Deep Neural Networks for Estimation and Inference

Max H. Farrell

with Tengyuan Liang & Sanjog Misra

Introduction & Outline

1. **Context: Nonparametric Estimation**

- Goal in this paper is to study a standard problem with a "new" tool
- How does deep learning do in a problem we understand well
- We not addressing: high dimensionality, adaptivity, double-descent, ...

2. **Main Results**

- Nonasymptotic high-probability bounds and implied rates
- "Industry standard" setup: multi-layer perceptron, ReLU activation
- Generic results for other architectures

3. **Deep Learning In Economics** (if time)

- Structured models for individual heterogeneity
- Structured network to match
- Semiparametric inference

Deep Feed-Forward Neural Networks:

Problem Set Up

Nonparametric Problem

General regression-type problem

- Outcome Y, d covariates $\boldsymbol{X} \in \mathbb{R}^d$
- Target is $f_\star = \arg\min \mathbb{E}\left[\ell\left(f, Y, \boldsymbol{X}\right)\right]$
- Any loss function such that:
  1. Lipschitz: $\left|\ell(f, y, \boldsymbol{x}) - \ell(g, y, \boldsymbol{x})\right| \leq C_\ell \left|f(\boldsymbol{x}) - g(\boldsymbol{x})\right|$
  2. Curvature: $c_1 \mathbb{E}\left[(f - f_\star)^2\right] \leq \mathbb{E}[\ell(f, Y, \boldsymbol{X})] - \mathbb{E}[\ell(f_\star, Y, \boldsymbol{X})] \leq c_2 \mathbb{E}\left[(f - f_\star)^2\right]$
  - Includes least squares, logistic, poisson, ...

DNN estimator

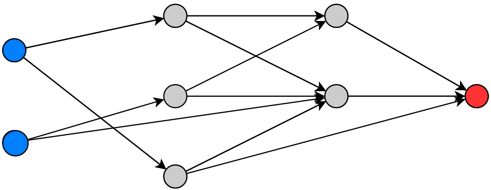
$$\widehat{f}_{\text{DNN}} := \underset{f_{\boldsymbol{\theta}} \in \mathcal{F}_{\text{DNN}}}{\arg\min} \sum_{i=1}^{n} \ell\left(f, y_i, \boldsymbol{x}_i\right), \qquad \text{e.g.} \quad \ell\left(f, y_i, \boldsymbol{x}_i\right) = \frac{1}{2}(y - f(\boldsymbol{x}))^2$$

- No optimization issues here
- Sometimes helpful to think of a linear sieve, but with learned basis: $\widehat{f}(\boldsymbol{x}) = \hat{\boldsymbol{p}}(\boldsymbol{x})'\hat{\boldsymbol{\gamma}}$

Feedforward Neural Networks

A set of units:

- $d = \dim(\boldsymbol{X})$ input units
- One output unit, Y
- U hidden units between



Units are arranged into layers:

- According to a directed, acyclic graph
- Unit is in layer l if it has a predecessor in $l-1$ and none for any $l' \geq l$
- For parameters $\boldsymbol{w}_{h,l}$ and $b_{h,l}$, unit h in layer l computes $\tilde{x}_{h,l+1} = \sigma(\tilde{\boldsymbol{x}}_l' \boldsymbol{w}_{h,l} + b_{h,l})$
  $\hookrightarrow$ layer l returns $\tilde{\boldsymbol{x}}_{l+1} = (\tilde{x}_{1,l+1}, \ldots, \tilde{x}_{H_l,l+1})$
- Dimension of $\tilde{\boldsymbol{x}}_l = H_l =$ *width*
- Number of layers $= L =$ *depth*
- Final layer outputs appropriate $\widehat{f}_{\text{DNN}}(\boldsymbol{x})$, e.g., for LS $\widehat{f} = \tilde{\boldsymbol{x}}_L(\boldsymbol{x})' \boldsymbol{w}_L + b_L$

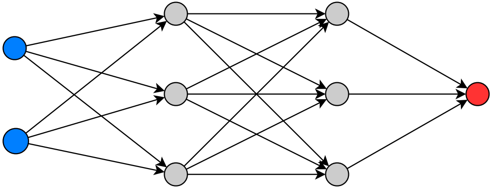
Computation

- Layer by layer, back-propagation implements chain rule
- $\boldsymbol{\theta} =$ all *weights* and *"biases"* $\{(\boldsymbol{w}_{h,l}', b_{h,l})\}$; W total parameters

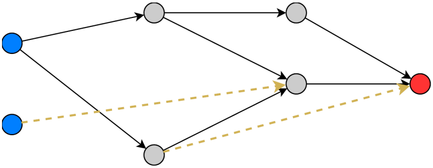
Architecture Choice 1: Multi-Layer Perceptron

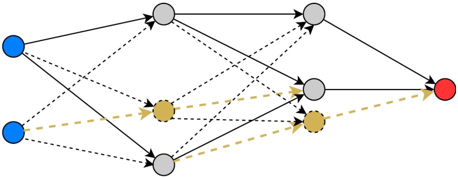
MLP a.k.a. Fully-connected feedforward network

- Fully connected to *adjacent* layers
- No hyperlinks
- Not necessarily “rectangle”
- Now common practice
  (though many others exist)



Embed any feedforward network in an MLP:

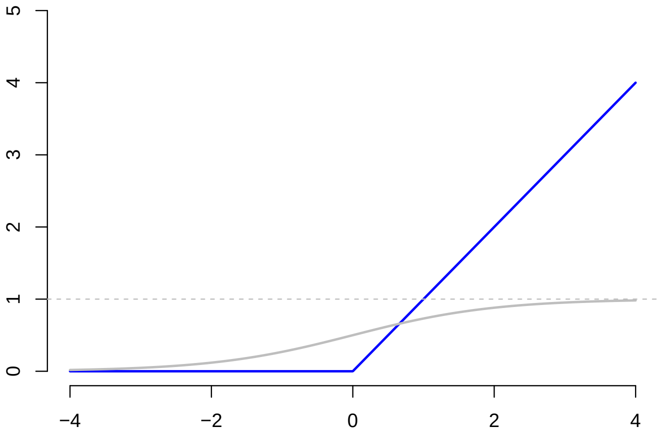




Architecture Choice 2: Rectified Linear Unit Activation Function

The activation function for each node: $\sigma(z) = \max(0, z)$

- Used at each node: $(\tilde{\boldsymbol{x}}_l' \boldsymbol{w}_l + b_l) \mapsto \sigma(\tilde{\boldsymbol{x}}_l' \boldsymbol{w}_l + b_l)$
- The ReLU activation is ubiquitous, replacing sigmoid-type
- Better optimization properties, does not get stuck in zero-gradient areas
- Conjecture: same results hold for any piecewise linear activation, slight modifications



Architecture Choice 3: Unbounded Weights

Parameters of the DNN are "slopes" $\boldsymbol{w}_{h,l}$ and "intercepts" $b_{h,l}$

- Each node $\tilde{\boldsymbol{x}}_l \mapsto (\tilde{\boldsymbol{x}}_l' \boldsymbol{w}_{h,l} + b_{h,l}) \mapsto \sigma(\tilde{\boldsymbol{x}}_l' \boldsymbol{w}_{h,l} + b_{h,l})$
- Common in theory to impost a bound on all parameters:

$$\max_{l \leq L} \max_{h \leq H_l} \|\boldsymbol{w}_{h,l}\|_\infty \vee |b_{h,l}| \leq B$$

- The reason: bounded weights make complexity arguments easier

Practically this seems innocuous

- Because empirically weights are often "small"
- And certainly optimizers don't return $\boldsymbol{w}_{h,l} = \infty$

But actually:

- Bounds cause corner solutions and other computational problems
- How do you set the bound?
- Is this some type of regularization?

What about in theory?

Architecture Choice 3: Unbounded Weights

For intuition, go back to a linear regression model:

$$y_i = x_i\beta + \varepsilon_i$$

The analogue of bounded weights in a NN is a bound on $|\hat{\beta}|$

Seems innocuous: **all** theory and **all** practice assume $|\beta|$ bounded

- Theory: $\mathbb{E}\left[\hat{\beta}_{\text{OLS}} \mid \boldsymbol{X}_n\right] = \beta$
- Practice: How many papers have a table with $\hat{\beta} = \infty$?

The argument goes like this:

1. If $|\beta| < \infty$ (which we all agree on), then there is some $B > 0$ such that $|\beta| < B$
2. The exact constant B doesn't matter, because I can always rescale:

$$y_i = x_i\beta + \varepsilon_i = y_i = \left(x_i B\right)\frac{\beta}{B} + \varepsilon_i$$

3. Therefore, without loss of generality, I restrict estimation to $\left|\hat{\beta}\right| \leq 1$

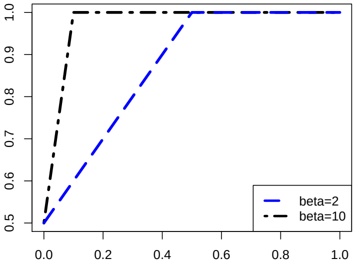
Wait, what?

Architecture Choice 3: Unbounded Weights

Same thing for **neural networks**: instead of *rescaling*, we do a *copying* trick

Simple example:

$$f_\star = \frac{\text{ReLU}(\beta x + 1) - \text{ReLU}(\beta x - 1)}{2}$$



With unbounded weights

1. 2 hidden units for any β
2. $\sqrt{1/n}$ rate for any β

With bounded weights ≤ 1

1. 2β units required
2. $\sqrt{\beta/n}$ rate

If $|\beta| < B$, then "asymptotically" this **copying** makes no difference

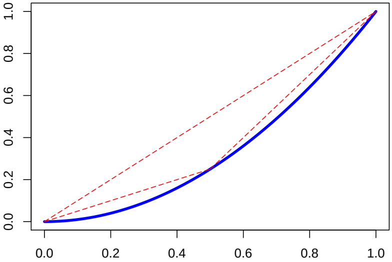
Architecture Choice 3: Unbounded Weights

For nonparametrics, the copying happens on the ideal approximating network

ReLU universal approximation idea:

1. Composition of ReLU = piecewise linear
2. Piecewise linear $\approx$ any polynomial
3. Smooth fcn $\approx$ Taylor series = polynomial

$\Rightarrow$ ReLU $\approx$ any smooth fcn (Yarotsky)



Now argue just like before

1. If $f_\star$ is smooth (say Hölder p) then $\max_{k \le p} \left\| f_\star^{(k)} \right\|_\infty < B$, some B
2. Therefore with bounded weights ≤ 1, need $2Bp =$ "$O(1)$" ReLU units to $\approx f_\star$

There's a **big** difference between $|\beta| < B$ for **some** B vs a **specified, known** B

Nonasymptotic High Probability Bounds

and

Implied Convergence Rates

Main Results

Generic Architecture

- Let $\mathcal{F}_{\text{DNN}}$ be a generic class of feed-forward DNNs with ReLU activation
- Depth $= L$, total parameters $= W$
- Assume $f_\star$ is bounded. Define the bias as $\epsilon_{\text{DNN}} := \inf_{f \in \mathcal{F}_{\text{DNN}}} \|f - f_\star\|_\infty$

Then we have the general result:

With probability at least $1 - e^{-\gamma}$:

$$\mathbb{E}_n\left[(\widehat{f}_{\text{DNN}} - f_\star)^2\right] \leq C\left(\frac{WL\log W}{n}\log n + \frac{\log\log n + \gamma}{n} + \epsilon_{\text{DNN}}^2\right)$$

Comments

- Nonasymptotic, high-probability bounds. Convergence rates follow immediately.
- Final bound depends on the architecture and the assumed function space

Main Results

Multi-Layer Perceptrons and **Smooth Functions**

- Matching standard practice: MLP + ReLU
- For MLP: $W = H_n^2 L_n$, $H_n =$ common width order
- Standard smoothness assumption: $f_\star$ is p-smooth

A leading, important case of the first result is then

$$\mathbb{E}_n\left[(\widehat{f}_{\mathrm{MLP}} - f_\star)^2\right] = O_{\mathbb{P}}\left(\frac{H_n^2 L_n^2 \log(H_n^2 L_n)}{n} \log n + \epsilon_{\mathrm{MLP}}^2\right)$$

$$\ldots\text{at best} = O_{\mathbb{P}}\left(n^{-\frac{p}{p+d}} \log^8 n\right)$$

Comments

- Best rate uses $H_n \asymp n^{\frac{d}{2(p+d)}} \log^2 n$ and $L_n \asymp \log n$
- Fast enough for semiparametrics ... but not optimal $(n^{-\mathbf{2}p/(\mathbf{2}p+d)})$
- Relies on best-known approximation results for MLP-ReLU: $\epsilon_{\mathrm{MLP}} = \epsilon_{\mathrm{MLP}}(H_n, L_n)$
- Loose bounds? Or really suboptimal?

Main Results

Two other interesting results

1. Optimal rate
   - A special, cooked-up architecture delivers $n^{-\frac{2p}{2p+d}}$, i.e. Stone's bound
   - Only of theoretical interest; architecture not practical

2. Fixed-width
   - An MLP, with $H = 2d + 10 \not\to \infty$ and $L_n \asymp n^{\frac{d}{2(2+d)}}$, can obtain $n^{-\frac{2}{2+d}}$
   - Most **different** from 1990s results
   - Interesting in transfer learning, but limited by rate (i.e. $p = 1$)

Comment: future results?

- New approximations (bounds on bias ϵ_{DNN}) can yield new rates immediately
- E.g.: fixed-width for $p > 1$ $\rightarrow$ Corollary 2 immediately sharpens
- E.g.: Sharper ϵ_{MLP} for MLP-ReLU $\rightarrow$ Faster rate on previous slide

Proof Intuition

Start with usual decomposition:

- Truth: $f_\star$
- Estimate: $\hat{f}$
- Best Approximation: $f_n \in \mathcal{F}_{\text{DNN}}$, $\arg\min \|f - f_\star\|_\infty$
- Usual bias-variance:

$$\|\hat{f} - f_\star\|^2_{L_2(X)} \lesssim \underbrace{(\mathbb{E} - \mathbb{E}_n)\left[\ell(\hat{f}, \boldsymbol{z}) - \ell(f_\star, \boldsymbol{z})\right]}_{\text{Empirical Process}} + \underbrace{\mathbb{E}_n\left[\ell(f_n, \boldsymbol{z}) - \ell(f_\star, \boldsymbol{z})\right]}_{\text{Bias}}$$

Key motivation in the proof

- Control the empirical process without bounding the weights

Proof Intuition: Empirical Process

Empirical Process Term

- Employ *localization* techniques, e.g. Bartlett, Bousquet, Mendelson (2005)
- Sharp VC dimension bounds for ReLU networks: Bartlett et al (2017)

Rademacher Complexity

$$R_n\mathcal{F} := \sup_{f\in\mathcal{F}} \frac{1}{n}\sum_{i=1}^{n} \eta_i f(x_i).$$

- Rademacher draws, $\eta_i = \pm 1$
- Intuitively measures how flexible the function class is for predicting random signs
- Have good control of *Empirical Rademacher Complexity* $\mathbb{E}_\eta[R_n\mathcal{F}]$ (via VC dim)

Key Idea: Recursive Improvement

- Suppose we knew $\|\hat{f} - f_\star\|^2_{L_2(X)} \le {r_0}^2$
- Let $\mathcal{G} := \{g = \ell(f, \boldsymbol{z}) - \ell(f_\star, \boldsymbol{z}) : f \in \mathcal{F}_{\text{DNN}}, \|f - f_\star\|^2_{L_2(X)} \le {r_0}^2\}$
- Use this to tighten the bound

Proof Intuition: Empirical Process

Step (i) Symmetrization: w.p. $\geq 1 - 2e^{-\gamma}$

$$(\mathbb{E} - \mathbb{E}_n)\left[\ell(\hat{f}, \boldsymbol{z}) - \ell(f_\star, \boldsymbol{z})\right] \lesssim \mathbb{E}_\eta R_n \mathcal{G} + \sqrt{\frac{r_0{}^2 \gamma}{n}} + \frac{\gamma}{n}$$

Step (ii) Complexity Bound:

$$\mathbb{E}_\eta R_n \mathcal{G} \lesssim r_0 \sqrt{\frac{\mathrm{Pdim}(\mathcal{F}_{\text{DNN}})}{n} \log n} \lesssim r_0 \sqrt{\frac{WL \log(W)}{n} \log n}$$

Step (iii) Improve the Initial Bound:

$$\begin{aligned}
\|\hat{f} - f_\star\|_{L_2(X)}^2 &\lesssim (\mathbb{E} - \mathbb{E}_n)\left[\ell(\hat{f}, \boldsymbol{z}) - \ell(f_\star, \boldsymbol{z})\right] + \mathbb{E}_n\left[\ell(f_n, \boldsymbol{z}) - \ell(f_\star, \boldsymbol{z})\right] \\
&\lesssim r_0 \cdot \left(\sqrt{\frac{WL \log W}{n} \log n} + \sqrt{\frac{\gamma}{n}}\right) + \epsilon_n^2 + \epsilon_n \sqrt{\frac{\gamma}{n}} + \frac{\gamma}{n} \\
&= r_0 o(r_0) \ll r_0{}^2 = \text{initial bound}
\end{aligned}$$

Step (iv) Stopping point:

This trick keeps working if $r_0 > \sqrt{\frac{WL \log W}{n} \log n}$ $\square$

Deep Learning In Economics:

Individual Heterogeneity

The Problem

1. We are interested in some **economic model** with heterogeneous effects:

$$\mathcal{L}\big(\text{data}\ ,\ \beta_i\big)$$

   However:
   - Only with long panels $\hat{\beta}_i \approx \beta_i$
   - Estimates may not conform to theory
   - And what about new individuals?

2. Instead we **balance** structure and flexibility, using covariates:

$$\mathcal{L}\big(\text{data}\ ,\ \beta(\boldsymbol{x}_i)\big)$$

   - Intuitively the same as using interaction terms, but $\beta(\boldsymbol{x})$ is a flexible function

3. Specifically we use **deep neural networks** to capture the heterogeneity:

$$\mathcal{L}\big(\text{data}\ ,\ \widehat{\beta}_{\text{DNN}}(\boldsymbol{x}_i)\big)$$

Framework for Heterogeneity

1. Model $\mathbb{E}[Y \mid \boldsymbol{x}, \boldsymbol{t}] = G\big(\alpha(\boldsymbol{x}) + \boldsymbol{\beta}(\boldsymbol{x})'\boldsymbol{t}\big)$

- Special case of $\mathcal{L}\big(\{y_i, \boldsymbol{t}_i, \boldsymbol{x}_i\}\ ,\ \boldsymbol{\beta}(\boldsymbol{x}_i)\big)$, just for today
- Old history in statistics, a.k.a. varying/functional/smooth coefficient model
- But here our motivation is heterogeneity + structure
  - \+ Interpretable model
  - \+ Economically meaningful
  - \+ Respects economic principles
  - \+ Fully flexible heterogeneity

Compare to:

- Classical parametric model: $\mathbb{E}[Y \mid \boldsymbol{x}, \boldsymbol{t}] = G\big(\alpha + \boldsymbol{\beta}'\boldsymbol{t} + \boldsymbol{\gamma}'\boldsymbol{x}\big)$
  - \+ Interpretable, meaningful
  - – Limited heterogeneity, no flexibility

- Fully nonparametric, naive ML model: $\mathbb{E}[Y \mid \boldsymbol{x}, \boldsymbol{t}] = G\big(u(\boldsymbol{t}, \boldsymbol{x})\big)$
  - \+ Fully heterogeneous and flexible
  - – Mostly uninterpretable, may not make economic sense
  - – Cannot recover second-stage objects

Why Deep Learning?

The **model** imposes economic structure, respects economic theory, ...

$$\mathbb{E}[Y \mid \boldsymbol{x}, \boldsymbol{t}] = G\big(\alpha(\boldsymbol{x}) + \boldsymbol{\beta}(\boldsymbol{x})'\boldsymbol{t}\big)$$

We need an estimator that is "**structurally compatible**"

- The model structure is baked directly into the estimation
- Global restriction, easy to impose

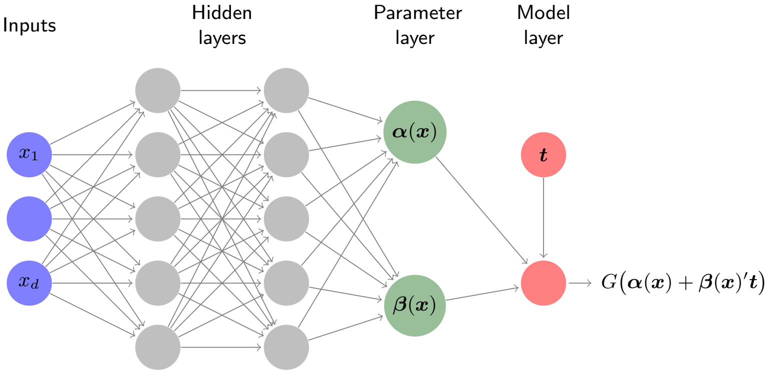
DNNS are an ideal method

- **Both** structurally compatible and capable of tackling modern problems
- Economic structure helps in implementing ML

Why Deep Learning?

Big picture point: implementing ML in economics, why are we ignoring economics?

- Avoid pure prediction: $\hat{y} = \hat{f}(\boldsymbol{x}, t)$, instead learn $\alpha(\boldsymbol{x})$ and $\boldsymbol{\beta}(\boldsymbol{x})$ **jointly**
- Point is not to adapt to structure
- We design our **novel architecture** to model heterogeneity, enforce structure:



Main Results

Key Results

- Only $d = \dim(\boldsymbol{x})$ affects the rate, i.e. dimension of heterogeneity
  - Naive ML approach have slow rate: $\dim(\boldsymbol{x}) + \dim(\boldsymbol{t})$
- Discrete variables handled seamlessly **and** don't affect rate
  - Former uses the same architecture idea, and is more novel; latter is standard

Main Rate Theorem

- Building on FLM1
- Say $\alpha(\cdot)$, $\beta(\cdot)$ are p-smooth in the d_c continuous covariates
- Set width $H_n \asymp n^{\frac{d_c}{2(p+d_c)}} \log^2 n$ and depth $L_n \asymp \log n$

$$\boxed{\mathbb{E}_n\left[(\widehat{\beta}_{\text{DNN}} - \beta)^2\right] \;=\; O_{\mathbb{P}}\left(n^{-\frac{p}{p+d_c}} \log^8 n\right)}$$

Framework for Heterogeneity

1. Model $\mathbb{E}[Y \mid \boldsymbol{x}, \boldsymbol{t}] = G\big(\alpha(\boldsymbol{x}) + \boldsymbol{\beta}(\boldsymbol{x})'\boldsymbol{t}\big)$

2. Inference Target $\theta_0 = \mathbb{E}\big[H(\boldsymbol{X}, \alpha(\boldsymbol{X}), \boldsymbol{\beta}(\boldsymbol{X}), \boldsymbol{t}^*)\big]$

Researcher chooses H function:

- Vectors are allowed
- So are implicit functions
- Not possible without structure
- Only require you to compute $\nabla H = (\partial H/\partial\alpha,\ \partial H/\partial\boldsymbol{\beta})'$

1+2 simultaneously cover many economically interesting contexts:

- Recovering existing results: ATE, partially linear model, ...
- Delivering **new** applications: choice models, average partial effects, continuous/multiple treatments, production functions, count data, Berry logit, ...
- Idea extends to IV, multinomial choice, systems, ...

Framework for Heterogeneity

1. Model $\mathbb{E}[Y \mid \boldsymbol{x}, \boldsymbol{t}] = G\big(\alpha(\boldsymbol{x}) + \boldsymbol{\beta}(\boldsymbol{x})'\boldsymbol{t}\big)$

2. Inference Target $\theta_0 = \mathbb{E}\big[H(\boldsymbol{X}, \alpha(\boldsymbol{X}), \boldsymbol{\beta}(\boldsymbol{X}), \boldsymbol{t}^*)\big]$

Main **theoretical result** of this section:

- **Single** influence function calculation, $\psi - \theta_0$, (our model structure + Newey 1994)

$$\psi = H(\alpha, \boldsymbol{\beta}, \boldsymbol{x}) \; + \; \nabla H(\alpha, \boldsymbol{\beta}, \boldsymbol{x})' \boldsymbol{\Lambda}(\boldsymbol{x})^{-1}(1, \boldsymbol{t})' \Big(y - G\big(\alpha(\boldsymbol{x}) + \boldsymbol{\beta}(\boldsymbol{x})'\boldsymbol{t}\big)\Big)$$

- One crucial piece is $\boldsymbol{\Lambda}(\boldsymbol{x}) = \mathrm{var}[\dot{G}^{1/2}\boldsymbol{T}|\boldsymbol{x}]$, $\dot{G} = \partial G(u)/\partial u$
  - Intuition: generalization of the inverse propensity part of the IF
  - Implementation: Estimate in observational data, but **compute** in experiments
  - For semiparametric inference in general: small denominators are a bear
- Inference then follows standard methods, e.g. cross-fitting (Chernozhukov et al 2018)

Seeing it All in Action:

What is Advertising Worth?

How does advertising content affect purchasing?

Bertrand, Karlan, Mullainathan, Shafir, Zinman (QJE, 2010)

- Advertising for shortish-term loans in South Africa
- $Y = \{0, 1\}$ = Applied for a loan
- $\boldsymbol{T}$ = 12 ad characteristics, **randomly** assigned (based on $\boldsymbol{x}$)
  - Interest rate: directly compute value
  - Other qualities (photos, tables, uses) can be valued
- $\boldsymbol{X}$ = 11 individual characteristics, some discrete
- Original questions: does advertising content matter? How much?
- Original results: marginal effects from probit

Applying the Framework

1. Structural model: $\mathbb{E}[Y \mid \boldsymbol{x}, \boldsymbol{t}] = \text{logit}\big(\alpha(\boldsymbol{x}) + \boldsymbol{\beta}(\boldsymbol{x})'\boldsymbol{t}\big)$
2. Average marginal effects:
$$\theta_0 = \mathbb{E}\left[\frac{\partial \mathbb{E}[Y|\boldsymbol{X}, t^*]}{\partial t}\right] = \mathbb{E}\big[\underbrace{\beta(\boldsymbol{X})G(1-G)}_{H(\alpha,\beta)}\big]$$
3. Estimate $\widehat{\alpha}_{\text{DNN}}(\boldsymbol{x}_i)$, $\widehat{\boldsymbol{\beta}}_{\text{DNN}}(\boldsymbol{x}_i)$
4. Inference via IF
   - Compute $\boldsymbol{\Lambda}(\boldsymbol{x})$: nontrivial, but computable

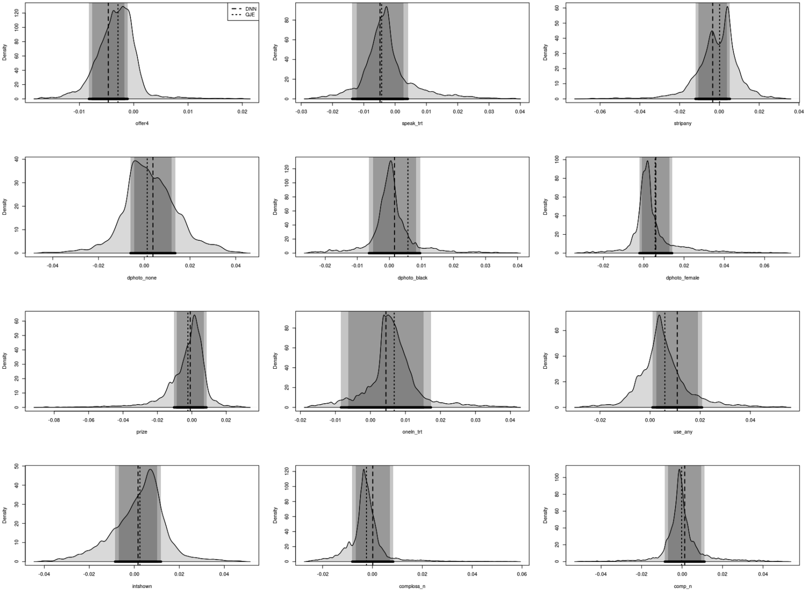
What couldn't we do before?

- Structural model allows for computation of marginal effects
- Deep learning means 11-dimensional nonparametrics feasible
  - **13!** nonparametric functions: $\alpha(\boldsymbol{x}), \boldsymbol{\beta}(\boldsymbol{x})$
- Standard errors rely on novel IF

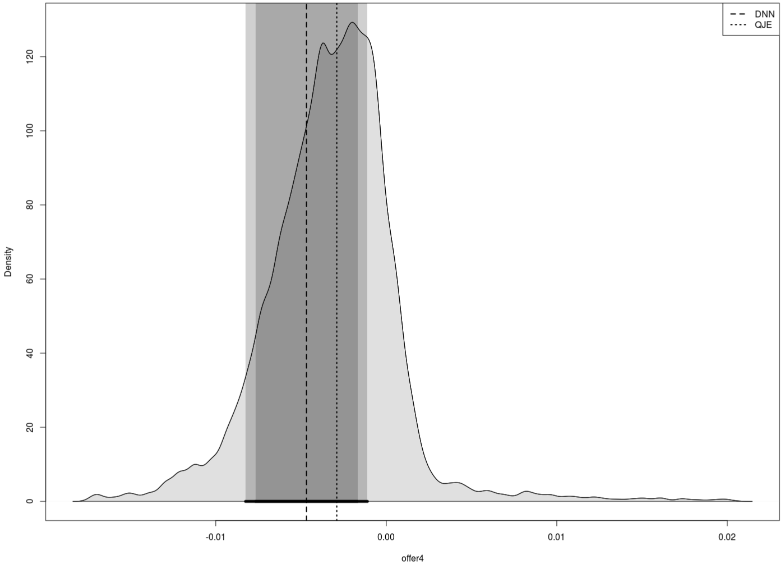
Results Table

Variable	QJE	DNN ME	95% CI		$\Pr(\beta(x) > 0)$	Coef. of Variation
Interest rate offer	-0.0029	-0.0047	-0.0083	-0.0011	0.1337	1.0211
We speak your language	-0.0043	-0.0048	-0.0137	0.0041	0.2533	2.0542
Special rate for you	0.0001	-0.0034	-0.0120	0.0053	0.5001	4.4506
No photo	0.0013	0.0038	-0.0060	0.0136	0.5723	3.4931
Black photo	0.0058	0.0016	-0.0064	0.0096	0.5402	5.1348
Female photo	0.0057	0.0060	-0.0021	0.0141	0.6820	2.3375
Cell phone raffle	-0.0023	-0.0009	-0.0104	0.0085	0.4812	17.0059
Example loan shown	0.0068	0.0044	-0.0084	0.0173	0.8631	1.9379
No loan use mentioned	0.0059	0.0108	0.0009	0.0207	0.7499	1.0936
Interest rate shown	0.0025	0.0017	-0.0085	0.0119	0.6289	7.9903
Loss comparison	-0.0024	0.0001	-0.0081	0.0083	0.2342	89.3606
Competitors rate shown	-0.0002	0.0013	-0.0085	0.0111	0.4107	9.6790

Estimates


DNN
QJE
Density
offer4
speak_trt
stripany
dphoto_none
dphoto_black
dphoto_female
prize
oneln_trt
use_any
intshown
comploss_n
comp_n

Offer Rate Coefficient



Beyond Marginal Effects: Optimal Offers

Some Real Economics

- Assume the firm wishes to maximize profits:

$$\pi_i = \max_{r=\text{rate}} L\left[rG\left(r\right)\right]\left[1 - D\left(r\right)\right]$$

  - $L =$ expected dollar loan amount, normalize to 1 (doesn't impact the rate)
  - $r =$ the interest rate offered
  - $G(r) =$ the probability of acceptance (depends on $\alpha(\boldsymbol{x}_i)$, $\boldsymbol{\beta}(\boldsymbol{x}_i)$)
  - $\left[1 - D\left(r\right)\right] =$ probability of non-default on the loan
(calibrated to match the results in the QJE using a logit kernel)

- Then it is straightforward to show that

$$\frac{\partial \pi}{\partial r} = \left(r\dot{G}\left(r\right)\beta_r + G\left(r\right)\right)\left[1 - D\left(r\right)\right] - rG\left(r\right)\dot{D}\left(r\right)\delta = 0$$

Beyond Marginal Effects: Optimal Offers

- Simplifying:

$$\frac{\partial \pi}{\partial r} = \left(r\left(1 - G\left(r\right)\right)\beta_r + 1\right)\left[1 - D\left(r\right)\right] - r\dot{D}\left(r\right)\delta = 0$$

- And using the logit structure:

$$r = \frac{1 + r\left(1 - G\left(r\right)\right)\beta_r}{D\left(r\right)\delta}$$

- There will a unique fixed point since the denominator of the RHS is decreasing in r for $\beta_r < 0$ and $\delta > 0$
- Therefore:

$$r^* = \frac{1 + r^*\left(1 - G\left(r^*\right)\right)\beta_r}{D\left(r^*\right)\delta}$$

Applying the Framework

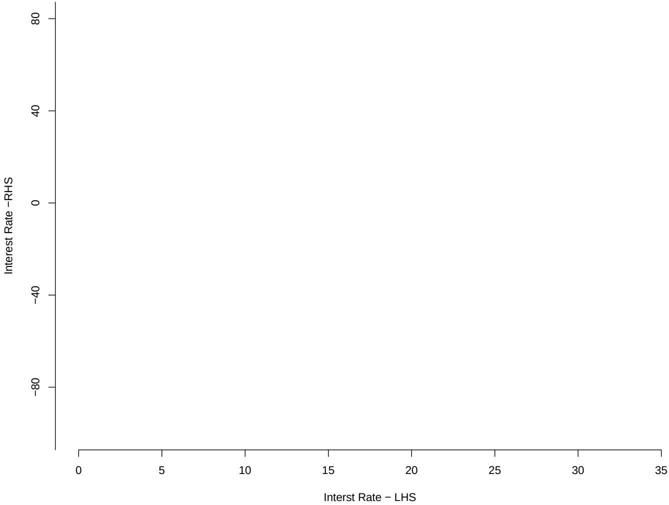
- Even if we don't have it in **closed form**, r^* is a smooth function of $\alpha(\boldsymbol{x}), \boldsymbol{\beta}(\boldsymbol{x})$

$\Rightarrow$ We can do inference on $\theta_0 = \mathbb{E}[H(\alpha, \boldsymbol{\beta}, r^*)]$

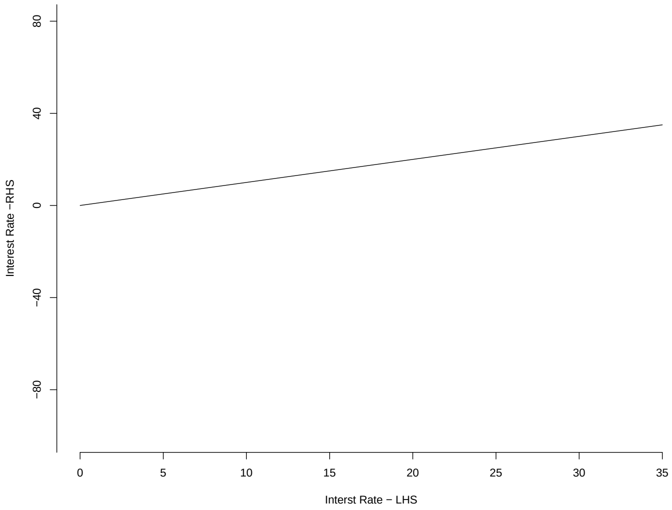
  - derivatives can be calculated via implicit differentiation or numerically: $\dfrac{\partial H}{\partial r^*}\dfrac{\partial r^*}{\partial \beta}$

- Impossible without our results and without exploiting heterogeneity

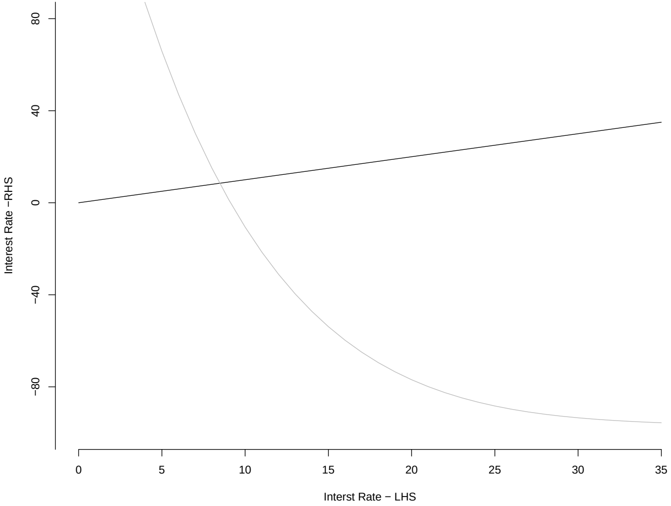
Optimal Offers



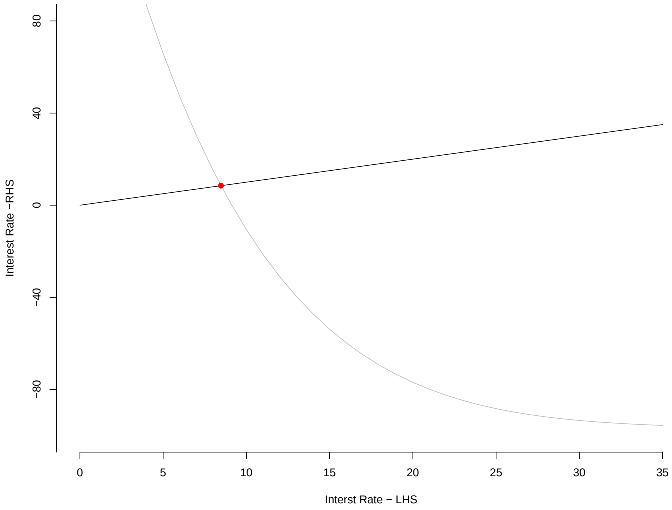
Optimal Offers



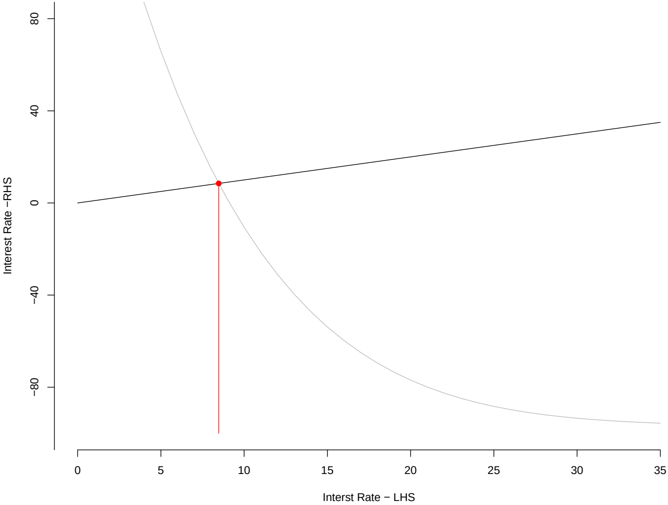
Optimal Offers



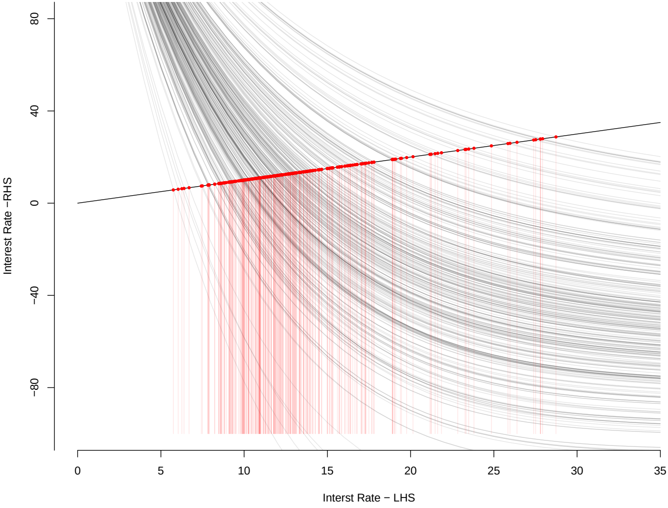
Optimal Offers



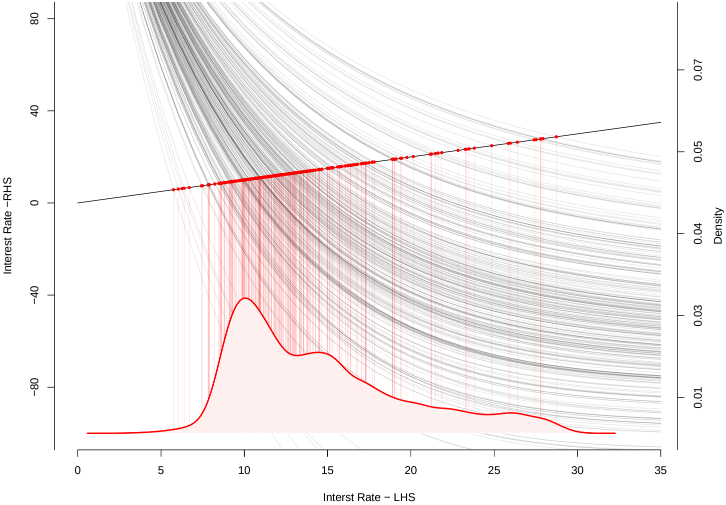
Optimal Offers



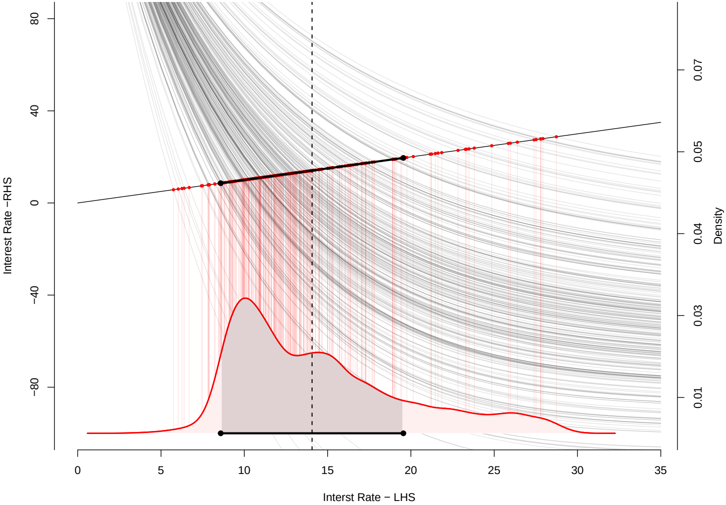
Optimal Offers



Optimal Offers



Optimal Offers



Wrapping Up

Wrapping Up

Deep Learning can have a place in the economics toolbox
Our results are practicable & broadly applicable

Contributions to deep learning itself

- Nonasymptotic bounds for general regression problems
- Matching common practice
  - Fully connected network, ReLU activation, unbounded weights
- Shows good empirical performance

Contribution to economics $\cap$ ML: structured models with heterogeneity

- Wide ranging methodology
- Novel architecture for structured models
- Inference via novel IF
- Big picture point: implementing ML in economics, don't ignore economics

Long way to go

- Computational issues: optimization, tuning, easy to use tools, ...
- Theoretical issues: optimal architectures, depth vs. width, different data types, ...

Thanks!