Training (and pre-training) a Neural Network for Structural Model Estimation

Max Wei[1] Zhenling Jiang[2]

[1]Marshall School of Business, University of Southern California

[2]Wharton School, University of Pennsylvania

Use Machine Learning to Estimate Structural Models

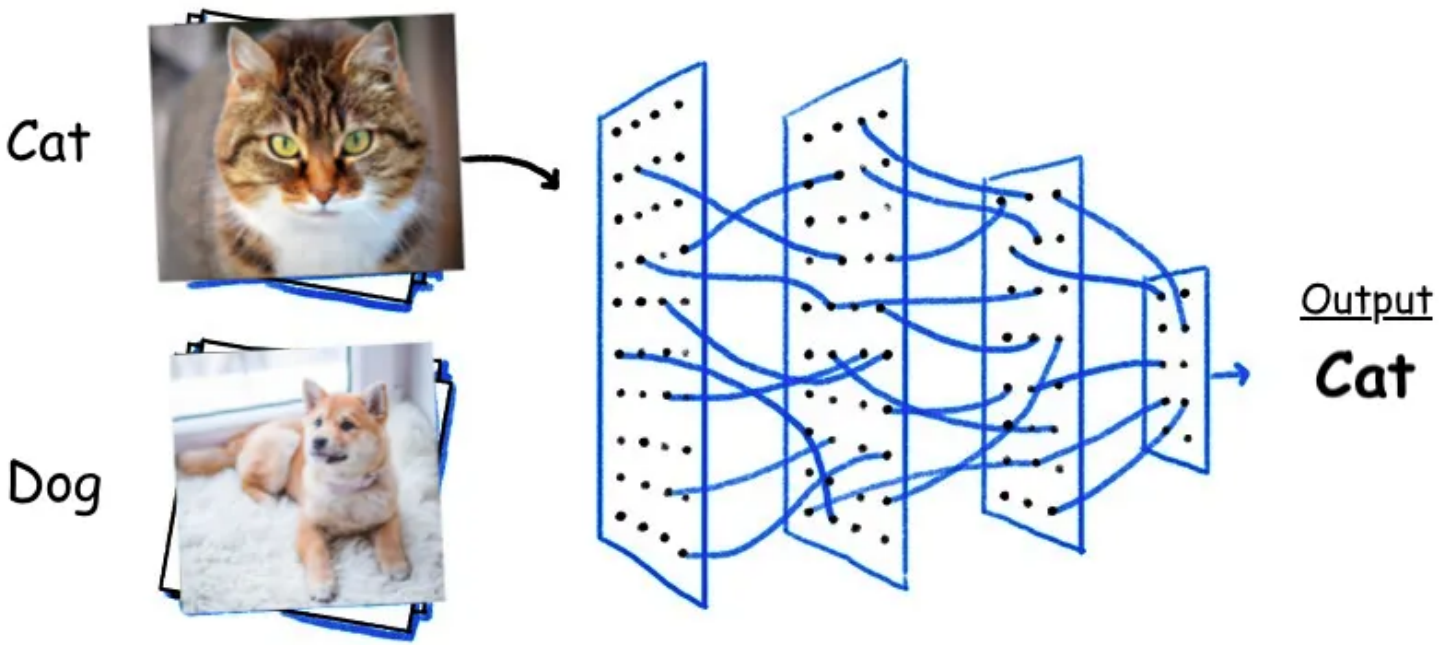
- Three broad types of empirical models:
  - Structural estimation
  - Machine learning
  - Reduced-form analysis
- Today at a high level:
  - Use machine learning techniques to make structural estimation as accessible as reduced form analysis.

Neural Networks

- Neural networks are good at predictive tasks.
- Example: image recognition

Pretrained Model for Image Recognition

- A neural net maps an image to the label “cat”.



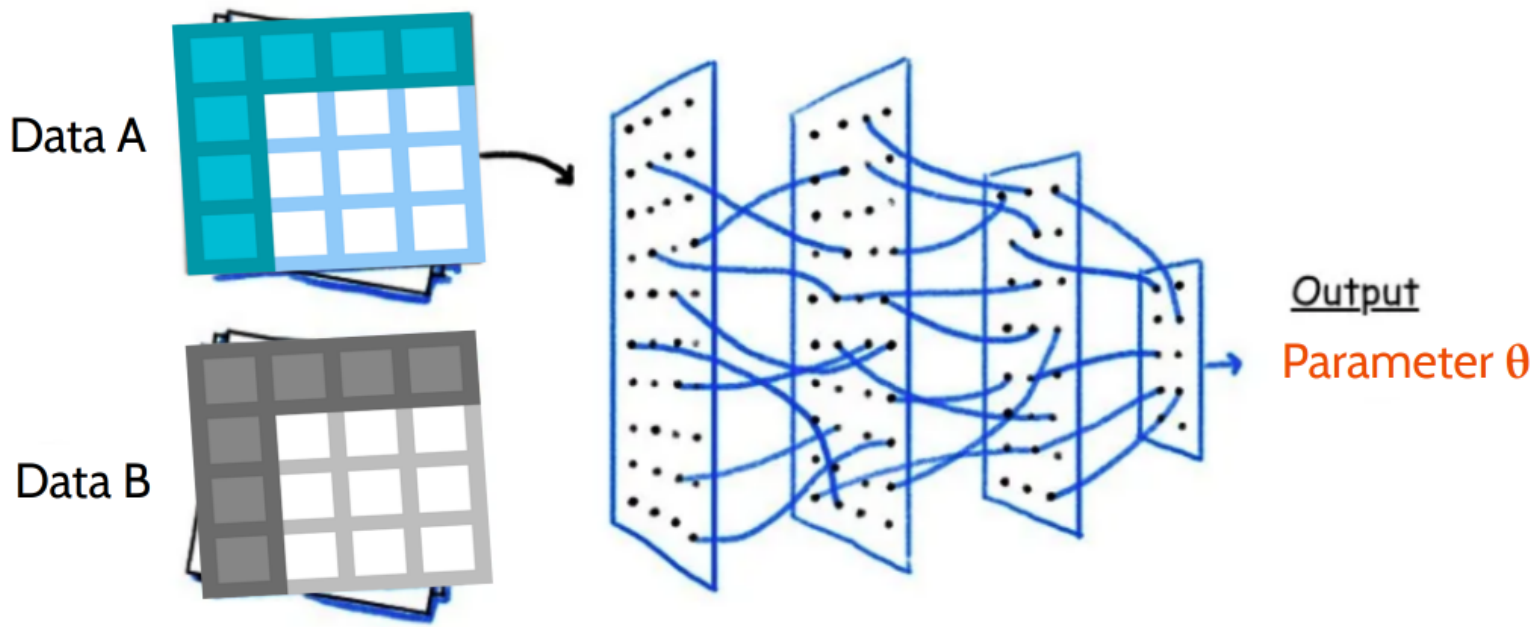
This Paper

Training (and pretraining) a neural network to estimate structural model.

Given your data $\mapsto$ parameter θ (based on a specific model)

Neural Network to Map Dataset to Model Parameter θ

- For model estimation: treat a dataset as "images", and model parameter θ as the label.



Some Intuitions

- Many structural papers show data patterns (e.g. reduced-form results), which are informative of model parameter θ.
  - Higher-ranked items are searched more $\longrightarrow$ search cost increases with ranking.
  - Lower-price items are searched more $\longrightarrow$ utility decreases with price.
- So an "**expert**" familiar with search model can guesstimate its parameter θ after seeing data patterns.
- We can train such an "**expert**" using neural net.

Some Intuitions

- Many structural papers show data patterns (e.g. reduced-form results), which are informative of model parameter θ.
  - Higher-ranked items are searched more $\rightarrow$ search cost increases with ranking.
  - Lower-price items are searched more $\rightarrow$ utility decreases with price.
- So an "**expert**" familiar with search model can guesstimate its parameter θ after seeing data patterns.
- We can train such an "**expert**" using neural net.

Structural Models and Computational Challenges

- Structural model: $\boldsymbol{y} = \boldsymbol{g}(\boldsymbol{x}, \varepsilon; \boldsymbol{\theta})$.
  - E.g., discrete choice, search model.
- $\boldsymbol{\theta}$ is typically estimated by (simulated) MLE/GMM.
  - Often high computational cost because of simulation burden, non-smooth objective functions, etc.
  - Steep learning / coding costs for researchers.

Overview

- A review of Neural Net Estimator (**NNE**)[1]
- Extend to a *pre-trained* estimator (**pNNE**)
- Conclusion

[1] Yanhao (Max) Wei, Zhenling Jiang (2024) Estimating Parameters of Structural Models Using Neural Networks. Marketing Science

NNE vs. pNNE

pNNE

like using a pre-trained image classifier

- "Off-the-shelf", as easy as running a regression and very fast.
- Trained for a specific structural model.
- Accommodate a wide range of datasets.
- Best for fairly standardized models.

NNE

like training own image classifier

- Requires coding by researchers. Compared to SMLE/SMM, can have lighter computational cost and robust to redundant moments.
- Users train for their own (structural) model and specific data.
- A wider range of applications.

Overview of NNE

- Key idea: use neural nets to learn the mapping $\{\boldsymbol{y}_i, \boldsymbol{x}_i\}_{i=1}^n \mapsto \boldsymbol{\theta}$.

$$\textit{train a neural net } f(\cdot) \quad \begin{cases} \boldsymbol{\theta}^{(1)} \xrightarrow{\boldsymbol{g}(\boldsymbol{x}_i, \boldsymbol{\varepsilon}_i^{(1)}; \boldsymbol{\theta}^{(1)})} \{\boldsymbol{y}_i^{(1)}, \boldsymbol{x}_i\}_{i=1}^n \xrightarrow{\textit{moments}} \boldsymbol{m}^{(1)} \xrightarrow{\textit{neural net}} \widehat{\boldsymbol{\theta}}^{(1)} \\ \boldsymbol{\theta}^{(2)} \longrightarrow \{\boldsymbol{y}_i^{(2)}, \boldsymbol{x}_i\}_{i=1}^n \longrightarrow \boldsymbol{m}^{(2)} \longrightarrow \widehat{\boldsymbol{\theta}}^{(2)} \\ \vdots \\ \boldsymbol{\theta}^{(L)} \longrightarrow \{\boldsymbol{y}_i^{(L)}, \boldsymbol{x}_i\}_{i=1}^n \longrightarrow \boldsymbol{m}^{(L)} \longrightarrow \widehat{\boldsymbol{\theta}}^{(L)} \end{cases}$$

$$\textit{apply } f(\cdot) \textit{ on real data} \qquad \{\underbrace{\boldsymbol{y}_i, \boldsymbol{x}_i}_{\textit{real data}}\}_{i=1}^n \xrightarrow{\textit{moments}} \boldsymbol{m} \xrightarrow{\textit{neural net}} \underbrace{\widehat{\boldsymbol{\theta}}}_{\textit{estimate}}$$

- Notation: $\boldsymbol{\theta}^{(\ell)}$ drawn from a space Θ; $\boldsymbol{y}_i = \boldsymbol{g}(\boldsymbol{x}_i, \varepsilon_i; \boldsymbol{\theta})$ a structural model; $\boldsymbol{y}^{(\ell)}$ simulated outcome; $\boldsymbol{m}^{(\ell)}$ simulated moments; $\widehat{\boldsymbol{\theta}}$ neural net prediction.

Generate Training Datasets

- Draw $\boldsymbol{\theta}^{(\ell)}$ uniformly from a parameter space Θ.
- Use the structural model to generate training datasets.
- Repeat L times to get L number of training datasets.

$$
\textit{train a neural net } f(\cdot) \left\{ \begin{array}{lllll}
\boldsymbol{\theta}^{(1)} & \xrightarrow{\boldsymbol{g}(\boldsymbol{x}_i, \boldsymbol{\varepsilon}_i^{(1)}; \boldsymbol{\theta}^{(1)})} & \{\boldsymbol{y}_i^{(1)}, \boldsymbol{x}_i\}_{i=1}^n & \xrightarrow{\textit{moments}} \boldsymbol{m}^{(1)} & \xrightarrow{\textit{neural net}} \widehat{\boldsymbol{\theta}}^{(1)} \\
\boldsymbol{\theta}^{(2)} & \longrightarrow & \{\boldsymbol{y}_i^{(2)}, \boldsymbol{x}_i\}_{i=1}^n & \longrightarrow \boldsymbol{m}^{(2)} & \longrightarrow \widehat{\boldsymbol{\theta}}^{(2)} \\
 & & \vdots & & \\
\boldsymbol{\theta}^{(L)} & \longrightarrow & \{\boldsymbol{y}_i^{(L)}, \boldsymbol{x}_i\}_{i=1}^n & \longrightarrow \boldsymbol{m}^{(L)} & \longrightarrow \widehat{\boldsymbol{\theta}}^{(L)}
\end{array} \right.
$$

Summarize with Data Moments

- Summarize $\{\boldsymbol{y}_i^{(\ell)}, \boldsymbol{x}_i\}_{i=1}^n$ with data moments $\boldsymbol{m}^{(\ell)}$.
  - Mean, variance, cross-covariance, etc.
  - The parameters need to be identified by the moments (same as in GMM).

$$\textit{train a neural net } f(\cdot) \quad \begin{cases} \boldsymbol{\theta}^{(1)} \xrightarrow{\boldsymbol{g}(\boldsymbol{x}_i, \boldsymbol{\varepsilon}_i^{(1)}; \boldsymbol{\theta}^{(1)})} \{\boldsymbol{y}_i^{(1)}, \boldsymbol{x}_i\}_{i=1}^n \xrightarrow{\textit{moments}} \boldsymbol{m}^{(1)} \xrightarrow{\textit{neural net}} \widehat{\boldsymbol{\theta}}^{(1)} \\ \boldsymbol{\theta}^{(2)} \longrightarrow \{\boldsymbol{y}_i^{(2)}, \boldsymbol{x}_i\}_{i=1}^n \longrightarrow \boldsymbol{m}^{(2)} \longrightarrow \widehat{\boldsymbol{\theta}}^{(2)} \\ \vdots \\ \boldsymbol{\theta}^{(L)} \longrightarrow \{\boldsymbol{y}_i^{(L)}, \boldsymbol{x}_i\}_{i=1}^n \longrightarrow \boldsymbol{m}^{(L)} \longrightarrow \widehat{\boldsymbol{\theta}}^{(L)} \end{cases}$$

Train a Neural Net $f : \boldsymbol{m} \mapsto \boldsymbol{\theta}$

- Train a neural net to predict $\boldsymbol{\theta}$ from moments $\boldsymbol{m}$ with MSE loss function $C = \sum_{\ell=1}^{L} \|\widehat{\boldsymbol{\theta}}^{(\ell)} - \boldsymbol{\theta}^{(\ell)}\|^2$.

$$\textit{train a neural net } f(\cdot) \begin{cases} \boldsymbol{\theta}^{(1)} \xrightarrow{\boldsymbol{g}(\boldsymbol{x}_i, \boldsymbol{\varepsilon}_i^{(1)}; \boldsymbol{\theta}^{(1)})} \{\boldsymbol{y}_i^{(1)}, \boldsymbol{x}_i\}_{i=1}^n \xrightarrow{\textit{moments}} \boldsymbol{m}^{(1)} \xrightarrow{\textit{neural net}} \widehat{\boldsymbol{\theta}}^{(1)} \\ \boldsymbol{\theta}^{(2)} \longrightarrow \{\boldsymbol{y}_i^{(2)}, \boldsymbol{x}_i\}_{i=1}^n \longrightarrow \boldsymbol{m}^{(2)} \longrightarrow \widehat{\boldsymbol{\theta}}^{(2)} \\ \vdots \\ \boldsymbol{\theta}^{(L)} \longrightarrow \{\boldsymbol{y}_i^{(L)}, \boldsymbol{x}_i\}_{i=1}^n \longrightarrow \boldsymbol{m}^{(L)} \longrightarrow \widehat{\boldsymbol{\theta}}^{(L)} \end{cases}$$

Obtain Estimates

- Once the neural net $\boldsymbol{f}$ is trained, plug real data moments to obtain estimates.

$$\textit{train a neural net } f(\cdot) \quad \begin{cases} \boldsymbol{\theta}^{(1)} \xrightarrow{\boldsymbol{g}(\boldsymbol{x}_i, \boldsymbol{\varepsilon}_i^{(1)}; \boldsymbol{\theta}^{(1)})} \{\boldsymbol{y}_i^{(1)}, \boldsymbol{x}_i\}_{i=1}^n \xrightarrow{\textit{moments}} \boldsymbol{m}^{(1)} \xrightarrow{\textit{neural net}} \widehat{\boldsymbol{\theta}}^{(1)} \\ \boldsymbol{\theta}^{(2)} \longrightarrow \{\boldsymbol{y}_i^{(2)}, \boldsymbol{x}_i\}_{i=1}^n \longrightarrow \boldsymbol{m}^{(2)} \longrightarrow \widehat{\boldsymbol{\theta}}^{(2)} \\ \vdots \\ \boldsymbol{\theta}^{(L)} \longrightarrow \{\boldsymbol{y}_i^{(L)}, \boldsymbol{x}_i\}_{i=1}^n \longrightarrow \boldsymbol{m}^{(L)} \longrightarrow \widehat{\boldsymbol{\theta}}^{(L)} \end{cases}$$

$$\textit{apply } f(\cdot) \textit{ on real data} \qquad \{\underbrace{\boldsymbol{y}_i, \boldsymbol{x}_i}_{\textit{real data}}\}_{i=1}^n \xrightarrow{\textit{moments}} \boldsymbol{m} \xrightarrow{\textit{neural net}} \underbrace{\widehat{\boldsymbol{\theta}}}_{\textit{estimate}}$$

Theoretical Results

- **Proposition**: Under conditions, as the number of training datasets $L \to \infty$, the trained neural net converges to limited-info Bayesian posterior mean $\mathbb{E}(\boldsymbol{\theta}|\boldsymbol{m})$ and covariance matrix $\mathbb{C}\text{ov}(\boldsymbol{\theta}|\boldsymbol{m})$.
  - "Limited information" because conditional on $\boldsymbol{m}$ not whole data.
  - This result holds for any fixed data size n.

Key Benefits Compared to Existing Methods

- Lighter simulation cost than traditional estimators.
  - No need to evaluate likelihood or moment functions; can be costly (many simulations at every θ trial) and introduce inaccuracy.
- Robust to redundant moments.
  - Learn which moments are useful during training.

NNE Recap

- Train a neural net to estimate structural models.
- Code available at nnehome.github.io.
- User still needs to
  - 1) simulate $\boldsymbol{y}_i = \boldsymbol{g}(\boldsymbol{x}_i, \varepsilon_i; \boldsymbol{\theta})$
  - 2) generate moments
  - 3) train a neural net

Overview

- A review of Neural Net Estimator (NNE)
- **Extend to a *pre-trained* estimator (pNNE)**
- Conclusion

Pretrained Estimator

- Benefit: Requires *minimal* coding and computation time of users.
- Scope:
  - Trained for a specific structural model (e.g., consumer search).
  - Applies to a wide range of data.
- Key innovations on top of NNE:
  - In training, $\boldsymbol{x}$ is simulated instead of taken from data.
  - $\boldsymbol{m}$ includes reduced-form regressions in addition to summary moments.
- Build a pretrained estimator for sequential search model as a proof of concept.

Consumer Sequential Search Model

- Consumer optimal sequential search model (Weitzman 1979).
  - Consumer utility: $u_{ij} = \beta' \boldsymbol{x}_{ij}^{(prod)} + e_{ij} + \epsilon_{ij}$
  - Outside option: $u_{i0} = \eta_0 + \boldsymbol{\eta}' \boldsymbol{x}_{i}^{(cons)} + \epsilon_{i0}$
  - Consumer pay search cost c_{ij} to know ϵ_{ij}: $log(c_{ij}) = \alpha_0 + \boldsymbol{\alpha}' \boldsymbol{x}_{ij}^{(ads)}$
- Optimal search strategy: which options to search, when to stop, and what to buy.
- Observe consumer search and purchase decisions.
- Estimate parameters: $\{\boldsymbol{y}, \boldsymbol{x}\} \mapsto [\alpha_0, \boldsymbol{\alpha}, \eta_0, \boldsymbol{\eta}, \beta]$.

Estimation via MLE

- Very challenging to estimate: millions of search-buy combinations for each consumer:

$$L = \prod_{i=1}^{N} \int \Big\{ z_{i1} > z_{i2} > \ldots > z_{iH} > \max_{j' \notin (1,\ldots,H)} (z_{ij'}),$$

$$max(u_{i0}, u_{i1}) < z_{i2}, \ldots, \; max(u_{i0}, u_{i1}, \ldots, u_{i,H-1}) < z_{iH}, \;\; max(u_{i0}, u_{i1}, \ldots, u_{iH}) > \max_{j' \notin (1,\ldots,H)} (z_{ij'}),$$

$$u_{ij\star} \geq max(u_{i0}, u_{i1}, \ldots, u_{iH}) \Big\} \prod_{j=1}^{J} dF(e_{ij}) \prod_{j=0}^{J} dF(\epsilon_{ij})$$

- Directly simulating likelihood is nearly impossible.
- Survey by Ursu, Seiler and Honka 2024[2]

[2] Ursu, Raluca and Seiler, Stephan and Honka, Elisabeth (2024), The Sequential Search Model: A Framework for Empirical Research. Quantitative Marketing and Economics.

pNNE Demo

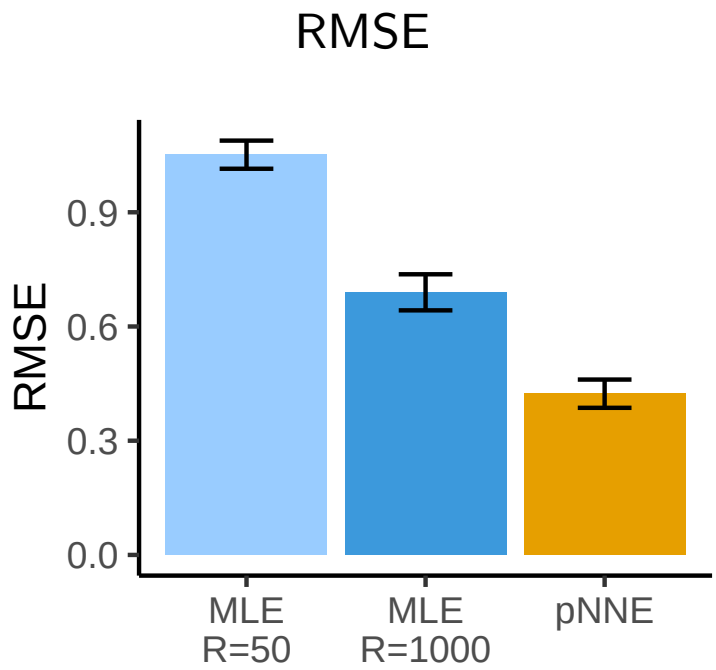
- Off-the-shelf: use on your data.
- (Beta-version) available at pnnehome.github.io.
- Python package coming very soon!

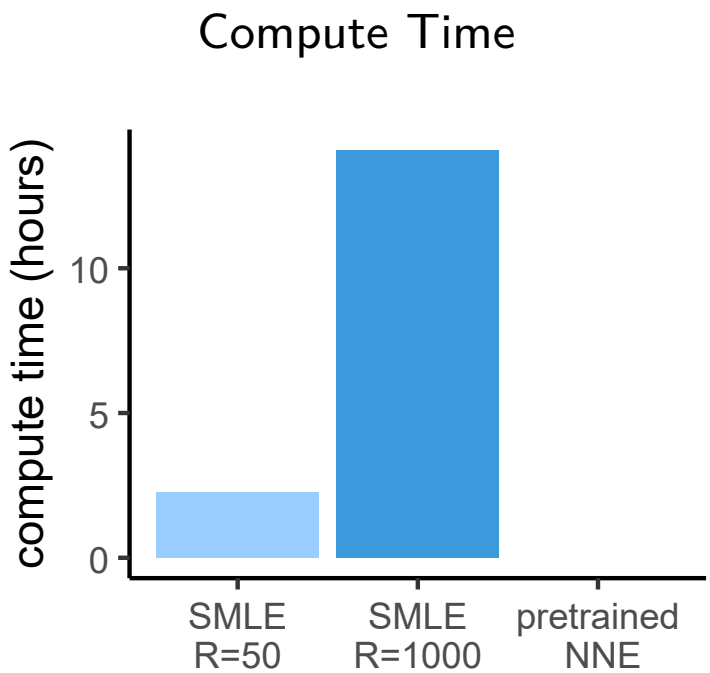
```
>>  load('trained_nne.mat', 'nne')
>>  tic
result = nne_estimate(nne, Y, Xp, Xa, Xc, consumer_idx);
toc
Elapsed time is 0.784314 seconds.
>> result
result = 8×2 table
    name           val
 ----------      -------
 "\alpha_0"      -6.4546
 "\alpha_1"     -0.11333
 "\eta_0"         5.929
 "\eta_1"      -0.048687
 "\eta_2"        0.22004
 "\eta_3"       0.052894
 "\beta_1"       0.25836
 "\beta_2"      -0.53811
```

As easy as running a reduced-form regression.

Monte Carlo

- Benchmark: MLE with a GHK-type estimator (Jiang et al. 2021[3]).
- pNNE has lower RMSE than MLE with the number of simulations S = 1000.
- pNNE is much faster (4-5 orders of magnitude difference).





[3] Jiang, Zhenling, Tat Chan, Hai Che, and Youwei Wang (2021). "Consumer search and purchase: An empirical investigation of retargeting based on consumer online behaviors." Marketing Science.

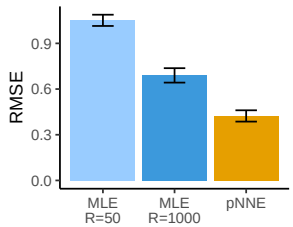
Monte Carlo: Fourteen Datasets from Different Domains

	number of consumers	product attributes	consumer attributes	"advertising" attributes
Taobao				
iPad mini	10,000	3	5	0
iPad 4	10,000	3	5	0
Expedia (Kaggle)				
Destination 1	1,258	6	2	1
Destination 2	8,97	6	2	1
Expedia (WCA)				
Cancun	2,135	6	0	1
Manhattan	1,369	6	0	1
Trivago				
PC channel	2,082	7	0	1
mobile channel	2,726	7	0	1
JD				
PC channel	5,896	4	2	0
mobile channel	4,069	4	2	0
app channel	10,000	4	2	0
Anonymous eCommerce				
laptop	10,000	6	0	0
vacuum	10,000	4	0	0
washer	10,000	5	0	0

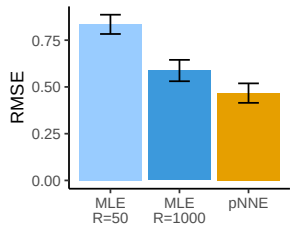
Monte Carlo Results

■: MLE (R=50) ■: MLE (R=1e3) ■: pretrained NNE

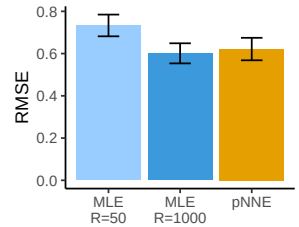
Taobao: iPad mini



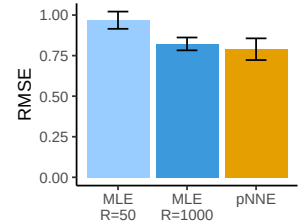
Taobao: iPad 4



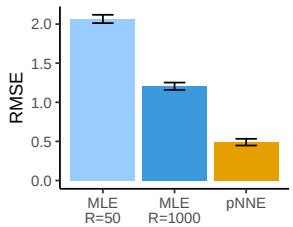
Expedia (Kaggle): Destination 1



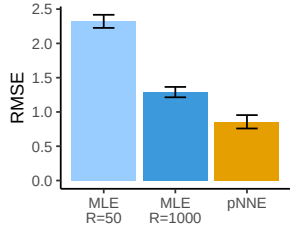
Expedia (Kaggle): Destination 2



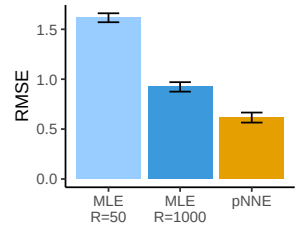
Expedia (WCA): Cancun



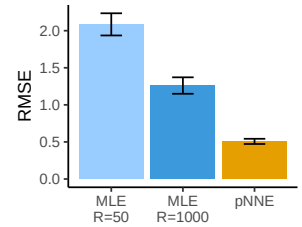
Expedia (WCA): Manhattan



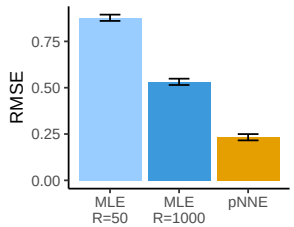
Trivago: PC channel



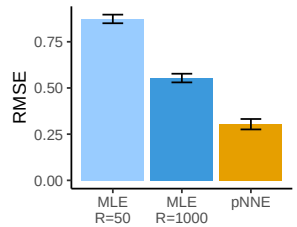
Trivago: mobile channel



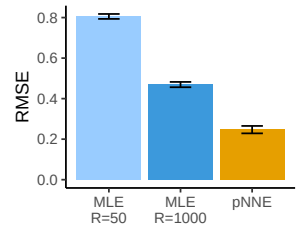
JD: PC channel



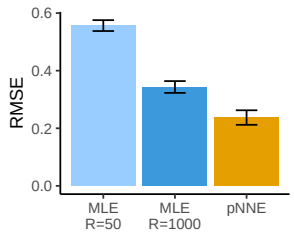
JD: mobile channel



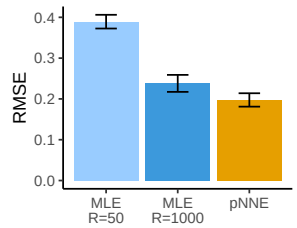
JD: app channel



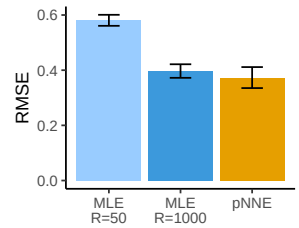
eCommerce: laptop



eCommerce: vacuum



eCommerce: washer



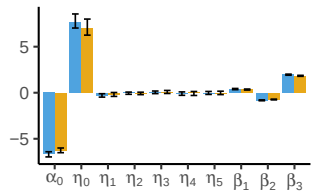
Real Data with Diverse Search Patterns

	number of consumers	buy rate	search rate	number of searches
Taobao				
iPad mini	10,000	1.16%	10.82%	1.14
iPad 4	10,000	1.23%	9.58%	1.12
Expedia (Kaggle)				
Destination 1	1,258	11.53%	8.66%	1.18
Destination 2	8,97	3.79%	6.02%	1.13
Expedia (WCA)				
Cancun	2,135	1.31%	30.82%	1.56
Manhattan	1,369	3.07%	28.71%	1.52
Trivago				
PC channel	2,082	2.83%	40.01%	2.03
mobile channel	2,726	2.46%	29.09%	1.62
JD				
PC channel	5,896	16.35%	26.46%	1.44
mobile channel	4,069	11.16%	12.17%	1.15
app channel	10,000	7.96%	12.88%	1.17
Anonymous eCommerce				
laptop	10,000	5.9%	21.5%	1.36
vacuum	10,000	8.5%	19.0%	1.29
washer	10,000	7.8%	22.1%	1.38

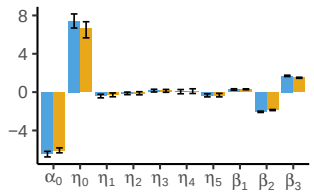
Real Data Estimation Results

■: MLE (R=1e3) ■: pretrained NNE

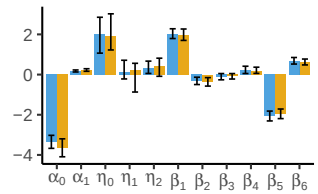
Taobao: iPad mini



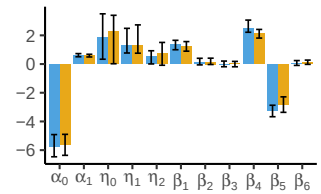
Taobao: iPad 4



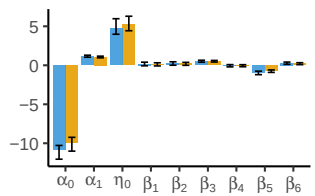
Expedia (Kaggle): Destination 1



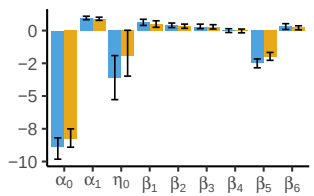
Expedia (Kaggle): Destination 2



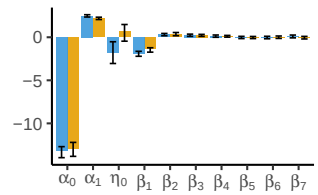
Expedia (WCA): Cancun



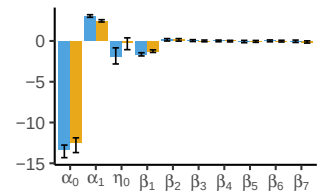
Expedia (WCA): Manhattan



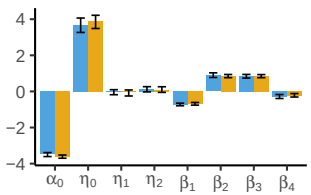
Trivago: PC channel



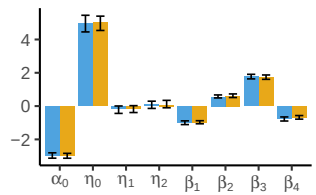
Trivago: mobile channel



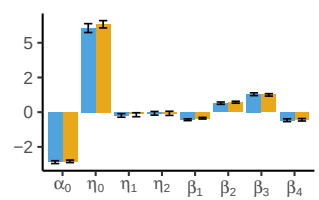
JD: PC channel



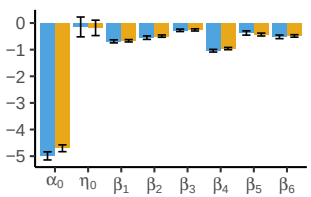
JD: mobile channel



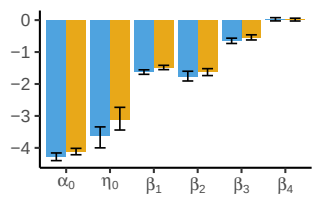
JD: app channel



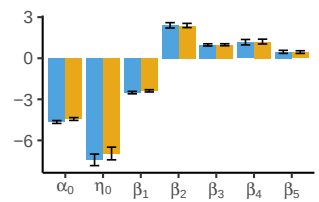
eCommerce: laptop



eCommerce: vacuum



eCommerce: washer



Overview of pNNE

- Generate training examples, construct data patterns, and train a neural net to predict $\boldsymbol{\theta}$.

$$
\textit{train pNNE} \quad \begin{cases}
\boldsymbol{\theta}^{(1)}, \boldsymbol{x}^{(1)} \xrightarrow{\boldsymbol{g}(\boldsymbol{x}_i^{(1)}, \boldsymbol{\varepsilon}_i^{(1)}; \boldsymbol{\theta}^{(1)})} \{\boldsymbol{y}_i^{(1)}, \boldsymbol{x}_i^{(1)}\}_{i=1}^n \xrightarrow[\text{regressions}]{\text{reduced form}} \boldsymbol{m}^{(1)} \xrightarrow{\text{neural net}} \widehat{\boldsymbol{\theta}}^{(1)} \\
\boldsymbol{\theta}^{(2)}, \boldsymbol{x}^{(2)} \longrightarrow \{\boldsymbol{y}_i^{(2)}, \boldsymbol{x}_i^{(2)}\}_{i=1}^n \longrightarrow \boldsymbol{m}^{(2)} \longrightarrow \widehat{\boldsymbol{\theta}}^{(2)} \\
\vdots \\
\boldsymbol{\theta}^{(L)}, \boldsymbol{x}^{(L)} \longrightarrow \{\boldsymbol{y}_i^{(L)}, \boldsymbol{x}_i^{(L)}\}_{i=1}^n \longrightarrow \boldsymbol{m}^{(L)} \longrightarrow \widehat{\boldsymbol{\theta}}^{(L)}
\end{cases}
$$

$$
\textit{apply to real data} \quad \underbrace{\{\boldsymbol{y}_i, \boldsymbol{x}_i\}_{i=1}^n}_{\textit{real data}} \xrightarrow[\text{regressions}]{\text{reduced form}} \boldsymbol{m} \xrightarrow{\text{neural net}} \underbrace{\widehat{\boldsymbol{\theta}}}_{\textit{estimate}}
$$

Overview of pNNE: Generate x

- Generate diverse $\boldsymbol{x}$ to cover possible values in real applications.

$$
\textit{train pNNE} \left\{ \begin{array}{llll}
\boldsymbol{\theta}^{(1)}, \boldsymbol{x}^{(1)} & \xrightarrow{\boldsymbol{g}(\boldsymbol{x}_i^{(1)}, \boldsymbol{\varepsilon}_i^{(1)}; \boldsymbol{\theta}^{(1)})} & \{\boldsymbol{y}_i^{(1)}, \boldsymbol{x}_i^{(1)}\}_{i=1}^n \xrightarrow[\textit{regressions}]{\textit{reduced form}} \boldsymbol{m}^{(1)} \xrightarrow{\textit{neural net}} \widehat{\boldsymbol{\theta}}^{(1)} \\
\boldsymbol{\theta}^{(2)}, \boldsymbol{x}^{(2)} & \longrightarrow & \{\boldsymbol{y}_i^{(2)}, \boldsymbol{x}_i^{(2)}\}_{i=1}^n \longrightarrow \boldsymbol{m}^{(2)} \longrightarrow \widehat{\boldsymbol{\theta}}^{(2)} \\
 & & \vdots \\
\boldsymbol{\theta}^{(L)}, \boldsymbol{x}^{(L)} & \longrightarrow & \{\boldsymbol{y}_i^{(L)}, \boldsymbol{x}_i^{(L)}\}_{i=1}^n \longrightarrow \boldsymbol{m}^{(L)} \longrightarrow \widehat{\boldsymbol{\theta}}^{(L)}
\end{array} \right.
$$

apply to real data

$$
\underbrace{\{\boldsymbol{y}_i, \boldsymbol{x}_i\}_{i=1}^n}_{\textit{real data}} \xrightarrow[\textit{regressions}]{\textit{reduced form}} \boldsymbol{m} \xrightarrow{\textit{neural net}} \underbrace{\widehat{\boldsymbol{\theta}}}_{\textit{estimate}}
$$

How to Generate Diverse x?

- Draw the data dimensionality from a range.

Specified Ranges for Data Dimensions

	$d^{(prod)}$	$d^{(ads)}$	$d^{(cons)}$	J	n
min	2	0	0	15	1e3
max	8	2	5	35	∞

- Draw from correlated multivariate normal distributions.
- Randomly transform to different non-normal distributions.
  - Dummy
  - A scale of 1 to 5
  - A skewed continuous distribution
- **Standardize** each attribute.

Overview of pNNE: Draw $\boldsymbol{\theta}$

- Draw $\boldsymbol{\theta}$ from a prior distribution.
  - Working with standardized $\boldsymbol{x}$ is very helpful.
  - Need the prior to be wide enough to cover possible values but not too wide.

train pNNE

$$\begin{cases} \boldsymbol{\theta}^{(1)}, \boldsymbol{x}^{(1)} \xrightarrow{\boldsymbol{g}(\boldsymbol{x}_i^{(1)}, \boldsymbol{\varepsilon}_i^{(1)}; \boldsymbol{\theta}^{(1)})} \{\boldsymbol{y}_i^{(1)}, \boldsymbol{x}_i^{(1)}\}_{i=1}^n \xrightarrow[\text{regressions}]{\text{reduced form}} \boldsymbol{m}^{(1)} \xrightarrow{\text{neural net}} \widehat{\boldsymbol{\theta}}^{(1)} \\ \boldsymbol{\theta}^{(2)}, \boldsymbol{x}^{(2)} \longrightarrow \{\boldsymbol{y}_i^{(2)}, \boldsymbol{x}_i^{(2)}\}_{i=1}^n \longrightarrow \boldsymbol{m}^{(2)} \longrightarrow \widehat{\boldsymbol{\theta}}^{(2)} \\ \vdots \\ \boldsymbol{\theta}^{(L)}, \boldsymbol{x}^{(L)} \longrightarrow \{\boldsymbol{y}_i^{(L)}, \boldsymbol{x}_i^{(L)}\}_{i=1}^n \longrightarrow \boldsymbol{m}^{(L)} \longrightarrow \widehat{\boldsymbol{\theta}}^{(L)} \end{cases}$$

apply to real data

$$\{\underbrace{\boldsymbol{y}_i, \boldsymbol{x}_i}_{\text{real data}}\}_{i=1}^n \xrightarrow[\text{regressions}]{\text{reduced form}} \boldsymbol{m} \xrightarrow{\text{neural net}} \underbrace{\widehat{\boldsymbol{\theta}}}_{\text{estimate}}$$

Overview of pNNE: Generate Training Data

- With $\boldsymbol{x}^{(l)}$ and $\boldsymbol{\theta}^{(l)}$, use the structural model to generate $\boldsymbol{y}^{(l)}$.
- Repeat L times to get L number of training datasets.

$$
\textit{train pNNE} \begin{cases}
\boldsymbol{\theta}^{(1)}, \boldsymbol{x}^{(1)} \xrightarrow{\boldsymbol{g}(\boldsymbol{x}_i^{(1)}, \boldsymbol{\varepsilon}_i^{(1)}; \boldsymbol{\theta}^{(1)})} \{\boldsymbol{y}_i^{(1)}, \boldsymbol{x}_i^{(1)}\}_{i=1}^n \xrightarrow[\text{regressions}]{\textit{reduced form}} \boldsymbol{m}^{(1)} \xrightarrow{\textit{neural net}} \widehat{\boldsymbol{\theta}}^{(1)} \\
\boldsymbol{\theta}^{(2)}, \boldsymbol{x}^{(2)} \longrightarrow \{\boldsymbol{y}_i^{(2)}, \boldsymbol{x}_i^{(2)}\}_{i=1}^n \longrightarrow \boldsymbol{m}^{(2)} \longrightarrow \widehat{\boldsymbol{\theta}}^{(2)} \\
\vdots \\
\boldsymbol{\theta}^{(L)}, \boldsymbol{x}^{(L)} \longrightarrow \{\boldsymbol{y}_i^{(L)}, \boldsymbol{x}_i^{(L)}\}_{i=1}^n \longrightarrow \boldsymbol{m}^{(L)} \longrightarrow \widehat{\boldsymbol{\theta}}^{(L)}
\end{cases}
$$

$$
\textit{apply to real data} \quad \{\underbrace{\boldsymbol{y}_i, \boldsymbol{x}_i}_{\textit{real data}}\}_{i=1}^n \xrightarrow[\text{regressions}]{\textit{reduced form}} \boldsymbol{m} \xrightarrow{\textit{neural net}} \underbrace{\widehat{\boldsymbol{\theta}}}_{\textit{estimate}}
$$

Overview of pNNE: Construct Data Patterns

- Summarize the key information with $\boldsymbol{m}$.

$$
\textit{train pNNE} \begin{cases}
\boldsymbol{\theta}^{(1)}, \boldsymbol{x}^{(1)} \xrightarrow{\boldsymbol{g}(\boldsymbol{x}_i^{(1)}, \boldsymbol{\varepsilon}_i^{(1)}; \boldsymbol{\theta}^{(1)})} \{\boldsymbol{y}_i^{(1)}, \boldsymbol{x}_i^{(1)}\}_{i=1}^n \xrightarrow[\textit{regressions}]{\textit{reduced form}} \boldsymbol{m}^{(1)} \xrightarrow{\textit{neural net}} \widehat{\boldsymbol{\theta}}^{(1)} \\
\boldsymbol{\theta}^{(2)}, \boldsymbol{x}^{(2)} \longrightarrow \{\boldsymbol{y}_i^{(2)}, \boldsymbol{x}_i^{(2)}\}_{i=1}^n \longrightarrow \boldsymbol{m}^{(2)} \longrightarrow \widehat{\boldsymbol{\theta}}^{(2)} \\
\vdots \\
\boldsymbol{\theta}^{(L)}, \boldsymbol{x}^{(L)} \longrightarrow \{\boldsymbol{y}_i^{(L)}, \boldsymbol{x}_i^{(L)}\}_{i=1}^n \longrightarrow \boldsymbol{m}^{(L)} \longrightarrow \widehat{\boldsymbol{\theta}}^{(L)}
\end{cases}
$$

$$
\textit{apply to real data} \quad \underbrace{\{\boldsymbol{y}_i, \boldsymbol{x}_i\}_{i=1}^n}_{\textit{real data}} \xrightarrow[\textit{regressions}]{\textit{reduced form}} \boldsymbol{m} \xrightarrow{\textit{neural net}} \underbrace{\widehat{\boldsymbol{\theta}}}_{\textit{estimate}}
$$

How to Construct Informative Data Patterns?

- Regression coefficients:
  - Search decisions: Logit regression of $y_{ij}^{(search)}$ on product and consumer attributes.
  - Purchase decisions: Multinomial logit regression of $y_{ij}^{(buy)}$ among searched products.
  - Similar regressions at the consumer level.
- Summary statistics:
  - Buy rate, search rate, average number of searches.

Overview of pNNE: Train a Neural Net

- Train a neural net to predict $\boldsymbol{\theta}$ from moments $\boldsymbol{m}$.
  - MSE loss function $C = \sum_{\ell=1}^{L} \|\widehat{\boldsymbol{\theta}}^{(\ell)} - \boldsymbol{\theta}^{(\ell)}\|^2$

$$\textit{train pNNE} \begin{cases} \boldsymbol{\theta}^{(1)}, \boldsymbol{x}^{(1)} \xrightarrow{\boldsymbol{g}(\boldsymbol{x}_i^{(1)}, \boldsymbol{\varepsilon}_i^{(1)}; \boldsymbol{\theta}^{(1)})} \{\boldsymbol{y}_i^{(1)}, \boldsymbol{x}_i^{(1)}\}_{i=1}^n \xrightarrow[\text{regressions}]{\text{reduced form}} \boldsymbol{m}^{(1)} \xrightarrow{\text{neural net}} \widehat{\boldsymbol{\theta}}^{(1)} \\ \boldsymbol{\theta}^{(2)}, \boldsymbol{x}^{(2)} \longrightarrow \{\boldsymbol{y}_i^{(2)}, \boldsymbol{x}_i^{(2)}\}_{i=1}^n \longrightarrow \boldsymbol{m}^{(2)} \longrightarrow \widehat{\boldsymbol{\theta}}^{(2)} \\ \vdots \\ \boldsymbol{\theta}^{(L)}, \boldsymbol{x}^{(L)} \longrightarrow \{\boldsymbol{y}_i^{(L)}, \boldsymbol{x}_i^{(L)}\}_{i=1}^n \longrightarrow \boldsymbol{m}^{(L)} \longrightarrow \widehat{\boldsymbol{\theta}}^{(L)} \end{cases}$$

$$\textit{apply to real data} \quad \{\underbrace{\boldsymbol{y}_i, \boldsymbol{x}_i}_{\textit{real data}}\}_{i=1}^n \xrightarrow[\text{regressions}]{\text{reduced form}} \boldsymbol{m} \xrightarrow{\text{neural net}} \underbrace{\widehat{\boldsymbol{\theta}}}_{\textit{estimate}}$$

Overview of pNNE: Obtain Estimates

- Users plug in their data to the pretrained model.

$$
\textit{train pNNE} \left\{ \begin{array}{lllll}
\theta^{(1)}, x^{(1)} & \xrightarrow{g(x_i^{(1)}, \varepsilon_i^{(1)}; \theta^{(1)})} & \{y_i^{(1)}, x_i^{(1)}\}_{i=1}^n & \xrightarrow[\text{regressions}]{\textit{reduced form}} & m^{(1)} \xrightarrow{\textit{neural net}} \widehat{\theta}^{(1)} \\
\theta^{(2)}, x^{(2)} & \longrightarrow & \{y_i^{(2)}, x_i^{(2)}\}_{i=1}^n & \longrightarrow & m^{(2)} \longrightarrow \widehat{\theta}^{(2)} \\
 & & \vdots & & \\
\theta^{(L)}, x^{(L)} & \longrightarrow & \{y_i^{(L)}, x_i^{(L)}\}_{i=1}^n & \longrightarrow & m^{(L)} \longrightarrow \widehat{\theta}^{(L)}
\end{array} \right.
$$

apply to real data

$$
\{ \underbrace{y_i, x_i}_{\textit{real data}} \}_{i=1}^n \xrightarrow[\text{regressions}]{\textit{reduced form}} m \xrightarrow{\textit{neural net}} \underbrace{\widehat{\theta}}_{\textit{estimate}}
$$

"Privacy-preserving" Structural Estimation

- Researchers only need to know the aggregated data moments $\boldsymbol{m}$ to estimate the model.

$$\textit{train pNNE} \begin{cases} \theta^{(1)}, \boldsymbol{x}^{(1)} \xrightarrow{g(x_i^{(1)}, \varepsilon_i^{(1)}; \theta^{(1)})} \{y_i^{(1)}, x_i^{(1)}\}_{i=1}^n \xrightarrow[\text{regressions}]{\text{reduced form}} \boldsymbol{m}^{(1)} \xrightarrow{\text{neural net}} \widehat{\theta}^{(1)} \\ \theta^{(2)}, \boldsymbol{x}^{(2)} \longrightarrow \{y_i^{(2)}, x_i^{(2)}\}_{i=1}^n \longrightarrow \boldsymbol{m}^{(2)} \longrightarrow \widehat{\theta}^{(2)} \\ \vdots \\ \theta^{(L)}, \boldsymbol{x}^{(L)} \longrightarrow \{y_i^{(L)}, x_i^{(L)}\}_{i=1}^n \longrightarrow \boldsymbol{m}^{(L)} \longrightarrow \widehat{\theta}^{(L)} \end{cases}$$

apply to real data

$$\underbrace{\{y_i, x_i\}_{i=1}^n \xrightarrow[\text{regressions}]{\text{reduced form}} \boldsymbol{m}}_{\textbf{company compiles moments}} \underbrace{\xrightarrow{\text{neural net}} \widehat{\theta}_{estimate}}_{\textbf{researchers apply pNNE}}$$

Conclusion

- One can train (and use a pre-trained) neural network model to estimate structural models.
- Key idea: learn the mapping from reduced-form patterns to parameters.
- pNNE is trained for a specific structural model (consumer search as a proof of concept) while NNE can be applied to a wide set of applications.

Potential Applications of pNNE

- Privacy sensitive scenarios: Estimation without access to individual-level data.
  - The (aggregated) reduced-form patterns serve as sufficient statistics for the structural model.
- A fast "package" for model estimation.
  - Scalable to larger data.
  - Larger models where search model is a component and needs to be estimated repeated (e.g., dynamic models).
  - Incorporate search model into algorithms (e.g., Bandit) that are updated in real time.