Counterfactuals via Deep IV

Matt Taddy (Chicago + MSR) Greg Lewis (MSR)
Jason Hartford (UBC) Kevin Leyton-Brown (UBC)

Endogenous Errors

$$y = g(p, \boldsymbol{x}) + e \text{ and } \mathbb{E}[\, p\, e\,] \neq 0$$

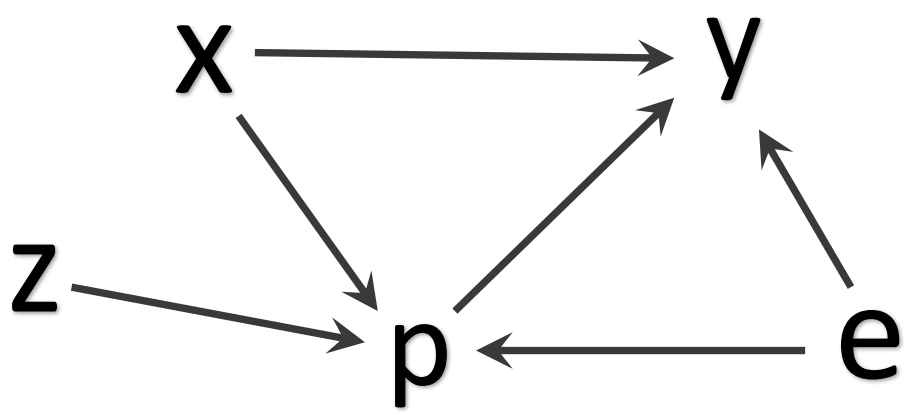
If you estimate this using naïve ML, you'll get

$$E[y|p, \boldsymbol{x}] = E_{e|p}[g(p, \boldsymbol{x}) + e] = g(p, \boldsymbol{x}) + E[e|p, \boldsymbol{x}]$$

This works for prediction. It doesn't work for counterfactual inference:

What happens if I change p independent of e?

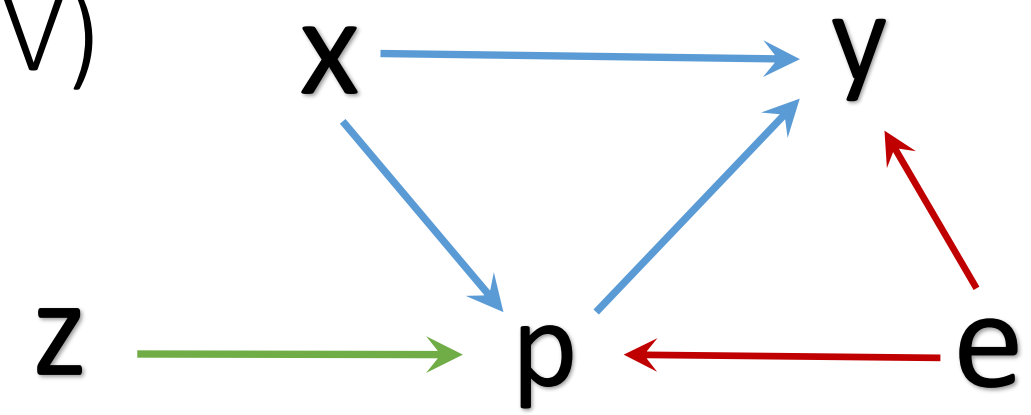
Instrumental Variables (IV)



In IV we have a special $z \perp e$ that influences policy p but not response y.

- Supplier costs that move price independent of demand (e.g., fish, oil)
- Any source of treatment randomization (intent to treat, AB tests, lottery)

Instrumental Variables (IV)



The *exclusion structure* implies

$$\mathbb{E}[y|x,z] = \int g(p,x)dF(p|x,z)$$

You can observe and estimate $\widehat{\mathbb{E}}[y|x,z]$ and $\hat{F}(p|x,z)$

$\Rightarrow$ to solve for *structural* $g(p,x)$ we have an inverse problem.

cf Newey+Powell 2003

$$\min_{g \in G} \sum \left(y_i - \int g(p, x_i) dF(p|x_i, z_i) \right)^2$$

2SLS: $p = \beta z + \nu$ and $g(p) = \tau p$ so that $\int g(p) dF(p|z) = \tau \mathbb{E}[p|z]$

So you first regress p on z then regress y on $\hat{p}$ to recover $\hat{\tau}$.

$$\min_{g \in G} \sum \left(y_i - \int g(p, x_i) dF(p|x_i, z_i) \right)^2$$

Or nonparametric sieves where $g(p, x_i) \approx \sum_k \gamma_k \varphi_k(p, x_i)$ and

$$\mathbb{E}_F[\varphi_k(p, x_i)] \approx \sum_j \alpha_{kj} \beta_j(x_i, z_i)$$ (Newey+Powell)

or

$$\mathbb{E}_F[\, y_i - \sum_k \gamma_k \varphi_k(p, x_i)\,] \approx \sum_j \alpha_j \beta_j(x_i, z_i)$$ (BCK, Chen+Pouzo)

Also Darolles et al (2011) and Hall+Horowitz (2005) for kernel methods.

But this requires careful crafting and will not scale with $\dim(x)$

$$\min_{g \in G} \sum \left(y_i - \int g(p, x_i) dF(p|x_i, z_i) \right)^2$$

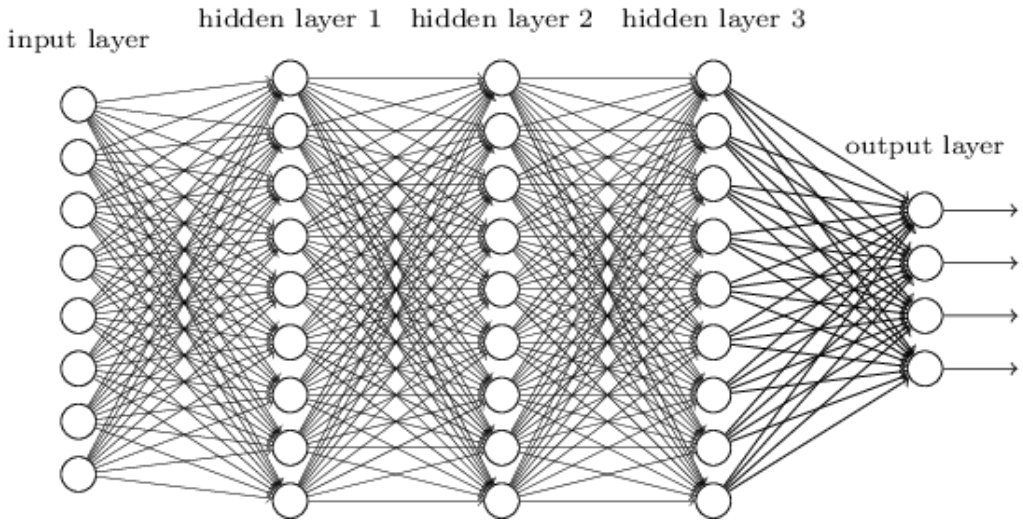
Instead, we propose to target the integral loss function directly

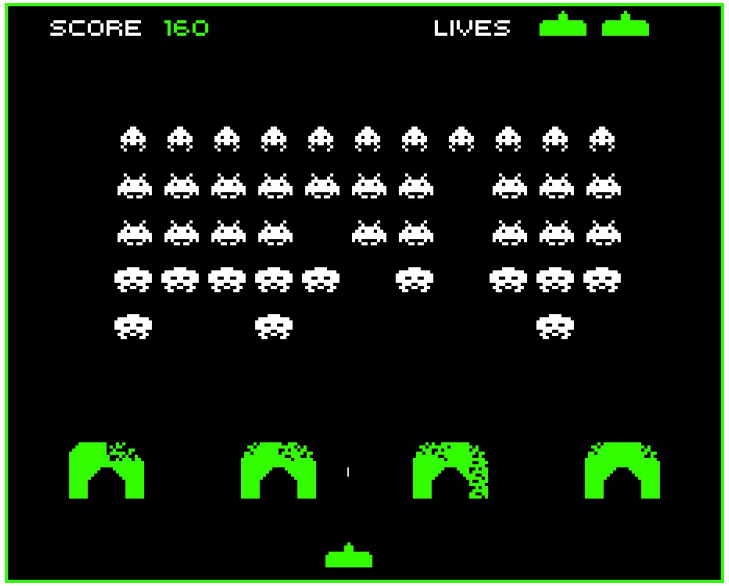
For discrete (or discretized) treatment

- Fit distributions $\hat{\mathrm{F}}(p|x_i, z_i)$ with probability masses $\hat{f}(p_b|x_i, z_i)$
- Train $\hat{g}$ to minimize $\left[\mathrm{y_i} - \sum_b g(\hat{p}_b, x_i) \hat{f}(p_b|x_i, z_i) \right]^2$

And you've turned IV into two *generic* machine learning tasks

Learning to love Deep Nets





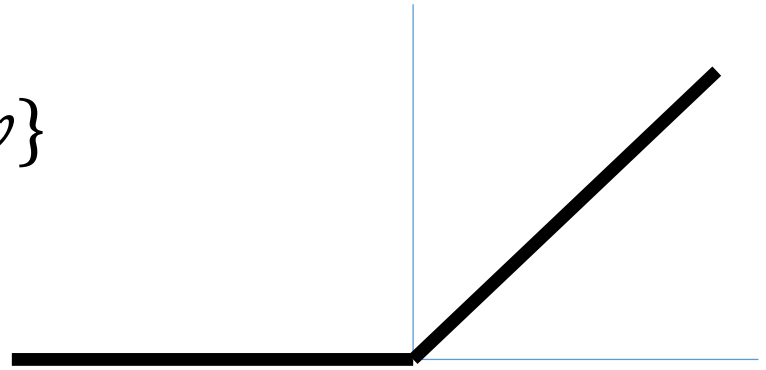
What is a deep net?

$$\hat{y}_i = \sum_k h_k^L\left(a_{ik}^L\right), \qquad a_{ik}^L = \boldsymbol{z}_{ik}^{L\prime} W^L, \qquad \boldsymbol{z}_{ik}^L = \sum_j h_k^{L-1}\left(a_{ik}^{L-1}\right), \ldots$$

And so-on until you get down to the input layer $\boldsymbol{a}_i = \boldsymbol{h}^0(\boldsymbol{x}_i')$

Many different variations here: recursive, convolutional, …

Apart from the bottom, usually $h(v) = \max\{0, v\}$



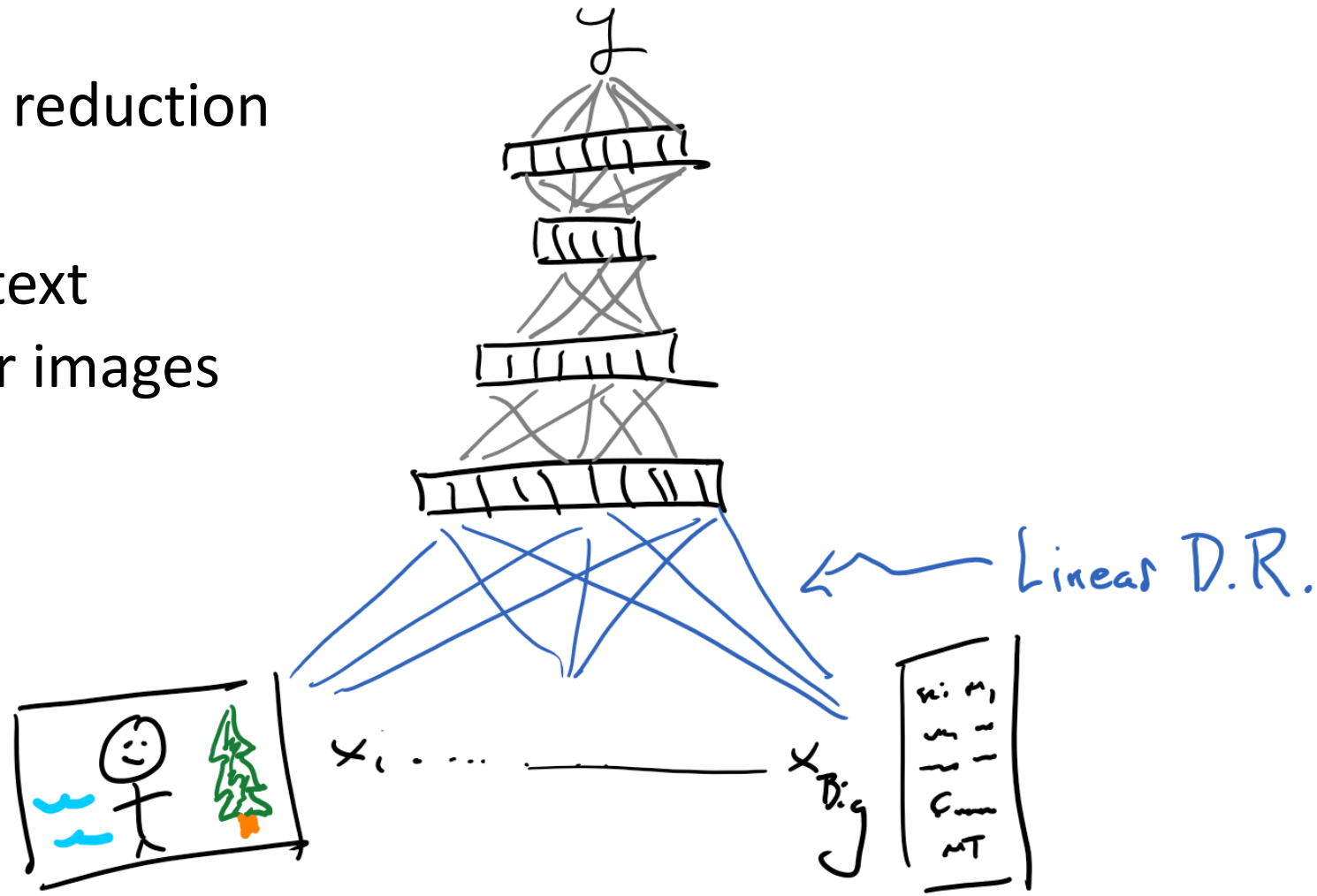
Deep nets are not really sieves

1st layer is a big dimension reduction

e.g.,

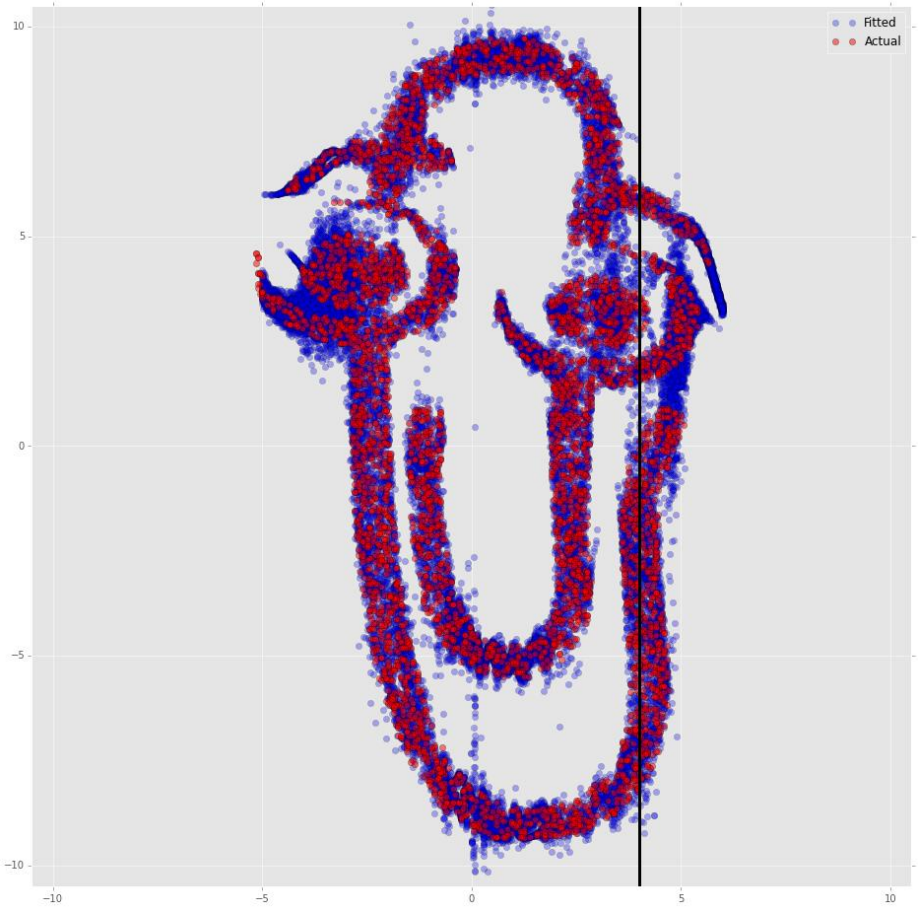
- word embedding for text
- matrix convolution for images

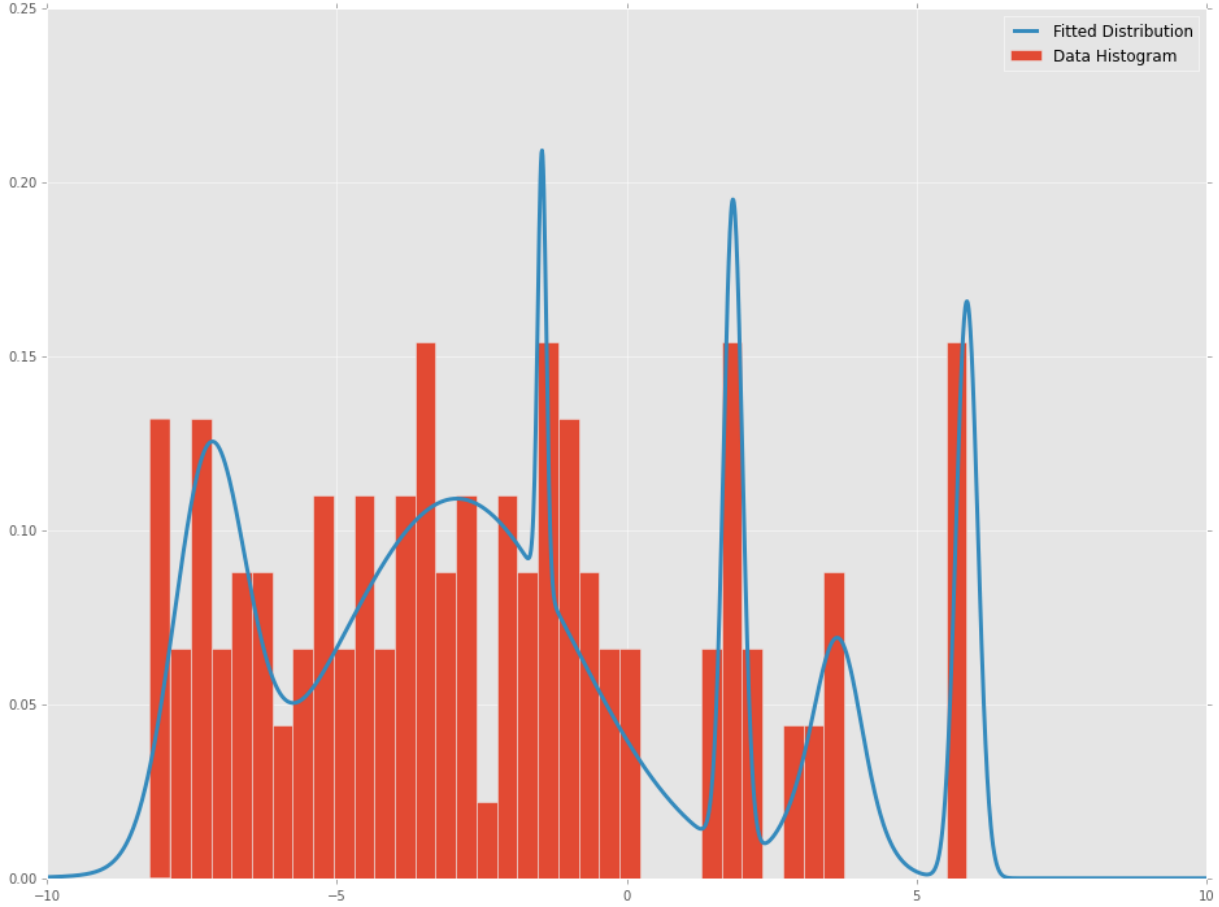
We need to study these…



e.g., first-stage learning for $\mathrm{F}(p|x_i, z_i)$

Bishop 96: Final layer of network parametrizes a mixture of Gaussians





Stage 2: Integral Loss

The second stage involves an integral loss function

If p is not discrete or can take many values, not easy!

Brute force just samples from $\hat{\mathrm{F}}(p|x_i, z_i)$ and you take gradients on

$$\frac{1}{N}\sum_i \left(y_i - \frac{1}{B}\sum_b g(p_b, x_i; \theta)\right)^2, \quad p_b \sim \hat{\mathrm{F}}(p|x_i, z_i)$$

This is what economists usually do, but this is super inefficient

Stochastic Gradient Descent

You have loss $L(\boldsymbol{D},\ \theta)$ where $\boldsymbol{D} = [\boldsymbol{d}_1\ \ldots \boldsymbol{d}_N]$

In the usual GD, you iteratively descend

$$\theta_t = \theta_{t-1} - \boldsymbol{C}_t \nabla L(\boldsymbol{D}, \theta_{t-1})$$

In SGD, you instead follow *noisy* but *unbiased* sample gradients

$$\theta_t = \theta_{t-1} - \boldsymbol{C}_t \nabla L(\{\boldsymbol{d}_{t_b}\}_{b=1}^{B}, \theta_{t-1})$$

SGD for integral loss functions

Our one-observation stochastic gradient is

$$\nabla L(d_i, \theta) = -2\left(y_i - \int g_\theta(p, x_i) d\hat{F}(p|x_i, z_i)\right) \int g_\theta{}'(p, x_i) d\hat{F}(p|x_i, z_i)$$

Do SGD by pairing each observation with *two independent* treatment draws

$$\nabla \hat{L}(d_i, \theta) = -2(y_i - g_\theta(\dot{p}, x_i))\, g'_\theta(\ddot{p}, x_i), \quad \dot{p}, \ddot{p} \sim \hat{\mathrm{F}}(p|x_i, z_i)$$

So long as the draws are independent, $\mathbb{E}\nabla \hat{L}(d_i, \theta) = \mathbb{E}\nabla L(d_i, \theta) = L(\boldsymbol{D}, \theta)$

Aside: we can use SGD more in econ …

There are a ton of setups where we use simulation to solve

$$\min_{\beta} \sum \left(y_i - \int g(x_i;\theta) d\mathrm{P}(\theta|\beta) \right)^2$$

Random coefficient models, simulate ML or simulate MM…

Monte Carlo SGD is a perfect fit here

Validation and model tuning

We can do *causal validation* via two OOS loss functions

Leave-out deviance on first stage

$$\sum_{i \in LO} -\log \hat{f}(p|x_i, z_i)$$

Leave-out loss on second stage (constrained fit of $\mathbb{E}[y|xz]$)

$$\sum_{i \in LO} \left(y_i - \int g_\theta(p, x_i) d\hat{F}(p|x_i, z_i)\right)^2$$

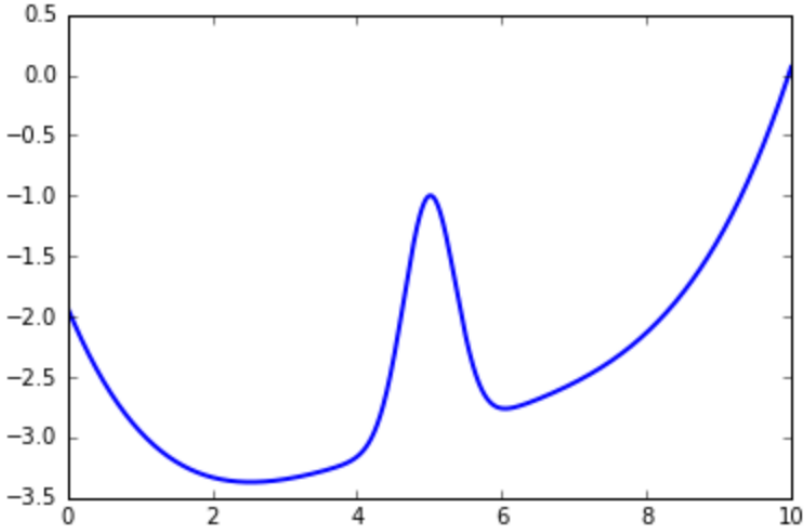
You want to minimize both of these (in order).

heterogeneous price effects

$$y = 100 + s\psi_t + (\psi_t - 2)p + e,$$

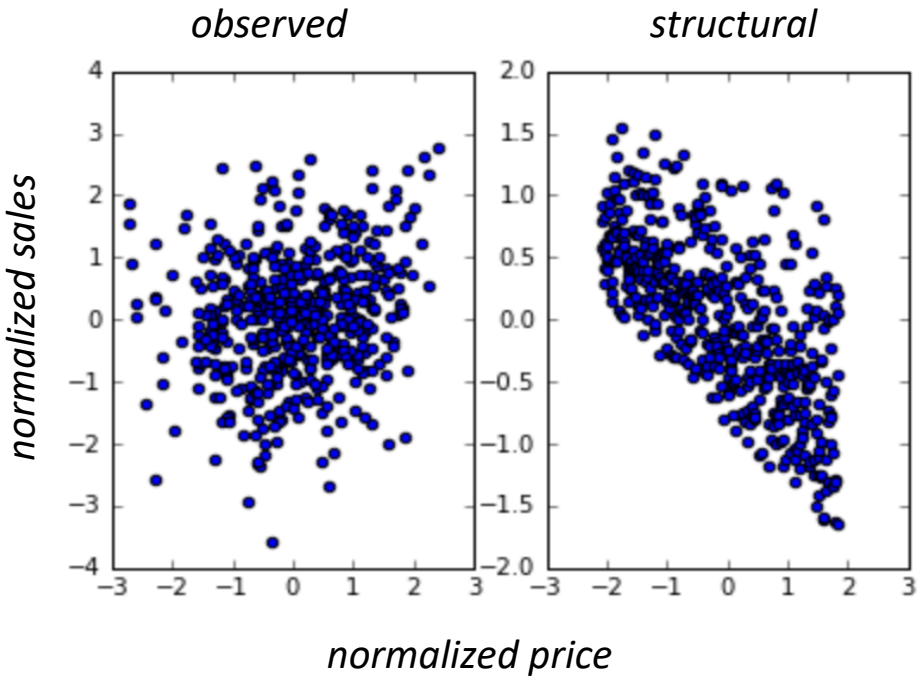
$$p = 25 + (z+3)\psi_t + v$$

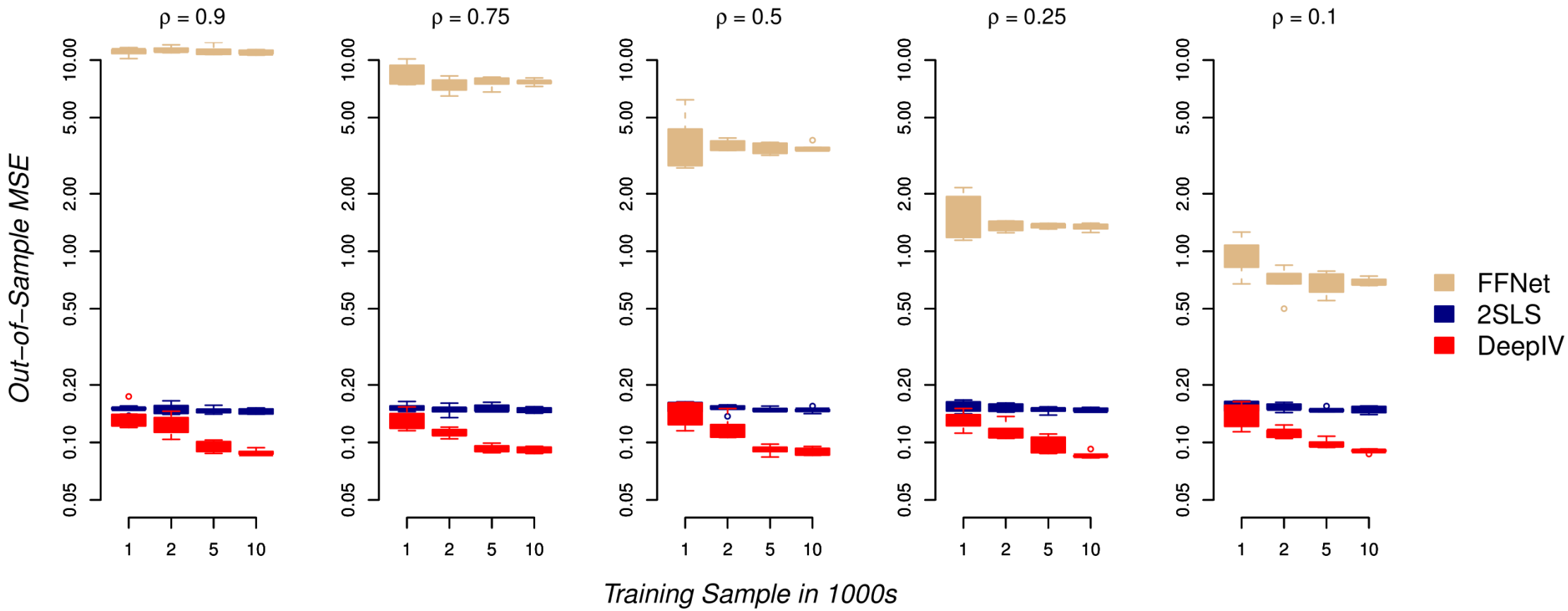
$$z,\ v \sim \mathrm{N}(0,1) \ \text{ and } \ e \sim \mathrm{N}(\rho v, 1-\rho^2),$$



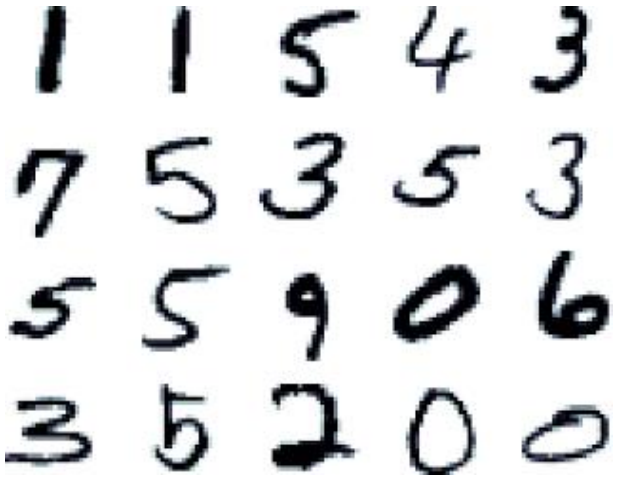
‘time’ dependent prices, sensitivity, utility

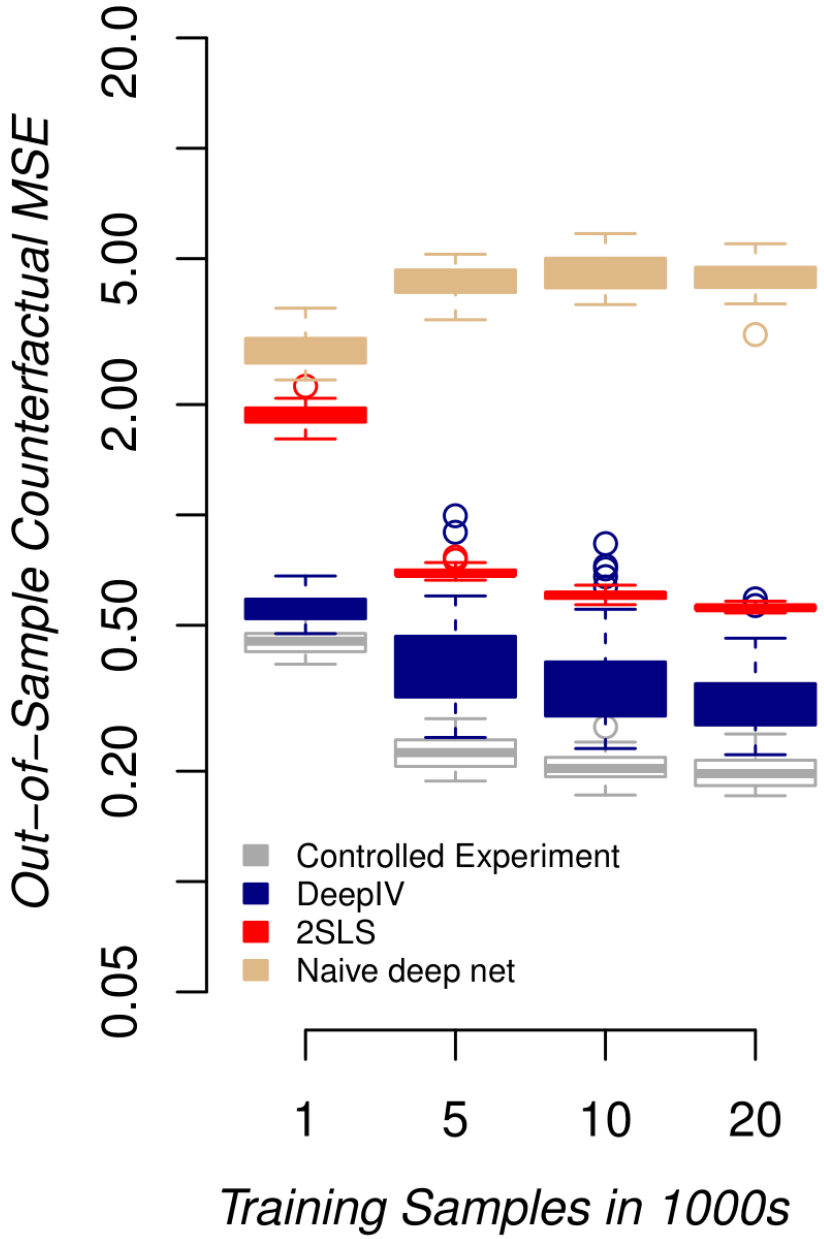
Customer ‘type’ 1-7 impacts demand




ρ = 0.9
ρ = 0.75
ρ = 0.5
ρ = 0.25
ρ = 0.1
Out−of−Sample MSE
0.05
0.10
0.20
0.50
1.00
2.00
5.00
10.00
1
2
5
10
Training Sample in 1000s
FFNet
2SLS
DeepIV

Make it harder…





Inference? Good question

Data split! Get top node values and averages on left-out data:

$$\eta_{ik} = \eta_k(x_i, p_i) \quad \text{and} \quad \bar{\eta}_{ik} = \mathbb{E}_{\hat{F}(p|x_i,z_i)}\eta_k(x_i, p)$$

Stack as instruments $\bar{\mathrm{H}} = [\bar{\eta}_1 \cdots \bar{\eta}_L]'$ and treatments $\mathrm{H} = [\eta_1 \cdots \eta_L]'$

Post-net 2SLS coefficients are $\hat{\beta} = (\bar{H}'H)^{-1}\bar{H}'y$ with variance V_β and

$$\mathrm{var}[\hat{g}(x, p)] = \boldsymbol{\eta}'(x, p)\, V_\beta\, \boldsymbol{\eta}(x, p)$$

Inference? Good question

Variational Bayes: fit q to minimize $\mathbb{E}_{\mathrm{q}}[\log q(W) - \log p(W|D]$

Diversion… in training we use dropout:

At each SGD update, calculate gradients against $W^l = \Xi^l \Omega^l$ at layer l where

$$\Xi^l = \mathrm{diag}\big(\, \xi_{l1} \;\; \ldots \xi_{lK_l}\big), \qquad \xi_{kj} \sim \mathrm{Bern}(c)$$

i.e., dropout randomly drops *rows* of each layer's weight matrix

Dropout is Variational Bayes!

VB minimizes $KL(q) \propto \mathbb{E}_{\mathrm{q}}[\log q(W) - \log p(\boldsymbol{D}|W) \ - \log p(W) \ ]$

If $q(W) = \prod_l \prod_k (c\mathbb{1}_{\left[W_k^l = \Omega_k^l\right]} + (1-c)\,\mathbb{1}_{\left[W_k^l = 0\right]})$ and $w \sim \mathrm{N}(0, \lambda^{-1})$,

$$KL(q) \propto \mathbb{E}_q L(\boldsymbol{D}|W) + c\lambda|W|_2 + \mathrm{K}\,[c\log c + (1-c)\log(1-c)]$$

(see also Gal and Ghahramani)

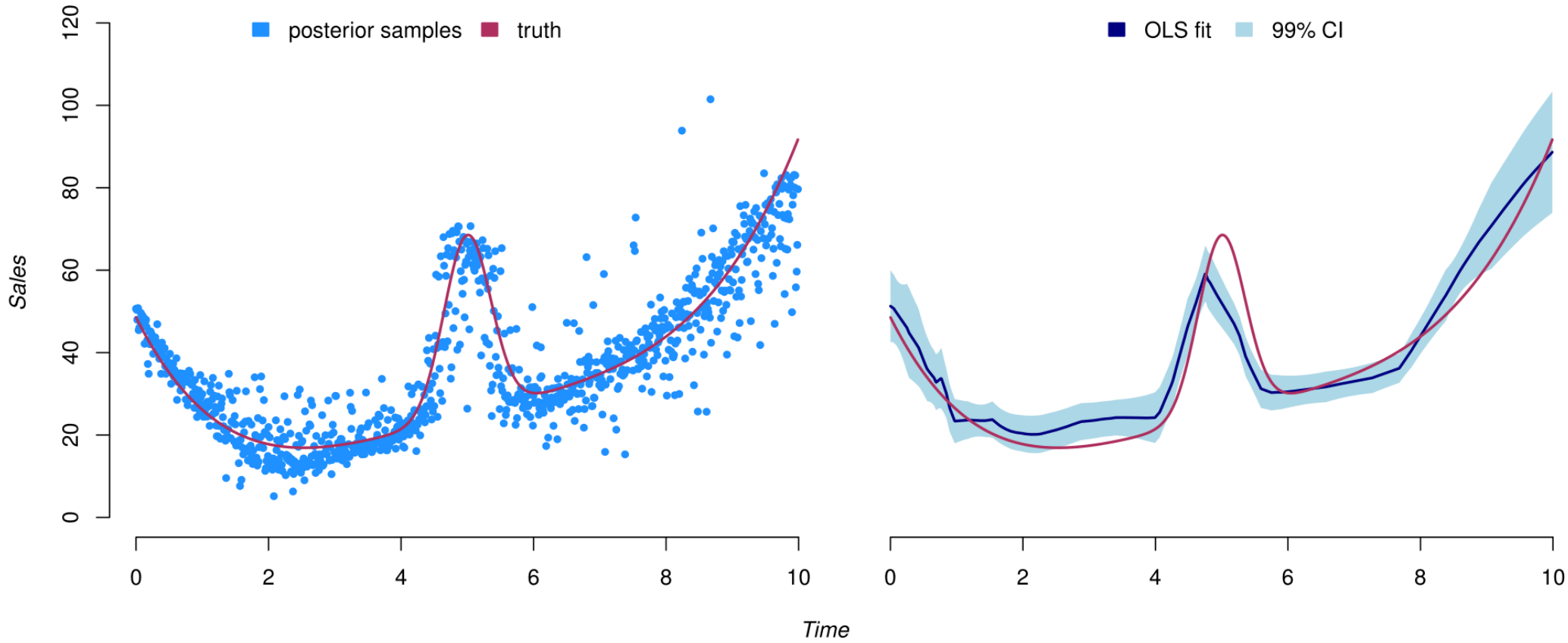
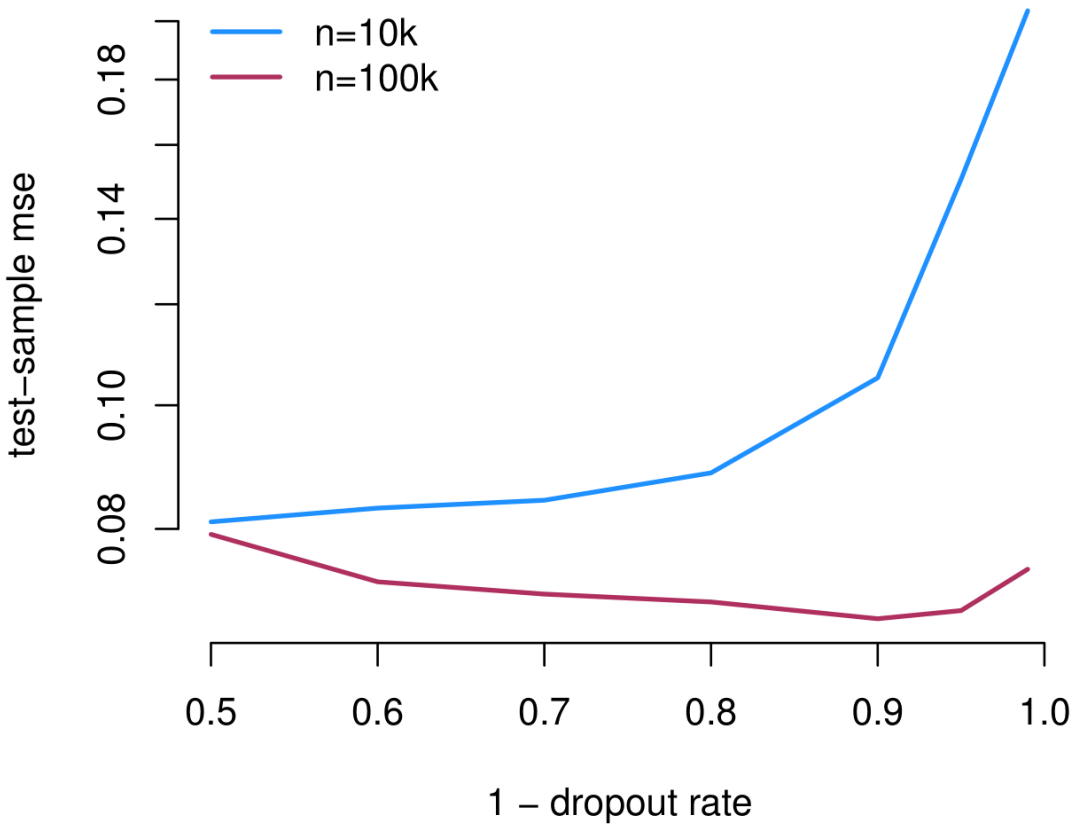
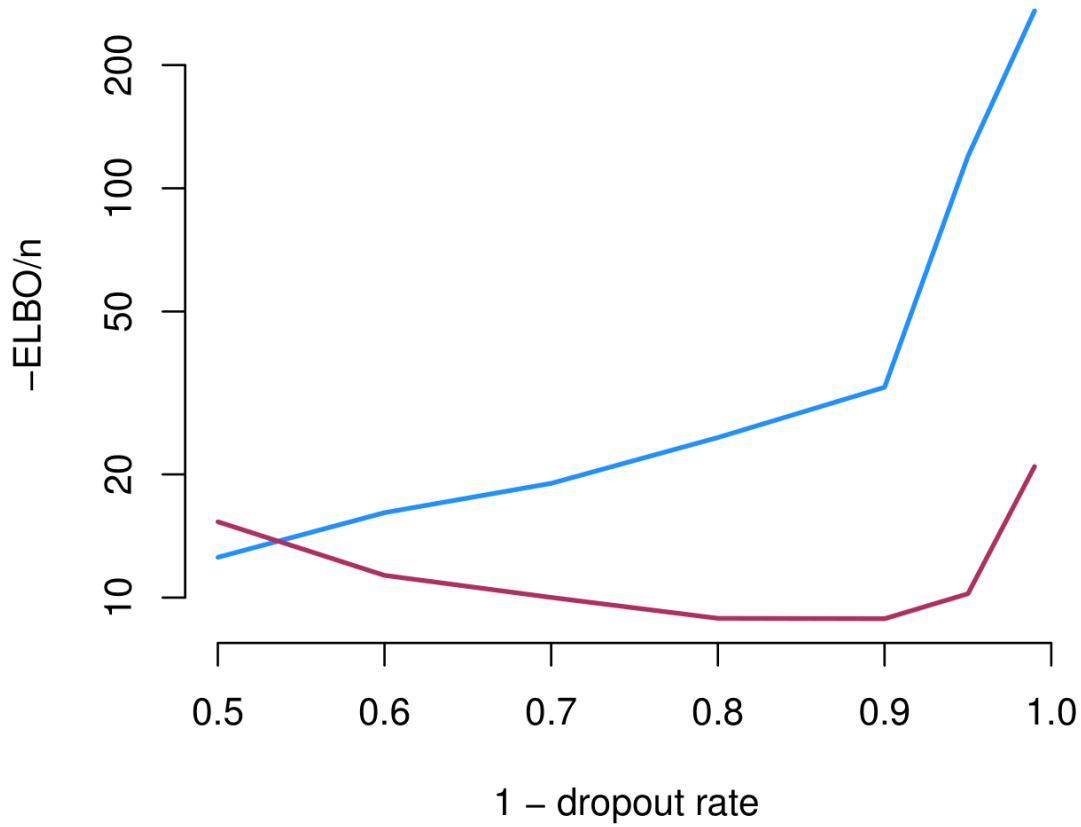


Figure 3: Bayesian (left) and Frequentist (right) inference for a central slice of the counterfactual function, taken at the average price and in our 4^{th} customer category. Since the price effect for a given customer at a specific time is constant in (27), the curves here are a rescaling of the customer *price sensitivity* function.

Tuning the dropout rate is like treating it as a variational parameter





Ads Application

Taken from Goldman and Rao (2014)

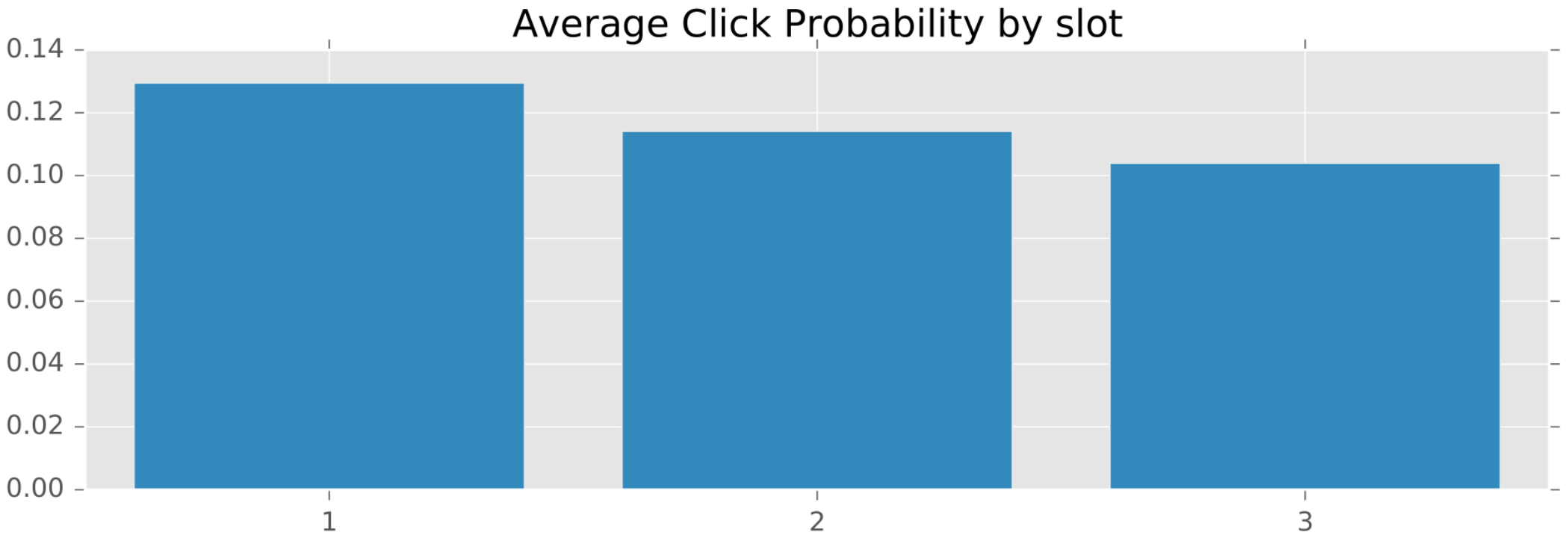
We have 74 mil click-rates over 4 hour increments for 10k search terms

Treatment: ad position 1-3

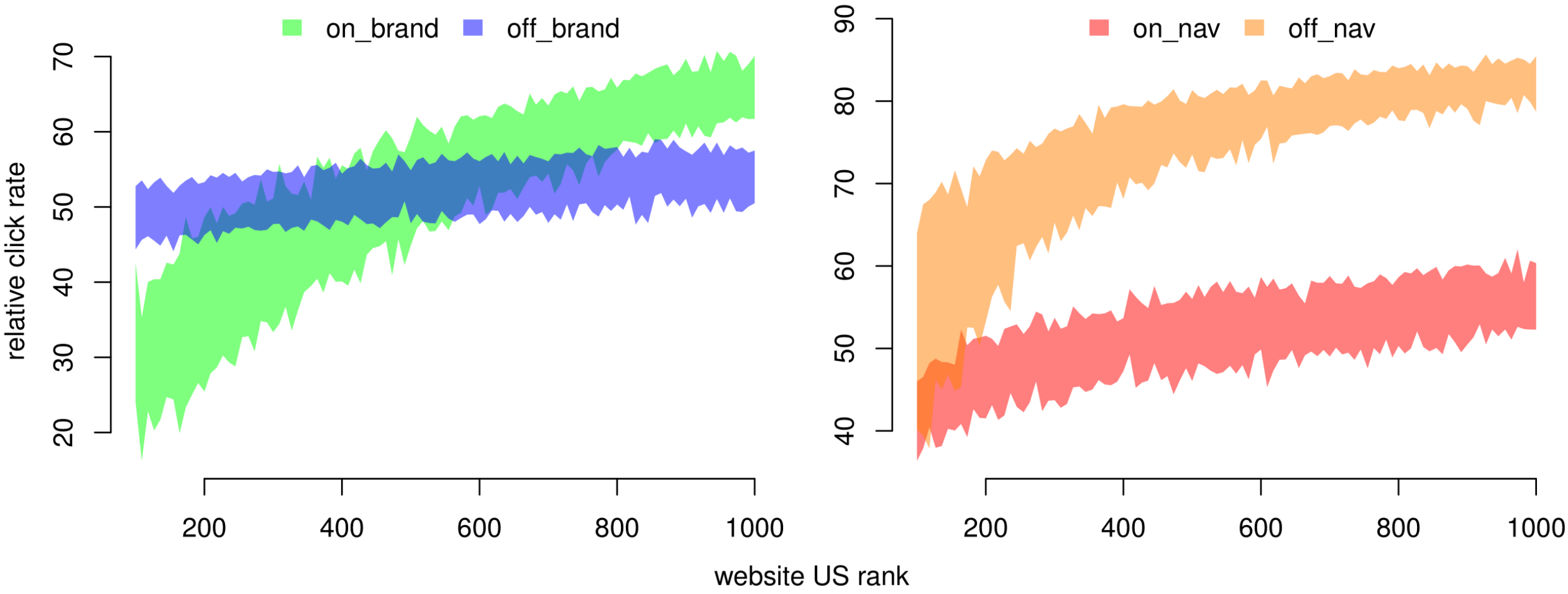
Instrument: background AB testing (bench of ~ 100 tests)

Covariates: advertiser id and ad properties, search text, time period

Average Treatment Effects



These compare to observed click probabilities of 0.33, 0.1, and 0.05.



Heterogeneity across advertiser and search

Automated Learning and Intelligence for Causation and Economics

We use economic theory to build systems of tasks that can be addressed with deep nets and other state-of-the-art ML.

This is the construction of systems for *Economic* AI