Estimating Nonlinear Heterogeneous Agents Models with Neural Networks

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

Deep Learning for Solving and Estimating Dynamic Models, August 25, 2023

[1]European Central Bank
[2]FRB Chicago, CEPR, University of Warwick
[3]Deutsche Bundesbank

Macroeconomic models of the future

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 Based Particle Filter**
   Dramatically reduce the **cost of likelihood evaluations**

Solution procedure using deep neural networks

- Building on Maliar, Maliar, and Winant (2021) Euler residual minimization
  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

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

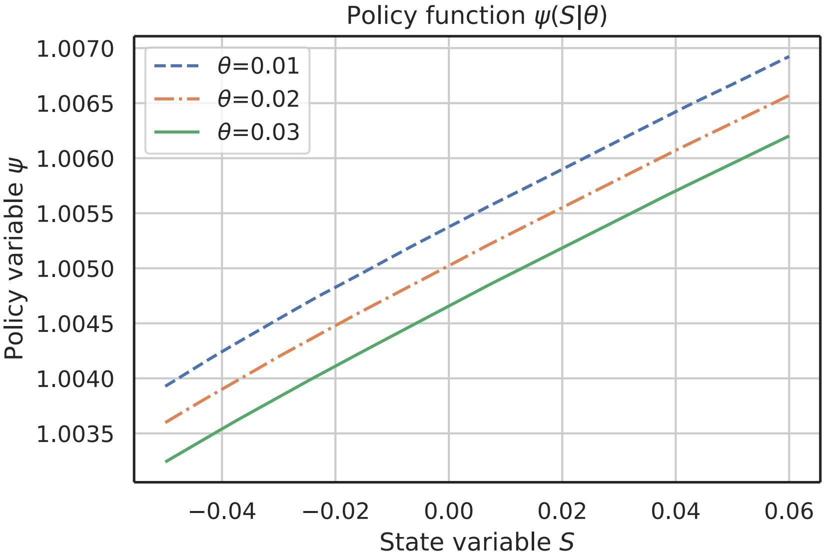
  Where $\mathbb{S}_t = \{\{\mathbb{S}_t^i\}_{i=1}^L, \mathbb{S}_t^A\}$ is a vector of state variables

  $\bar{\Theta}$ **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



For nonlinear models we can obtain the likelihood using the **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 Based Particle Filter

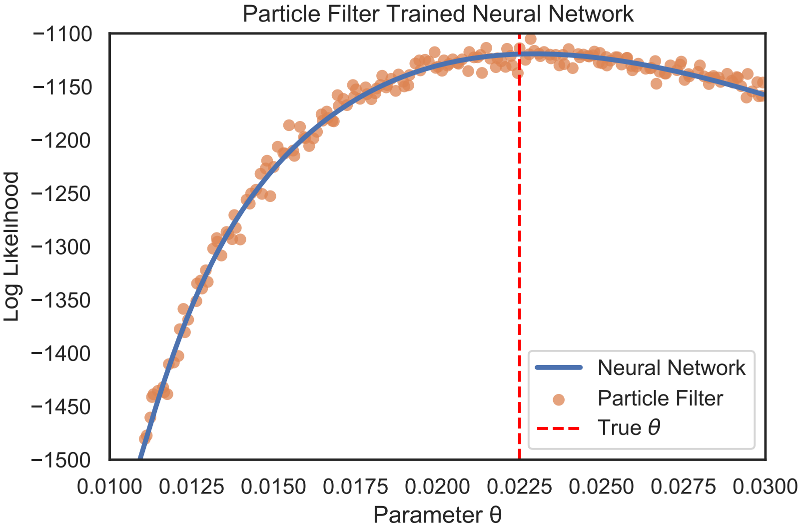
Create a surrogate model for the particle filter:

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
  $\Rightarrow$ **Allows for a large number of draws in Metropolis-Hastings algorithm**
- Easily parallelized
  $\Rightarrow$ **We can use multiple GPUs to create a bigger dataset**
- Smooths out noise from the particle filter
  $\Rightarrow$ **We can use less particles in the filter**

Neural Network Based Particle Filter - one parameter



Recap

Two key innovations

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

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 conventional method
    - Simple nonlinear RANK model with a ZLB

   Estimation results are very similar

3. Estimating a nonlinear HANK model

   Scales to larger models

Linearized NK model

- Small linearized three equation NK model with a TFP shock

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

$$\hat{\Pi}_t = \kappa\hat{X}_t + \beta E_t\hat{\Pi}_{t+1} \qquad \text{(NKPC)}$$

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

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

- Analytical solution:

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

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

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 500 (parallel worlds)
   - 100 000 iterations
     1. **Draw parameters** from a bounded parameter space:

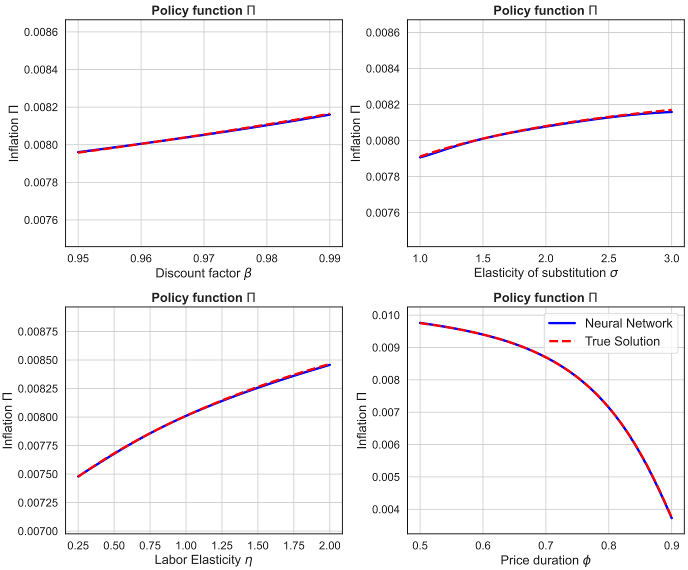
Parameters		LB	UB	Parameters		LB	UB
β	Discount factor	0.95	0.99	θ_Π	MP inflation response	1.25	2.5
σ	Relative risk aver.	1	3	θ_Y	MP output response	0.0	0.5
η	Inverse Frisch elas.	1	4	ρ_A	Persistence TFP shock	0.8	0.95
ϕ	Price duration	0.5	0.9	σ_A	Std. dev. TFP shock	0.02	0.1

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

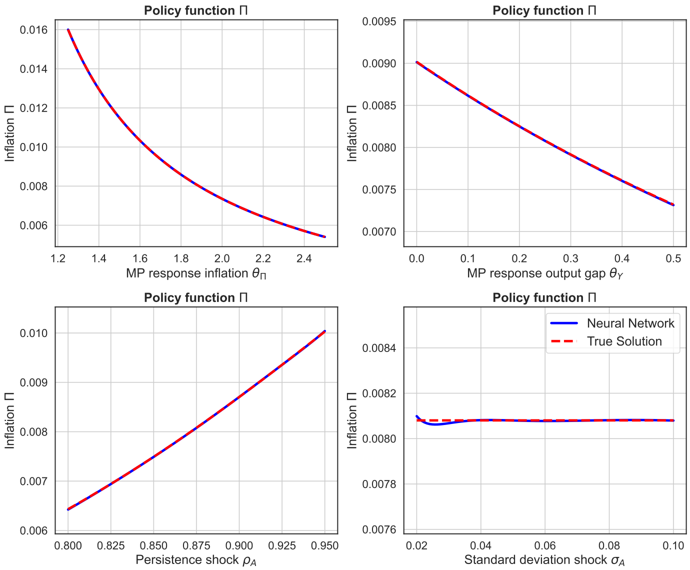
$$\hat{R}_t^F = \rho_A \hat{R}_{t-1}^F + \sigma(\rho_A - 1)\omega\sigma_A\epsilon_t^A$$

     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 conventional method

- Simple RANK model
  - Only a preference shock
  - Zero lower bound
- Interesting laboratory:
  1. Simple enough to solve and estimate with conventional methods
  2. No solution if the volatility of the demand shock is too large

Estimation comparison

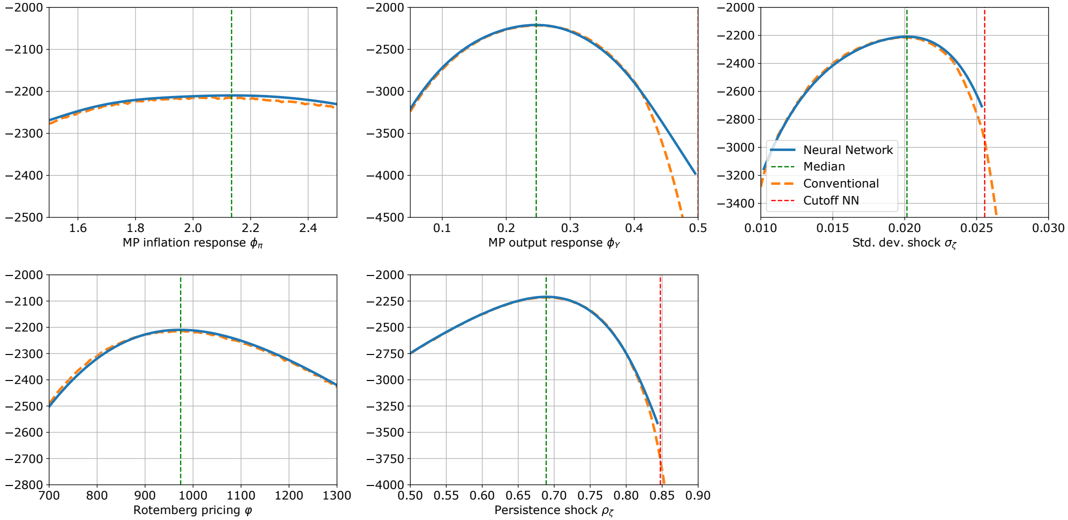
- Use the model to create time series for: output growth, inflation, interest rate
- Recover 5 parameter values using:
  1. Neural networks based approach (**extended NN, NN based PF**, RWMH)
  2. Standard approach (time iteration, regular particle filter, RWMH)

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 based PF**, RWMH)
  2. Standard approach (time iteration, regular particle filter, RWMH)

			Neural Network Posterior			Conventional Approach Posterior		
Parameter		True value	Median	5%	95%	Median	5%	95%
θ_Π	Inflation resp.	2.0	2.02	1.87	2.17	2.06	1.94	2.20
θ_Y	Output resp.	0.25	0.251	0.238	0.263	0.248	0.237	0.258
φ	Rotemberg	1000	988.6	935.1	1036.7	973.7	911.2	1037.2
ρ_ζ	Persistence	0.8	0.686	0.669	0.701	0.691	0.670	0.710
σ^ζ	Std. dev.	0.02	0.020	0.020	0.021	0.020	0.019	0.020

Estimation comparison: posterior



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 conventional method~~
   - Simple nonlinear RANK model with a ZLB

   Estimation results are very similar

3. **Estimating a nonlinear HANK model**
   - Using aggregate time-series for the US (few parameters)
   - Using simulated data from the model (more parameters)

Estimating a nonlinear HANK model

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

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

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

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

  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
- Monetary policy is constrained by the **zero lower bound**

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

Estimating a nonlinear HANK model using US data

- US time-series data from 1984Q1 to 2019:Q4
  - GDP per capita growth
  - GDP deflator
  - Federal funds rate
- Measurement equation

$$\begin{bmatrix} \text{Output growth} \\ \text{Inflation} \\ \text{Interest rate} \end{bmatrix} = \begin{bmatrix} 100\left(\frac{Y_t}{Y_{t-1}/g_t} - 1\right) \\ 400\left(\Pi_t - 1\right) \\ 400\left(R_t - 1\right) \end{bmatrix} + u_t, \text{ where } u_t \sim \mathcal{N}(0, \Sigma_u)$$

Priors for the estimated parameters

Estimation					
Parameter	Prior				
	Type	Mean	Std	Lower Bound	Upper Bound
Idiosyncratic income σ_s	Trc.N	4.5%	1.0%	1.0%	5.0%
Preference σ_ζ	Trc.N	1.5%	10.0%	1.0%	2.2%
Growth rate σ_g	Trc.N	0.4%	0.1%	0.2%	0.6%
Monetary policy σ_{mp}	Trc.N	0.13%	0.01%	0.05%	0.38%

Calibrated Parameters

Estimating a nonlinear HANK model - Extended NN part

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

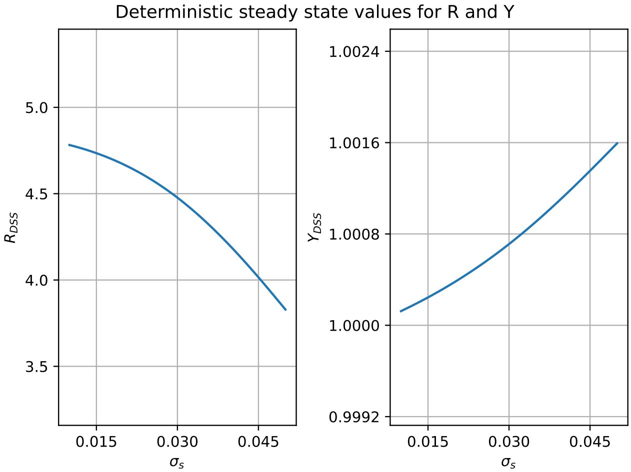
0. Instead of continuum of agents there are $L = 100$ agents
1. Policy functions parameterized by deep neural networks
   - Aggregate: inflation and wage
   - Individual: labor choice
   - **207 state variables**
     - 200 individual, 3 aggregate and 4 pseudo (parameters) states
2. Loss function is 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 ...)

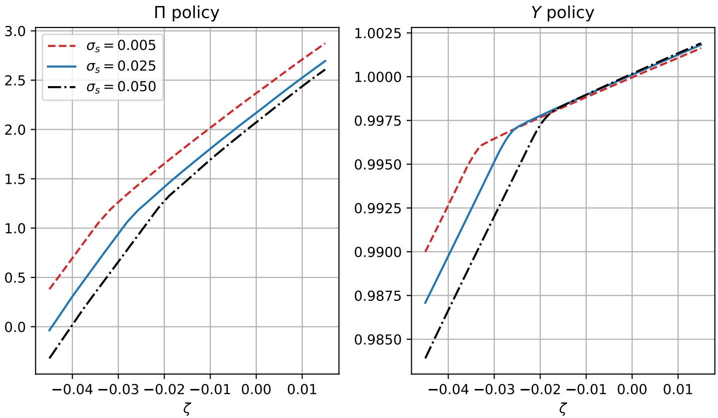
3.a. Deterministic Steady State

- Increase in idiosyncratic risk lowers market clearing rate



3.b. Aggregate policy functions

- Policy functions for inflation and output for varying preference shock
  - Zero lower bound creates nonlinearity
  - Degree of nonlinearity depends on the amount of idiosyncratic risk

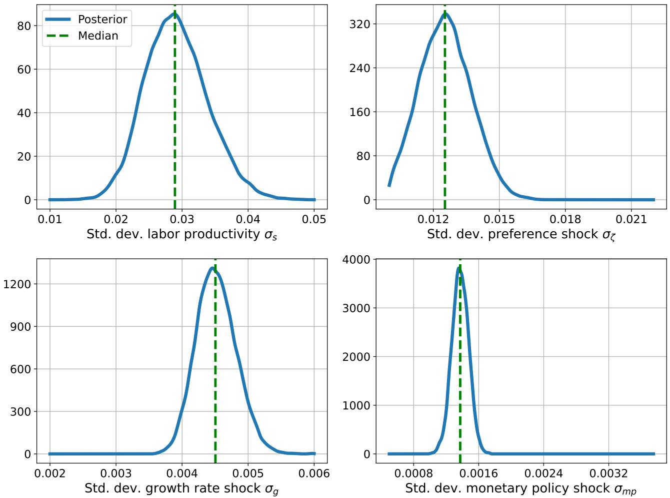


Estimating a nonlinear HANK model - NN particle filter and RWMH

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 Results: Posterior Distribution

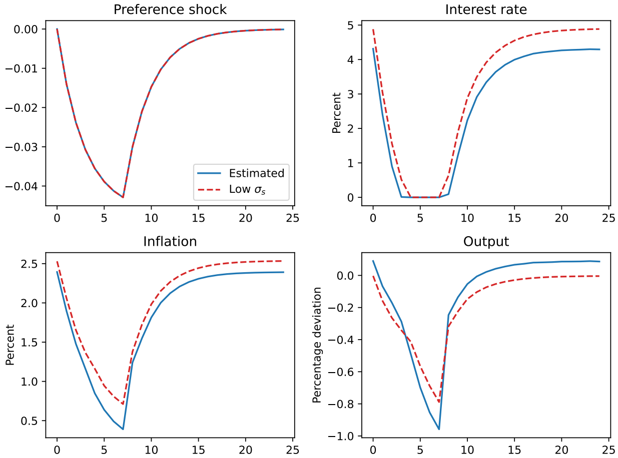
- Aggregate data helps to pin down degree of idiosyncratic risk indirectly



Table

Results: Nonlinearities and Heterogeneity

- Comparison of model at posterior median to a version with low idiosyncratic risk

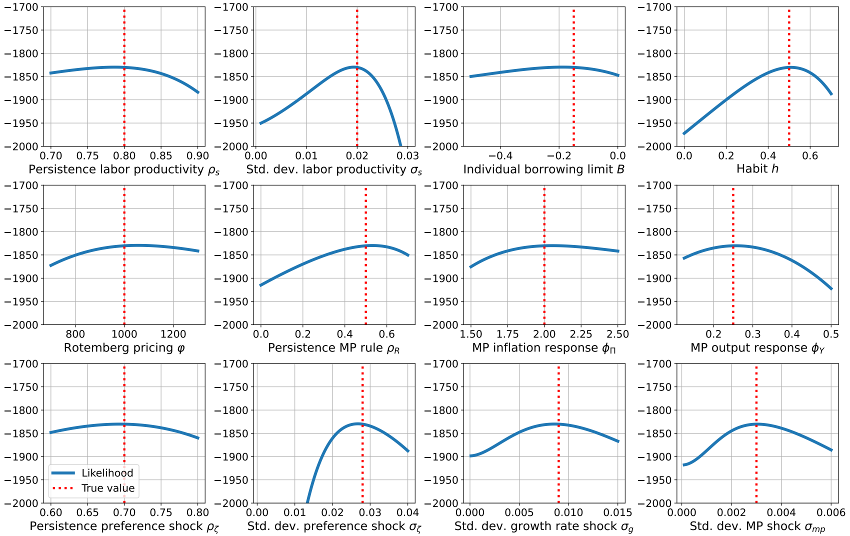


Estimating HANK with using simulated data

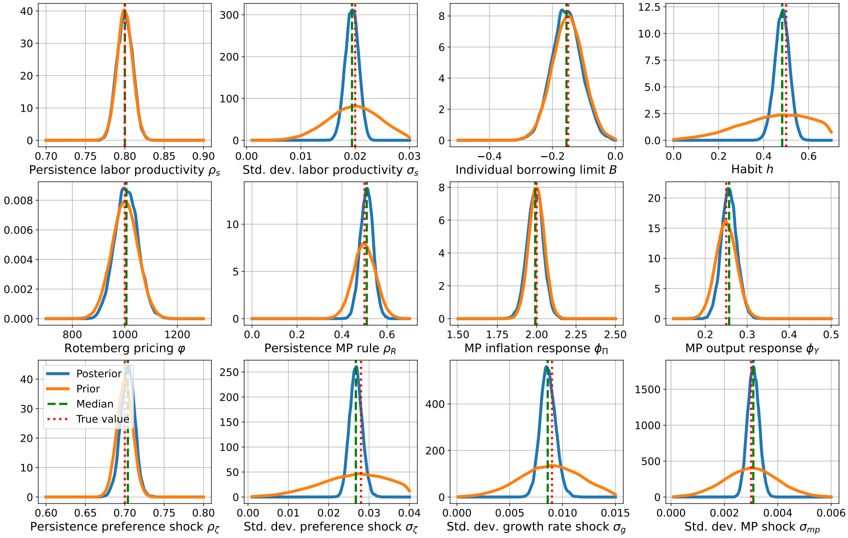
Estimation experiment:

- Use the calibrated model to create time series for:
  - Output growth
  - Inflation
  - Interest rate
- Recover 12 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

Output of the Neural Network Based Particle Filter



Estimated posterior distributions of parameters



Conclusion

Novel estimation procedure based on neural networks

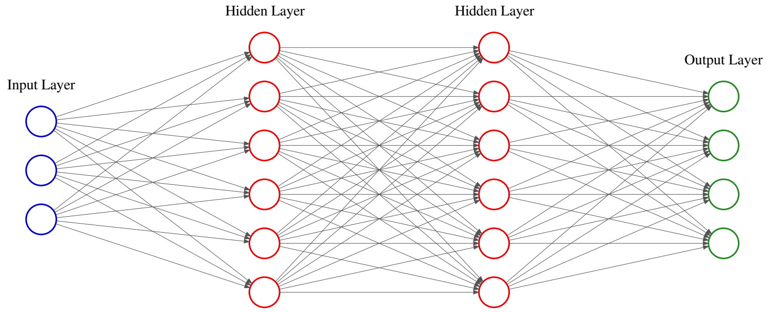
1. **Extended Neural Network** - avoid repeated solving
2. **Neural Network Based Particle Filter** - fast likelihood evaluations

- 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

Thank you!

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:

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

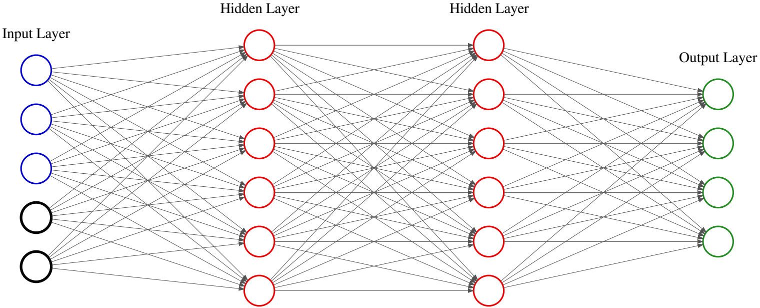
- The entire network with L hidden layers:

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

- 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

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

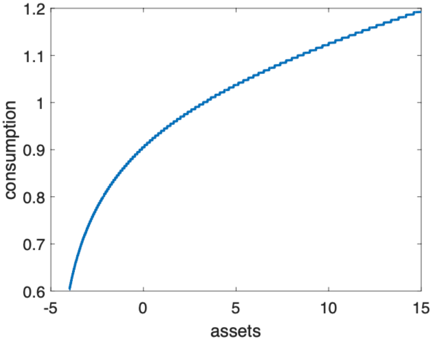
- We could **extend** the states by treat parameters Θ as additional input

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

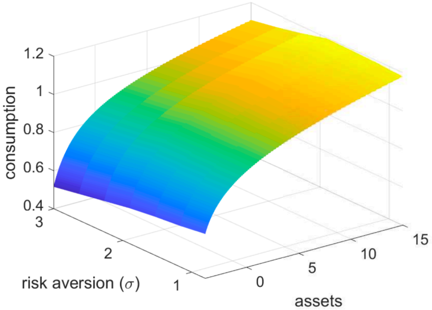
- With $\psi(\mathbb{S}_t, \Theta)$ we can quickly get the policy for different parameter values

Extended (policy function approximated by) Neural Networks

- Simple incomplete markets consumption-savings problem
- Consumption as a function of assets (and risk aversion)
- $\psi_t = \psi(\mathbb{S}_t | \Theta)$



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



from a discussion by Kieran Larkin

Extended (policy function approximated by) Neural Networks

- 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

back

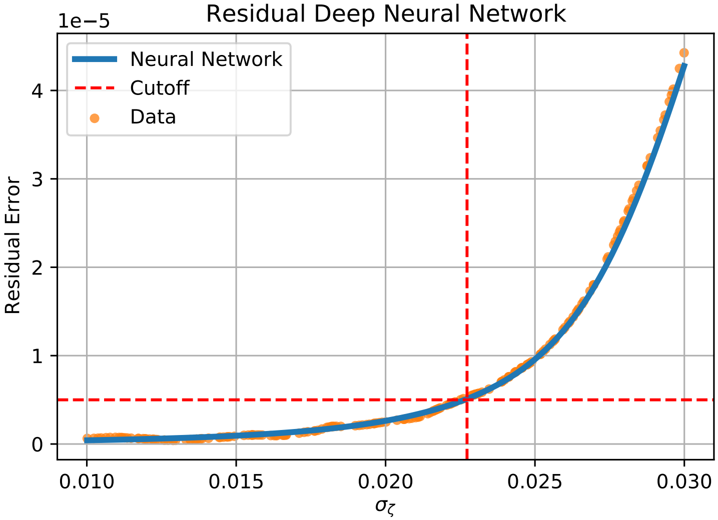
What if there is no solution?

- The Neural Network that approximates the policy functions is trained by minimizing weighted mean of squared residuals
- For parameter values where there is no solution:
  - Conventional solution method: **an error**
  - Extended Neural Network: **a value, but the residual is larger**

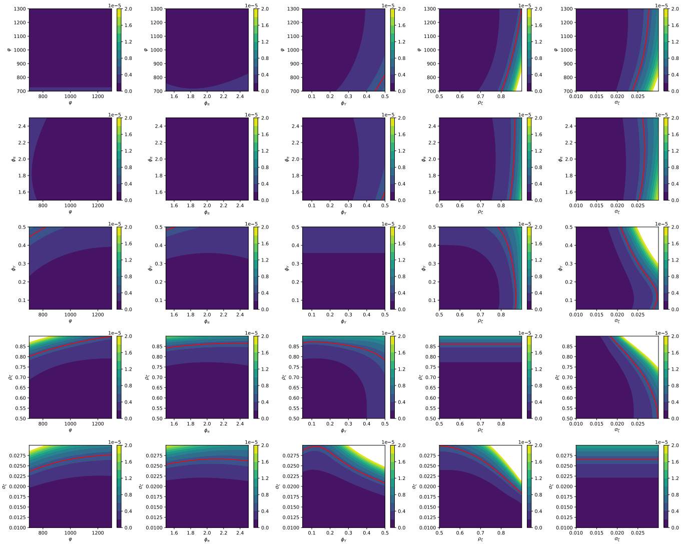
Introduce additional neural network that maps parameters to residual error

1. Create a dataset of parameter values and residuals
2. Split the dataset into training and validation samples
3. Train a neural network on the training sample
4. Pick a cutoff value to **discard bad solutions**

Solution to "What if there is no solution?"



Solution to "What if there is no solution?"



Calibration of the HANK model

Calibration		
Parameters	Description	Value
β	Discount factor	0.99825
σ	Relative risk aversion	1
η	Inverse Frisch elasticity	1
ϵ	Price elasticity demand	11
χ	Disutility labor	0.91
g	Average growth rate	1.0039
γ^{τ}	Tax progressivity	0.18
θ_{Π}	MP inflation response	2.6
θ_Y	MP output response	0.98
D	Government debt	0.25
$\underline{B}$	Individual borrowing limit	-0.15
φ	Rotemberg pricing	100
Π	Inflation target	1.00625
ρ_s	Persistence labor productivity	0.8
ρ_{ζ}	Persistence preference shock	0.7

HANK Estimation Results

Estimation								
Par.	Prior					Neural Network		
	Type	Mean	Std	Lower Bound	Upper Bound	Posterior Median	Posterior 5%	Posterior 95%
σ_s	Trc.N	4.5%	1.0%	1.0%	5.0%	2.89%	2.19%	3.73%
σ_ζ	Trc.N	1.5%	10.0%	1.0%	2.2%	1.25%	1.07%	1.45%
σ_g	Trc.N	0.4%	0.1%	0.2%	0.6%	0.45%	0.40%	0.50%
σ_{mp}	Trc.N	0.13%	0.01%	0.05%	0.38%	0.14%	0.12%	0.15%

Back