Deep Learning for Macroeconomists

Jesús Fernández-Villaverde[1]

November 4, 2021

[1]University of Pennsylvania

Background

- Presentation based on joint work with different coauthors:

  1. Solving High-Dimensional Dynamic Programming Problems using Deep Learning, with Galo Nuño, Roberto Rafael Maura, George Sorg-Langhans, and Maximilian Vogler.
  2. Exploiting Symmetry in High-Dimensional Dynamic Programming, with Mahdi Ebrahimi Kahou, Jesse Perla, and Arnav Sood.
  3. Financial Frictions and the Wealth Distribution, with Galo Nuño and Samuel Hurtado.
  4. Programming FPGAs for Economics, with Bhagath Cheela, André DeHon, and Alessandro Peri.
  5. Structural Estimation of Dynamic Equilibrium Models with Unstructured Data, with Sara Casella and Stephen Hansen.

- All the papers share a common thread: how to compute and take to the data the aggregate dynamics of models with heterogeneous agents.

Resources

- These slides are available at:

  https://www.sas.upenn.edu/~jesusfv/deep-learning_chicago.pdf

- My teaching slides:

  https://www.sas.upenn.edu/~jesusfv/Continuous_Time_2.pdf

- Examples and code at:

  1. https://colab.research.google.com/drive/1_4wL6lqA-BsgGWjDZv4J7UtnuZk6TpqW?usp=sharing
  2. https://github.com/jesusfv/financial-frictions

Deep learning as a tool for solving models

- Solving models in macro (and I.O., international trade, finance, game theory, ...) is, at its core, a functional approximation problem.

- Given some states $\mathbf{x} = \{x_1, x_2, ..., x_N\}$, we want to compute a function (or, more generally, an operator):

$$y = h(\mathbf{x})$$

  such as a value function, a policy function, a best response function, a pricing kernel, an allocation, a probability distribution, ...

- We need to find a function that satisfies some optimality/equilibrium conditions.

- Usually, we do not know much about the functional form of $h(\cdot)$ or $\mathbf{x}$ is highly dimensional.

- Deep learning provides an extremely powerful framework to work through these problems.

Why is deep learning is a great functional approximation tool?

- Theory reasons:

  1. Deep learning is a universal nonlinear approximator. For example, it can tackle correspondences (e.g., multiplicity of equilibria).

  2. Deep learning breaks the "curse of dimensionality" (compositional approximations that require only "local" computations instead of additive ones).

- Practical reasons $\Rightarrow$ deep learning involves algorithms that are:

  1. Easy to code.

  2. Implementable with state-of-the-art libraries.

  3. Stable.

  4. Scalable through massive parallelization.

  5. Can take advantage of dedicated hardware.

A basic example

- Take the canonical RBC model:

$$\max \mathbb{E}_0 \sum_{t=0}^{\infty} \beta^t u\left(c_t, l_t\right)$$

$$c_t + k_{t+1} = e^{z_t} k_t^{\alpha} l_t^{1-\alpha} + \left(1-\delta\right) k_t, \ \forall \ t > 0$$

$$z_t = \rho z_{t-1} + \sigma \varepsilon_t, \ \varepsilon_t \sim \mathcal{N}(0,1)$$

- Examples of objects we are interested in approximating:

1. Decision rules: $c_t = h(k_t, z_t)$.
2. Conditional expectations: $h(k_t, z_t) = \mathbb{E}_t \left\{ \frac{u'(c_{t+1}, l_{t+1})}{u'(c_t, l_t)} \left(1 + \alpha e^{z_{t+1}} k_{t+1}^{\alpha-1} l_{t+1}^{1-\alpha} - \delta\right) \right\}$.
3. Value functions: $h\left(k_t, z_t\right) = \max_{\{c_t, l_t\}} \left\{ u\left(c_t, l_t\right) + \beta \mathbb{E}_t h\left(k_{t+1}, z_{t+1}\right) \right\}$.

A parameterized solution

- General idea: substitute $h(\mathbf{x})$ by $h^j(\mathbf{x},\theta)$ where θ is a vector of coefficients to be determined by satisfying some criterium indexed by j.
- Two classical approaches based on the addition of functions:

1. Perturbation methods:

$$h^{PE}(\mathbf{x},\theta) = \vec{\theta_0} + \vec{\theta_1}(\mathbf{x}-\mathbf{x_0}) + (\mathbf{x}-\mathbf{x_0})'\vec{\theta_2}(\mathbf{x}-\mathbf{x_0}) + H.O.T.$$

   We use implicit-function theorems to find θ.

2. Projection methods:

$$h^{PR}(\mathbf{x},\theta) = \theta_0 + \sum_{m=1}^{M} \theta_m \phi_m(\mathbf{x})$$

   where ϕ_m is, for example, a Chebyshev polynomial.

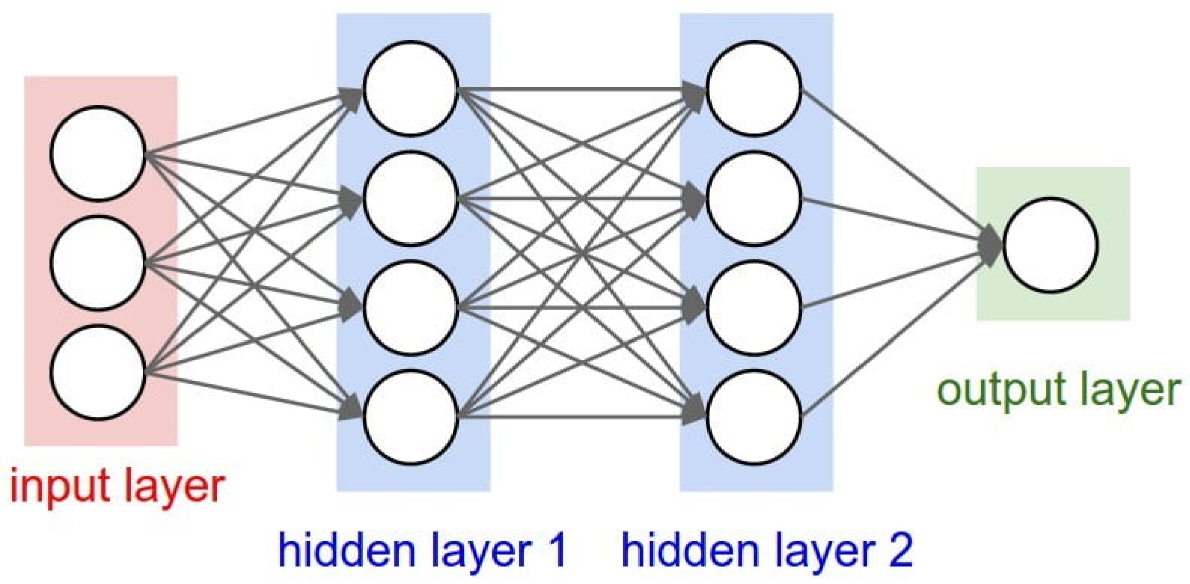
   We pick a basis $\{\phi_m(\mathbf{x})\}_{i=0}^{\infty}$ and "project" the optimality/equilibrium conditions against that basis to find θ.

A neural network

- A (one-layer) neural network approximates $h(\mathbf{x})$ by using M times an activation function $\varphi(\cdot)$:

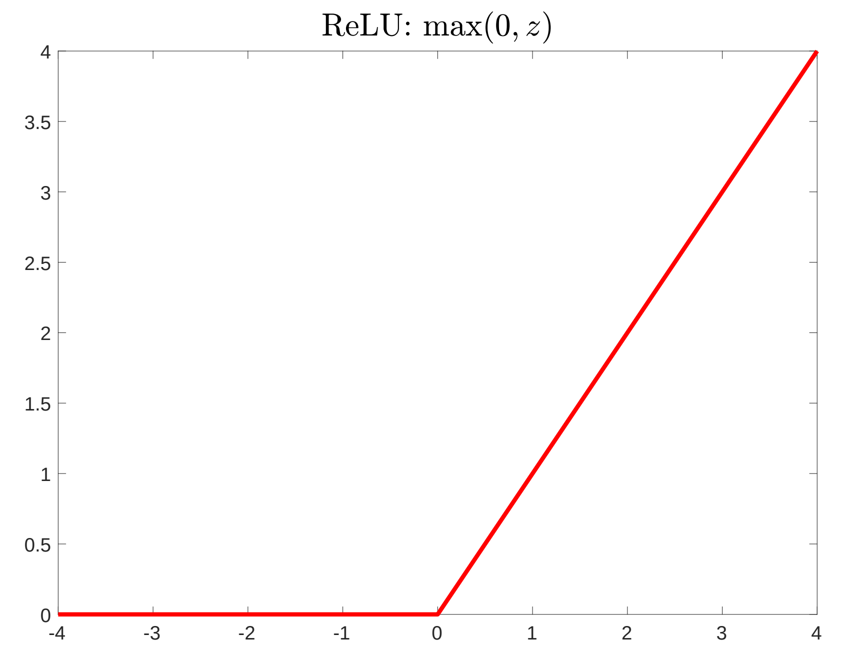
$$y = h(\mathbf{x}) \cong h^{NN}(\mathbf{x}; \theta) = \theta_0 + \sum_{m=1}^{M} \theta_m \varphi \left(\underbrace{\theta_{0,m} + \sum_{n=1}^{N} \theta_{n,m} x_n}_{z_m} \right)$$

- M is the width of the network.
- We can add more layers (i.e., x_n is transformed into x_n^1 by a similar composition of an activation function, and so on), but notation becomes heavy.
- The number of layers J is the depth of the network.
- We select θ such that $h^{NN}(\mathbf{x}; \theta)$ is as close to $h(\mathbf{x})$ as possible given some relevant metric (e.g., L^2).
- This is called "training" the network (a lot of details need to be filled in here!).


input layer
hidden layer 1
hidden layer 2
output layer

Architecture of the network

- We pick the simplest activation function possible:
  - Easier to take derivatives (key while training the network).
- We select M and J following the same ideas that one uses to select how many grid points we use in value function iteration or how many polynomials with Chebyshev polynomials:
  1. We start with some default M and J and train the network. We assess approximation error.
  2. We vary M and J until the approximation error is minimized.


ReLU: max(0, z)
4
3.5
3
2.5
2
1.5
1
0.5
0
-4
-3
-2
-1
0
1
2
3
4

Two classic (yet remarkable) results

Universal approximation theorem: Hornik, Stinchcombe, and White (1989)

A neural network with at least one hidden layer can approximate any Borel measurable function mapping finite-dimensional spaces to any desired degree of accuracy.

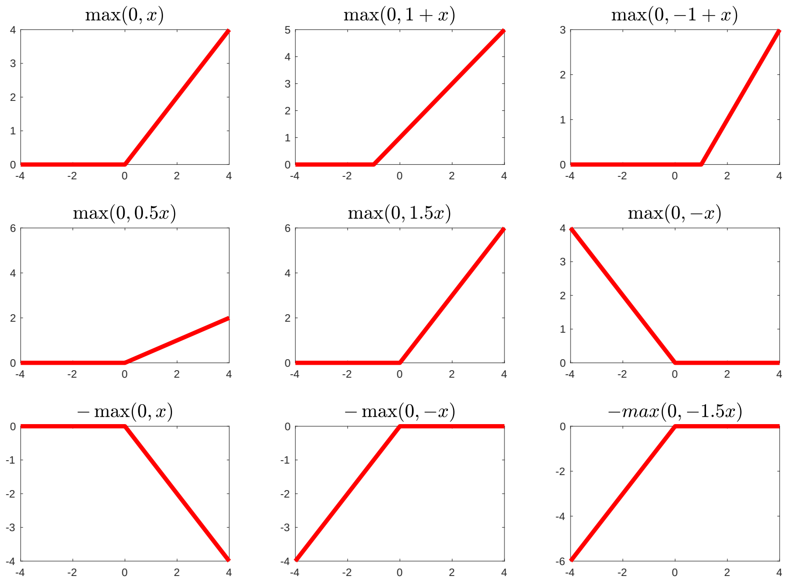
- Under some additional technical conditions:

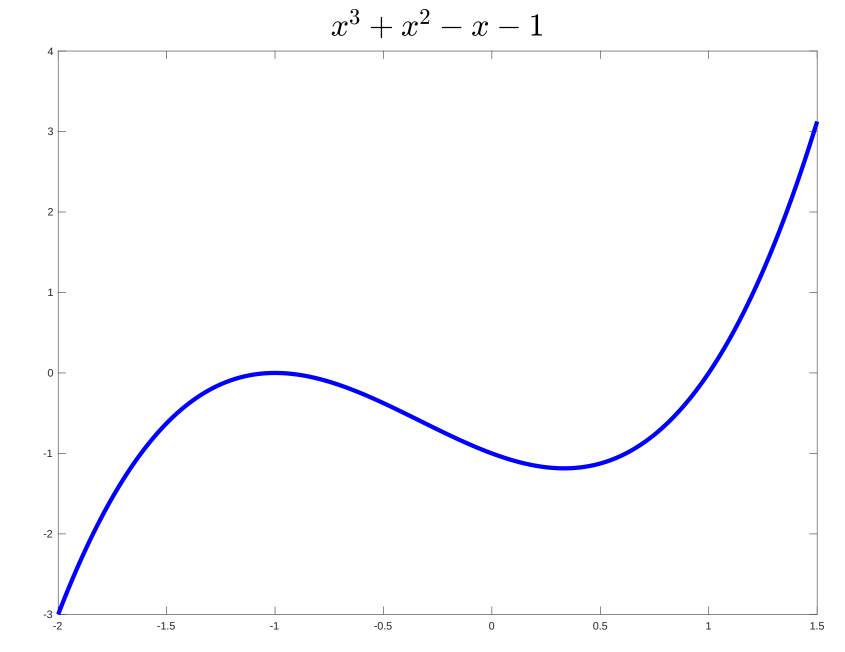
Breaking the curse of dimensionality: Barron (1993)

A one-layer neural network achieves integrated square errors of order $\mathcal{O}(1/M)$, where M is the number of nodes. In comparison, for series approximations, the integrated square error is of order $\mathcal{O}(1/(M^{2/N}))$ where N is the dimensions of the function to be approximated.

- We can rely on more general theorems by Leshno et al. (1993) and Bach (2017).

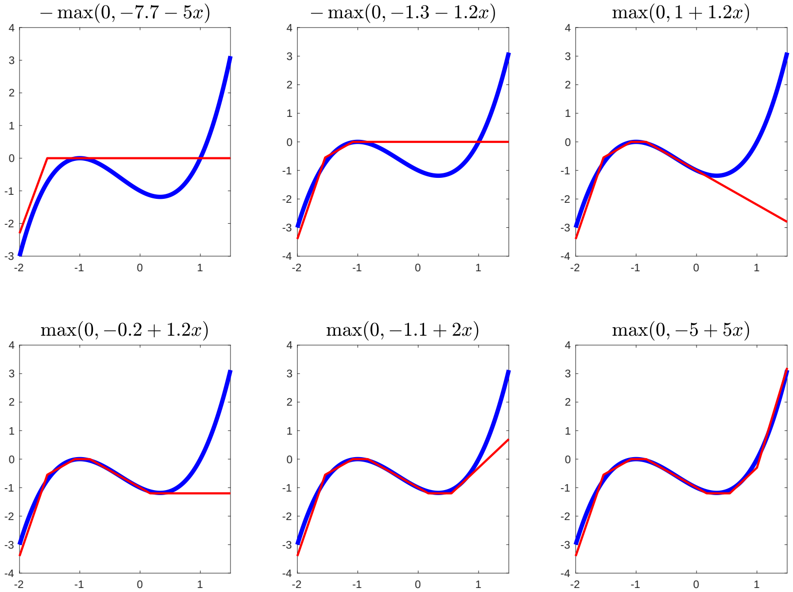
Different ReLUs: $\theta_i \max(0, \theta_{i,0} + \theta_{i,1} x)$





$x^3 + x^2 - x - 1$
4
3
2
1
0
-1
-2
-3
-2
-1.5
-1
-0.5
0
0.5
1
1.5

A six ReLUs approximation



Practical reasons

- Deep learning involves algorithms that are:

    1. Easy to code.
    2. Implementable with state-of-the-art libraries.
    3. Stable.
    4. Scalable through massive parallelization.
    5. Can take advantage of dedicated hardware.

PyTorch Lightning

PyTorch

scikit learn

Keras

fast.ai

TensorFlow

Programming field-programmable gate arrays for economics

Dynamic programming

Solving high-dimensional dynamic programming problems using Deep Learning

- Solving High-Dimensional Dynamic Programming Problems using Deep Learning.
- Our goal is to solve the recursive continuous-time Hamilton-Jacobi-Bellman (HJB) equation globally:

$$\rho V(\mathbf{x}) = \max_{\boldsymbol{\alpha}} r(\mathbf{x}, \boldsymbol{\alpha}) + \nabla_x V(\mathbf{x}) f(\mathbf{x}, \boldsymbol{\alpha}) + \frac{1}{2} tr(\sigma(\mathbf{x}))^T \Delta_x V(\mathbf{x}) \sigma(\mathbf{x}))$$

$$\text{s.t.} \quad G(\mathbf{x}, \boldsymbol{\alpha}) \leq \mathbf{0} \quad \text{and} \quad H(\mathbf{x}, \boldsymbol{\alpha}) = \mathbf{0},$$

- Think about the case where we have many state variables.
- Why continuous time?
- Alternatives for this solution?

- We define four neural networks:
  1. $\tilde{V}(\mathbf{x}; \mathbf{\Theta}^V)$ to approximate the value function $V(\mathbf{x})$.
  2. $\tilde{\alpha}(\mathbf{x}; \mathbf{\Theta}^\alpha)$ to approximate the policy function $\boldsymbol{\alpha}$.
  3. $\tilde{\mu}(\mathbf{x}; \mathbf{\Theta}^\mu)$ and $\tilde{\lambda}(\mathbf{x}; \mathbf{\Theta}^\lambda)$ to approximate the Karush-Kuhn-Tucker (KKT) multipliers $\boldsymbol{\mu}$ and $\boldsymbol{\lambda}$.
- To simplify notation, we accumulate all weights in the matrix $\mathbf{\Theta} = (\mathbf{\Theta}^V, \mathbf{\Theta}^\alpha, \mathbf{\Theta}^\mu, \mathbf{\Theta}^\lambda)$.

Error criterion I

- The HJB error:

$$err_{HJB}(\mathbf{x};\boldsymbol{\Theta}) \equiv r(\mathbf{x},\tilde{\alpha}(\mathbf{s};\boldsymbol{\Theta}^{\alpha})) + \nabla_x \tilde{V}(\mathbf{x};\boldsymbol{\Theta}^{V}) f(\mathbf{x},\tilde{\alpha}(\mathbf{x};\boldsymbol{\Theta}^{\alpha})) +$$

$$+ \frac{1}{2} tr[\sigma(\mathbf{x})^T \Delta_x \tilde{V}(\mathbf{x};\boldsymbol{\Theta}^{V}) \sigma(\mathbf{x})] - \rho \tilde{V}(\mathbf{x};\boldsymbol{\Theta}^{V})$$

- The policy function error:

$$err_{\alpha}(\mathbf{x};\boldsymbol{\Theta}) \equiv \frac{\partial r(\mathbf{x},\tilde{\alpha}(\mathbf{x};\boldsymbol{\Theta}^{\alpha}))}{\partial \boldsymbol{\alpha}} + D_{\alpha} f(\mathbf{x},\tilde{\alpha}(\mathbf{x};\boldsymbol{\Theta}^{\alpha}))^T \nabla_x \tilde{V}(\mathbf{x};\boldsymbol{\Theta}^{V})$$

$$- D_{\alpha} G(\mathbf{x},\tilde{\alpha}(\mathbf{x};\boldsymbol{\Theta}^{\alpha}))^T \tilde{\mu}(\mathbf{x};\boldsymbol{\Theta}^{\mu}) - D_{\alpha} H(\mathbf{x},\tilde{\alpha}(\mathbf{x};\boldsymbol{\Theta}^{\alpha})) \tilde{\lambda}(\mathbf{x};\boldsymbol{\Theta}^{\lambda}),$$

where $D_{\alpha} G \in \mathbb{R}^{L_1 \times M}$, $D_{\alpha} H \in \mathbb{R}^{L_2 \times M}$, and $D_{\alpha} f \in \mathbb{R}^{N \times M}$ are the submatrices of the Jacobian matrices of G, H and f respectively containing the derivatives with respect to α.

Error criterion II

- The constraint error is itself composed of the primal feasibility errors:

$$err_{PF_1}(\mathbf{x};\boldsymbol{\Theta}) \equiv \max\{0, G(\mathbf{x},\tilde{\alpha}(\mathbf{x};\boldsymbol{\Theta}^{\alpha}))\}$$
$$err_{PF_2}(\mathbf{x};\boldsymbol{\Theta}) \equiv H(\mathbf{x},\tilde{\alpha}(\mathbf{x};\boldsymbol{\Theta}^{\alpha})),$$

  the dual feasibility error:

$$err_{DF}(\mathbf{x};\boldsymbol{\Theta}) = \max\{0, -\tilde{\mu}(\mathbf{x};\boldsymbol{\Theta}^{\mu}\},$$

  and the complementary slackness error:

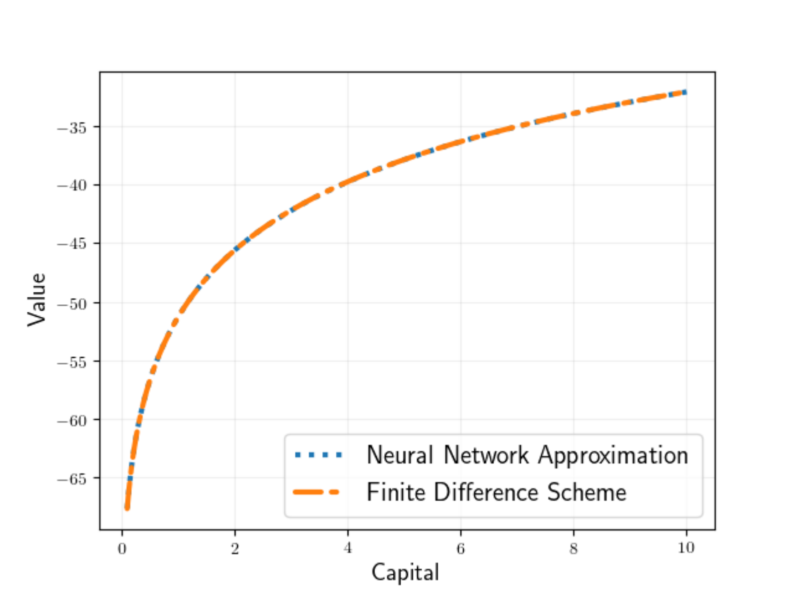
$$err_{CS}(\mathbf{x};\boldsymbol{\Theta}) = \tilde{\mu}(\mathbf{x};\boldsymbol{\Theta})^{T} G(\mathbf{x},\tilde{\alpha}(\mathbf{x};\boldsymbol{\Theta}^{\alpha})).$$

- We combine these four errors by using the squared error as our loss criterion:

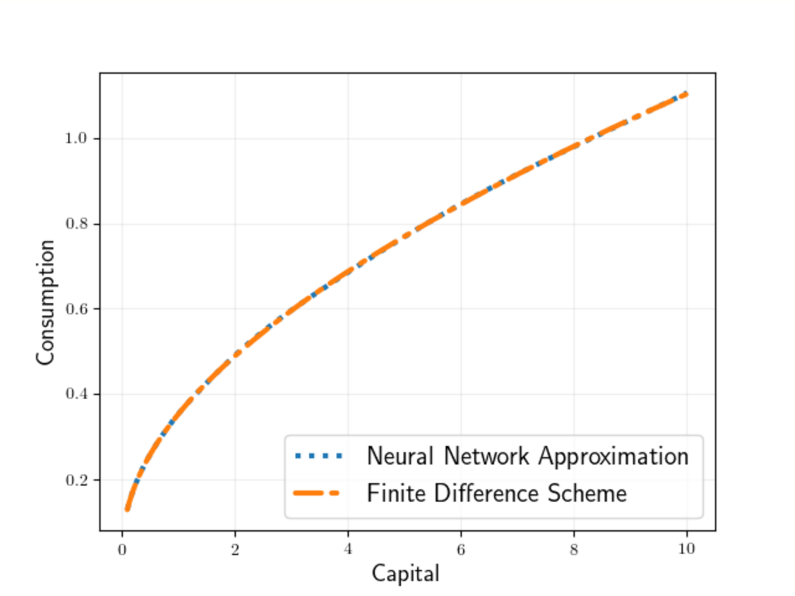
$$\mathcal{E}(\mathbf{x};\boldsymbol{\Theta}) \equiv \left|\left|err_{HJB}(\mathbf{x};\boldsymbol{\Theta})\right|\right|_2^2 + \left|\left|err_{\alpha}(\mathbf{x};\boldsymbol{\Theta})\right|\right|_2^2 + \left|\left|err_{PF_1}(\mathbf{x};\boldsymbol{\Theta})\right|\right|_2^2 + $$
$$+ \left|\left|err_{PF_2}(\mathbf{x};\boldsymbol{\Theta})\right|\right|_2^2 + \left|\left|err_{DF}(\mathbf{x};\boldsymbol{\Theta})\right|\right|_2^2 + \left|\left|err_{CS}(\mathbf{x};\boldsymbol{\Theta})\right|\right|_2^2$$

Training

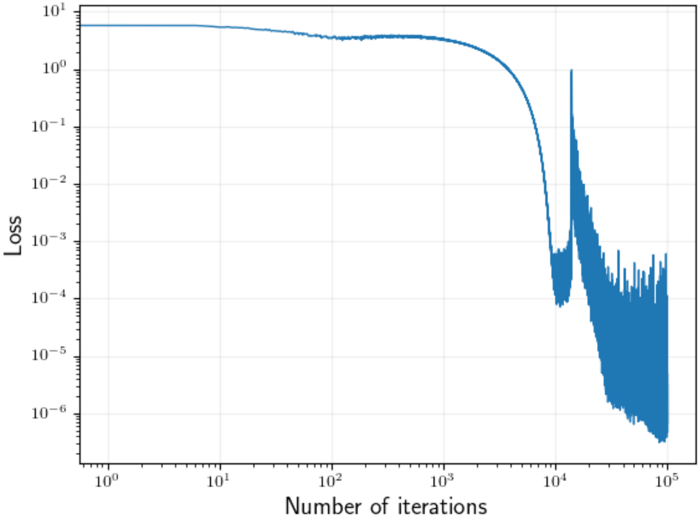
- We train our neural networks by minimizing the error criterion through mini-batch gradient descent over points drawn from the ergodic distribution of the state vector.
- We start by initializing our network weights, and we perform K learning steps called epochs.
- For each epoch, we draw I points from the state space by simulating from the ergodic distribution.
- Then, we randomly split this sample into B mini-batches of size S. For each mini-batch, we define the mini-batch error, by averaging the loss function over the batch.
- Finally, we perform mini-batch gradient descent for all network weights, with η_k being the learning rate in the k-th epoch.



(a) Value with closed-form policy



(c) Consumption with closed-form policy



(e) HJB error with closed-form policy

A coda

- Exploiting Symmetry in High-Dimensional Dynamic Programming.
- We introduce the idea of permutation-invariant dynamic programming.
- Intuition.
- Solution has a symmetry structure we can easily exploit using representation theorems, concentration of measure, and neural networks.
- More in general: how do we tackle models with heterogeneous agents?

Models with heterogeneous agents

The challenge

- To compute and take to the data models with heterogeneous agents, we need to deal with:

  1. The distribution of agents G_t.

  2. The operator $H(\cdot)$ that characterizes how G_t evolves:

     $$G_{t+1} = H(G_t, S_t)$$

     or

     $$\frac{\partial G_t}{\partial t} = H(G_t, S_t)$$

     given the other aggregate states of the economy S_t.

- How do we track G_t and compute $H(G_t, S_t)$?

A common approach

- If we are dealing with N discrete types, we keep track of $N - 1$ weights.
- If we are dealing with continuous types, we extract a finite number of features from G_t:
    1. Moments.
    2. Q-quantiles.
    3. Weights in a mixture of normals...
- We stack either the weights or features of the distribution in a vector μ_t.
- We assume μ_t follows the operator $h(\mu_t, S_t)$ instead of $H(G_t, S_t)$.
- We parametrize $h(\mu_t, S_t)$ as $h^j(\mu_t, S_t; \theta)$.
- We determine the unknown coefficients θ such that an economy where μ_t follows $h^j(\mu_t, S_t; \theta)$ replicates as well as possible the behavior an economy where G_t follows $H(\cdot)$.

Example: Basic Krusell-Smith model

- Two aggregate variables: aggregate productivity shock and household distribution $G_t(a, z)$ where:

$$\int G_t(a, z)da = K_t$$

- We summarize $G_t(a, \cdot)$ with the log of its mean: $\mu_t = \log K_t$ (extending to higher moments is simple, but tedious).

- We parametrize $\underbrace{\log K_{t+1}}_{\mu_{t+1}} = \underbrace{\theta_0(s_t) + \theta_1(s_t)\log K_t}_{h^j(\mu_t, s_t; \theta)}$.

- We determine $\{\theta_0(s_t), \theta_1(s_t)\}$ by OLS run on a simulation.

Problems

- No much guidance regarding feature and parameterization selection in general cases.
  - Yes, keeping track of the log of the mean and a linear functional form work well for the basic model. But, what about an arbitrary model?
  - Method suffers from "curse of dimensionality": difficult to implement with many state variables or high N/higher moments.
  - Lack of theoretical foundations (Does it converge? Under which metric?).

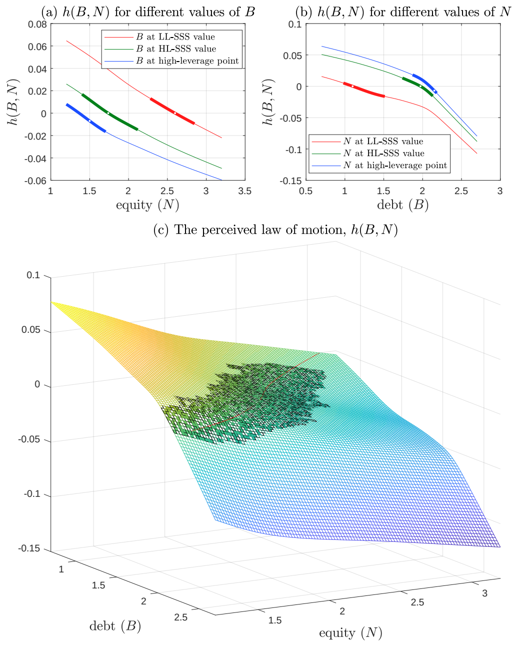
How can deep learning help?

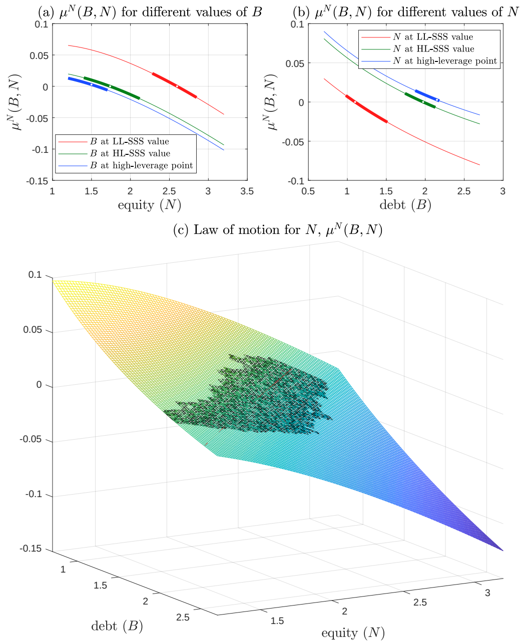
- Deep learning addresses challenges:
  1. How to extract features from an infinite-dimensional object efficiently.
  2. How to parametrize the non-linear operator mapping how distributions evolve.
  3. How to tackle the "curse of dimensionality."
- Given time limitations, today I will discuss the last two points.
- In our notation of $y = h(\mathbf{x})$:
  1. $y = \mu_{t+1}$.
  2. $\mathbf{x} = (\mu_t, S_t)$.

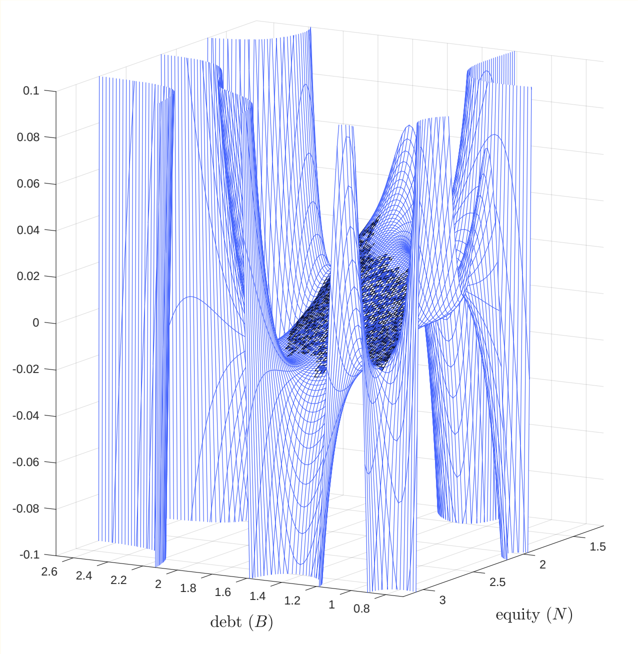
The algorithm in "Financial Frictions and the Wealth Distribution"

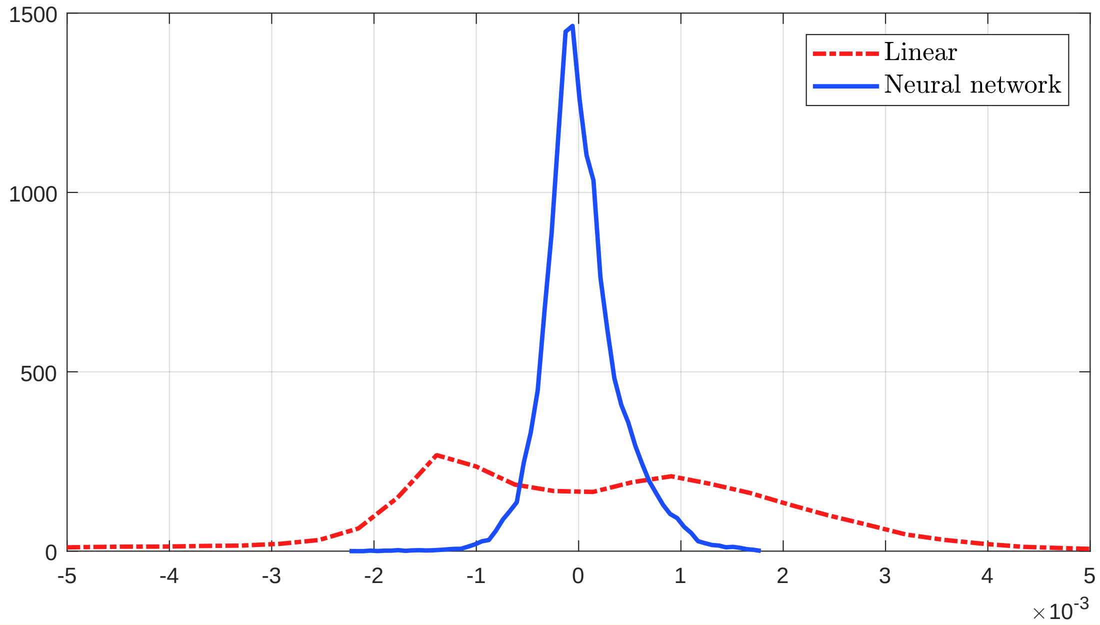
- Model with a distribution of households over asset holding and labor productivity.
- We keep track, as in Krusell-Smith, of moments of distribution.
- Algorithm:

  1) Start with h_0, an initial guess for h.
  2) Using current guess h_n, solve for agent's problem.
  3) Construct time series $\mathbf{x} = \{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_J\}$ and $\widehat{\mathbf{y}} = \{\widehat{\mathbf{y}}_1, \widehat{\mathbf{y}}_2, ..., \widehat{\mathbf{y}}_J\}$ for aggregate variables by simulating J periods the cross-sectional distribution of agents (starting at a steady state and with a burn-in).
  4) Use $(\widehat{\mathbf{y}}, \mathbf{x})$ to train h_{n+1}, a new guess for h.
  5) Iterate steps 2)-4) until h_{n+1} is sufficiently close to h_n.


(a) $h(B,N)$ for different values of B
B at LL-SSS value
B at HL-SSS value
B at high-leverage point
$h(B,N)$
equity (N)
(b) $h(B,N)$ for different values of N
N at LL-SSS value
N at HL-SSS value
N at high-leverage point
$h(B,N)$
debt (B)
(c) The perceived law of motion, $h(B,N)$
debt (B)
equity (N)


(a) $\mu^N(B,N)$ for different values of B
$\mu^N(B,N)$
equity (N)
B at LL-SSS value
B at HL-SSS value
B at high-leverage point
0.1
0.05
0
-0.05
-0.1
-0.15
1
1.5
2
2.5
3
3.5
(b) $\mu^N(B,N)$ for different values of N
$\mu^N(B,N)$
debt (B)
N at LL-SSS value
N at HL-SSS value
N at high-leverage point
0.1
0.05
0
-0.05
-0.1
0.5
1
1.5
2
2.5
3
(c) Law of motion for N, $\mu^N(B,N)$
0.1
0.05
0
-0.05
-0.1
-0.15
debt (B)
1
1.5
2
2.5
equity (N)
1.5
2
2.5
3

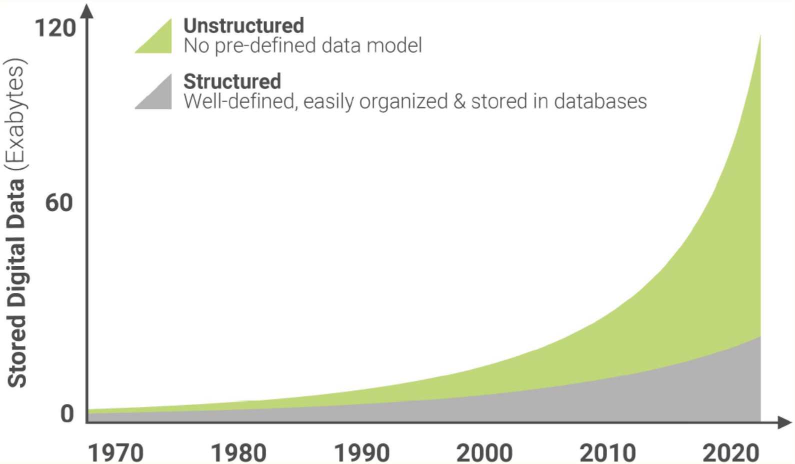

0.1
0.08
0.06
0.04
0.02
0
-0.02
-0.04
-0.06
-0.08
-0.1
2.6
2.4
2.2
2
1.8
1.6
1.4
1.2
1
0.8
debt (B)
1.5
2
2.5
3
equity (N)


Linear
Neural network
1500
1000
500
0
-5
-4
-3
-2
-1
0
1
2
3
4
5
$\times 10^{-3}$

Structural estimation with unstructured data

New data

- Unstructured data: Newspaper articles, business reports, congressional speeches, FOMC meetings transcripts, satellite data, ...



Motivation

- Unstructured data carries information on:
    1. Current state of the economy (Thorsrud, 2017, Bybee et al., 2019).
    2. Beliefs about current and future states of the economy.
- An example:

From the minutes of the FOMC meeting of September 17-18, 2019

Participants agreed that consumer spending was increasing at a strong pace. They also expected that, in the period ahead, household spending would likely remain on a firm footing, supported by strong labor market conditions, rising incomes, and accommodative financial conditions. [...]

Participants judged that trade uncertainty and global developments would continue to affect firms' investment spending, and that this uncertainty was discouraging them from investing in their businesses. [...]

Our goal

- Since:

  1. This information might go over and above observable macro series (e.g., agents' expectations and sentiment).

  2. And it might go further back in history, is available for developing countries, or in real time.

- **How do we incorporate unstructured data in the estimation of structural models?**

- Potential rewards:

  1. Determine more accurately the latent structural states.

  2. Reconcile agents' behavior and macro time series.

  3. Could change parameters values (medium-scale DSGE models typically poorly identified).

Application and data

- Our application:

  Text Data: Federal Open Market Committee (FOMC) meeting transcripts.

  Model: New Keynesian dynamic stochastic general equilibrium (NK-DSGE) model.

- Our strategy:

  Right Now:
  1. Latent Dirichlet Allocation (LDA) for dimensionality reduction $\Rightarrow$ from words to topic shares.
  2. Cast the linearized DSGE solution in a state-space form.
  3. Use LDA output as additional observables in the measurement equation.
  4. Estimation with Bayesian techniques.

  Going Forward: Model the data generating process for text and macroeconomic data *jointly*.

Preliminary findings

1. Using FOMC data for estimation sharpens the likelihood.
2. Posterior distributions more concentrated.
3. Especially true for parameters related to the hidden states of the economy and to fiscal policy.
4. FOMC data carries extra information about fiscal policy and government intentions

Latent Dirichlet Allocation (LDA) ‣ LDA Details

- **How does it work?**
  - LDA is a Bayesian statistical model.
  - Idea: (i) **each document is described by a distribution of K (latent) topics** and (ii) **each topic is described by a distribution of words**.
  - Use word co-occurrence patterns + priors to assign probabilities.
  - Key of LDA dimensionality reduction **topic shares** $\varphi_t \Rightarrow$ amount of time document spends talking about each topic k.
- **Why do we like it?**
  - Tracks well attention people devote to different topics.
  - Automated and easily scalable.
  - Bayesian model natural to combine with structural models.

DSGE state space representation

Log-linearized DSGE model solution has the form of a generic state-space model:

- Transition equation:

$$\underbrace{s_{t+1}}_{\text{Structural States}} = \Phi_1(\theta)s_t + \Phi_\epsilon(\theta)\epsilon_t, \quad \epsilon_t \sim N(0, I)$$

- Measurement equation:

$$\underbrace{Y_t}_{\text{Macroeconomic Observables}} = \Psi_0(\theta) + \Psi_1(\theta)s_t$$

- θ vector that stacks all the structural parameters.

▸ System Matrices

Topic dynamic factor model

- Allow the topic time series φ_t to depend on the model states:

$$\underbrace{\varphi_t}_{\text{Topic Shares}} = T_0 + T_1 \underbrace{s_t}_{\text{Structural States}} + \Sigma \underbrace{u_t}_{\text{Measurement Error}}, \quad u_t \sim N(0, I)$$

- Interpretable as a **dynamic factor model** in which the structure of the DSGE model is imposed on the latent factors.
- Akin to Boivin and Giannoni (2006) and Kryshko (2011).

Augmented measurement equation

- Augmented measurement equation

$$\underbrace{\begin{pmatrix} Y_t \\ \varphi_t \end{pmatrix}}_{\text{New vector of observables}} = \begin{pmatrix} \Psi_0(\theta) \\ T_0 \end{pmatrix} + \begin{pmatrix} \Psi_1(\theta) \\ T_1 \end{pmatrix} s_t + \begin{pmatrix} 0_{4\times 4} & 0_{4\times k} \\ 0_{k\times 4} & \Sigma \end{pmatrix} u_t, \quad u_t \sim N(0, I)$$

- If text data carries relevant information, its use should make the estimation more efficient.
- General approach: any numerical machine learning output and structural model (DSGE, IO, labor...) will work.

FOMCs transcripts data

- Available for download from the Federal Reserve website.
- Provide a nearly complete account of every FOMC meeting from the mid-1970s onward.
- Transcripts are divided into two parts:
  - **FOMC1**: members talk about their reading of the current economic situation.
  - **FOMC2**: talk about monetary policy strategy.
- We are interested in the information on the current state of the economy and the beliefs that policymakers have on it $\Rightarrow$ focus on FOMC1 section from 1987 to 2009.
- Total of 180 meetings.

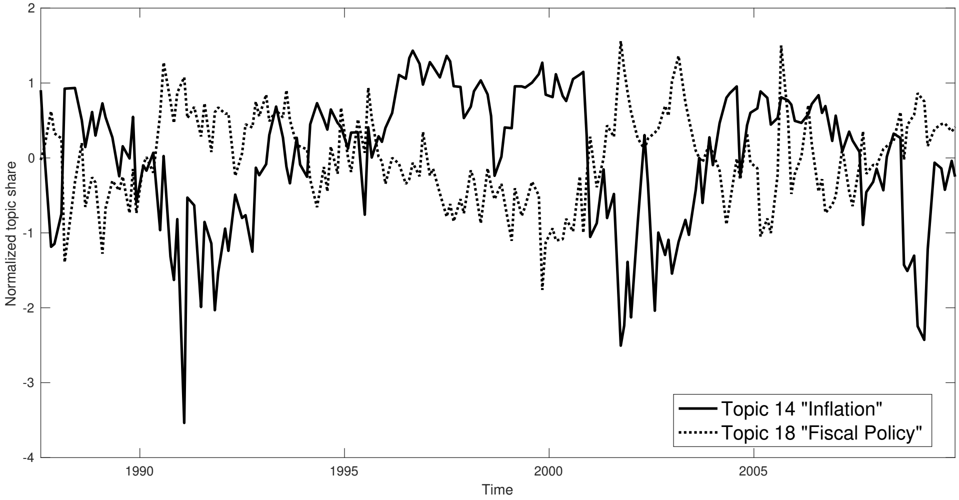
Topic composition

- Example of two topics with $K = 20$.

(a) Topic 8

(b) Topic 14

Topic shares



New Keynesian DSGE model

Conventional New Keynesian DSGE model with price rigidities:

Agents:
- Representative household.
- Perfectly competitive final good producer.
- Continuum of intermediate good producers with market power.
- Fiscal authority that taxes and spends.
- Monetary authority that sets the nominal interest rate.

States:
- 4 Exogenous states.
- Demand, productivity, government expenditure, monetary policy.
- Modelled as AR(1)s

Parameters:
- 12 Structural parameters to estimate.

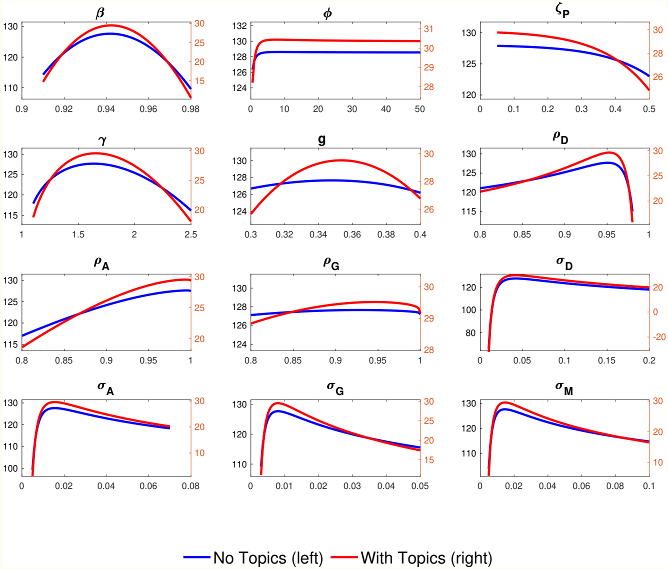
▸ Equilibrium Equations ▸ Exogenous Processes ▸ Structural Parameters Recap

Bayesian estimation

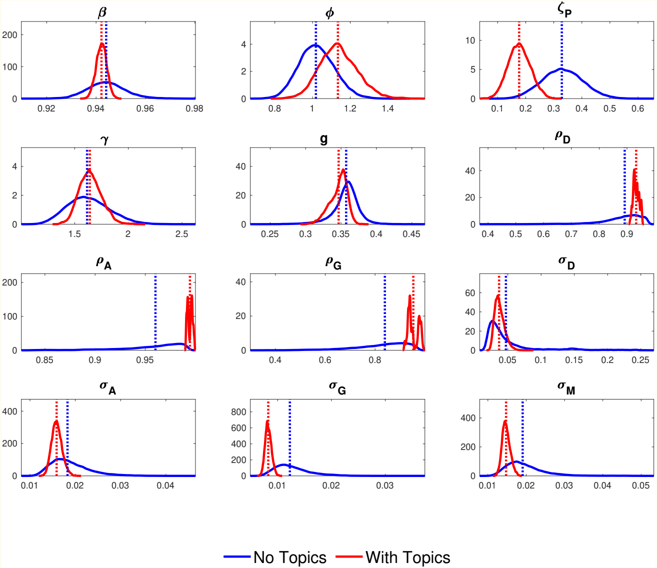
- We estimate two models for comparison:
  - New Keynesian DSGE model alone (standard).
  - New Keynesian DSGE Model + measurement equation with topic shares (new).
- Total parameters to estimate: 12 structural parameters (θ) + 120 topic dynamic factor model parameters (T_0, T_1, Σ assuming covariances are 0).
- Pick standard priors on structural parameters.
- Priors on topic related parameters?
  - Use MLE to get an idea of where they are.
  - Use conservative approach: center all parameters quite tightly around 0.
- Random-walk Metropolis Hastings to obtain draws from the posterior.

▸ Priors

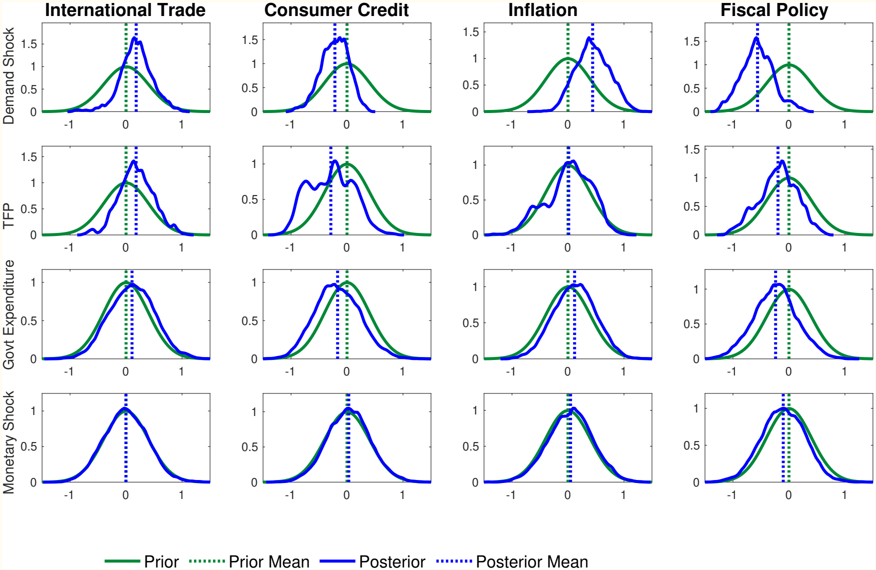
Likelihood comparison



Posterior distributions for structural parameters



Posterior distributions for selected topic parameters



Going forward: Joint model

- Extend the model in two ways:
  1. Model the dependence of latent topics on the hidden structural states directly.
  2. Allow for autocorrelation among topic shares.
- Instead of first creating φ_t and then using it for estimation, want to model the generating process of both text and macroeconomic observables together.
- Why a joint model?
  - Conjecture the topic composition and topic share will be more precise and more interpretable as a result.
  - Properly take into account the uncertainty around the topic shares.

Going forward: Joint model

- New vector of augmented states:

$$\widetilde{s}_t = \begin{pmatrix} s_t \\ \varphi_{t-1} \end{pmatrix}$$

- New vector of observables:

$$\widetilde{Y}_t = \begin{pmatrix} Y_t \\ \mathbf{w}_t \end{pmatrix}$$

- Stacks both the "traditional" observables Y_t and the text $\mathbf{w}_t$.

Joint state space representation

- Transition equation (linear):

$$\begin{aligned} \mathbf{s}_{t+1} &= \Phi(\theta)\mathbf{s}_t + \Phi_\epsilon(\theta)\epsilon && \leftarrow \text{same as before} \\ \varphi_t &= T_s\mathbf{s}_t + T_\varphi\varphi_{t-1} + \Sigma_e e_t && \leftarrow \text{new} \end{aligned}$$

- Measurement equation (non-linear):

$$\widetilde{Y}_t \sim p\left(\widetilde{Y}_t | \widetilde{s}_t\right) = \begin{pmatrix} \Psi_1(\theta)s_t \\ p(\mathbf{w}_t | \widetilde{s}_t) \end{pmatrix}$$

- Challenges: algorithm for estimation, impact of choice of priors.

Details on $p(\mathbf{w}_t | \widetilde{s}_t)$

LDA details

Back

LDA assumes the following generative process for each document W of length N:

1. Choose topic proportions $\varphi \sim \text{Dir}(\vartheta)$. Dimensionality K of the Dirichlet distribution. Thus, dimensionality of the topic variable is assumed to be known and fixed.
2. For each word $n = 1, ..., N$:
    2.1 Choose one of K topics $z_n \sim$ Multinomial (φ).
    2.2 Choose a term w_n from $p\left(w_n | z_n, \beta\right)$, a multinomial probability conditioned on the topic z_n. β is a $K \times V$ matrix where $\beta_{i,j} = p\left(w^j = 1 | z^i = 1\right)$. We assign a symmetric Dirichlet prior with V dimensions and hyperparameter η to each β_k.

Posterior:

$$p(\varphi, z, \beta | W, \vartheta, \eta) = \frac{p(\varphi, z, W, \beta | \vartheta, \eta)}{p(W | \vartheta, \eta)}$$

Model equilibrium equations (log-linearized) Back

$$\widehat{c}_t - \widehat{d}_t = \mathbb{E}_t\{\widehat{c}_{t+1} - \widehat{d}_{t+1} - \widehat{R}_t + \widehat{\Pi}_{t+1}\}$$

$$\widehat{\Pi}_t = \kappa\left((1+\phi c)\,\widehat{c}_t + \phi g \widehat{g}_t - (1+\phi)\,\widehat{A}_t\right) + \beta \mathbb{E}_t \widehat{\Pi}_{t+1}$$

$$\widehat{R}_t = \gamma \widehat{\Pi}_t + m_t$$

$$\widehat{l}_t = \widehat{y}_t - \widehat{A}_t = c\widehat{c}_t + g\widehat{g}_t - \widehat{A}_t$$

Exogenous States Processes

Back

$$\log d_t = \rho_d \log d_{t-1} + \sigma_d \varepsilon_{d,t}$$
$$\log A_t = \rho_A \log A_{t-1} + \sigma_A \varepsilon_{A,t}$$
$$\log g_t = \rho_g \log g_{t-1} + \sigma_g \varepsilon_{g,t}$$
$$m_t = \sigma_m \varepsilon_{mt}$$

Parameters Recap

Back

Param	Description
Steady-state-related parameters	
β	Discount factor
g	SS govt expenditure/GDP
Endogenous propagation parameters	
ϕ	Inverse Frisch elasticity
ζ_P	Fraction of fixed prices
γ	Taylor rule elasticity

Param	Description
Exogenous shocks parameters	
ρ_D	Persistence demand shock
ρ_A	Persistence TFP
ρ_G	Persistence govt expenditure
σ_D	s.d. demand shock innovation
σ_A	s.d. TFP shock
σ_G	s.d. govt expenditure shock
σ_G	s.d. monetary shock

State space matrices

Back

Law of motion for the states of the economy is:

$$\underbrace{\begin{pmatrix} \hat{d}_{t+1} \\ \hat{A}_{t+1} \\ \hat{g}_{t+1} \\ m_{t+1} \end{pmatrix}}_{s_{t+1}} = \underbrace{\begin{pmatrix} \rho_d & 0 & 0 & 0 \\ 0 & \rho_A & 0 & 0 \\ 0 & 0 & \rho_g & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}}_{\Phi_1(\theta)} \underbrace{\begin{pmatrix} \hat{d}_t \\ \hat{A}_t \\ \hat{g}_t \\ m_t \end{pmatrix}}_{s_t} + \underbrace{\begin{pmatrix} \sigma_d & 0 & 0 & 0 \\ 0 & \sigma_A & 0 & 0 \\ 0 & 0 & \sigma_g & 0 \\ 0 & 0 & 0 & \sigma_m \end{pmatrix}}_{\Phi_\epsilon(\theta)} \underbrace{\begin{pmatrix} \varepsilon_{d,t+1} \\ \varepsilon_{A,t+1} \\ \varepsilon_{g,t+1} \\ \varepsilon_{m,t+1} \end{pmatrix}}_{\epsilon_t}$$

and for observables:

$$\underbrace{\begin{pmatrix} \log c_t \\ \log \Pi_t \\ \log R_t \\ \log l_t \end{pmatrix}}_{Y_t} = \underbrace{\begin{pmatrix} \log(1-g) \\ 0 \\ -\log\beta \\ 0 \end{pmatrix}}_{\Psi_0(\theta)} + \underbrace{\begin{pmatrix} a_1 & a_2 & a_3 & a_4 \\ b_1 & b_2 & b_3 & b_4 \\ \gamma b_1 & \gamma b_2 & \gamma b_3 & 1+b_4 \\ ca_1 & ca_2-1 & 1+ca_3 & ca_4 \end{pmatrix}}_{\Psi_1(\theta)} \underbrace{\begin{pmatrix} \hat{d}_t \\ \hat{A}_t \\ \hat{g}_t \\ m_t \end{pmatrix}}_{s_t}$$

Prior distribution I

Back

Param	Description	Domain	Density	Param 1	Param 2
Steady-state-related parameters					
β	Discount factor	$(0,1)$	Beta	0.95	0.05
g	SS govt expenditure/GDP	$(0,1)$	Beta	0.35	0.05
Endogenous propagation parameters					
ϕ	Inverse Frisch elasticity	$\mathbb{R}_+$	Gamma	1	0.1
ζ_P	Fraction of fixed prices	$(0,1)$	Beta	0.4	0.1
γ	Taylor rule elasticity	$\mathbb{R}_+$	Gamma	1.5	0.25

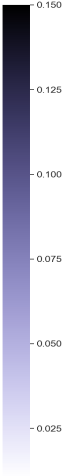
Prior distribution II

Back

Exogenous shocks parameters					
ρ_D	Persistence demand shock	$(0,1)$	Uniform	0	1
ρ_A	Persistence TFP	$(0,1)$	Uniform	0	1
ρ_G	Persistence govt expenditure	$(0,1)$	Uniform	0	1
σ_D	s.d. demand shock innovation	$\mathbb{R}_+$	InvGamma	0.05	0.2
σ_A	s.d. TFP shock	$\mathbb{R}_+$	InvGamma	0.05	0.2
σ_G	s.d. govt expenditure shock	$\mathbb{R}_+$	InvGamma	0.05	0.2
σ_G	s.d. monetary shock	$\mathbb{R}_+$	InvGamma	0.05	0.2
Topic parameters					
$T_{0,k}$	Topic baseline	$\mathbb{R}$	Normal	0	0.1
$T_{1,k,s}$	Topic elasticity to states	$\mathbb{R}$	Normal	0	0.4
σ_k	s.d. measurement error	$\mathbb{R}_+$	InvGamma	0.2	0.1

Topics ranked by pro-cyclicality

													Pro-cyclicality
Topic0	inflat	price	increas	product	wage	cost	rise	growth	trend	labor	core	pressur	0.053
Topic1	percent	year	quarter	rate	growth	forecast	last	month	greenbook	inflat	project	expect	0.014
Topic2	statement	meet	chang	will	risk	word	discuss	polici	market	issu	view	languag	0.011
Topic3	forecast	model	inflat	rate	greenbook	term	differ	chang	use	assumpt	shock	question	0.01
Topic4	district	nation	continu	area	region	remain	employ	manufactur	economi	report	activ	sector	0.007
Topic5	move	data	can	look	number	will	evid	signific	may	quit	economi	point	0.007
Topic6	move	chairman	mr	support	direct	point	recommend	agre	asymmetr	prefer	tighten	eas	0.004
Topic7	question	know	want	someth	thing	look	realli	peopl	tri	number	talk	ask	0.002
Topic8	dollar	unitedstates	import	export	countri	foreign	price	growth	forecast	oil	effect	japan	0.002
Topic9	forecast	quarter	project	data	spend	inventori	will	revis	recent	expect	anticip	month	0.002
Topic10	presid	ye	governor	parri	okay	thank	break	stern	vice	hoenig	minehan	laughter	0.001
Topic11	year	line	right	panel	shown	chart	expect	percent	project	next	middl	left	0.0
Topic12	period	committe	run	consist	monetari	rate	might	aggreg	target	rang	econom	borrow	0.0
Topic13	polici	rate	inflat	might	market	economi	expect	tighten	may	term	committe	eas	-0.001
Topic14	year	report	busi	sale	product	increas	price	industri	compani	contact	continu	firm	-0.007
Topic15	bank	market	credit	rate	incom	debt	financi	loan	consum	fund	interest	household	-0.015
Topic16	economi	will	can	seem	time	problem	believ	know	rather	much	world	may	-0.015
Topic17	side	littl	look	seem	much	realli	term	lot	pretti	quit	certainli	concern	-0.017
Topic18	risk	economi	continu	growth	seem	may	recoveri	will	busi	confid	remain	outlook	-0.02
Topic19	will	effect	fiscal	ta	cut	term	budget	time	uncertainti	probabl	state	spend	-0.04



Details on $p(\mathbf{w}_t|\widetilde{s}_t)$ ◂ Back

We assume the following generative process for a document:

1. Draw $\varphi_t|\varphi_{t-1}, s_t, T_\varphi, T_s, \Sigma^e \sim N(\varphi_0 + T_\varphi \varphi_{t-1} + T_s s_t, \Sigma^e)$.
2. To map this representation of topic shares into the simplex, use the softmax function: $f(\varphi_t) = \exp \varphi_{t,i} / \sum_j \exp \varphi_{t,j}$.
3. For each word $n = 1, ..., N$:
   3.1 Draw topic $z_{n,t} \sim \text{Mult}(f(\varphi_t))$.
   3.2 Draw term $w_{n,t} \sim \text{Mult}(\beta_{t,z})$.

Then, the distribution of text conditional on the states, $p(\mathbf{w}_t|\widetilde{s}_t)$, is given by:

$$p(\mathbf{w}_t|\widetilde{s}_t) = \int p(\varphi_t|\widetilde{s}_t) \left(\prod_{n=1}^{N} \sum_{z_{n,t}} p(z_{n,t}|\varphi_t) p(w_{n,t}|z_{n,t}, \beta_t) \right) d\varphi_t$$

Conclusions

- Deep learning has tremendous potential for macroeconomics.
- Great theoretical and practical reasons for it.
- Many potential applications.