Estimating Nonlinear Heterogeneous Agent Models with Neural Networks

Hanno Kase[1], Leonardo Melosi[2], Matthias Rottner[3]

Numerical Methods in Macroeconomics, October 23, 2024

[1]European Central Bank
[2]European University Institute, University of Warwick, De Nederlandsche Bank, CEPR
[3]Bank for International Settlements, Deutsche Bundesbank

The views in this presentation are solely those of the authors and should not be interpreted as reflecting the views of the De Nederlandsche Bank, the Deutsche Bundesbank, the European Central Bank, or any other person associated with the Eurosystem.

Motivation

HANK models have gained more popularity:

- social inequality matters for dynamics of the economy and monetary policy
- aggregate policies shape income and wealth distribution

Hard to solve because of their elevated complexity

- Heterogeneous agents facing idiosyncratic risks
- Aggregate uncertainty and nonlinearities

Difficult to estimate, usually requires repeated solving

This paper

- **Develop estimation procedure based on neural networks**
- **Apply to nonlinear HANK model**

Key innovations

There are two key innovations tackling different estimation bottlenecks

1. **Extended Neural Network** more

   Allows us to **avoid repeated solving** the model

2. **Neural Network Particle Filter**

   Dramatically reduce the **cost of likelihood evaluations**

Solution procedure using deep neural networks

- Euler residual minimization method (Maliar et al. 2021)

0. Instead of continuum of agents, there are L agents
1. Parameterize individual and aggregate policy functions with deep neural networks

$$\psi_t^i = \psi_{NN}^I(\mathbb{S}_t^i, \mathbb{S}_t|\Theta) \quad \text{and} \quad \psi_t^A = \psi_{NN}^A(\mathbb{S}_t|\Theta)$$

   Where $\mathbb{S}_t = \{\{\mathbb{S}_t^i\}_{i=1}^L, \mathbb{S}_t^A\}$ is a vector of state variables

   Θ is the set of parameters of the model

2. Construct loss function - weighted mean of squared residuals
3. Train the deep neural networks using stochastic optimization
   - Minimize the loss for points drawn from the state space
   - Simulate model forward to generate a new draw from the state space

Training the neural networks repeatedly would take too long for estimation

Avoid repeated solving - Extended Neural Network

- **Treat model parameters as pseudo state variables**

0. Instead of continuum of agents, there are L agents
1. Parameterize individual and aggregate policy functions with deep neural networks

$$\psi_t^i = \psi_{NN}^I(\mathbb{S}_t^i, \mathbb{S}_t, \tilde{\boldsymbol{\Theta}} | \bar{\Theta}) \quad \text{and} \quad \psi_t^A = \psi_{NN}^A(\mathbb{S}_t, \tilde{\boldsymbol{\Theta}} | \bar{\Theta})$$

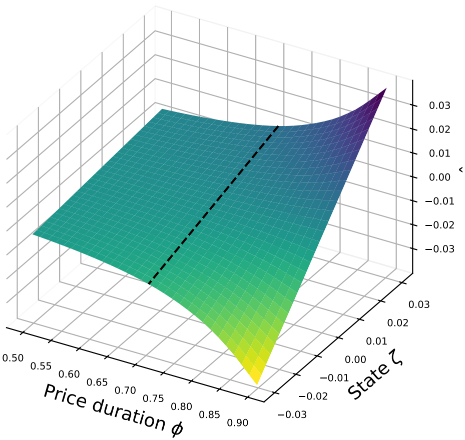
   Where $\mathbb{S}_t = \{\{\mathbb{S}_t^i\}_{i=1}^L, \mathbb{S}_t^A\}$ is a vector of state variables

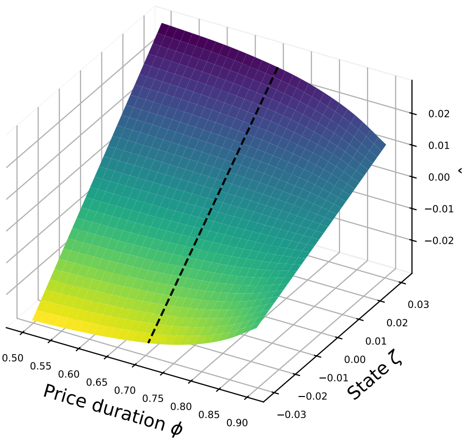
   $\bar{\Theta}$ **is the set of calibrated and** $\tilde{\boldsymbol{\Theta}}$ **estimated parameters of the model**

2. Construct loss function - weighted sum of mean of squared residuals
3. Train the deep neural networks using stochastic optimization
   - Minimize the loss for points drawn from the state space
   - **Draw new values for parameters $\tilde{\boldsymbol{\Theta}}$ we are interested in estimating**
   - Simulate model forward to generate a new draw from the state space

More complex problem, but we only need to train the networks ONCE!

Extended Neural Network - output from ONE neural network


Policy function $\hat{\Pi}_t$ conditioned on ϕ and ζ


Policy function $\hat{X}_t$ conditioned on ϕ and ζ

Black dashed line is what we get with standard solution methods.

Costly likelihood evaluation - Neural Network Particle Filter

For nonlinear models we can obtain the likelihood using a **particle filter**

- Model needs to be **evaluated** for thousands of particles and multiple time periods
- Particle filter becomes the bottleneck for estimation

Costly likelihood evaluation - Neural Network Particle Filter

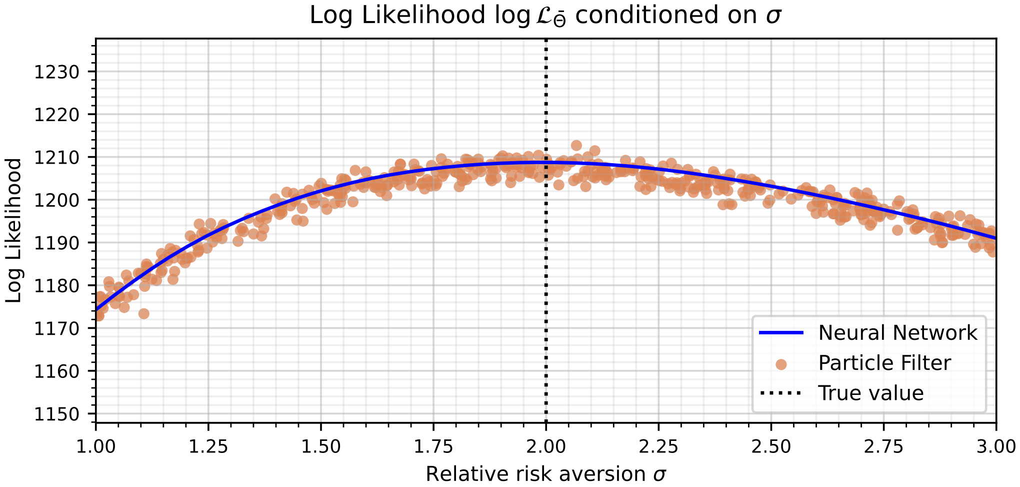
Train a neural network to directly map from parameters to log-likelihood

1. Create a dataset of parameter values and log-likelihoods
2. Split the dataset into training and validation samples
3. Train a neural network on the training sample
    - Use the validation sample to avoid overfitting

Benefits:

- Single likelihood evaluation can be done almost instantly
- Smooths out noise from the particle filter

Neural Network Particle Filter - example with one parameter



Proof of the pudding is in the eating

1. Compare the extended NN based solution to a benchmark
    - Linearized three equation NK model with an analytical solution

   Extended Neural Network matches the true solution

2. Compare the estimation results to a standard method
    - Simple nonlinear RANK model with a ZLB

   Estimation results are very similar

3. Estimating a nonlinear HANK model
    - Using simulated data from the model results
    - Using aggregate time-series for the US from 1990 to 2019

   Scales to larger models

Linearized NK model

- Small linearized three equation NK model with a TFP shock

$$\hat{X}_t = E_t \hat{X}_{t+1} - \sigma^{-1} \left(\phi_\Pi \hat{\Pi}_t + \phi_Y \hat{X}_t - E_t \hat{\Pi}_{t+1} - \hat{R}_t^F \right) \tag{IS}$$

$$\hat{\Pi}_t = \kappa \hat{X}_t + \beta E_t \hat{\Pi}_{t+1} \tag{NKPC}$$

$$\hat{R}_t^F = \rho_A \hat{R}_{t-1}^F + \sigma(\rho_A - 1)\omega\sigma_A \epsilon_t^A$$

Where $\hat{X}$: output gap, $\hat{\Pi}$: inflation, R^F: risk free rate, ϵ^A: TFP shock

- Analytical solution:

$$\begin{aligned} \hat{X}_t &= \frac{1 - \beta\rho_A}{(\sigma(1-\rho_A) + \theta_Y)(1 - \beta\rho_A) + \kappa(\theta_\Pi - \rho_A)} \hat{R}_t^F, \\ \hat{\Pi}_t &= \frac{\kappa}{(\sigma(1-\rho_A) + \theta_Y)(1 - \beta\rho_A) + \kappa(\theta_\Pi - \rho_A)} \hat{R}_t^F. \end{aligned}$$

Solving the linearized NK model with an Extended Neural Network

1. Parametrize the policy function with a deep neural network:

$$\begin{pmatrix} \hat{X}_t \\ \hat{\Pi}_t \end{pmatrix} = \psi(\underbrace{\hat{R}_t^F}_{\mathbb{S}_t}, \underbrace{\beta, \sigma, \eta, \phi, \theta_\Pi, \theta_Y, \rho_A, \sigma_A}_{\tilde{\Theta}}) \approx \psi_{NN}\left(\hat{R}_t^F, \beta, \sigma, \eta, \phi, \theta_\Pi, \theta_Y, \rho_A, \sigma_A\right)$$

2. Construct the loss function:

$$ERR_{IS} = \hat{X} - \left(E_t \hat{X}_{t+1} - \sigma^{-1}\left(\phi_\Pi \hat{\Pi}_t + \phi_Y \hat{X}_t - E_t \hat{\Pi}_{t+1} - \hat{R}_t^F\right)\right)$$

$$ERR_{NKPC} = \hat{\Pi}_t - \left(\kappa \hat{X}_t + \beta E_t \hat{\Pi}_{t+1}\right)$$

$$\mathcal{L} = w_1 \frac{1}{B} \sum_{I=1}^{B} (ERR_{IS}^i)^2 + w_2 \frac{1}{B} \sum_{i=1}^{B} (ERR_{NKPC}^i)^2 \quad \text{, where } B \text{ is the batch size}$$

3. Train the deep neural networks using stochastic optimization ...

Solving the linearized NK model with an Extended Neural Network

3. Train the deep neural networks using stochastic optimization
    - Batch size of 1000 (parallel worlds) for 500 000 iterations

   At each iteration:

   1. **Draw parameters** from a bounded parameter space:

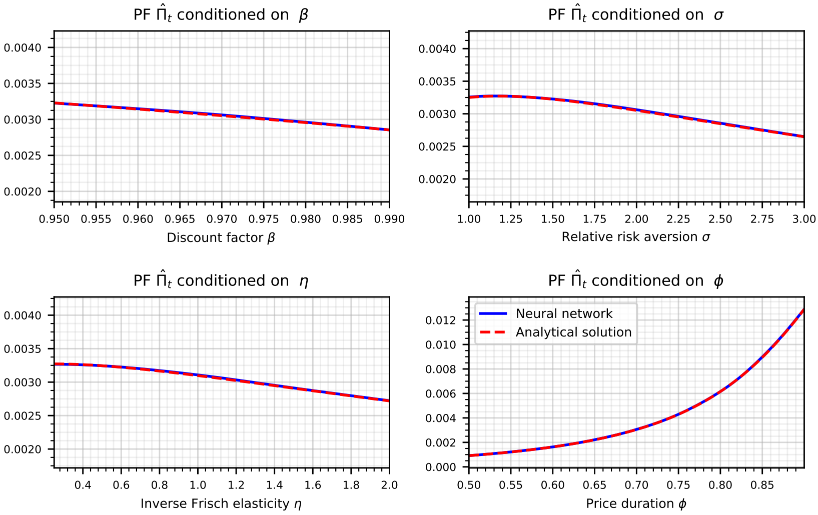
Parameters		LB	UB	Parameters		LB	UB
β	Discount factor	0.95	0.99	θ_{Π}	Mon.pol. inflation response	1.25	2.5
σ	Relative risk aversion	1	3	θ_Y	Mon.pol. output response	0.0	0.5
η	Inverse Frisch elasticity	1	4	ρ_A	Persistence TFP shock	0.8	0.95
φ	Price duration	0.5	0.9	σ_A	Std. dev. TFP shock	0.02	0.1

   2. **Draw points from the state space** by simulating the model:

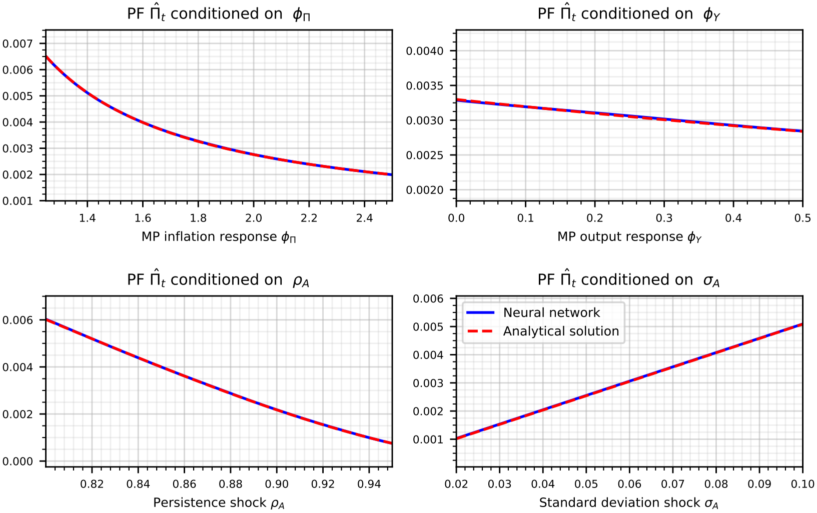
$$\hat{R}^F_t = \rho_A \hat{R}^F_{t-1} + \sigma(\rho_A - 1)\omega\sigma_A\epsilon^A_t$$

   3. **Compute the loss** $\mathcal{L}$
   4. **Optimizer step** (ADAM) to adjust the weights of the NN to minimize $\mathcal{L}$

Extended Neural Network: Inflation over the parameter space



Extended Neural Network: Inflation over the parameter space



Compare the estimation results to a standard method

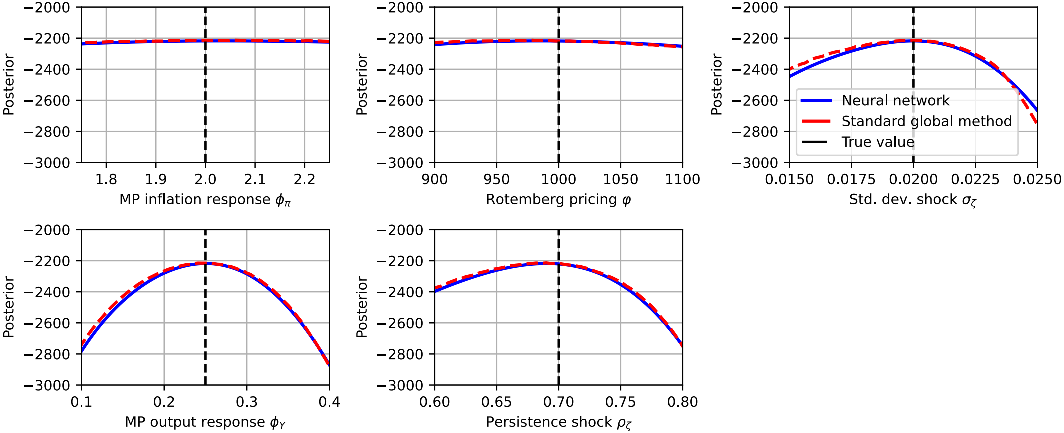
- Simple RANK model
  - Only a preference shock
  - Zero lower bound
- Interesting laboratory:
  1. Simple enough to solve and estimate with standard methods

Estimation comparison

- Use the model to create time series for: output growth, inflation, interest rate
- Recover 5 parameter values using:
  1. Neural networks based approach (**extended NN, NN PF**, RWMH)
  2. Standard approach (time iteration, regular particle filter, RWMH)

Estimation											
Par.	Prior					NN			Alternative Approach		
	Type	Mean	Std	Lower Bound	Upper Bound	Posterior			Posterior		
						Median	5%	95%	Median	5%	95%
θ_Π	Trc.N	2.0	0.1	1.5	2.5	2.04	1.92	2.15	2.06	1.93	2.20
θ_Y	Trc.N	0.25	0.05	0.05	0.5	0.250	0.240	0.260	0.248	0.237	0.260
φ	Trc.N	1000	50	700	1300	985	921	1047	970	909	1033
ρ_ζ	Trc.N	0.7	0.05	0.5	0.9	0.69	0.671	0.707	0.688	0.670	0.707
σ^ζ	Trc.N	0.02	0.0025	0.01	0.025	0.020	0.019	0.021	0.020	0.019	0.021

Estimation comparison: posterior



Estimating a nonlinear HANK model

- Households face idiosyncratic income risk s_t^i and a **borrowing limit** $\underline{B}$

$$E_0 \sum_{t=0}^{\infty} \beta^t \exp(\zeta_t^D) \left[\left(\frac{1}{1-\sigma} \right) \left(\frac{C_t}{Z_t} \right)^{1-\sigma} - \chi \left(\frac{1}{1+\eta} \right) (H_t^i)^{1+\eta} \right]$$

$$\text{s.t. } C_t^i + B_t^i = \tau_t \left(\frac{W_t}{Z_t} \exp(s_t^i) H_t^i \right)^{1-\gamma_\tau} + \frac{R_{t-1}}{\Pi_t} B_{t-1}^i + Div_t \exp(s_t^i)$$

$$B_t^i \geq \underline{\mathbf{B}}$$

where idiosyncratic risk follows an AR(1) process: $s_t^i = \rho_s s_{t-1}^i + \sigma_s \epsilon_t^i$

- Aggregate shocks: preference ζ^D, growth rate g_t and monetary policy mp_t
- Monopolistically competitive firms and Rotemberg pricing
- Monetary policy is constrained by the **zero lower bound**

$$R_t = \mathbf{max} \left[\mathbf{1}, R \left(\frac{\Pi_t}{\Pi} \right)^{\theta_\Pi} \left(\frac{Y_t}{Z_t Y} \right)^{\theta_Y} \exp(mp_t) \right]$$

Estimating a nonlinear HANK model - Extended NN part

We are interested in finding the **policy functions over parameter ranges**

0. Instead of continuum of agents there are $L = 100$ agents
1. Policy functions parameterized by deep neural networks
    - Aggregate: inflation and wage
    - Individual: labor choice
    - **213 state variables**
        - 200 individual, 3 aggregate and 10 pseudo (parameters) states
2. Loss function is a weighted sum of squared residuals of:
    - Fisher-Burmeister eq. (Euler residual and individual borrowing limit)
    - NKPC
    - Bond market clearing
    - Product market clearing
3. Train the deep neural networks ...

Estimating a nonlinear HANK model - Extended NN part

3. Train the deep neural networks ... in two steps

   a. **Deterministic steady state** (DSS) - model without agg. shocks

      - We need nominal rate and output for the **Taylor rule**
      - DSS network: R_{DSS} and Y_{DSS}
      - Individual network: labor choice
      - Slightly different loss function (no NKPC error, $Y - Y_{DSS}$)

   b. **Full nonlinear HANK** - agg. and idiosyncratic shocks

      - Start from individual network from the previous step (transfer learning)
      - Use DSS network (stays fixed)
      - Aggregate network: inflation and wage
      - Curriculum learning (HANK $\rightarrow$ HANK with ZLB ...)

Estimating a nonlinear HANK model - NN particle filter and RWMH

1. Neural network particle filter
    - Create a dataset:
        - Draw parameters (Sobol sequence)
        - Use particle filter to calculate model log-likelihood
    - Train a NN that maps from parameters to model log-likelihoods
2. Random Walk Metropolis-Hastings Algorithm
    - Computational costs are frontloaded
    - Very fast to generate a large number of draws

Estimation with US data

US time-series data from 1990:Q1 to 2019:Q4

- GDP growth rate per capita
- GDP deflator
- Shadow interest rate

Measurement equation:

$$\begin{bmatrix} \text{Output Growth} \\ \text{Inflation} \\ \text{Interest Rate} \end{bmatrix} = \begin{bmatrix} 400\left(\frac{Y_t}{Y_{t-1}/g_t} - 1\right) \\ 400\left(\Pi_t - 1\right) \\ 400\left(R_t - 1\right) \end{bmatrix} + u_t,$$

Measurement error $u_t \sim \mathcal{N}(0, \Sigma_u)$ is 5% of the variance of each observable

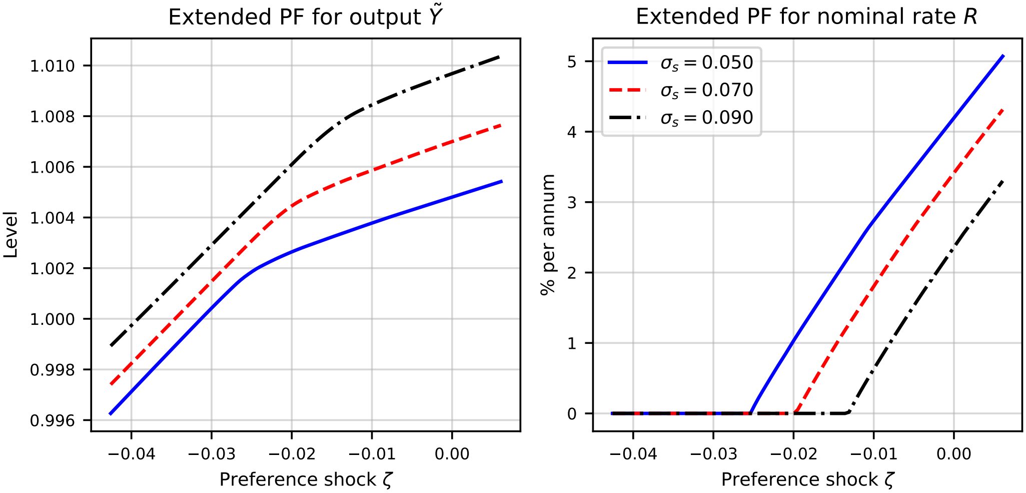
Estimation with US data

Moments

Estimation

Par.	Prior					NN		
	Type	Mean	Std	Lower Bound	Upper Bound	Posterior		
						Median	5%	95%
Parameters affecting the DSS								
$100\sigma_s$	Trc.N	5.00	1.000	2.50	10.0	7.04	5.67	8.10
$\underline{B}$	Trc.N	−0.50	0.010	−0.65	−0.35	−0.50	−0.54	−0.46
Other parameters								
φ	Trc.N	100	5.000	70	120	101	94	107
θ_Π	Trc.N	2.25	0.125	1.75	2.75	2.43	2.20	2.67
θ_Y	Trc.N	1.00	0.025	0.75	1.25	0.96	0.92	1.00
ρ_z	Trc.N	0.40	0.025	0.2	0.6	0.43	0.39	0.47
ρ_m	Trc.N	0.90	0.005	0.85	0.95	0.91	0.90	0.91
$100\sigma_\zeta$	Trc.N	1.50	0.100	1.00	2.00	1.22	1.10	1.33
$100\sigma_z$	Trc.N	0.40	0.100	0.30	0.60	0.47	0.43	0.53
$100\sigma_m$	Trc.N	0.06	0.010	0.05	0.20	0.15	0.14	0.16

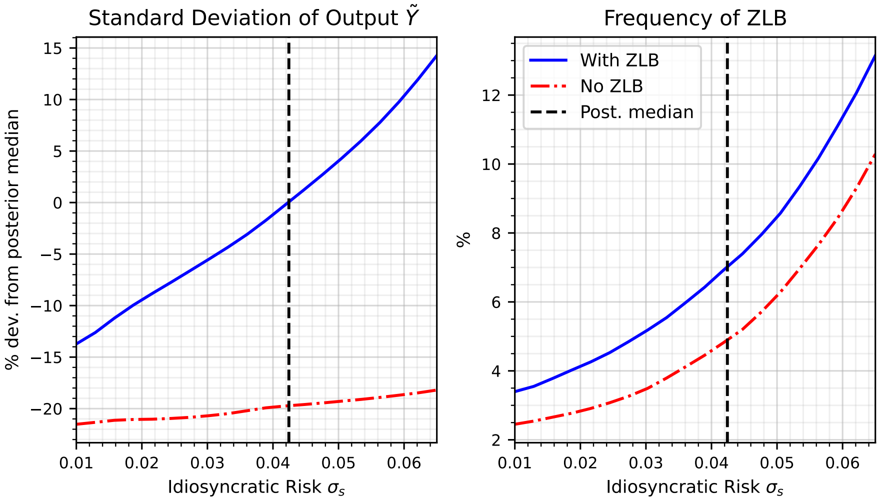
Interaction between heterogeneity and nonlinearities



Extended PF for output $\tilde{Y}$

Extended PF for nominal rate R

Idiosyncratic risk, ZLB frequency, and aggregate output volatility



Conclusion

Novel estimation procedure based on neural networks

1. **Extended Neural Network** - avoid repeated solving
2. **Neural Network Particle Filter** - fast likelihood evaluations

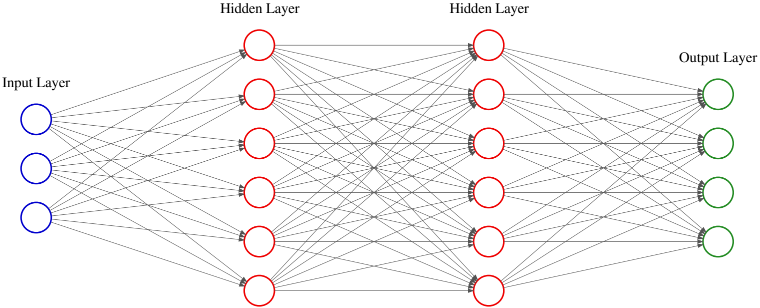
- Estimation of a HANK model with individual and aggregate nonlinearities
    - Two proof-of-concept models to demonstrate accuracy
- Opens up new exciting avenues for future research questions
    - Work with more realistic high-dimensional models
    - A framework to think about monetary policy strategy and inequality

Code for the analytical example!

https://github.com/tseep/estimating-hank-nn

Appendices

Neural Network



Neural Network

- Single neuron i in layer l with width H_l and activation function σ:

$$x_i^l = \sigma\left(\sum_j W_{ij}^l x_j^{l-1} + b_i^l\right), \quad 1 \leq i \leq H_l, \quad 1 \leq j \leq H_{l-1}$$

- Single layer:

$$\mathbf{x}^l = \sigma\left(\mathcal{A}^l(\mathbf{x}^{l-1})\right), \quad \mathcal{A}^l\left(\mathbf{x}^{l-1}\right) = \mathbf{W}^l\mathbf{x}^{l-1} + \mathbf{b}^l$$

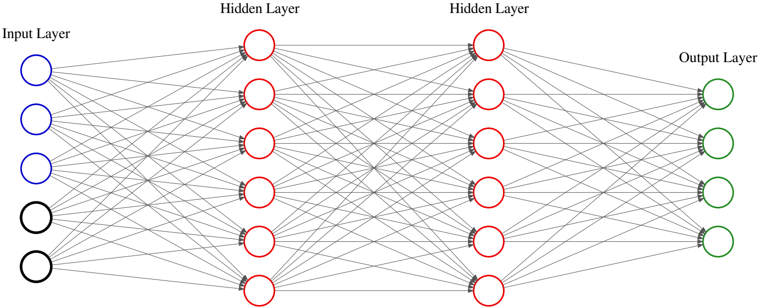
- The entire network with L hidden layers:

$$\psi(\mathbf{x}) = \left(\mathcal{A}^{L+1} \circ \sigma \circ \mathcal{A}^L \circ \sigma \circ \mathcal{A}^{L-1} \circ ... \circ \sigma \circ \mathcal{A}^1\right)(\mathbf{x})$$

- Weights and biases of the network:

$$\theta = \{\mathbf{W}^l, b^l\}_{l=1}^{L+1}$$

Extended Neural Network



Training a Neural Network

- Suppose we want to approximate $\mathbf{f} : \mathbf{x} \mapsto \mathbf{y}$ using our neural network $\psi(\mathbf{x}; \theta)$
- We have a dataset of pairwise samples $\mathcal{S} = \{(\mathbf{x}_i, \mathbf{y}_i) : 1 \leq i \leq N\}$
- **Training** is adjusting θ so that $\psi(\mathbf{x}, \theta)$ starts approximating $\mathbf{f}$:

$$\theta^* = \arg\min_\theta \mathcal{L}(\theta), \quad \text{where} \quad \mathcal{L}(\theta) = \frac{1}{N}\sum_{i=1}^{N}(y_i - \psi(\mathbf{x}; \theta))$$

- Usually done using (some variation of) gradient decent algorithm:

$$\theta_{k+1} = \theta_k - \eta \frac{\partial \mathcal{L}}{\partial \theta}(\theta_k)$$

- Where η is the learning rate
- The gradients are efficiently calculated using backpropagation algorithm

Extended (policy function approximated by) Neural Networks

- We usually solve models and find policies as a function of the state

$$\psi_t = \psi(\mathbb{S}_t|\Theta)$$

- We could **extend** the states by treat parameters Θ as additional input

$$\psi_t = \psi(\mathbb{S}_t, \Theta)$$

- With $\psi(\mathbb{S}_t, \Theta)$ we can quickly get the policy for different parameter values

Extended (policy function approximated by) Neural Networks back

- Infeasible using standard methods
  - Severe curse-of-dimensionality $N_{\mathbb{S}} \times N_{\Theta}$
  - Standard methods grow exponentially with dimensions

- **Neural networks** can tame the curse-of-dimensionality
  - Number of neurons required grows linearly with dimensions
  - Scale to models with large number of state variables
  - Can resolve local features accurately (kinks)
  - Can capture irregularly shaped domain

Estimating HANK with simulated data back

Estimation experiment:

- Use the calibrated model to create time series for:
    - Output growth
    - Inflation
    - Interest rate
- Recover 10 parameters
    1. Generate a dataset of parameter values and corresponding log-likelihoods
    2. Train the Neural Network Particle Filter
    3. Run the Random Walk Metropolis Hastings algorithm

Estimation with simulated data Calibrated parameters

Estimation									
Par.	True	Prior					NN		
	Value	Type	Mean	Std	Lower Bound	Upper Bound	Posterior Median	5%	95%
Parameters affecting the DSS									
$100\sigma_s$	5.00	Trc.N	5.00	1.000	2.50	10.0	4.28	3.17	5.31
$\underline{B}$	-0.50	Trc.N	-0.50	0.010	-0.65	-0.35	-0.50	-0.54	-0.46
Other parameters									
φ	100	Trc.N	100	5.000	70	120	100	92	108
θ_Π	2.25	Trc.N	2.25	0.125	1.75	2.75	2.40	2.25	2.55
θ_Y	1.00	Trc.N	1.00	0.025	0.75	1.25	1.01	0.97	1.05
ρ_z	0.40	Trc.N	0.40	0.025	0.20	0.60	0.40	0.37	0.45
ρ_m	0.90	Trc.N	0.90	0.005	0.85	0.95	0.90	0.89	0.91
$100\sigma_\zeta$	1.50	Trc.N	1.50	0.100	1.00	2.00	1.45	1.34	1.57
$100\sigma_z$	0.40	Trc.N	0.40	0.100	0.30	0.60	0.36	0.32	0.40
$100\sigma_m$	0.06	Trc.N	0.06	0.010	0.05	0.20	0.06	0.05	0.07

Calibrated parameters of the nonlinear HANK

back

Calibrated Parameter Values

Parameters		Value	Target/Source
β	Discount factor	0.9975	4% nominal interest rate
η	Inverse Frisch elasticity	0.72	Chetty et al. (2011)
σ	Relative risk aversion	1	Log utility
$\bar{a}$	Average growth rate	1.0033	Real GDP growth = 0.33% (quarterly)
χ	Disutility labor	0.74	Labor supply is approximately 1
γ^τ	Tax progressivity	0.18	Heathcote et al. (2017)
D	DSS government debt	1.0	Wealth share=25% GDP (Kaplan et al., 2018)
Π	Inflation target	1.00625	Inflation target = 2.5% (annualized)
ρ_s	Persistence labor prod.	0.9	Share of borrowers = 34%
ρ_ζ	Persistence pref. shock	0.7	Frequency of ZLB = 15%

How good is what we got? back

Standard deviations			Autocorrelations			Avg. Gini coef.		
	Model	Data		Model	Data		Model	Data
GDP	0.6947	0.5831	GDP	0.1355	0.4050	Wealth	0.8793	0.8410
Inflation	1.1511	0.9045	Inflation	0.8146	0.5456			
FFR	2.561	2.7537	FFR	0.7219	0.9707			