not always work.
- Several recent papers use linearization and perturbation to simplify the analysis of equilibrium in heterogeneous-agent models, including Reiter (2010), McKay and Reis (2016), Childers (2016), Boppart et al. (2018), Mertens and Judd (2017), Ahn et al (2018), Winberry (2018), Bayer and Luetticke (2020)
- Reiter (2019) provides for a thoughtful discussion of that literature.

DLC method

- 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.
- It 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.

Relation to the literature on deep learning in economics

- Our DLC method is related to recent papers on deep learning, including Duarte (2018), Villa and Valaitis (2019), Fernández-Villaverde, Hurtado, and Nuño (2019), Azinović, Luca and Scheidegger (2019), Lepetyuk, Maliar and Maliar (2020) and especially, Maliar, Maliar and Winant (2018, 2019, 2021).
- However, this literature does not analyze models with discrete choices, which is the main subject of the present paper.

Relation to the literature on discrete choices

- 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.
- The macro literature with discrete choices includes Chang and Kim (2007) and Chang, Kim, Kwon and Rogerson (2019) who solve a similar model by using Krusell and Smith (1998) analysis.
- Iskhakov, Jørgensen, Rust and Schjerning (2017) developed an endogenous grid method with taste shocks that is designed to deal with discrete choices in dynamic environment.
- In the context of Carroll's (2005) analysis, that paper suggests to apply logistic smoothing to the kinks by transferring the problem into the choice probability space via the taste shocks.
- In contrast, 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.

Applications: Krusell and Smith's (1998) model

- a version of Krusell and Smith's (1998) model with continuous choices (i.e., divisible labor);
- an indivisible-labor version with 2 discrete labor states (employed and unemployed);
- an indivisible-labor version with 3 discrete labor states (employed, unemployed and part-time employed agent).

The model

- Heterogeneous agents $i = 1, \dots, \ell$. Each agent i solves

$$\begin{aligned} & \max_{\{c_t^i, k_{t+1}^i, n_t^i\}_{t=0}^{\infty}} E_0 \left[\sum_{t=0}^{\infty} \beta^t u \left(c_t^i, n_t^i \right) \right] \\ & \text{s.t. } c_t^i + k_{t+1}^i = R_t k_t^i + W_t v_t^i n_t^i, \\ & \quad n_t \in N, \\ & \ln v_{t+1}^i = \rho_v \ln v_t^i + \sigma_v \epsilon_t^i \text{ with } \epsilon_t^i \sim \mathcal{N}(0, 1), \\ & \quad k_{t+1}^i \geq \bar{k}, \end{aligned}$$

where c_t^i , n_t^i , k_t^i and v_t^i are consumption, hours worked, capital and idiosyncratic labor productivity; $\beta \in (0, 1)$ is the discount factor; $\rho_v \in (0, 1)$ and $\sigma_v \geq 0$; and initial condition (k_0^i, v_0^i) is given. The capital choice is restricted by a borrowing limit $\bar{k} \leq 0$.

- The three different versions of the model are distinguished by the set of allowable labor choices N .

Production side

- The production side of the economy is described by a Cobb-Douglas production function $\exp(z_t) k_t^{\alpha} h_t^{1-\alpha}$, 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,

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

where $\rho_z \in (0, 1)$ and $\sigma_z \geq 0$.

- The interest rate R_t and wage W_t are given by

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

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

Kuhn-Tucker condition

- The Kuhn-Tucker condition with respect to capital is

$$\mu_t^i \delta_t^i = 0,$$

where $\delta_t^i \equiv k_{t+1}^i - \bar{k} \geq 0$ is the distance to the borrowing limit, and $\mu_t^i \geq 0$ is the Lagrange multiplier

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

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., $k_{t+1}^i > \bar{k}$, 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 $\delta_t^i = 0$

Three different version of the model

We consider three versions of the model that differ in the set of allowable labor choices $n_t \in N$:

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

Divisible labor model

- To characterize labor choice, we assume that the utility function takes the form

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

where $\eta > 0$ and L is the total time endowment.

- We normalize time 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.
- The labor choice is characterized by a FOC

$$n_t^i = L \left[\frac{c_t^i W_t v_t^i}{B} \right]^{1/\eta}.$$

Indivisible labor model with 2 states

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

$$\begin{aligned}n_t^i &= \bar{n} \text{ if } V^E = \max \{V^E, V^U\} \\n_t^i &= 0 \text{ otherwise.}\end{aligned}$$

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

Indivisible labor model with 3 states

The three employment states, $n_t^i = \bar{n}$, $n_t^i = \underline{n}$ and $n_t^i = 0$, correspond to full-time unemployment, part-time employment and unemployment, respectively,

$$n_t^i = \bar{n} \text{ if } V^{FT} = \max \{V^U, V^{FT}, V^{PT}\}$$

$$n_t^i = \underline{n} \text{ if } V^{PT} = \max \{V^U, V^{FT}, V^{PT}\}$$

$$n_t^i = 0 \text{ otherwise}$$

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

Deep learning method for divisible labor model

Deep learning method for divisible labor model

The state space of Krusell and Smith's (1998) model has $2\ell + 1$ state variables; for example, with $\ell = 1,000$, the state space has 2,001 state variables. To deal with so large dimensionality, we rely on a combination of techniques introduced in Maliar et al. (2018, 2019, 2021), including:

1. stochastic simulation that allows us to restrict attention to the ergodic set in which the solution "lives";
2. multilayer neural networks that perform model reduction and help deal with multicollinearity;
3. a (batch) stochastic gradient descent method that reduces the number of function evaluations by operating on random grids;
4. a Fischer-Burmeister function that effectively approximates the kink;
5. 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.
6. 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.

Stochastic simulation - ergodic set domain

- Under normally distributed shocks, stochastic simulation typically have a shape of a hypersphere (hyperoval)

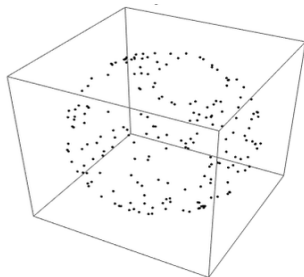


Figure 1. Hypercube versus hypersphere.

- The ratio of a volume of a hypersphere to that of an enclosing hypercube is an infinitesimally small number in high-dimensional applications; for example, for a 30-dimensional case, it is 10^{-14} ; see Judd, Maliar and Maliar (2011) for a discussion.

Neural networks

We use neural networks for parameterizing decision and value functions instead of more conventional approximation families like polynomial functions:

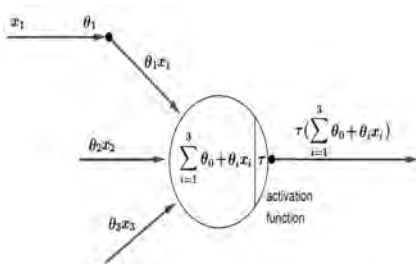


Figure 2a. Artificial neuron.

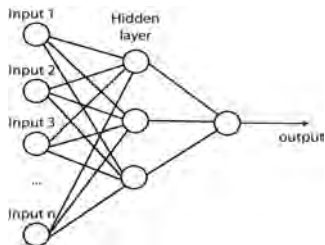


Figure 2b. Neural network.

In Figure 1a, the circle represents an artificial neuron that receives 3 signals (inputs) x_1 , x_2 and x_3 . In Figure 1b, we combine multiple neurons into a neural network.

Activation functions

The activation function that we use in our benchmark experiments is a sigmoid function $\sigma(z) = \frac{1}{1+e^{-z}} = \frac{1}{1+e^{-\theta_0+\theta_1x_1+\theta_2x_2+\dots+\theta_nx_n}}$.

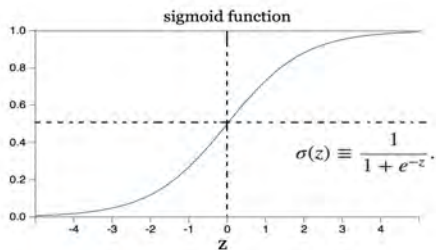


Figure 3. Sigmoid function.

The sigmoid function has two properties: First, its derivative can be inferred from the function itself $\sigma'(x) = \sigma(x)(1 - \sigma(x))$. Second, it maps a real line into a unit interval $\sigma : \mathbb{R}^n \rightarrow [0, 1]$ which makes it bounded between 0 and 1.

Parameterization of decision functions

- We solve for two decision functions—hours worked $\frac{n_t^i}{L}$ and the fraction of wealth that goes to consumption $\frac{c_t^i}{w_t^i}$ which we parameterized by a sigmoid function

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

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 and ζ_0 is a constant term.

- In addition, we parameterize the Lagrange multiplier μ_t^i associated with the borrowing constraint using an exponential activation function

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

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 DLC solution method aims at solving models with thousands of state variables by using 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.
- 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, Maliar and Maliar (2011) for a discussion of model reduction using principal-component analysis.
- Krusell and Smith (1998) proposed one specific model reduction method, namely, they approximate the distribution with just one moment – the mean.
- If Krusell and Smith's (1998) analysis is the most efficient representation of the state space, the neural network will also find it.
- However, the neural network will consider many other possible ways of extracting the information from the distributions and condensing it in a relatively small set of hidden layers trying to find the best one.

Objective function for deep learning

- The objective is to minimize the squared residuals in three model's conditions:

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

where $K \equiv (k^1, \dots, k^\ell)$ and $Y \equiv (v^1, \dots, v^\ell)$ are state variables; z_t is aggregate productivity; $\Sigma_{t+1} \equiv (\epsilon_{t+1}^1, \dots, \epsilon_{t+1}^\ell)$ the individual productivity shocks; ϵ_{t+1} is the aggregate productivity shock; and

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

is a $\Psi^{FB}(a, b) = 0$ is a Fisher-Burmeister objective function is equivalent to Kuhn Tucker conditions.

All in one expectation operator

- The constructed objective function $\Xi(\theta)$ is not convenient because it contains a square of expectation $\left[E_{(\Sigma_{t+1}, \epsilon_{t+1})} [\cdot]\right]^2$ nested inside another expectation $E_{(K_t, Y_t, z_t)} [\cdot]$.
- Constructing two nested expectation operators is costly because the inner expectation operator $E_{(\Sigma_{t+1}, \epsilon_{t+1})} [\cdot]$ has high dimensionality; if $\ell = 1,000$, it is 1,001-dimensional integral.
- This task would be simplified enormously if we could combine the two expectation operators but it is not possible

$$E_{(K_t, Y_t, z_t)} \left[\left[E_{(\Sigma_{t+1}, \epsilon_{t+1})} [\cdot] \right]^2 \right] \neq E_{(K_t, Y_t, z_t)} E_{(\Sigma_{t+1}, \epsilon_{t+1})} \left[[\cdot]^2 \right].$$

- 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.
- They 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 $(\Sigma'_{t+1}, \epsilon'_{t+1})$ and $(\Sigma''_{t+1}, \epsilon''_{t+1})$.
- Since the two random draws are uncorrelated, the expectation operator can be taken outside of the expectation function.

The objective function under AiO expectation operator

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

Thus, we are able to represent the studied model as an expectation function across a vector of random variables

$(K_t, Y_t, z_t, \Sigma'_{t+1}, \epsilon'_{t+1}, \Sigma''_{t+1}, \epsilon''_{t+1})$; see Maliar et al. (2021) for a discussion and further applications of the AiO expectation operator.

Training: gradient descent, batches and parallel computing

- Given that Ξ 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.
- 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

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

where θ and λ are the parameter vector and learning rate, respectively.

- Thus, we implement a cheap computation of the gradient of the integrand instead of computing far more expensive gradient of the expectation function. TensorFlow and PyTorch can compute such a gradient using a symbolic differentiation, which facilitates an the implementation of parallel computation.

Dealing with multicollinearity

- In the arguments of approximating functions, the state variables of agent i appear twice $\varphi\left(k_t^i, v_t^i, \{k_t^i, v_t^i\}_{i=1}^\ell, z_t; \theta\right)$ 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.

Algorithm 1: Deep learning for divisible labor model

Algorithm 1: Deep learning for divisible labor model.

Step 0: (Initialization).

Construct initial state of the economy $\left(\{k_0^i, v_0^i\}_{i=1}^{\ell}, z_0\right)$ and parameterize three decision functions by a neural network with three outputs

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

where $w_t^i \equiv R_t k_t^i + W_t v_t^i n_t^i$ is wealth; μ_t^i is Lagrange multiplier associated with the borrowing constraint; $\varphi(\cdot)$ is a neural network; $\sigma(z) = \frac{1}{1+e^{-z}}$ is a sigmoid (logistic) function; ζ_0 is a constant; θ is a vector of coefficients.

Algorithm 1: Deep learning for divisible labor model (cont)

Algorithm 1: Deep learning for divisible labor model.

Step 1: (Evaluation of decision functions).

Given state $\left(k_t^i, v_t^i, \{k_t^i, v_t^i\}_{i=1}^\ell, z_t\right) \equiv s_t^i$, compute $n_t^i, \mu_t^i, \frac{c_t^i}{w_t^i}$ from the neural networks, find the prices R_t and W_t ; and find k_{t+1}^i from the budget constraint for all agents $i = 1, \dots, \ell$.

Step 2: (Construction of Euler residuals).

Draw two random sets of individual productivity shocks $\Sigma_1 = (\epsilon_1^1, \dots, \epsilon_1^\ell)$, $\Sigma_2 = (\epsilon_2^1, \dots, \epsilon_2^\ell)$ and two aggregate shocks ϵ_1, ϵ_2 , and construct Euler residuals

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

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

Algorithm: Deep learning for divisible labor model (cont.)

Algorithm 1: Deep learning for divisible labor model.

Step 3: (Training).

Train the neural network coefficients θ to minimize 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 $n = 1, \dots, N$ denotes batches.

Step 4: (Simulation).

Move to $t + 1$ by using endogenous and exogenous variables of Step 3 under $\Sigma_1 = (\epsilon_1^1, \dots, \epsilon_1^{\ell})$ and ϵ_1 as a next-period state $(\{k_{t+1}^i, v_t^i\}_{i=1}^{\ell}, z_{t+1})$.

Calibration

- 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 ϵ and η of the utility function by performing a sensitivity analysis..

Training errors and running time

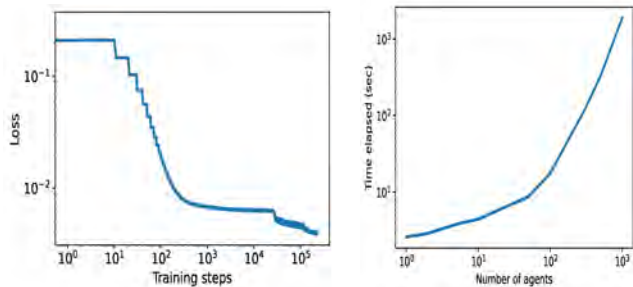


Figure 4. Training errors and running time for divisible labor model.

The solution for divisible labor model

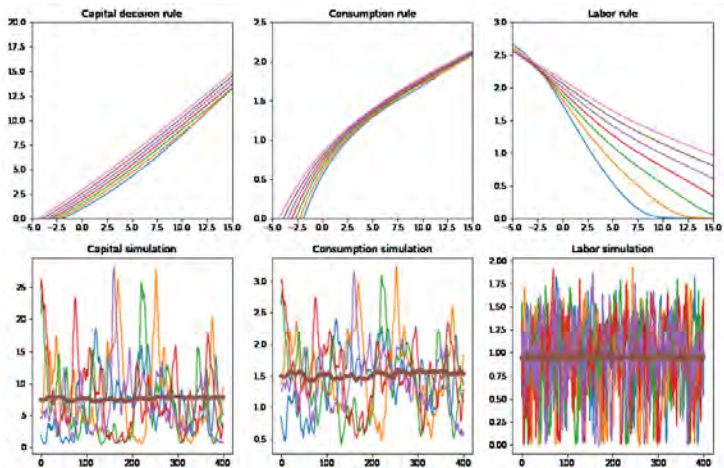


Figure 5. Solution to divisible labor mode.

Deep learning method for indivisible labor model

Logistic regression

Let us consider a typical classification problem. We have a collection of ℓ data points $\{X^i, y^i\}_{i=1}^{\ell}$ where $X^i \equiv (1, x_1^i, x_2^i, \dots)$ is a collection of dependent variables (features) and y^i is a categorical independent variable (label) that takes values 0 and 1. The goal is to construct a dashed line that separates the known examples of the two types.

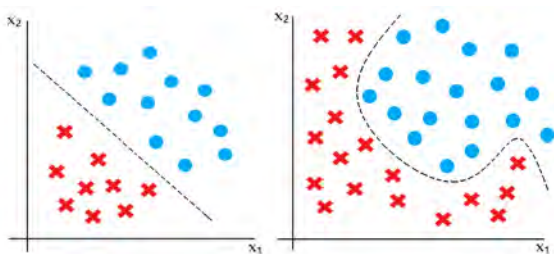


Figure 6. Examples of binary classification.

We restrict attention to one technique – logistic regression – which is simple, general and can be conveniently combined with our deep learning analysis.

A hypothesis

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,

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

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

Estimation

- 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

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

where p is the probability that a data point with characteristics $X \equiv (1, x_1, x_2, x_1^2, \dots)$ belongs to class 1, and $\theta \equiv (\theta_0, \theta_1, \dots, \theta_m, \dots, \theta_M)$ 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 $X\theta \rightarrow -\infty$ and $X\theta \rightarrow +\infty$, we have that $p \rightarrow 0$ and $p \rightarrow 1$, respectively.

Probability of an observation

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

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

Indeed, if $y = 1$, we have $\text{Prob}(y = 1 | X; \theta) = (p)^1 (1 - p)^0 = p$; and if $y = 0$, we have $\text{Prob}(y = 0 | X; \theta) = (p)^0 (1 - p)^1 = 1 - p$.

Likelihood function

We search for the coefficient vector θ that maximizes the (log)likelihood of the event such that a given matrix of features $\{X^i\}_{i=1}^{\ell}$ produces the given output realizations $\{y^i\}_{i=1}^{\ell}$, i.e.,

$$\begin{aligned} \max_{\theta} \ln L(\theta) &= \ln \prod_{i=1}^{\ell} (p(X^i; \theta))^{y^i} (1 - p(X^i; \theta))^{1 - y^i} = \\ &= \sum_{i=1}^{\ell} [y^i \ln (p(X^i; \theta)) + (1 - y^i) \ln (1 - p(X^i; \theta))], \end{aligned}$$

where the probability $p(X^i; \theta) \equiv \frac{1}{1 + \exp(-X^i \theta)}$ is given by a logistic function.

Constructing a maximizer

To find the maximizer, we compute the first-order conditions with respect to all coefficients θ_m for $m = 0, \dots, M$,

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

where x_m^i is the feature m of agent i .

The constructed gradient $\nabla \ln L_{\theta}(\theta) \equiv \left[\frac{\partial \ln L(\theta)}{\partial \theta_1}, \dots, \frac{\partial \ln L(\theta)}{\partial \theta_M} \right]'$ can be used for implementing the gradient descent-style method

$$\theta \leftarrow \theta - \lambda \nabla \ln L_{\theta}(\theta).$$

Decisions in divisible versus indivisible labor

- In the divisible labor model, we construct a policy function that determines the hours worked $\frac{n_t^i}{L}$.
- In the indivisible labor model studied here, we construct a decision boundary $\varphi(s_t^i; \theta) = 0$ that separates the employment and unemployment choices conditional on state $s_t^i \equiv \left(k_t^i, v_t^i, \{k_t^i, v_t^i\}_{i=1}^\ell, z_t \right)$.
- Whenever $\varphi(s_t^i; \theta) \geq 0$, the agent is employed $n_t^i = \bar{n}$ 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.

Decisions in divisible versus indivisible labor

- 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 coefficients of the neural network (weights and biases) by formulating a logistic regression,

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

- 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:

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

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. (Here, we also parameterize the Lagrange multiplier.

Decisions in divisible versus indivisible labor

- The function p_t^i , allows us to infer the indivisible labor choice directly, specifically, an agent is employed $n_t^i = \bar{n}$ 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 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.
- 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 $V^E(s_t^i; \theta^E)$ and $V^U(s_t^i; \theta^U)$.
- We next construct the labor choice \hat{n}_t^i implied by these two value functions

$$\hat{n}_t^i = \begin{cases} \bar{n} & \text{if } V^E = \max \{V^E, V^U\}, \\ 0 & \text{otherwise.} \end{cases}$$

Decisions in divisible versus indivisible labor

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

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

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

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

We then maximize the likelihood function by using a conventional / stochastic / batch stochastic gradient descent methods. We iterate on the decision functions p_t^i , $\frac{c_t^i}{w_t^i}$ and μ_t^i until convergence.

Implementation difference in construction of divisible and indivisible labor.

- There is an important implementation 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 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): intensive and extensive margins

- Prescott et al. (2009) propose a clever 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.
- They allow for intensive and extensive margins by "discretizing" the FOC. To be specific, they assume that the labor choice is divisible as long as it is above a given threshold \bar{n}_f but it jumps to zero whenever the labor choice falls below \bar{n}_f (i.e., the agent becomes unemployed):

$$\hat{n}_t^i = \begin{cases} L \left[\frac{c_i^\square W_t \exp(v_t^i)}{B} \right]^{\square 1/\eta} & \geq \bar{n}_f, \\ 0 & \text{otherwise.} \end{cases}$$

Determining indivisible labor: value functions versus "discretized" FOC

- We borrow from Prescott et al. (2009) 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 \bar{n} (employed):

$$\hat{n}_t^i = \begin{cases} \bar{n} & \text{if } L \leq \left[\frac{c_t^i W_t \exp(v_t^i)}{B} \right]^{1/\eta} \geq \bar{n}_f, \\ 0 & \text{otherwise.} \end{cases}$$

- 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.

Algorithm 2: Deep learning for indivisible labor model

Algorithm 2: Deep learning for the indivisible labor model.

Step 0: (Initialization).

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

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

where p_t^i is the probability of being employed.

Algorithm 2: Deep learning for indivisible labor model (cont.)

Algorithm 2: Deep learning for the indivisible labor model.

Step 1: (Evaluation of decision functions).

Given $(k_t^i, v_t^i, \{k_t^i, v_t^i\}_{i=1}^\ell, z_t)$, compute $n_t^i = \bar{n}$ 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 $\frac{c_t^i}{w_t^i}$, and find R_t and W_t ; and find k_{t+1}^i from the budget constraint for all agents $i = 1, \dots, \ell$.

Option 1: Construct V^E and V^U and find $\hat{n}_t^i = \begin{cases} \bar{n} & \text{if } V^E = \max\{V^E, V^U\}, \\ 0 & \text{otherwise.} \end{cases}$

Option 2: Use the discretized FOC $\hat{n}_t^i = \begin{cases} \bar{n} & \text{if } L \square \left[\frac{c_i^\square W_t \exp(v_t^i)}{B} \right]^{\square 1/\eta} \geq \bar{n}_f, \\ 0 & \text{otherwise.} \end{cases}$

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

Algorithm 2: Deep learning for divisible labor model (cont.)

Algorithm 2: Deep learning for the indivisible labor model.

Step 2: (Construction of Euler residuals).

Draw two random sets of individual productivity shocks $\Sigma_1 = (\epsilon_1^1, \dots, \epsilon_1^\ell)$, $\Sigma_2 = (\epsilon_2^1, \dots, \epsilon_2^\ell)$ and two aggregate shocks ϵ_1, ϵ_2 , to construct

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

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).

Training errors and running time

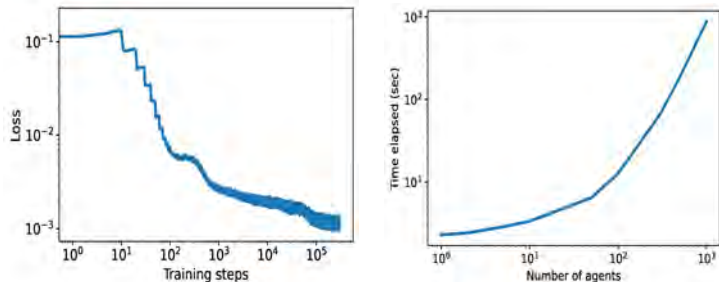


Figure 7. Training errors and running time for indivisible labor model.

The solution for divisible labor model

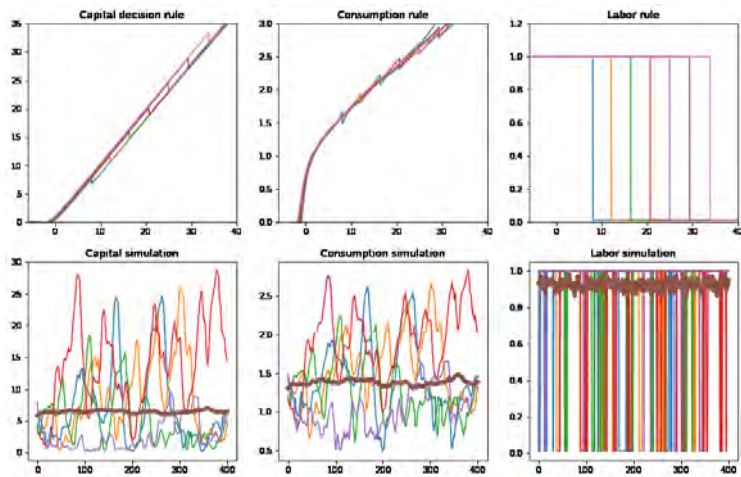


Figure 8. Solution to indivisible labor model under $\beta = 1$ and $\eta = 1$.

Deep learning method for the model with 3 states

Multiclass classification problem

We again have a collection of ℓ data points $\{X^i, y^i\}_{i=1}^{\ell}$ where $X^i \equiv (1, x_1^i, x_2^i, \dots)$ is composed of dependent variables (features) but now y^i is a categorical independent variable (label) that takes K values. Our goal is to construct the lines that separate the classes 1, 2 and 3.

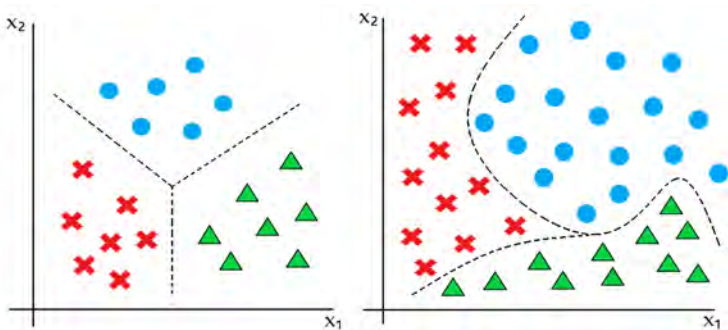


Figure 9. Examples of multiclass classification.

From multiclass to binary classification problem

- A popular approach in machine learning is to reformulate a multiclass classification problem as a collection of binary classification problems.
- The key assumption behind this approach is the hypothesis of an independence of irrelevant alternatives.
- In our analysis, that means that the choice between $\{\times\}$ and $\{\triangle\}$ is independent of the availability of $\{\circ\}$, the choice between $\{\triangle\}$ and $\{\circ\}$ is independent of the availability of $\{\times\}$ and the choice between $\{\circ\}$ and $\{\times\}$ is independent of the availability of $\{\triangle\}$.
- Two binary reformulations of a multiclass classification problems are the *one-versus-one* and *one-versus-rest* (or *one-versus-all*) classifiers,

$$\ln \frac{p(\times)}{p(\circ)} = X\theta^{(1)} \quad \ln \frac{p(\triangle)}{p(\circ)} = X\theta^{(2)} \quad \ln \frac{p(\triangle)}{p(\times)} = X\theta^{(3)},$$

$$\ln \frac{p(\times)}{p(\circ)+p(\triangle)} = X\theta^{(1)} \quad \ln \frac{p(\triangle)}{p(\circ)+p(\times)} = X\theta^{(2)} \quad \ln \frac{p(\circ)}{p(\triangle)+p(\times)} = X\theta^{(3)},$$

where $\theta^{(1)}$, $\theta^{(2)}$ and $\theta^{(3)}$ are the regression coefficients and X is the matrix of features.

Training multi class classifiers

- 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 $p(\times) = p(\circ) \exp(X\theta^{(1)})$ and $p(\Delta) = p(\circ) \exp(X\theta^{(2)})$ so that $p(\circ) \left(1 + \exp(X\theta^{(1)}) + \exp(X\theta^{(2)})\right) = 1$.
- In turn, for the one-versus-rest classifier, in the first regression, we replace $p(\circ) + p(\Delta)$ with $1 - p(\times)$ and in the second regression, we replace $p(\circ) + p(\times)$ with $1 - p(\Delta)$.

Consequently, we can re-write two classifiers as

$$p(\times) = \exp(X\theta^{(1)}) p(\circ), \quad p(\Delta) = \exp(X\theta^{(2)}) p(\circ) \quad ,$$
$$p(\circ) = \frac{1}{1 + \exp(X\theta^{(1)}) + \exp(X\theta^{(2)})},$$

$$p(\times) = \frac{1}{1 + \exp(X\theta^{(1)})} \quad p(\Delta) = \frac{1}{1 + \exp(X\theta^{(2)})} \quad p(\circ) = 1 - p(\times) - p(\Delta).$$

Symmetric one-versus-rest classifier

- Note that in the above expressions, we treat the normalizing class $\{\mathbf{o}\}$ differently from the other two classes $\{\Delta, \times\}$.
- 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 $\ln p(\times) = X\theta^{(1)}$, $\ln p(\Delta) = X\theta^{(2)}$, $\ln p(\mathbf{o}) = X\theta^{(3)}$ and by normalizing the exponential function ex-post by their sum.
- This classifier is called softmax and it is a generalization of a logistic function to multiple dimensions,

$$\begin{aligned} p(\times) &= \frac{1}{\Sigma} \exp\left(X\theta^{(1)}\right) \\ p(\Delta) &= \frac{1}{\Sigma} \exp\left(X\theta^{(2)}\right) \quad , \\ p(\mathbf{o}) &= \frac{1}{\Sigma} \exp\left(X\theta^{(3)}\right) \end{aligned}$$

where $\Sigma = \exp\left(X\theta^{(1)}\right) + \exp\left(X\theta^{(2)}\right) + \exp\left(X\theta^{(3)}\right)$.

- The symmetric treatment is convenient in deep learning analysis because it allows us to use a neural network with K symmetric outputs.

Likelihood function for softmax classifier

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

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

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 by using a gradient descent style method, $\theta \leftarrow \theta - \lambda \nabla \ln L_{\theta}(\theta)$.

Discrete choice in three state model

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

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

where $\varphi(\cdot)$ is a multilayer neural network parameterized by a vector of coefficients θ (weights and biases), $\Sigma \equiv p_t^i(\bar{n}) + p_t^i(\underline{n}) + p_t^i(0)$ 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 $\frac{p_t^i(\bar{n})}{\Sigma}$, $\frac{p_t^i(\underline{n})}{\Sigma}$ and $\frac{p_t^i(0)}{\Sigma}$ are bounded in the interval $[0, 1]$, and ζ_0 is a constant term. (In addition, we also parameterize the Lagrange multiplier).

Verifying the optimality conditions

- 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 $\frac{p_t^i(\bar{n})}{\Sigma}$, $\frac{p_t^i(\underline{n})}{\Sigma}$, $\frac{p_t^i(0)}{\Sigma}$, $\frac{c_t^i}{w_t^i}$ 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 \hat{n}_t^i implied by such value functions,

$$\hat{n}_t^i = \begin{cases} \bar{n} & \text{if } V^E = \max \{V^E, V^{PT}, V^U\}, \\ \underline{n} & \text{if } V^{PT} = \max \{V^E, V^{PT}, V^U\}, \\ 0 & \text{otherwise.} \end{cases}$$

- In the solution, the labor choice implied by the value function \hat{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.

Training the model

- To this purpose, we construct the categorical variable $y_t^i \equiv (y_t^{i,1}, y_t^{i,2}, y_t^{i,3})$ such that

$$y_t^i = \begin{cases} (1, 0, 0) & \text{if } \hat{n}_t^i = \bar{n}, \\ (0, 1, 0) & \text{if } \hat{n}_t^i = \underline{n}, \\ (0, 0, 1) & \text{otherwise.} \end{cases}$$

- We then formulate the (log)likelihood function

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

- We train the model to maximize the likelihood function by using a conventional / stochastic / batch stochastic gradient descent method.
- We iterate on the decision functions $p_t^i(\bar{n})$, $p_t^i(\underline{n})$, $p_t^i(0)$, $\frac{c_t^i}{w_t^i}$ and μ_t^i until convergence.

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:

$$\hat{n}_t^i = \begin{cases} \bar{n} & \text{if } L \square \left[\frac{c_i^\square W_t \exp(v_t^i)}{B} \right]^\square 1/\eta \geq \bar{n}_f \\ \underline{n} & \text{if } L \square \left[\frac{c_i^\square W_t \exp(v_t^i)}{B} \right]^\square 1/\eta \in [\bar{n}_p, \bar{n}_f] \\ 0 & \text{otherwise} \end{cases}$$

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

Algorithm 3: Deep learning for model with full and part-time employment

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

Step 0: (Initialization).

Construct initial state $\left(\{k_0^i, v_0^i\}_{i=1}^{\ell}, z_0\right)$ and parameterize the decision functions by $\left\{\frac{p_t^i(\bar{n})}{\Sigma}, \frac{p_t^i(\underline{n})}{\Sigma}, \frac{p_t^i(0)}{\Sigma}, \frac{c_t^i}{w_t^i}\right\} = \sigma\left(\zeta_0 + \varphi\left(k_t^i, v_t^i, \{k_t^i, v_t^i\}_{i=1}^{\ell}, z_t; \theta\right)\right)$, where $p_t^i(\bar{n})$, $p_t^i(\underline{n})$ and $p_t^i(0)$ are the probabilities to be full- and part-time employed and unemployed, respectively; and $\Sigma \equiv p_t^i(\bar{n}) + p_t^i(\underline{n}) + p_t^i(0)$ is a normalization of probability to one.

Algorithm 3: Deep learning for model with full and part-time employment (cont.)

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

Step 1: (Evaluation of decision functions).

Given state $\left(k_t^i, v_t^i, \{k_t^i, v_t^i\}_{i=1}^{\ell}, z_t\right)$, set $n_t^i = \bar{n}$, $n_t^i = \underline{n}$ and $n_t^i = 0$ depending on which probability $p_t^i(\bar{n})$, $p_t^i(\underline{n})$ and $p_t^i(0)$ is the largest. Compute $w_t^i, \frac{c_t^i}{w_t^i}$ from the decision rules and find k_{t+1}^i from the budget constraint for all agents $i = 1, \dots, \ell$. Reconstruct V^E , V^{PT} and V^U , respectively.

Find $\hat{n}_t^i = \begin{cases} \bar{n} & \text{if } V^E = \max\{V^E, V^{PT}, V^U\}, \\ \underline{n} & \text{if } V^{PT} = \max\{V^E, V^{PT}, V^U\}, \\ 0 & \text{otherwise.} \end{cases}$

and define $y_t^i = \begin{cases} (1, 0, 0) & \text{if } \hat{n}_t^i = \bar{n}, \\ (0, 1, 0) & \text{if } \hat{n}_t^i = \underline{n}, \\ (0, 0, 1) & \text{otherwise.} \end{cases}$ for each s_t^i .

Algorithm 3: Deep learning for model with full and part-time employment (cont)

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

Option 1: Construct V^E, V^{PT}, V^U and $\hat{n}_t^i = \begin{cases} \bar{n} & \text{if } V^E = \max \{V^E, V^{PT}, V^U\} \\ \underline{n} & \text{if } V^{PT} = \max \{V^E, V^{PT}, V^U\} \\ 0 & \text{otherwise.} \end{cases}$

Option 2: From discretized FOC $\hat{n}_t^i = \begin{cases} \bar{n} & \text{if } L \square \left[\frac{c_i^\square W_t \exp(v_t^i)}{B} \right]^{\square 1/\eta} \geq \bar{n}_f \\ \underline{n} & \text{if } L \square \left[\frac{c_i^\square W_t \exp(v_t^i)}{B} \right]^{\square 1/\eta} \in [\bar{n}_p, \bar{n}_f] \\ 0 & \text{otherwise} \end{cases}$

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

Algorithm 3: Deep learning for model with full and part-time employment (cont)

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

Step 2: (Construction of Euler residuals).

Draw two random sets of individual productivity shocks $\Sigma_1 = (\epsilon_1^1, \dots, \epsilon_1^\ell)$, $\Sigma_2 = (\epsilon_2^1, \dots, \epsilon_2^\ell)$ and two aggregate shocks ϵ_1, ϵ_2 , and construct the residuals

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

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).

Training errors and running time

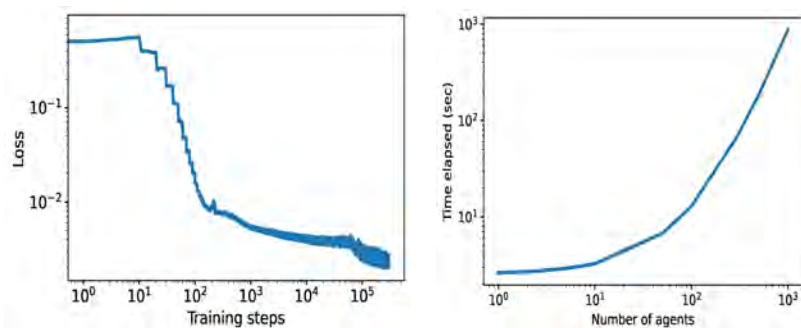


Figure 10. Training errors and running time for three-state employment m

The solution for divisible labor model

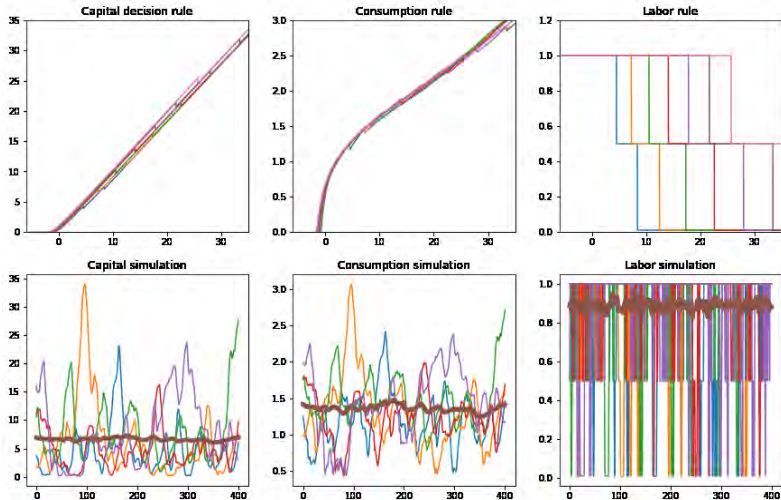


Figure 11. Solution to the three-state employment model.

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.

Machine learning, artificial intelligence and deep learning methods for dynamic economic models:

Conclusion

Lilia Maliar, Serguei Maliar

November 2022

Minicourse

Conclusion: Enthusiastic view

Data scientists:

- developed powerful AI techniques for data analysis (e.g., deep learning)
- created software for their efficient implementation (e.g., TensorFlow)

Economists mostly develop own method & codes

- We show how to cast the broad class of rational-expectation models into the form suitable for the state-of-the-art DL methods and software
- We hope that this approach augments vastly the set of models that is tractable in economics

Grain of salt, black magic and no-free-lunch (1)

- We offer AI-based technology but do not know whether it is best possible for all economic models.

First, it is not clear if **neural network** is useful or not.

Several reasons to doubt it:

- Neural networks are promising and universal approximators but have a large number of parameters and are highly non-linear.
- Convergence to global solution is not guaranteed.
- Solutions in economic models are often well described by simple polynomials.

Grain of salt, black magic and no-free-lunch (2)

Second, **Monte Carlo integration** lying in the basis of AI technology is simple but has a low, square-root rate of convergence.

- Mindless drawing of random number is not necessarily the best.
- More efficient outcomes can be obtained if we engineer better sequences or grids to enhance interpolation or integration results.

Grain of salt, black magic and no-free-lunch (3)

Third, **stochastic optimization** is magical: it replaces numerical approximation of integral over a large number of points with an evaluation of its integrand in few of such points.

- But there is no free lunch: its convergence rate is slower and not guaranteed.
- The latter method is more expensive than stochastic gradient and may be intractable for complex approximating functions.

Grain of salt, black magic and no-free-lunch (4)

Finally, there is a variety of AI technologies that can be used for solving dynamic economic models.

- Alphazero plays Go but not chess and Deepblue plays chess but not Go.
- Again, there is no free lunch: for one application where a method works, there is another where it does not.
- The idea that there is an all-purpose procedure, which will remove the curse of dimensionality, without any trade-offs, for any kind of models, and will produce highly precise and uniformly accurate solution belongs to magical thinking.
- The human input is still critical to see trade-offs

Computational economists may still be useful

- The bottom line is that we probably should not retire all computational economists at the moment.
- At least, not until they figure out what type of AI works best for economic models.

Thank you!