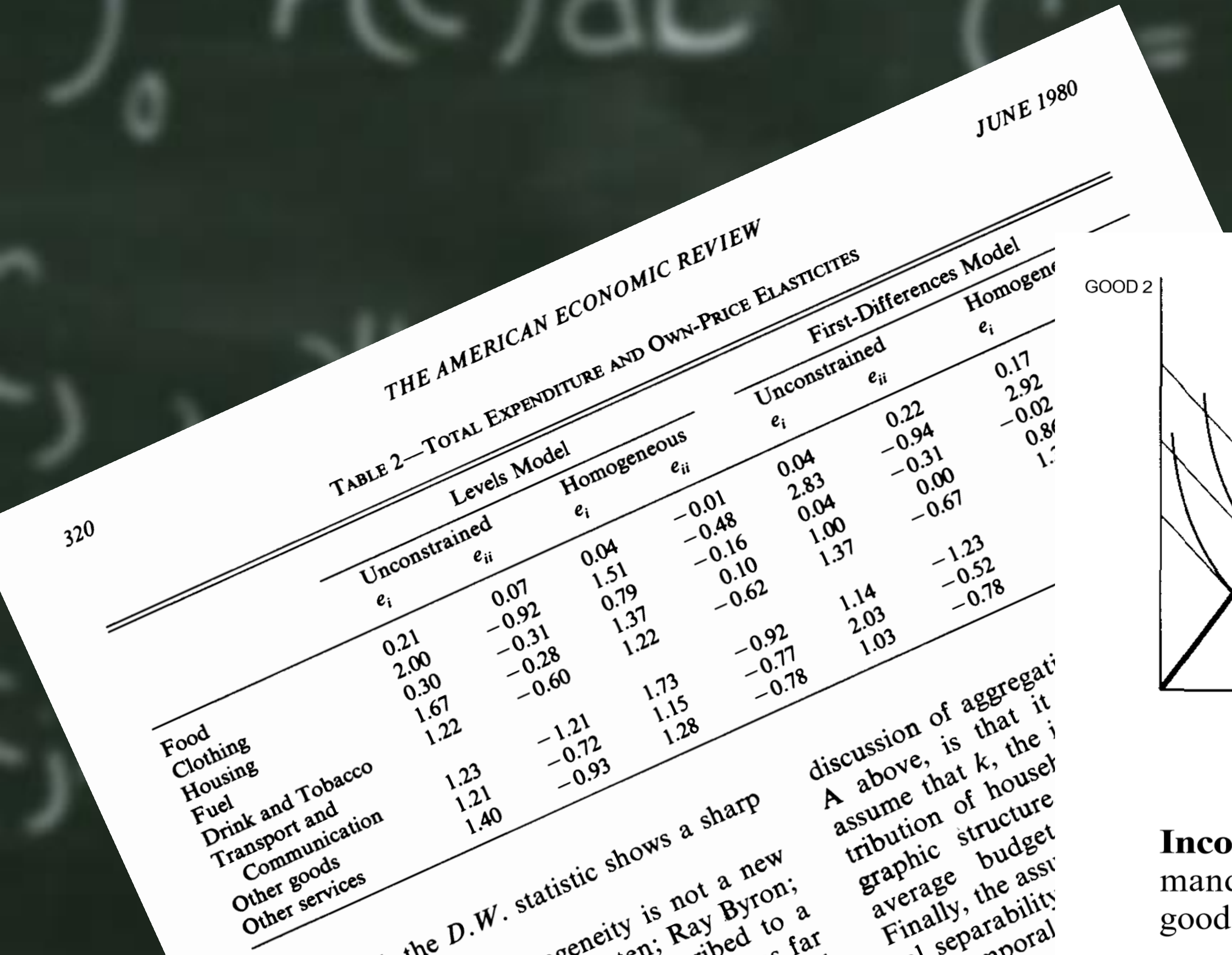
Economic AI

Matt Taddy

Microsoft and Chicago

What do economists do?



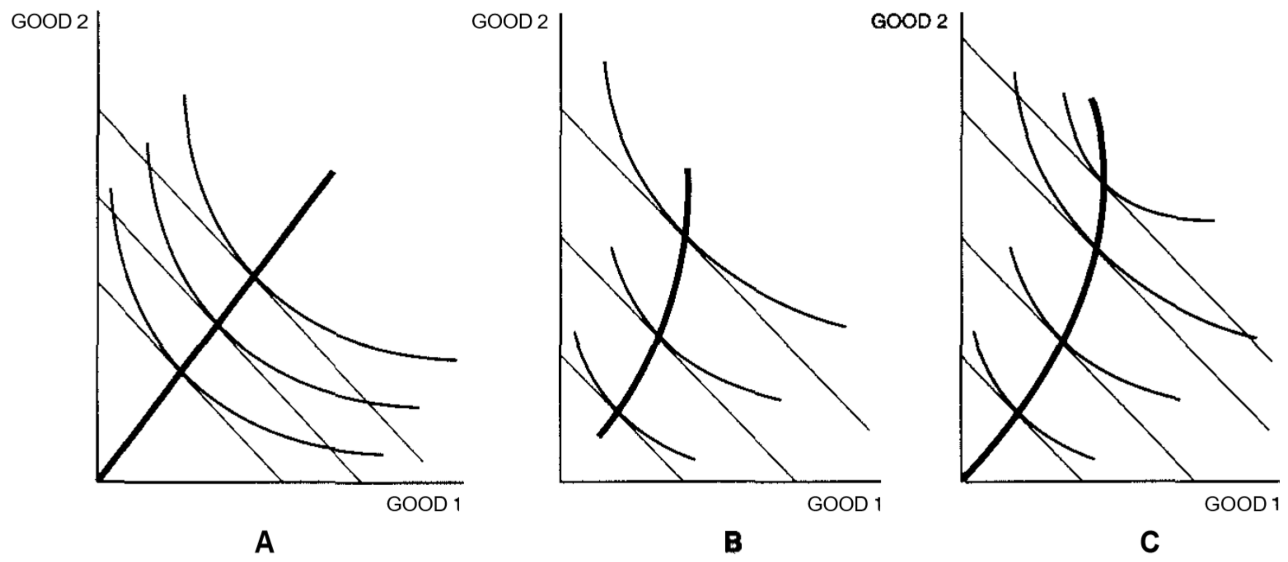
320 THE AMERICAN ECONOMIC REVIEW JUNE 1980

TABLE 2—TOTAL EXPENDITURE AND OWN-PRICE ELASTICITES

	Levels Model				First-Differences Model			
	Unconstrained		Homogeneous		Unconstrained		Homogene[illegible]	
	e_i	e_{ii}	e_i	e_{ii}	e_i	e_{ii}	e_i	[illegible]
Food	0.21	0.07	0.04	−0.01	0.04	0.22	0.17	[illegible]
Clothing	2.00	−0.92	1.51	−0.48	2.83	−0.94	2.92	[illegible]
Housing	0.30	−0.31	0.79	−0.16	0.04	−0.31	−0.02	[illegible]
Fuel	1.67	−0.28	1.37	0.10	1.00	0.00	[illegible]	[illegible]
Drink and Tobacco	1.22	−0.60	1.22	−0.62	1.37	−0.67	[illegible]	[illegible]
Transport and Communication	1.23	−1.21	1.73	−0.92	1.14	−1.23	[illegible]	[illegible]
Other goods	1.21	−0.72	1.15	−0.77	2.03	−0.52	[illegible]	[illegible]
Other services	1.40	−0.93	1.28	−0.78	1.03	−0.78	[illegible]	[illegible]

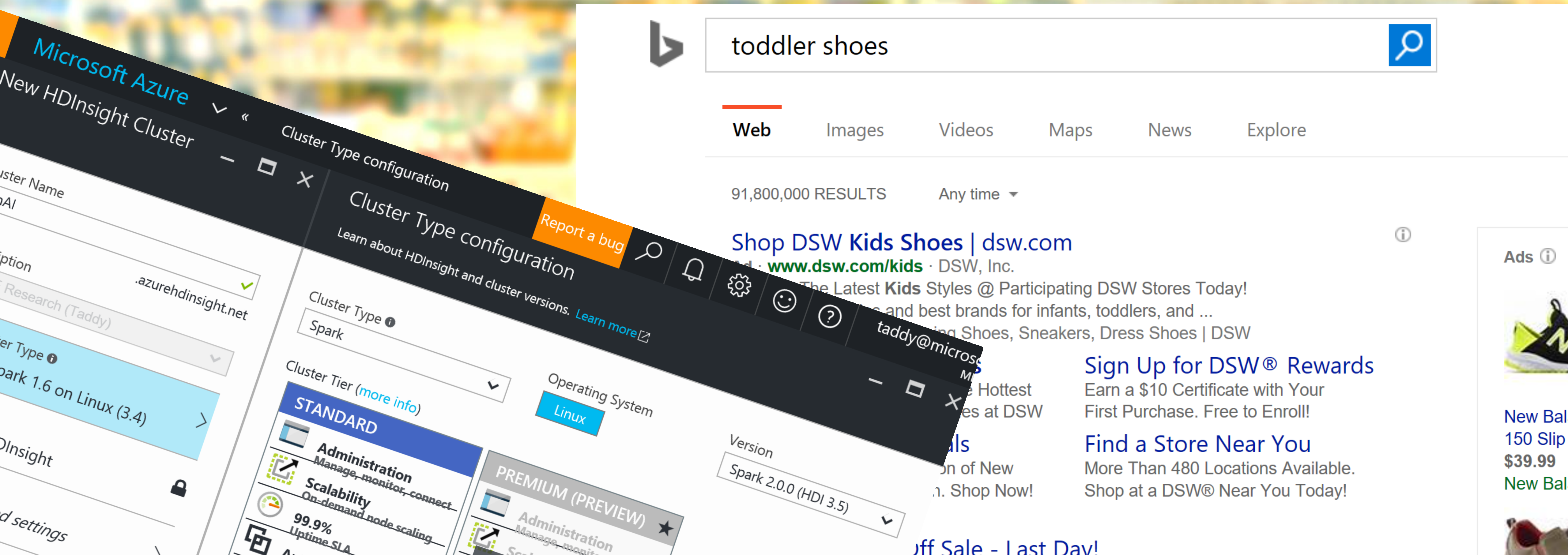
… the D.W. statistic shows a sharp … geneity is not a new … ten; Ray Byron; … ribed to a … of far

discussion of aggregati… A above, is that it … assume that k, the i… tribution of househ… graphic structure … average budget… Finally, the assu… l separability … mporal



Income expansion paths. Panel **A** depicts unit elastic demands, in panel B good 2 is a luxury good, and in panel C, good 1 is an inferior good.

What do they need to do today?



What can we (AI/ML) do to help?

The dimension and complexity of the problem space has exploded

We can develop ML to navigate this space: stay safe and automate

Econ AI is about systems of ML tasks for econometric questions

Example: Demand System

Suppose that you have transactions 't' on products 'j'.

Write the quantity bought 'q' as

$$q_{tj} = \alpha_{tj} + \gamma_j p_{tj} + e_{tj}$$

a function of utility we can (α_{tj}) and can't (e_{tj}) see, plus price p_{tj}.

You need to have a model like this to target customers or set prices.

But it's a system!

For example: There many different products

Demand for j depends on substitutes and complements

Or: where does price come from?

$$\log p_{tj} = \varphi_{tj} + \psi_j q^{\star}_{tj} + \nu_{tj}$$

and the *demand system* is in equilibrium when $q^{\star}_{tj} = q_{tj}$

Moving to a demand system (AID)

It's *almost* ideal:

$$\boldsymbol{s}_t \ = \ \boldsymbol{\alpha} \ + \ \boldsymbol{\Gamma}\log(\boldsymbol{p}_t) \ + \ \boldsymbol{\beta}\log\frac{e_t}{\phi_t} \ + \ \boldsymbol{\varepsilon}_t$$

s_{tj} is the budget share for product j in basket t and e_t is the budget

($e_t = \sum_j \$_{tj}$ and $s_{tj} = \$_{tj}/e_t$)

ϕ_t is the translog price index $\sum_j \log p_{tj} \left[\alpha_j + \sum_k \gamma_{jk} \log p_{tk}\right]$

But not so easy…

In grocery stores, we have 100,000s of products

Each week, at each store, for each product,
we will have different demand and elasticity and income effects.

$$\mathbb{E}s_{tj} = \alpha_{tj} + \sum_k \gamma_{tjk} \log p_{tj} + \beta_{tj} \frac{e_t}{\phi_t}$$

Featurize and Embed

Grab massive $\boldsymbol{x}_{tj}$ that *hierarchically* encode products and transactions

$$\mathbb{E}s_{tj} = \boldsymbol{x}'_{tj}\boldsymbol{\alpha} + \boldsymbol{x}'_{tj}\boldsymbol{\delta}\log p_{tj} + \boldsymbol{x}'_{tj}\boldsymbol{\beta}\log\left(\frac{e_t}{\phi_t(\boldsymbol{x}_{tj})}\right) + \boldsymbol{\Gamma}\log\boldsymbol{p}_t$$

and use a symmetric square matrix factorization $\boldsymbol{\Gamma} = \boldsymbol{U}\boldsymbol{V}' + \boldsymbol{V}\boldsymbol{U}'$

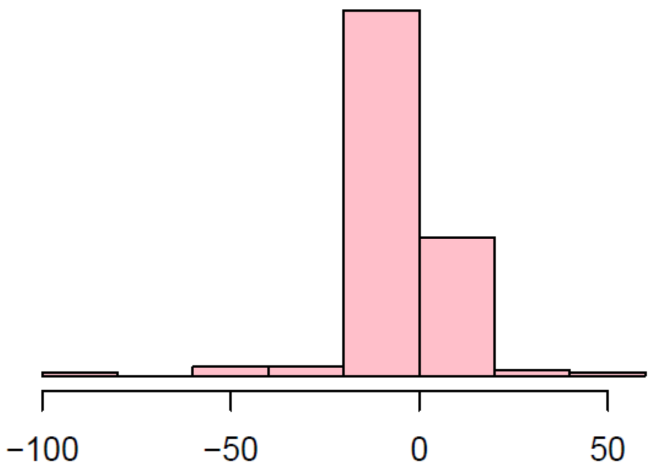
Note we are using the actual trans-log and solving a bilinear system

Beer Store own-price (compensated) elasticities

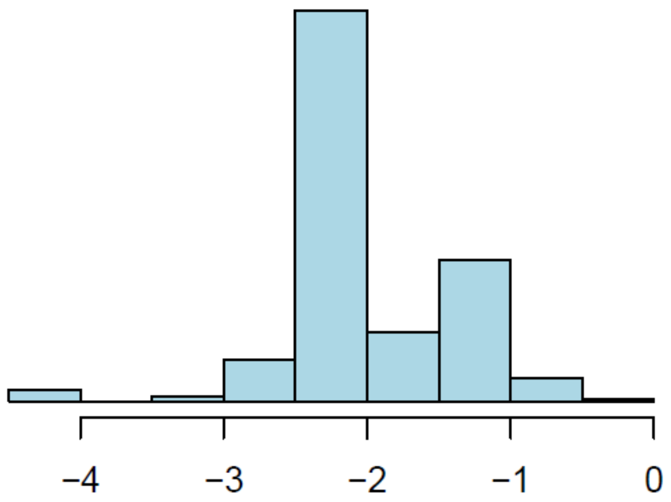
one shared: $x_{tj} = 1$

$$\frac{dq}{dp}\frac{p}{q} = -0.23$$

brand-specific: $x_{tjk} = \mathbb{1}_{[k=j]}$

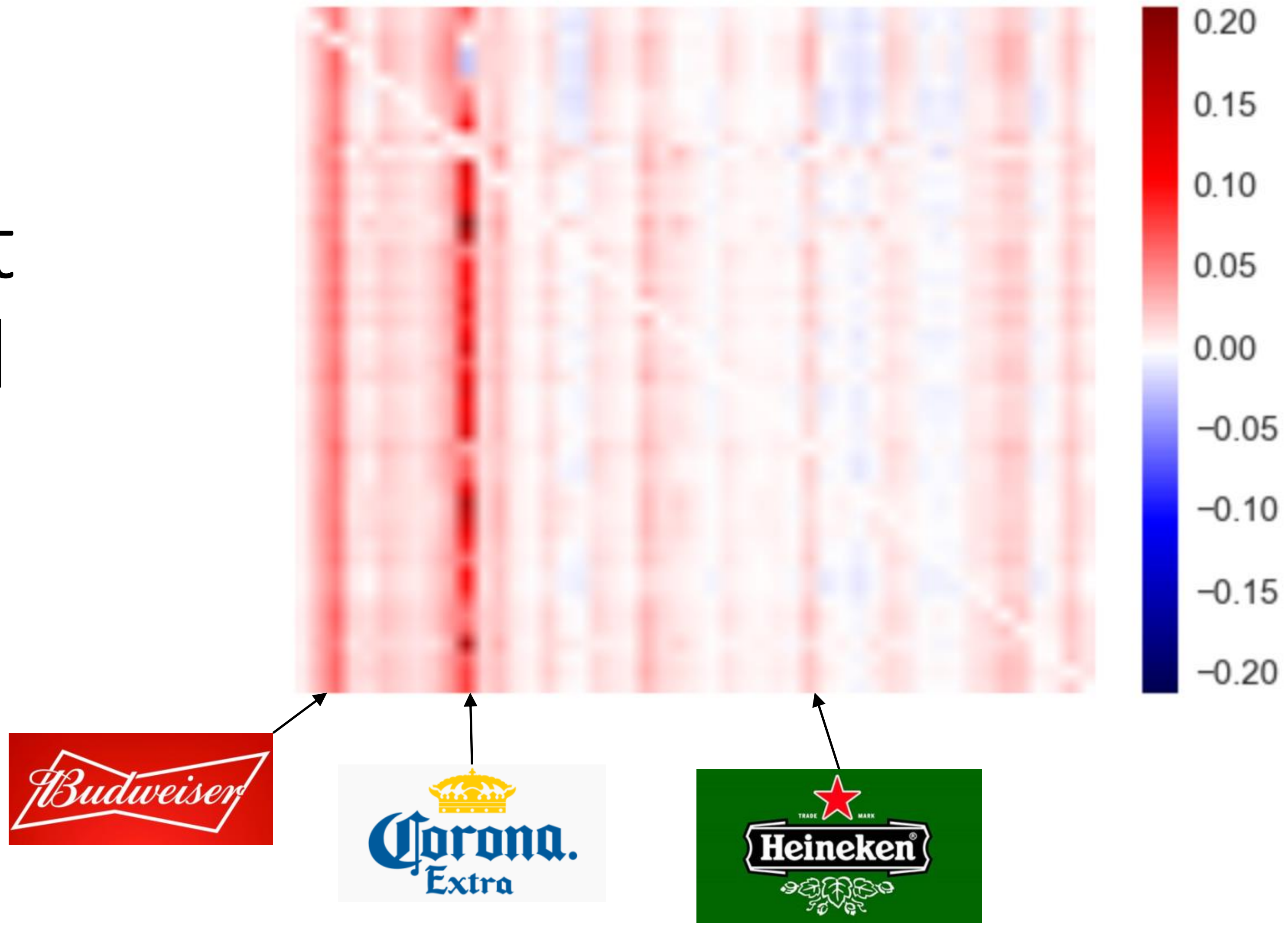


$\boldsymbol{x}_{tj}$= *featurized description*



Beer Store cross-product compensated elasticities

Elasticity matrix (omitting diagonal)



But wait… it's still a system

Recall: where does price come from?

$$\log p_{tj} \ = \ \varphi_{tj} \ + \ \psi_j q^{\star}_{tj} \ + \ \nu_{tj}$$

and the *demand system* is in equilibrium when $q^{\star}_{tj} = q_{tj}$

This equilibrium introduces `price endogeneity': $\mathbb{E}[p_{tj} e_{tj}] \neq 0$

Endogenous Errors

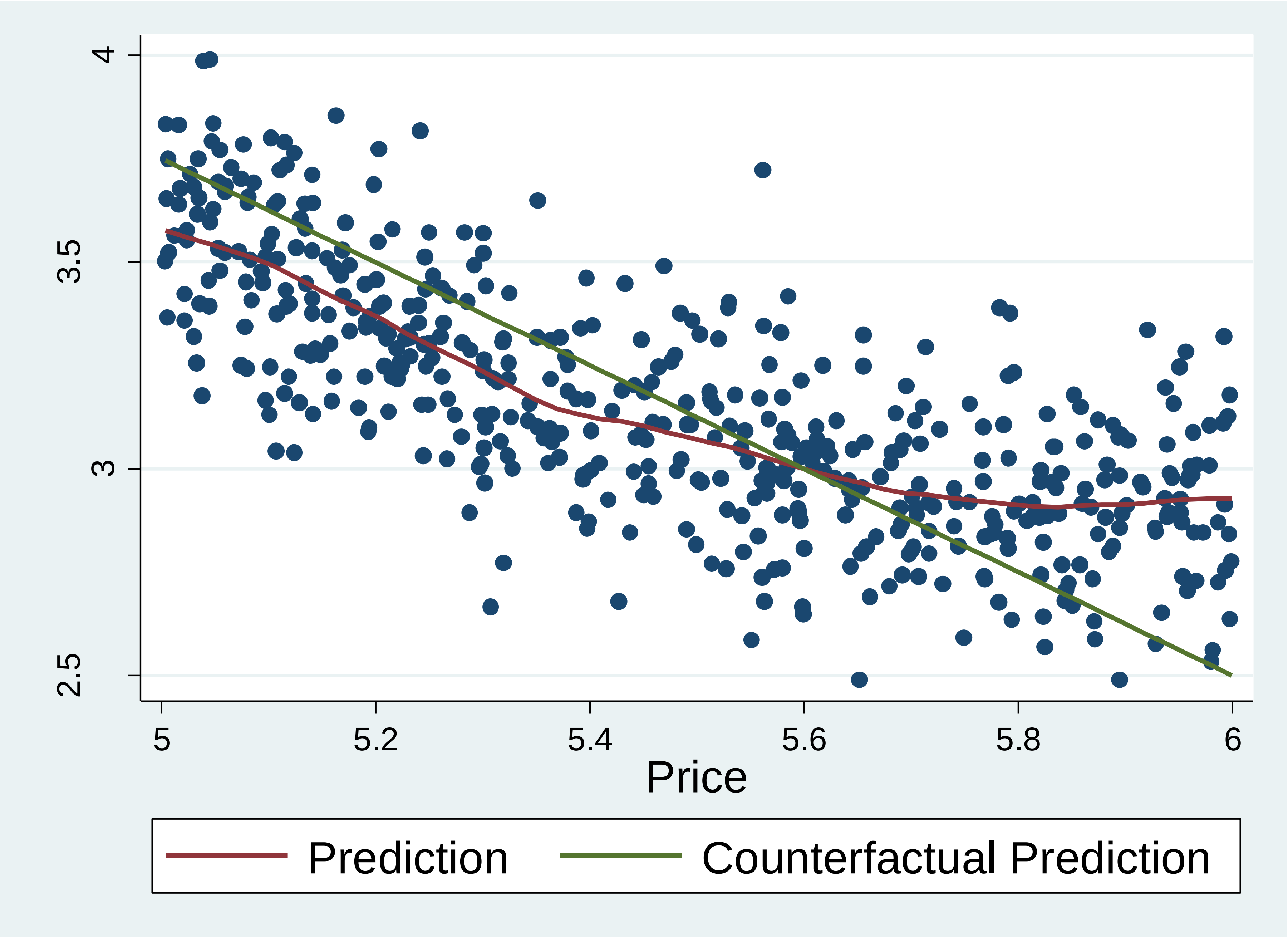
$$y = g(p, \boldsymbol{x}) + e \;\text{ and }\; \mathbb{E}[\,p\,e\,] \neq 0$$

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

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

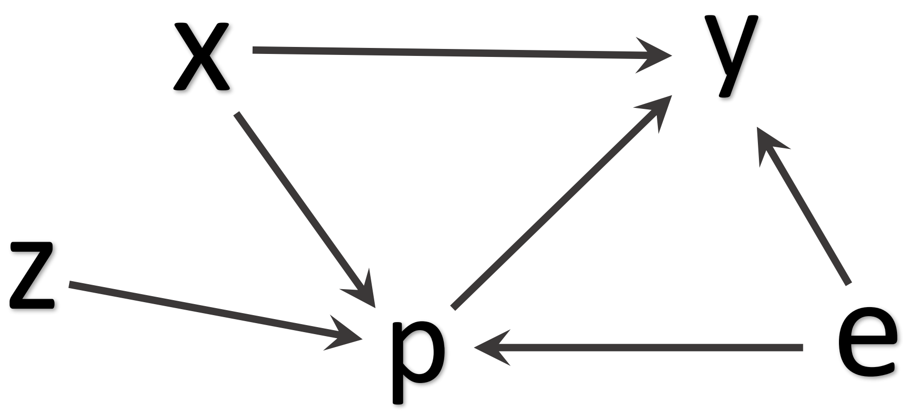
This works for prediction. It doesn't work for counterfactual inference:

What happens if I change p independent of e?



4
3.5
3
2.5
5
5.2
5.4
5.6
5.8
6
Price
Prediction
Counterfactual Prediction

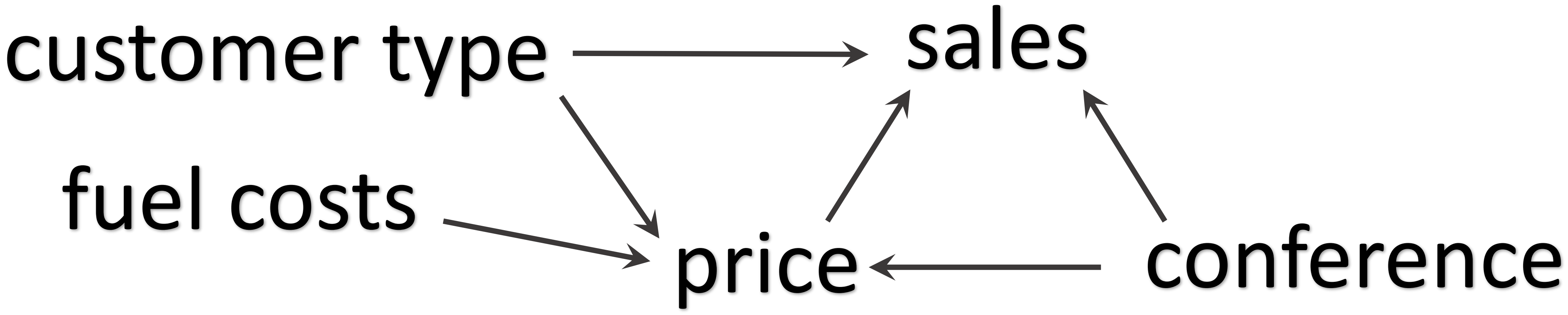
Instrumental Variables (IV)



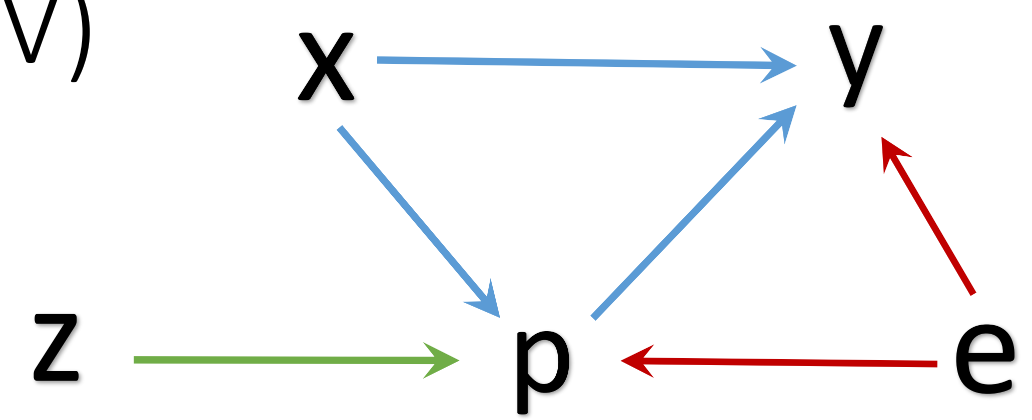
In IV we have a special $z \perp e$ that influences policy p but not response y.

- 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)



Instrumental Variables (IV)



The *exclusion structure* implies

$$E[y|x,z] = E[g(p,x)|x,z] + E[e|x] = \int g(p,x)d\mathrm{F}(p|x,z)$$

So to solve for *structural* $h(p,x)$ we have a new learning problem

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

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

2SLS:

$p = \beta z + \nu$ and $g(p) = \tau p$ so that $\int g(p) d\mathrm{P}(p|z) = \tau \hat{p} = \tau \hat{\beta} z$

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

This requires strict assumptions and homogeneous treatment effects.

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

Or look to nonparametric 2SLS like in Newey and Powell:

$$g(p, x_i) \approx \sum_k \varphi_k(p, x_i) \text{ and } \varphi_k(p, x_i) \approx \sum_j \phi_{kj}(x_i, z_i)$$

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

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

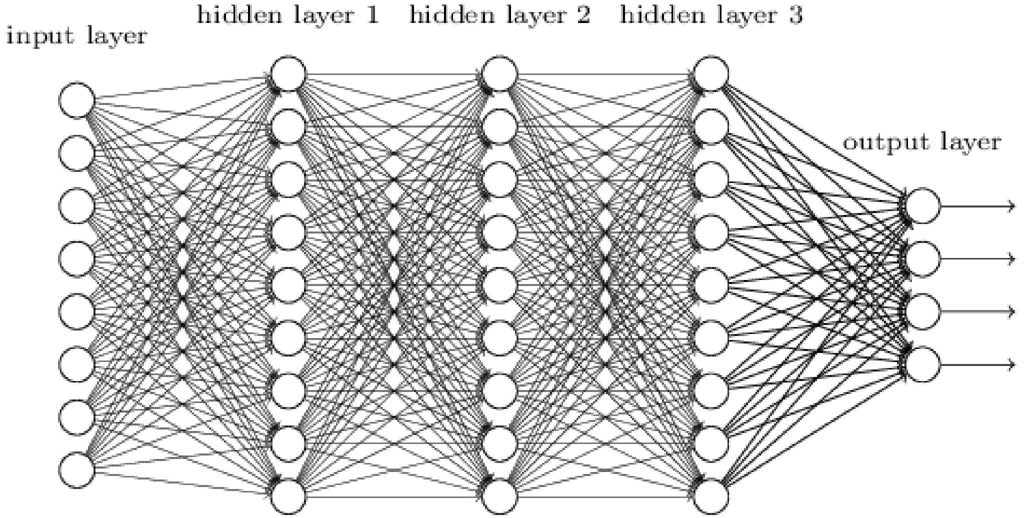
Instead, we propose to target the integral loss function directly

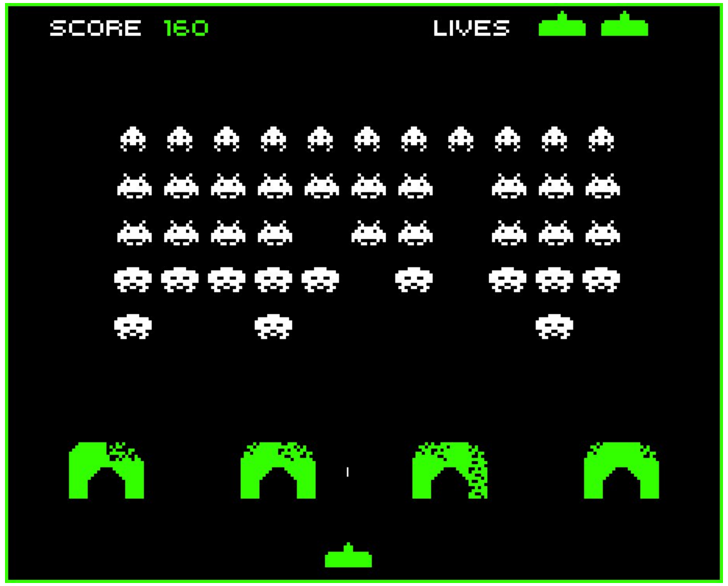
For discrete (or discretized) treatment

- 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





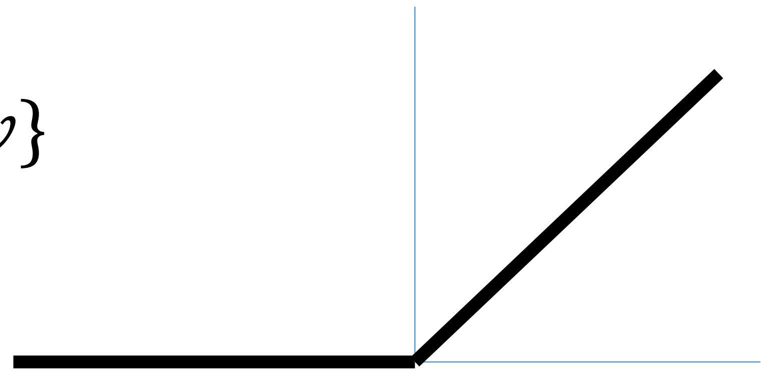
What is a deep net?

$$\hat{y}_i = \sum_k \eta_k(a_{ik}), \qquad a_{ik} = \sum_j w_{kj} z_j\,, \quad z_j = \sum_l h_l^1(b_{il})\,, \ldots$$

And so-on until you get down to a bottom layer $\{f_l(x_i)\}_l$

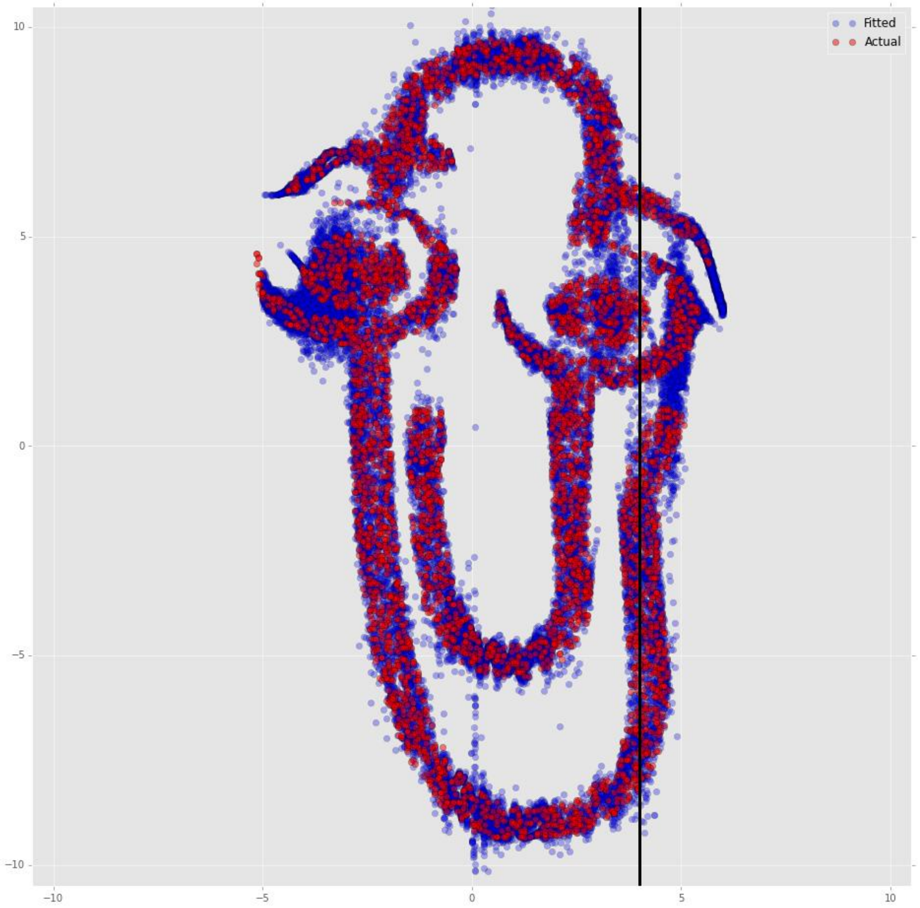
Many different variations here: recursive, convolutional, …

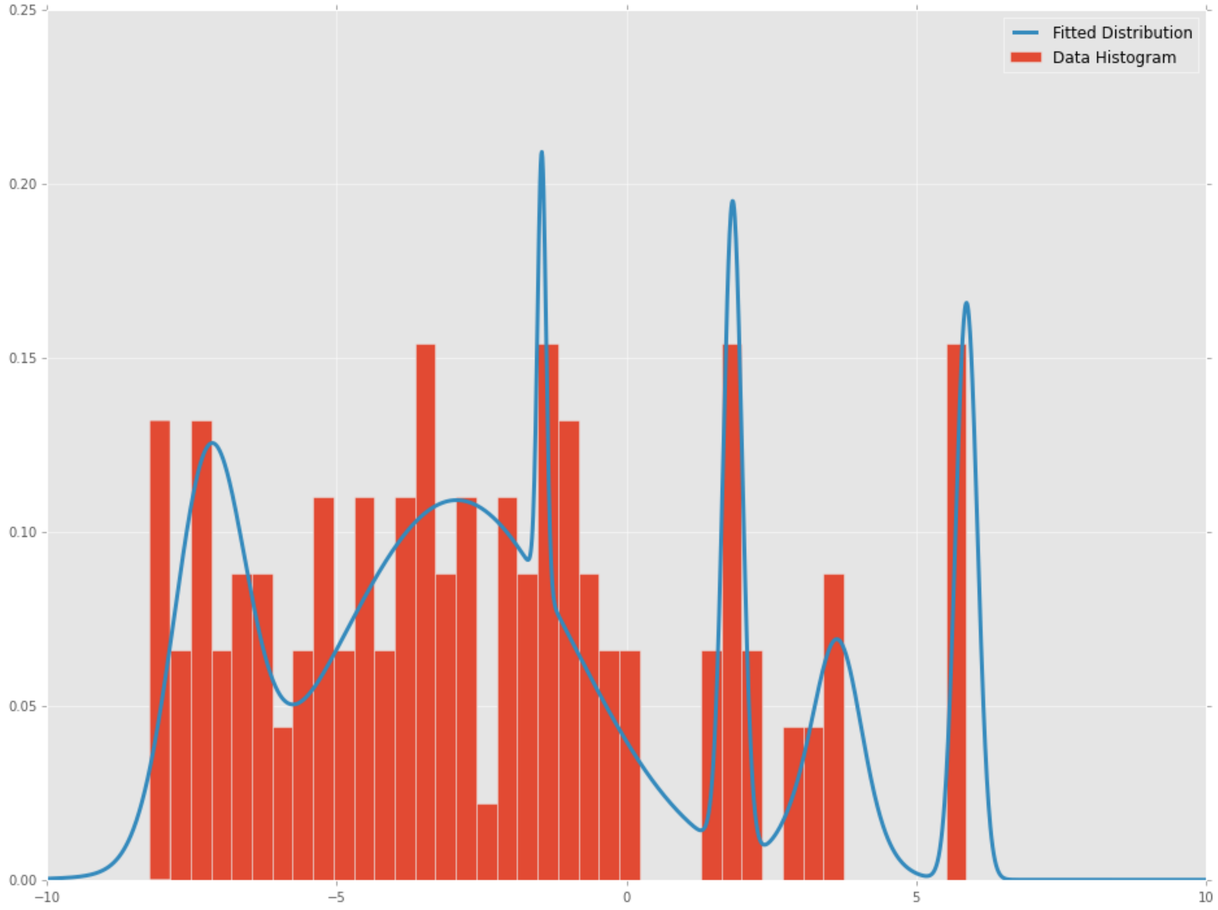
Apart from the bottom, usually $h(v) = \max\{0, v\}$



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

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





The second stage involves an integral loss function

If p is not discrete or can take many values, we can't just integrate

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

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

But this is inefficient

And more generally, it's inefficient even *without* the integral…

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

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

Why SGD? You get what you need faster

Table 2. Asymptotic equivalents for various optimization algorithms: gradient descent (GD, eq. 2), second order gradient descent (2GD, eq. 3), stochastic gradient descent (SGD, eq. 4), and second order stochastic gradient descent (2SGD, eq. 5). Although they are the worst optimization algorithms, SGD and 2SGD achieve the fastest convergence speed on the expected risk. They differ only by constant factors not shown in this table, such as condition numbers and weight vector dimension.

	GD	**2GD**	**SGD**	**2SGD**
Time per iteration :	n	n	1	1
Iterations to accuracy ρ :	$\log \frac{1}{\rho}$	$\log\log \frac{1}{\rho}$	$\frac{1}{\rho}$	$\frac{1}{\rho}$
Time to accuracy ρ :	$n \log \frac{1}{\rho}$	$n \log\log \frac{1}{\rho}$	$\frac{1}{\rho}$	$\frac{1}{\rho}$
Time to excess error $\mathcal{E}$:	$\frac{1}{\mathcal{E}^{1/\alpha}} \log^2 \frac{1}{\mathcal{E}}$	$\frac{1}{\mathcal{E}^{1/\alpha}} \log \frac{1}{\mathcal{E}} \log\log \frac{1}{\mathcal{E}}$	$\frac{1}{\mathcal{E}}$	$\frac{1}{\mathcal{E}}$

$$\mathcal{E} \;=\; \mathcal{E}_{\mathrm{app}} + \mathcal{E}_{\mathrm{est}} + \mathcal{E}_{\mathrm{opt}} \;\sim\; \mathcal{E}_{\mathrm{app}} + \left(\frac{\log n}{n}\right)^{\alpha} + \rho$$

Learning rates

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

What is $\boldsymbol{C}_t$? Hopefully not $\mathrm{c_t}\mathbf{I}$ …
Ideally, $\boldsymbol{C}_t \rightarrow t^{-1}\boldsymbol{C}$ and $\boldsymbol{C}^{-1} = -\nabla\nabla L(\boldsymbol{D}, \theta^\star)$, but that's infeasible

ADAGRAD (for convex loss)

$$\boldsymbol{C}_t = \mathrm{diag}\left(\sqrt{\sum_t \nabla L_{tj}^2}\right) \Rightarrow \sum_t \big(err(\theta_t) - err(\theta^\star)\big) = O(\|\theta^\star\| \mathrm{tr}(\mathbf{C}_t))$$

We use heuristic innovations on this (ADAM, momentum, etc)

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

$$\nabla \hat{L}(d_i, \theta) = -2(y_i - g_\theta(\hat{p}_{i1}, x_i))\, g_\theta'(\hat{p}_{i2}, x_i), \qquad \hat{p}_{ib} \sim \hat{\mathrm{F}}(p|x_i, z_i)$$

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

Lower variance grad ⇒ faster/better convergence

e.g., in most architectures you want to fit F and g in two-stages

And with MC integration, consider:

$$\nabla\hat{L}\left(\{\boldsymbol{d}_b\}_{b=1}^{B},\theta\right) = \sum_{b=1}^{B} \left(y_b - g_\theta(\hat{p}_{b1},x_b)\right) g'_\theta(\hat{p}_{b2},x_b)$$

$$\nabla\tilde{L}^B(\boldsymbol{d},\theta) = \left(y - B^{-1}\sum_{b=1}^{B} g_\theta(\hat{p}_{b1},x\,)\right)B^{-1}\sum_{b=1}^{B} g'_\theta(\hat{p}_{b2},x\,)$$

Both involve the same number of operations, but

$$\mathrm{var}\nabla\hat{L}(\{\boldsymbol{d}_b\}_{b=1}^{B},\theta) = \frac{\mathrm{var}\nabla\hat{L}(\boldsymbol{d},\theta)}{B} \quad \text{while} \quad \mathrm{var}\nabla\tilde{L}^B(\boldsymbol{d},\theta) \approx \frac{E\left[\mathrm{var}\nabla\tilde{L}^1(\boldsymbol{d},\theta)\right]}{B} + \mathrm{var}(\nabla L(\boldsymbol{d},\boldsymbol{\theta}))$$

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, or wider class in Pakes and Pollard

Sampling 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

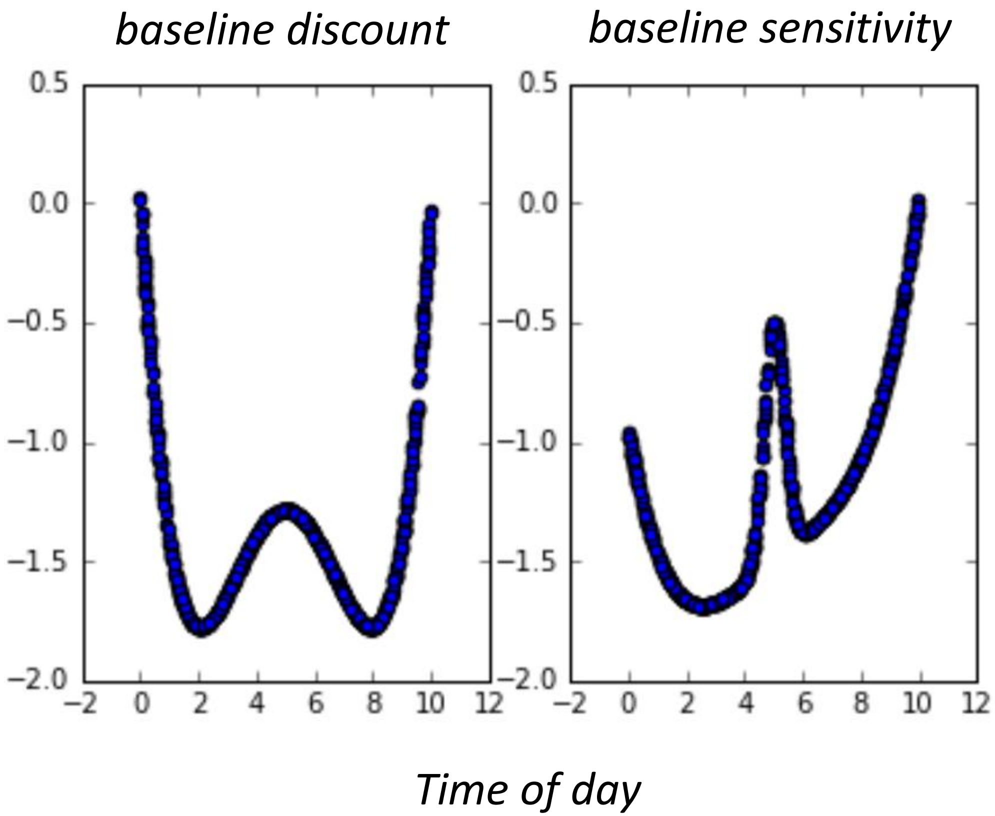
$$\sum_{i \in LO} -\log \hat{f}(p|x_i, z_i)$$

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

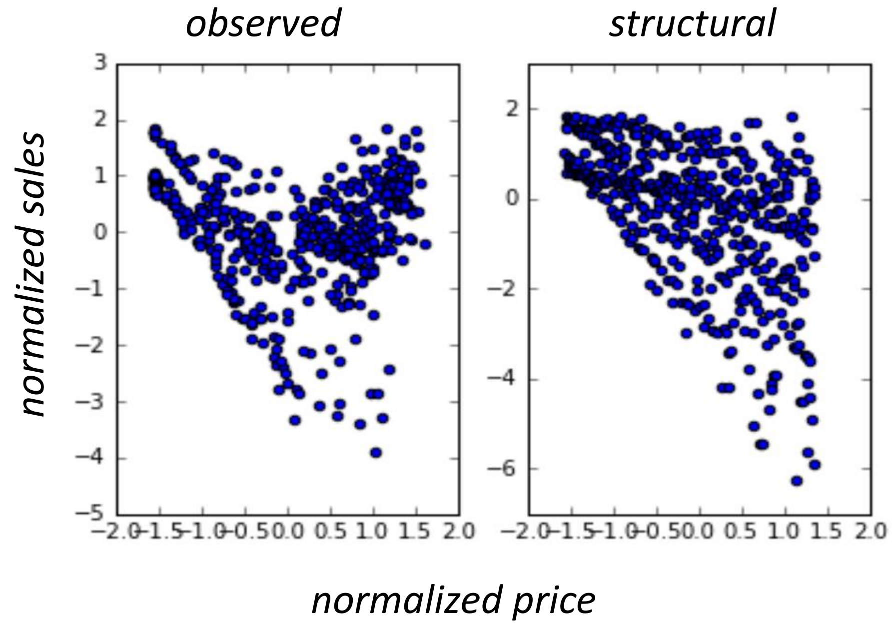
$$\sum_{i \in LO} \left(y_i - g_\theta(\dot{p}_i, x_i)\right)^2, \;\; \dot{p}_i \sim \hat{\mathrm{F}}(p|x_i, z_i)$$

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

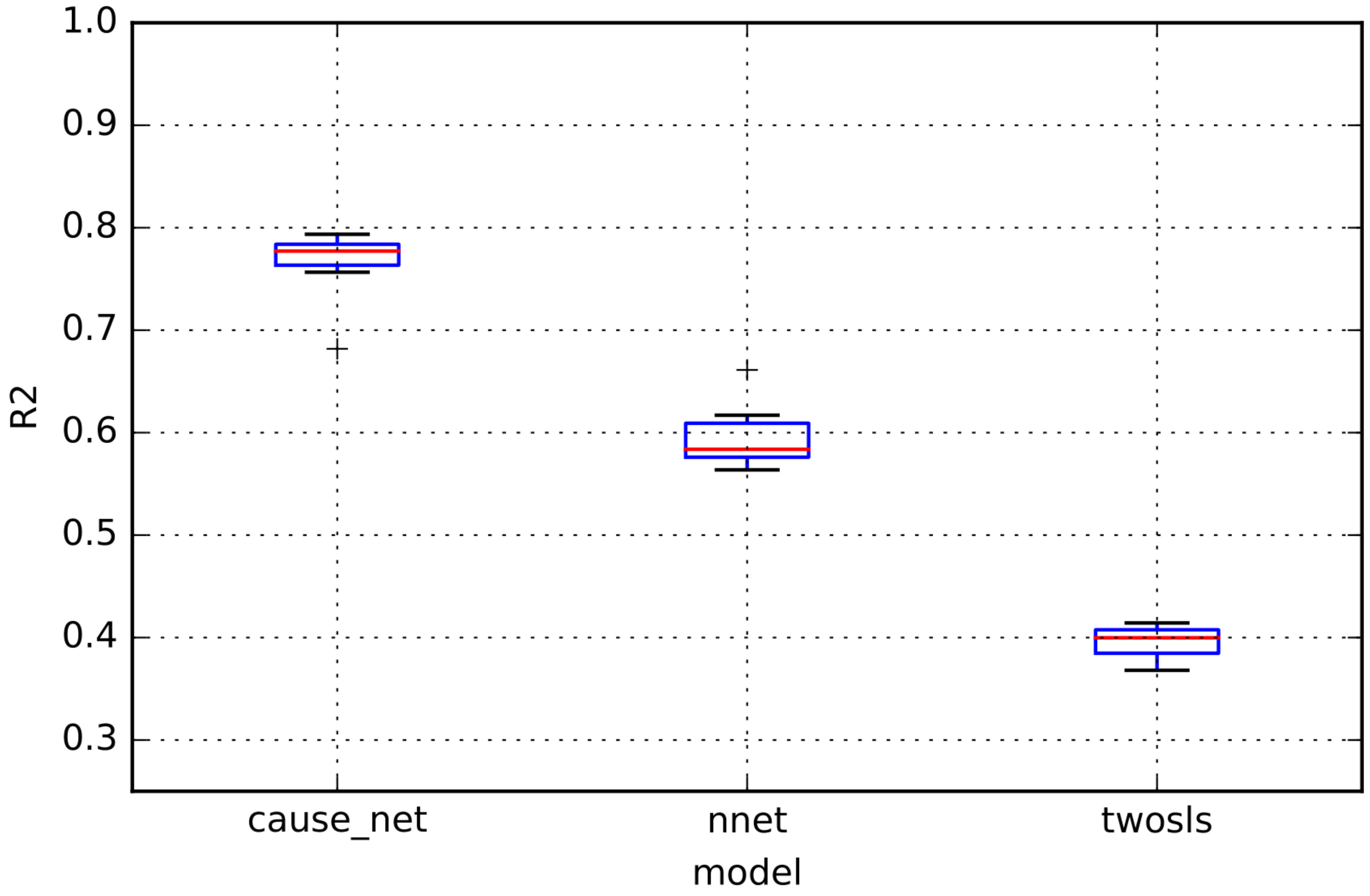
heterogeneous price effects



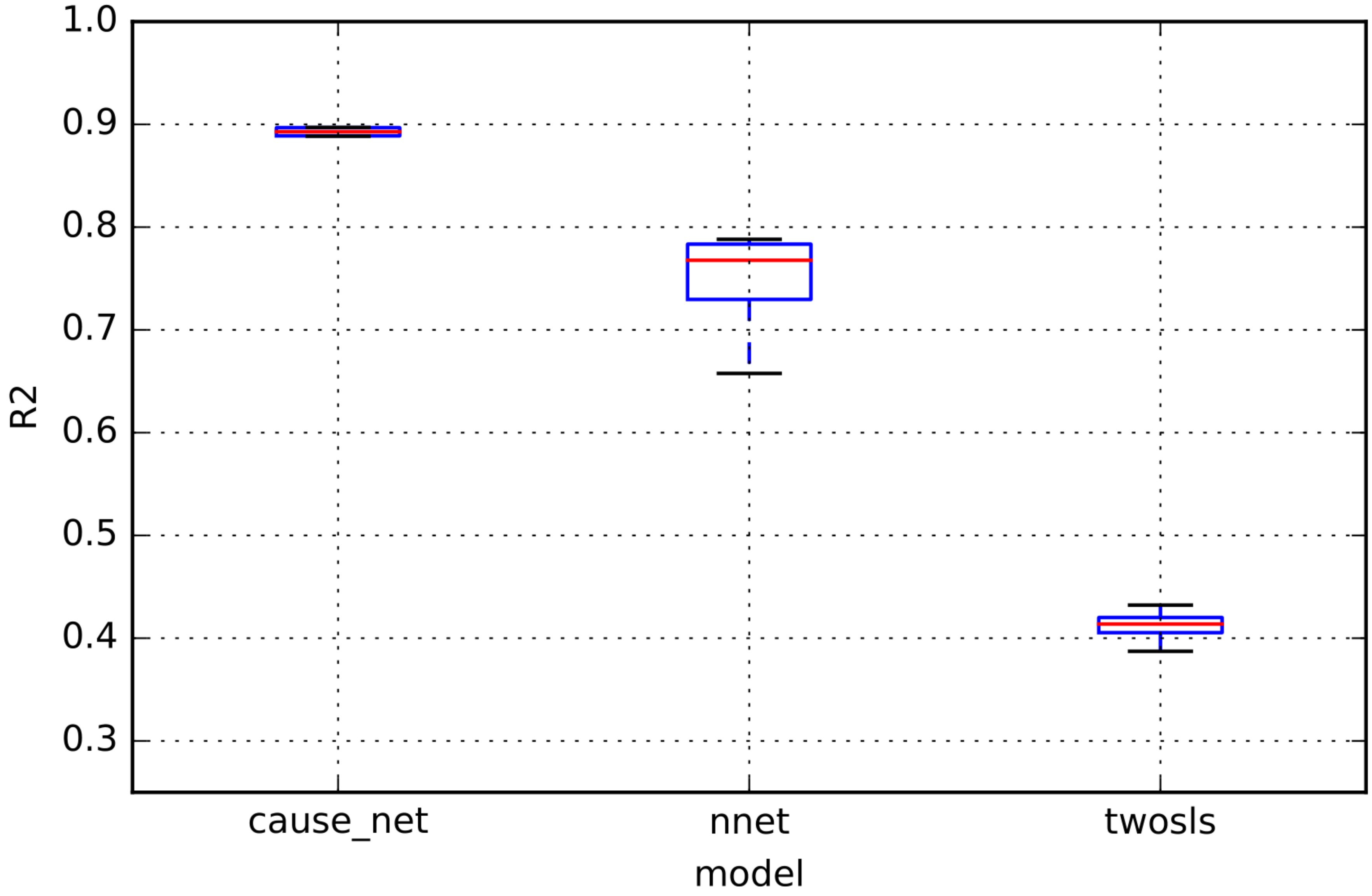
- 'time' dependent prices and elasticity
- 7 different customer types multiply discounts, sensitivities, and demand



N=1000



N=5000



Identification

The truth g_0 is identified if this equation has a unique solution. Let $T[g](x,z) = \int g(p,x)dF(p|x,z)$. Since T is a linear operator on the function space G, uniqueness of the solution is equivalent to the kernel of T being the singleton zero function:

$$\{g \in G : T(g) = 0\} = \{0\}$$

This can equivalently be stated as the "completeness condition" of Newey and Powell (2003): for all measurable g, we have:

$$E[g(t,x)|x,z] = 0 \; \forall (x,z) \in supp(X,Z) \;\; \Rightarrow g(t,x) = 0$$

where the equivalence comes from the fact that the LHS function is just $T(g)$.

Consistency

Our is just a minimum distance estimator conditional upon draws from the 1st stage

Consistency has $||\bar{g} - \bar{g}^{\star}|| \to 0$ after averaging over the instrument distribution

The relevant theory is in Newey and Powell (03) and Chen and Pouzo (12)

Promise to grow both nets at rates $k(n) \to \infty$, $k/n \to 0$, then approximation theorems for neural nets (e.g. Hornik 91) imply density around the truth

For treatment (and 1st stage) continuous, also need # mixture components to grow

Ads Application

Taken from Goldman and Rao (2014)

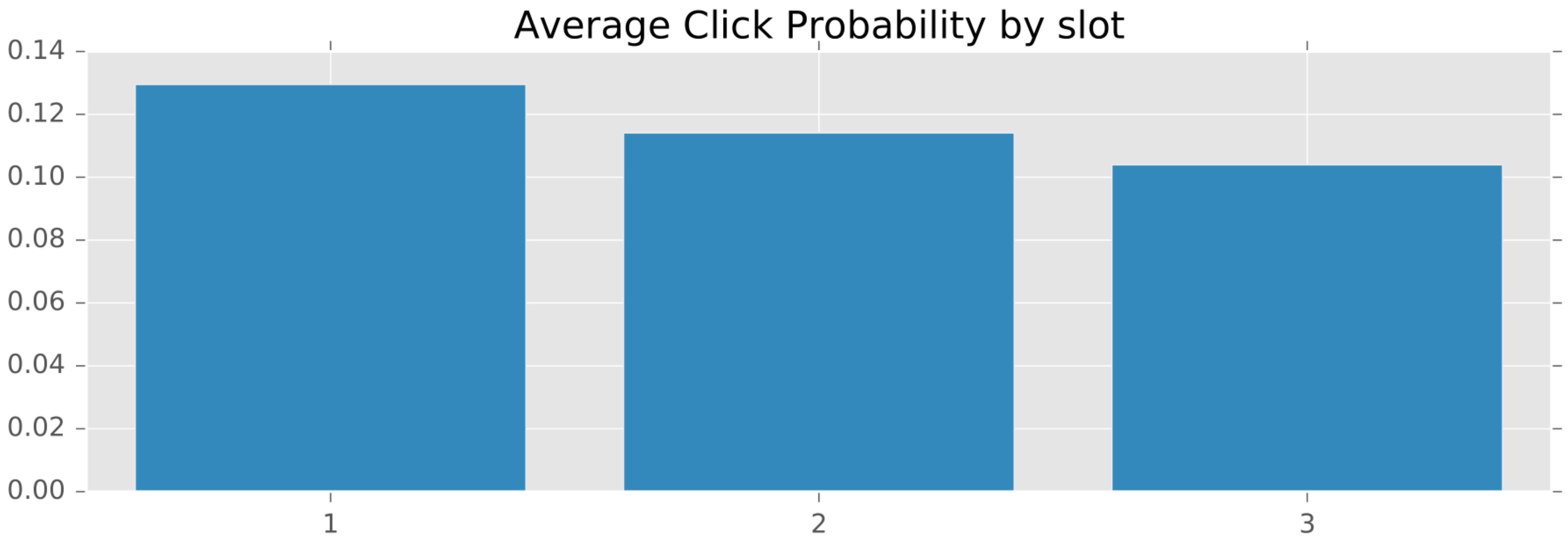
We have 74 mil click-rates over 4 hour increments for 10k search terms

Treatment: ad position 1-3

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

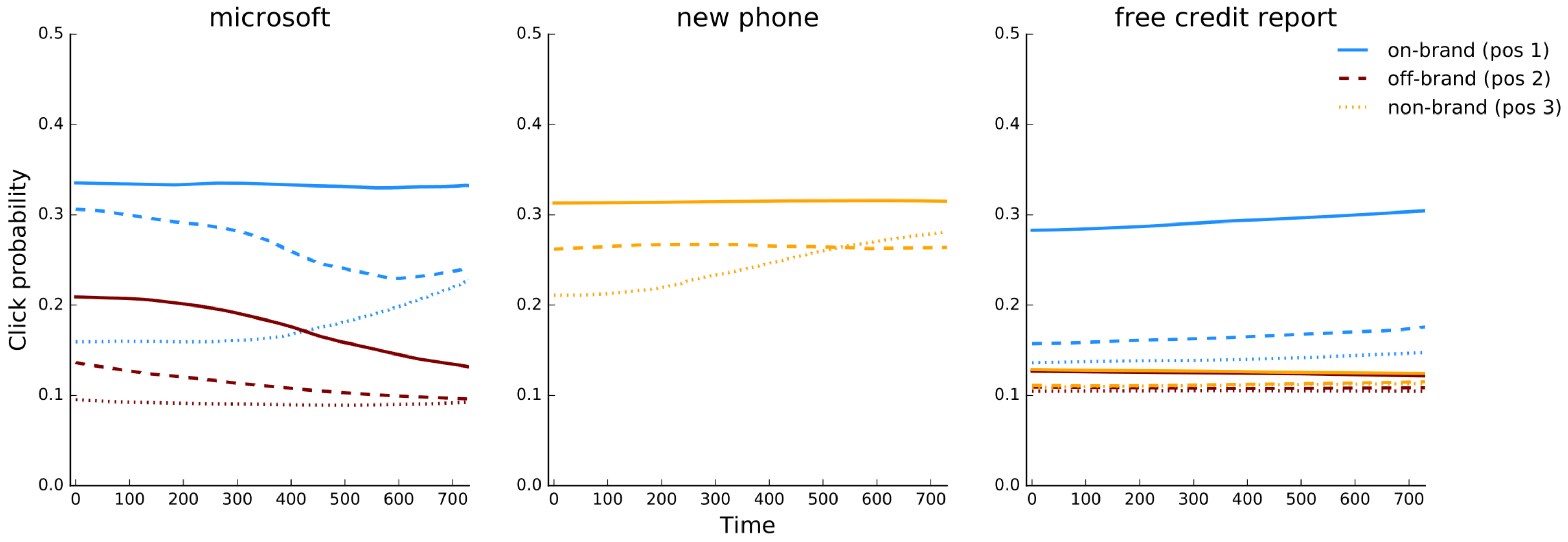
Covariates: advertiser id and ad properties, search text, time period

Average Treatment Effects



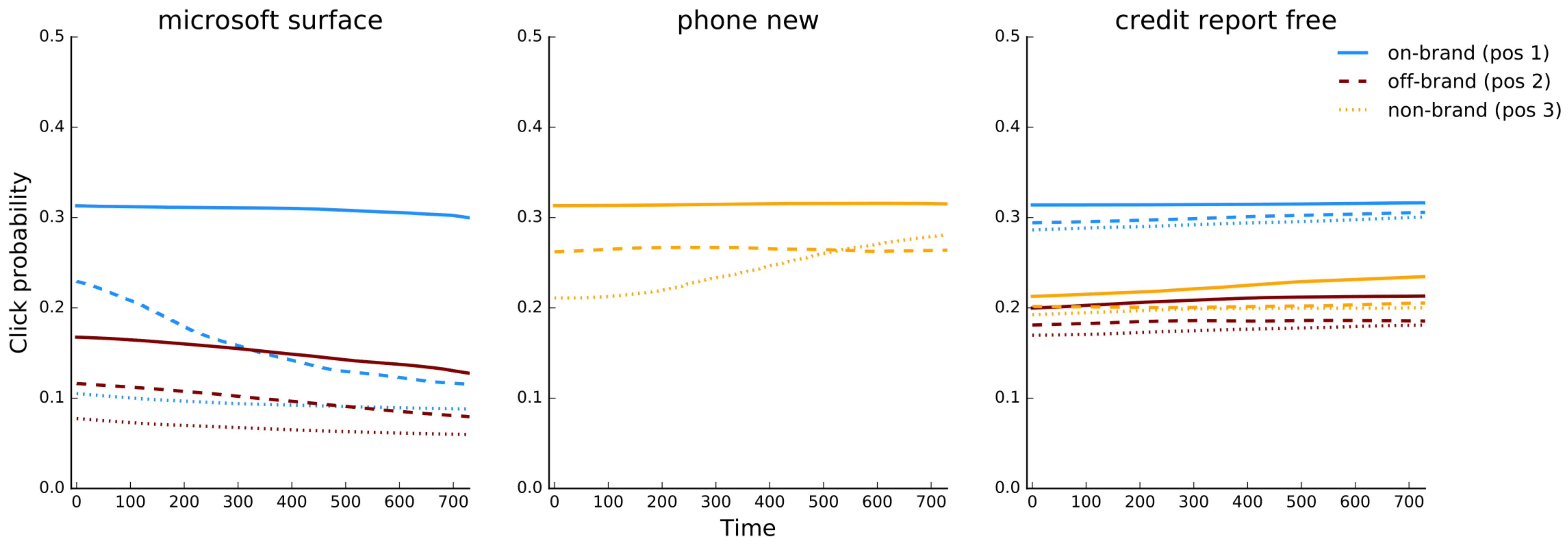
These compare to observed click probabilities of 0.33, 0.1, and 0.05.

Heterogeneous Treatment Effects



The *gaps* between ad-position lines can be interpreted *causally.*

Heterogeneous Treatment Effects



The *gaps* between ad-position lines can be interpreted *causally.*

Inference? Good question

For frequentist results we can do data splitting

Take the trained DNNs, and on *left-out* data do:

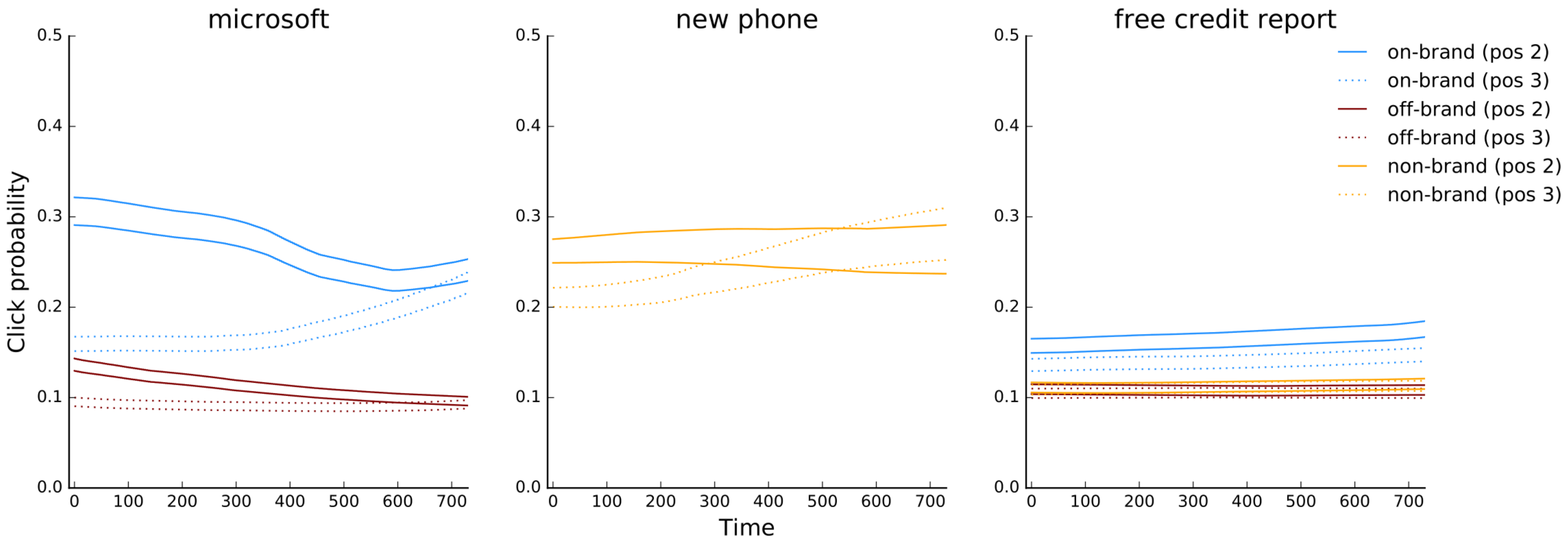
- Calculate conditional averages for each node in the top layer

$$\bar{\eta}_{ik} = E_{\hat{F}(\dot{p}|x_i,z_i)}\eta_k(x_i,\dot{p})$$

- Stack these as a design $\overline{\mathbf{H}} = [\bar{\boldsymbol{\eta}}_{\mathbf{1}} \cdots \bar{\boldsymbol{\eta}}_L]'$ and use OLS to fit $\boldsymbol{y} \approx \overline{\mathbf{H}}\boldsymbol{\beta}$
- Calculate *structural* residuals $e_i = y_i - \sum_k \hat{\beta}_k \eta_k(x_i, p_i)$
- Use the sandwich formula $\mathrm{var}\left(\widehat{\boldsymbol{\beta}}\right) = (\overline{\mathbf{H}}'\overline{\mathbf{H}})^{-\mathbf{1}}\overline{\mathbf{H}}'\mathrm{diag}(\mathbf{e})\overline{\mathbf{H}}(\overline{\mathbf{H}}'\overline{\mathbf{H}})^{-\mathbf{1}}$

Then, e.g., $\mathrm{var}(\,\hat{y}(x,p)) = \boldsymbol{\eta}_{\boldsymbol{k}}'(x,p)\mathrm{var}\left(\widehat{\boldsymbol{\beta}}\right)\boldsymbol{\eta}_{\boldsymbol{k}}(x,p)$

Heterogeneous Treatment Effects



+/- 2 *post selection* SD

Inference? Good question

Or Bayes: recall the deep net

$$\hat{y}_i = \sum_k \eta_k(a_{ik}), \qquad a_{ik} = \sum_j w_{kj} z_j\,, \quad z_j = \sum_l h_l^1(b_{il})\,, \ldots$$

When training with SGD, we actually use dropout for regularization

At each update, calculate gradients against $\tilde{\theta}$ where

$$\tilde{w}_{kj} = w_{kj}\delta_{kj}, \qquad \delta_{kj} \sim \mathrm{Bern}(\pi)$$

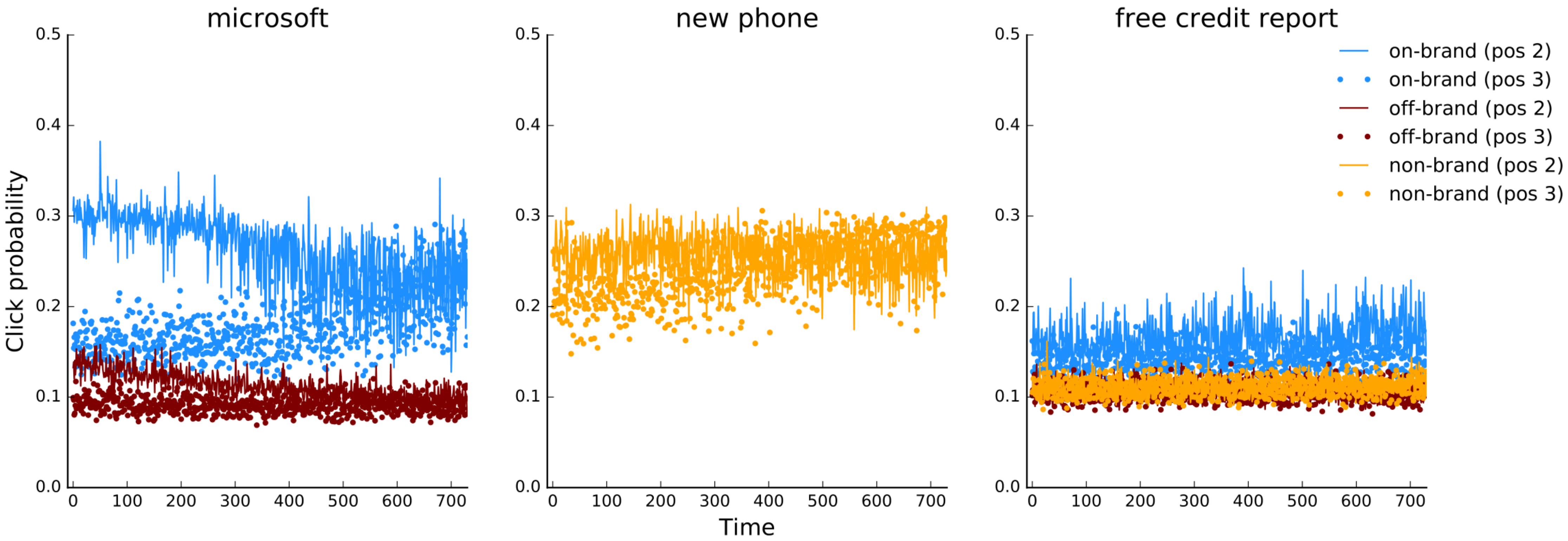
Bayesian inference via dropout

Variational Bayes minimizes

$$KL(q \mid p) = \mathbb{E}_{\mathrm{q}}[\log q(y|x)] \ - \mathbb{E}[\log p(y|x, \boldsymbol{D})]$$

Gal and Ghahramani show that *dropout* SGD is optimizing variational distribution q, with uncertainty parametrized by the Bernoulli weights

Posterior Treatment Effect Samples



Each point/dash is an independent draw from the `posterior'

Economics and Artificial Intelligence

We have a track record pointing ML at questions of science + causation.
We're going to replicate this success *at scale on unstructured data*

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* Artificial Intelligence.

Endogenous Errors

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

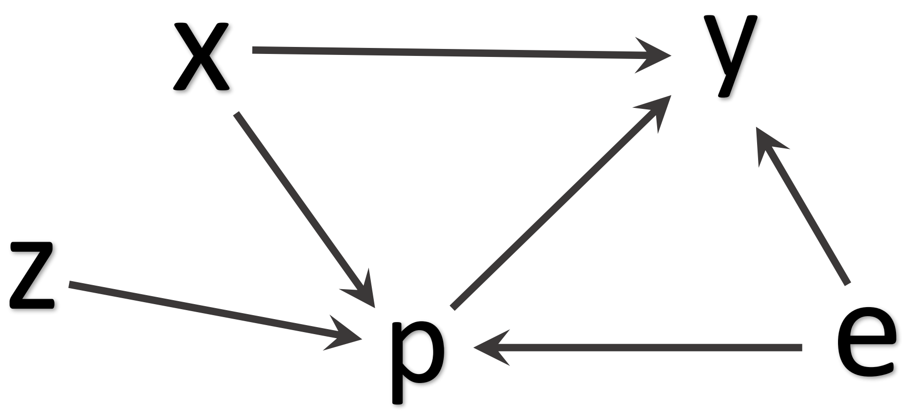
If you estimate this using naïve ML, you'll get

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

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

What happens if I change p independent of e?

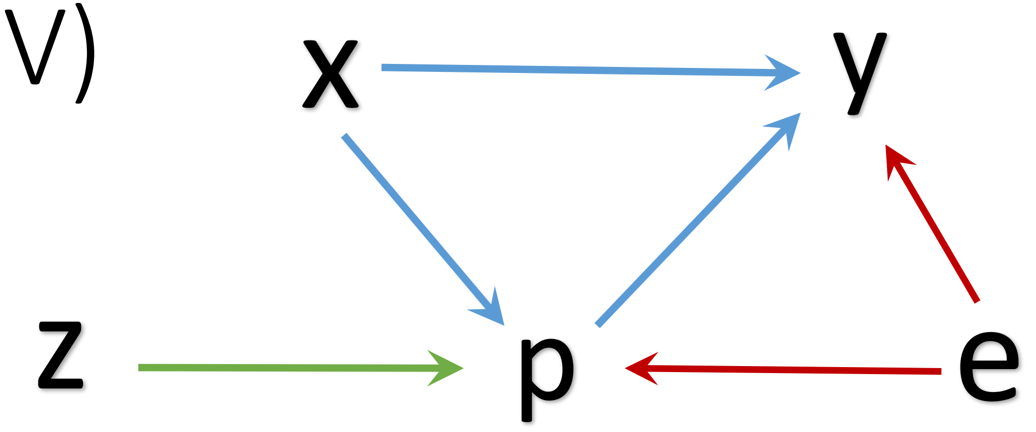
Instrumental Variables (IV)



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 $\hat{\mathbb{E}}[y|x,z]$ 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: $p = \beta z + v$ 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 $\dim(x)$

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

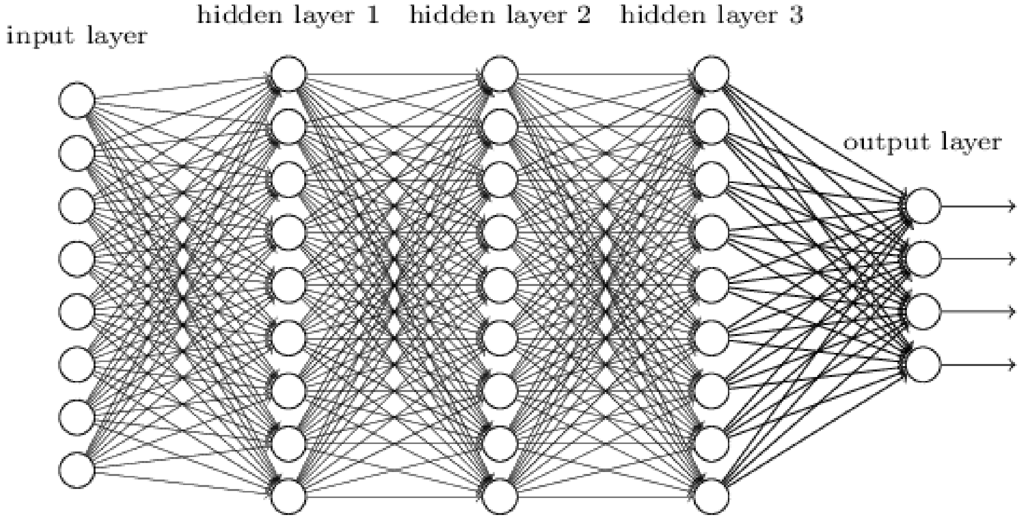
Instead, we propose to target the integral loss function directly

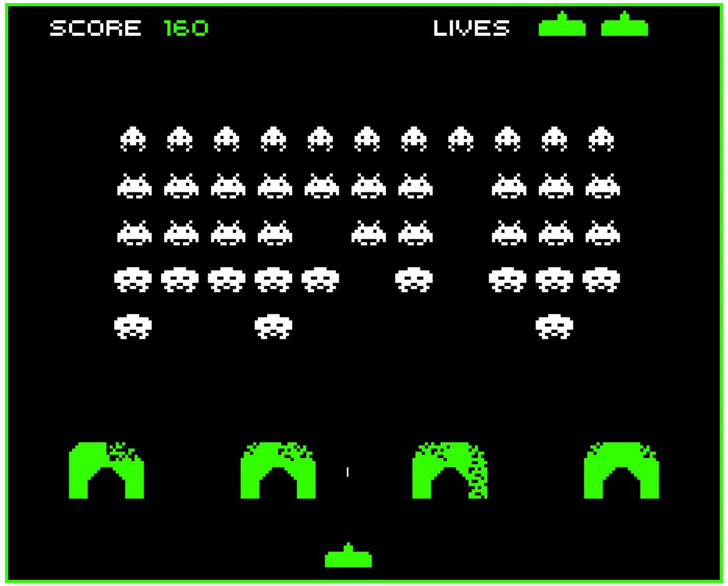
For discrete (or discretized) treatment

- 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





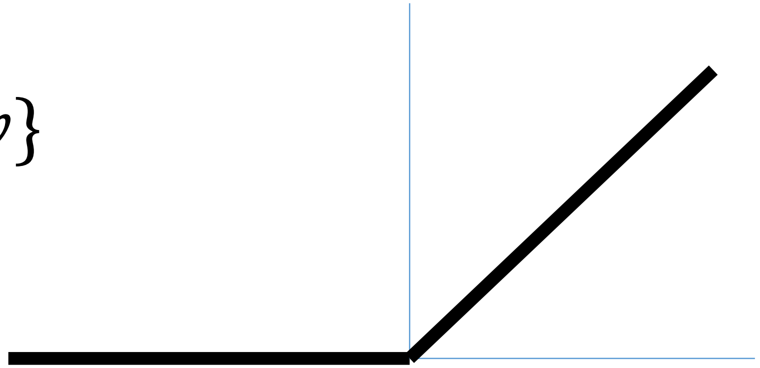
What is a deep net?

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

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 $h(v) = \max\{0, v\}$



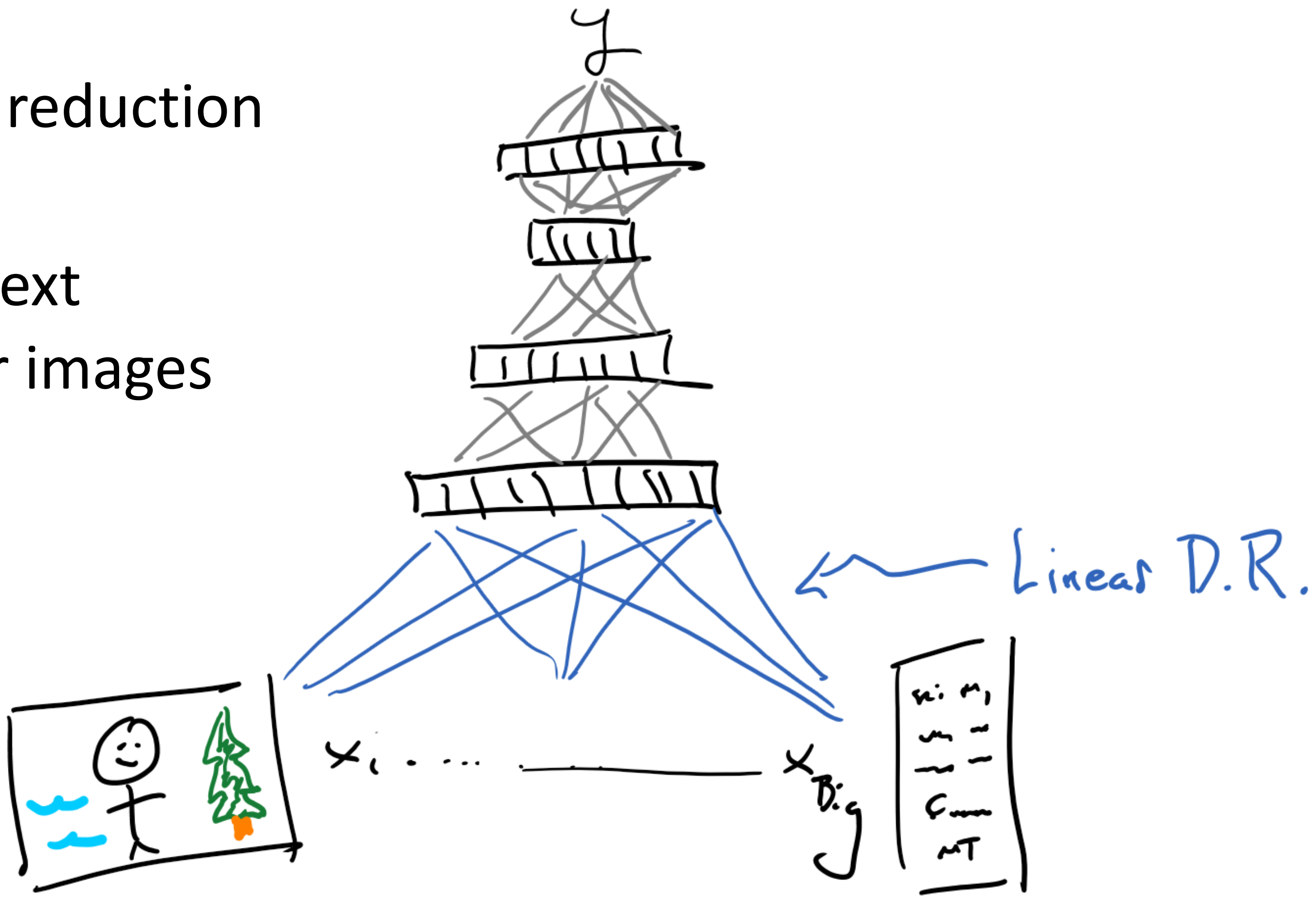
Deep nets are not really sieves

1st layer is a big dimension reduction

e.g.,

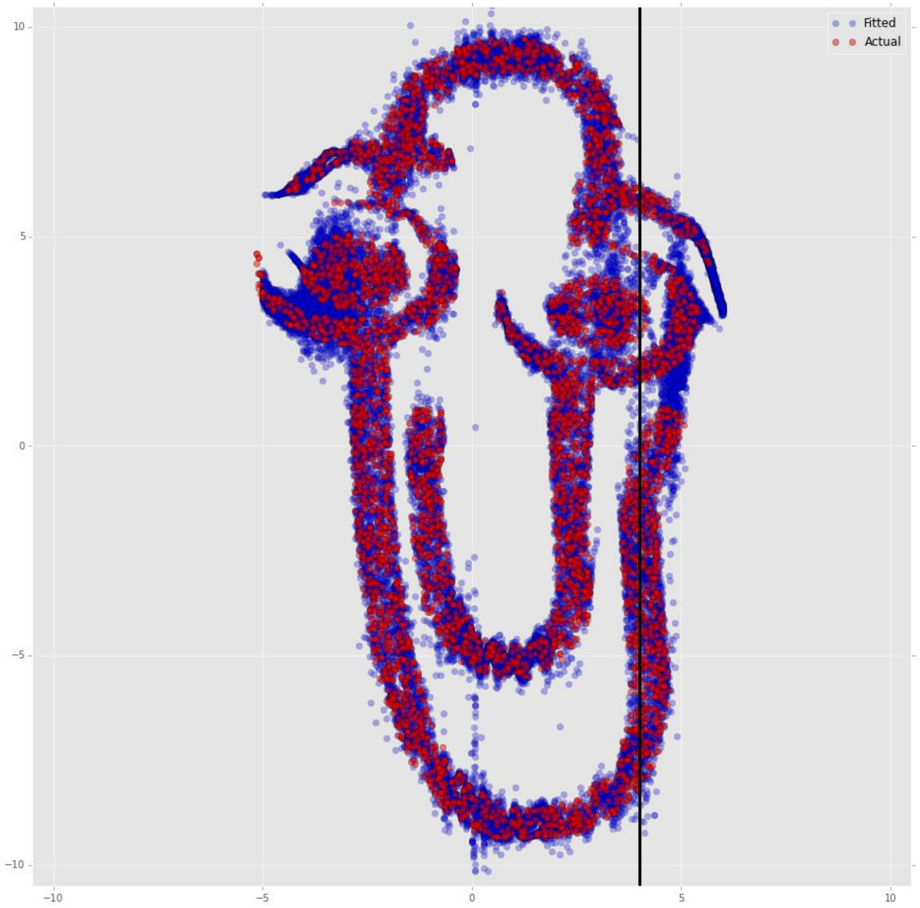
- word embedding for text
- matrix convolution for images

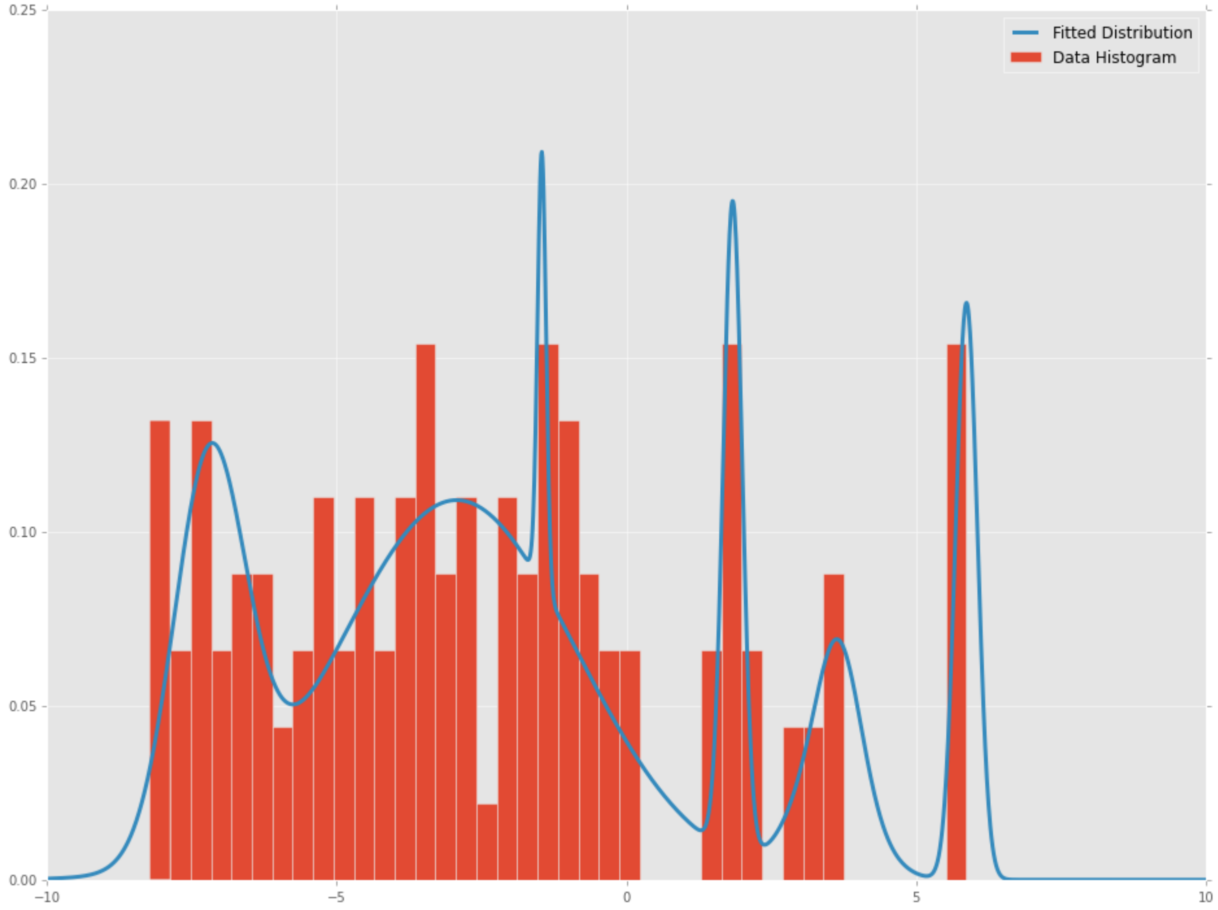
We need to study these…



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

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

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

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

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

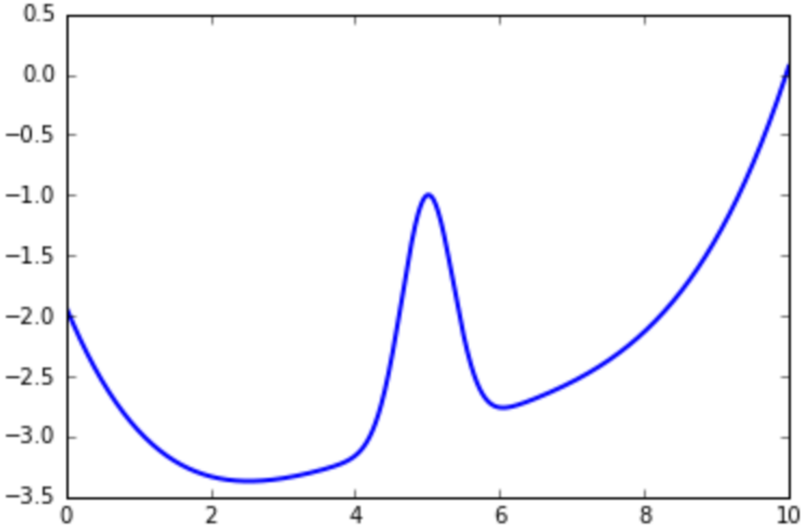
$$y = 100 + s\psi_t + (\psi_t - 2)p + e,$$

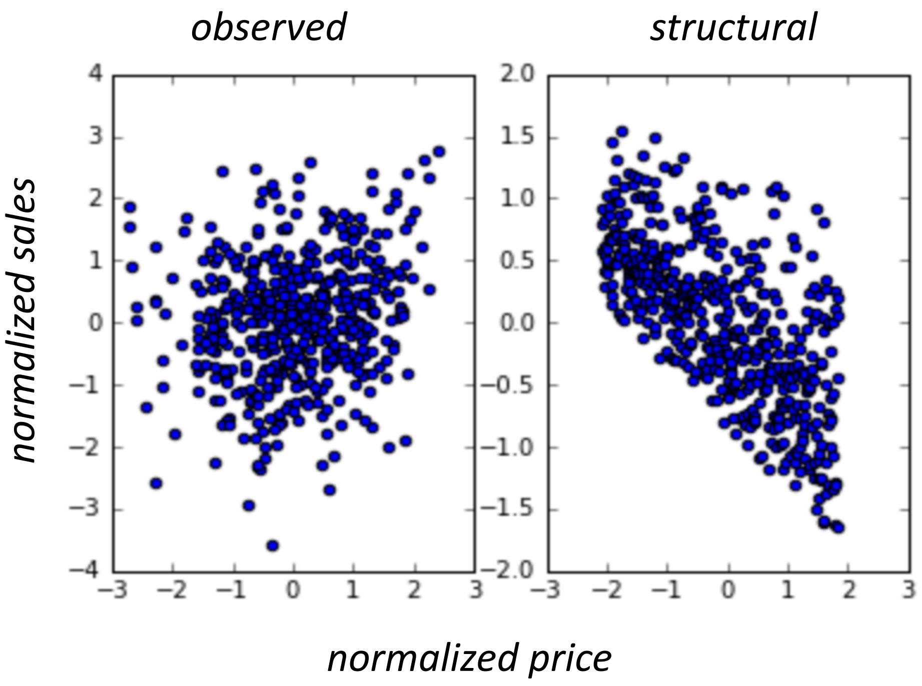
$$p = 25 + (z + 3)\psi_t + v$$

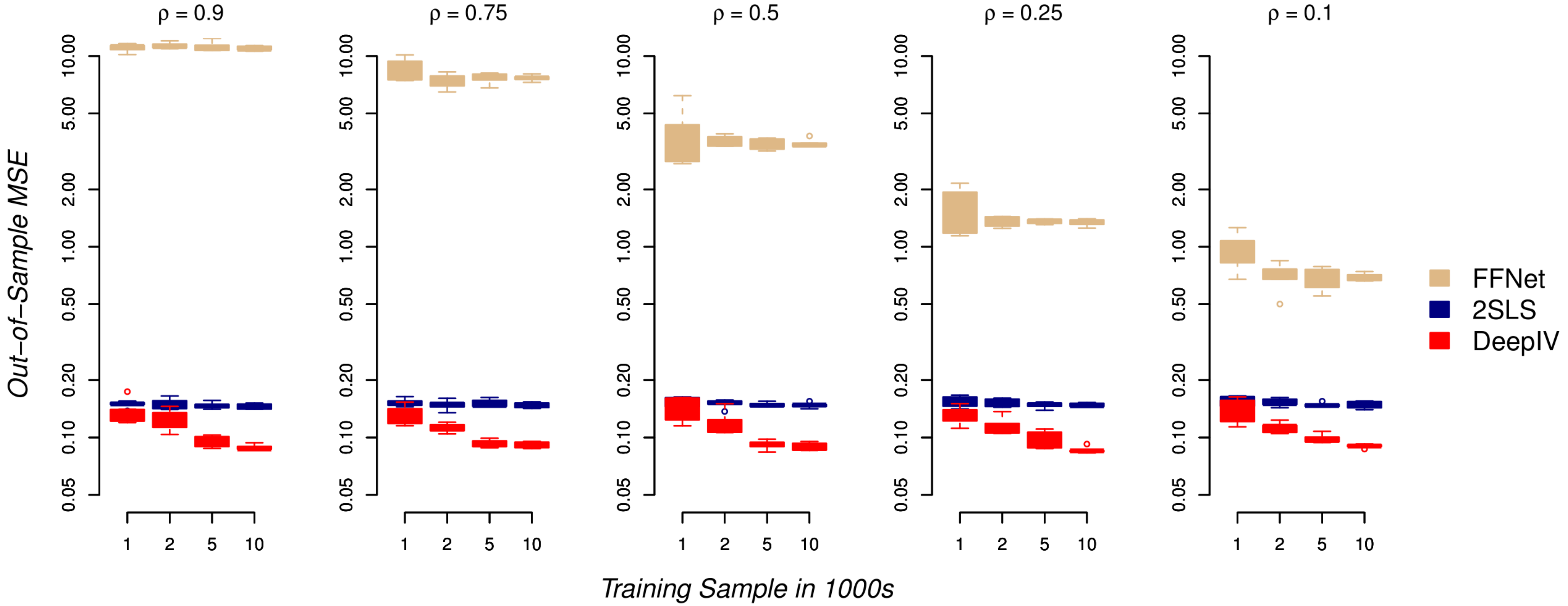
$$z,\ v \sim \mathrm{N}(0,1) \ \text{ and } \ e \sim \mathrm{N}(\rho v, 1-\rho^2),$$

'time' dependent prices, sensitivity, utility

Customer 'type' 1-7 impacts demand

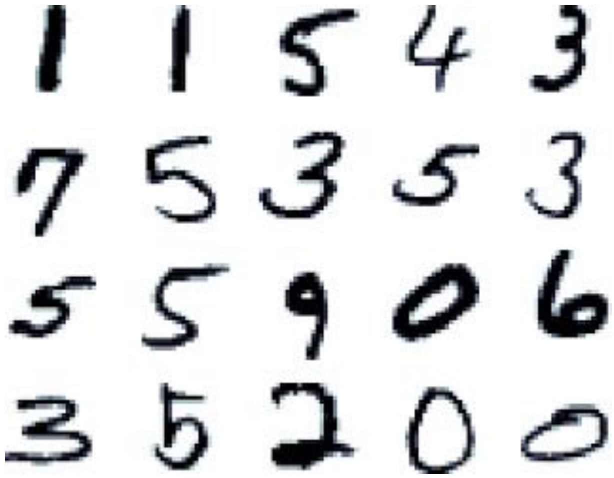


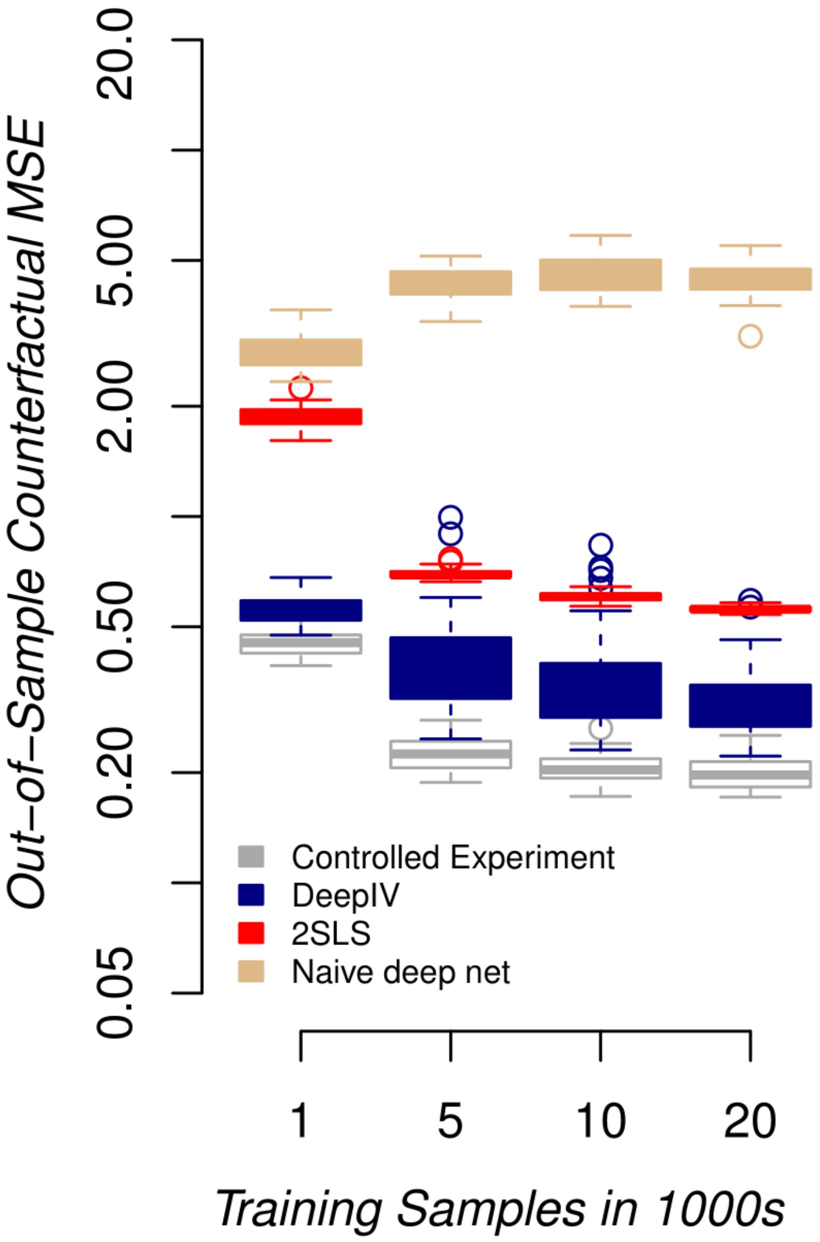




ρ = 0.9
ρ = 0.75
ρ = 0.5
ρ = 0.25
ρ = 0.1
Out-of-Sample MSE
10.00
5.00
2.00
1.00
0.50
0.20
0.10
0.05
1
2
5
10
Training Sample in 1000s
FFNet
2SLS
DeepIV

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

Post-net 2SLS coefficients are $\hat{\beta} = (\overline{H}'H)^{-1}\overline{H}'y$ 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

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

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 $w \sim \mathrm{N}(0, \lambda^{-1})$,

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

(see also Gal and Ghahramani)

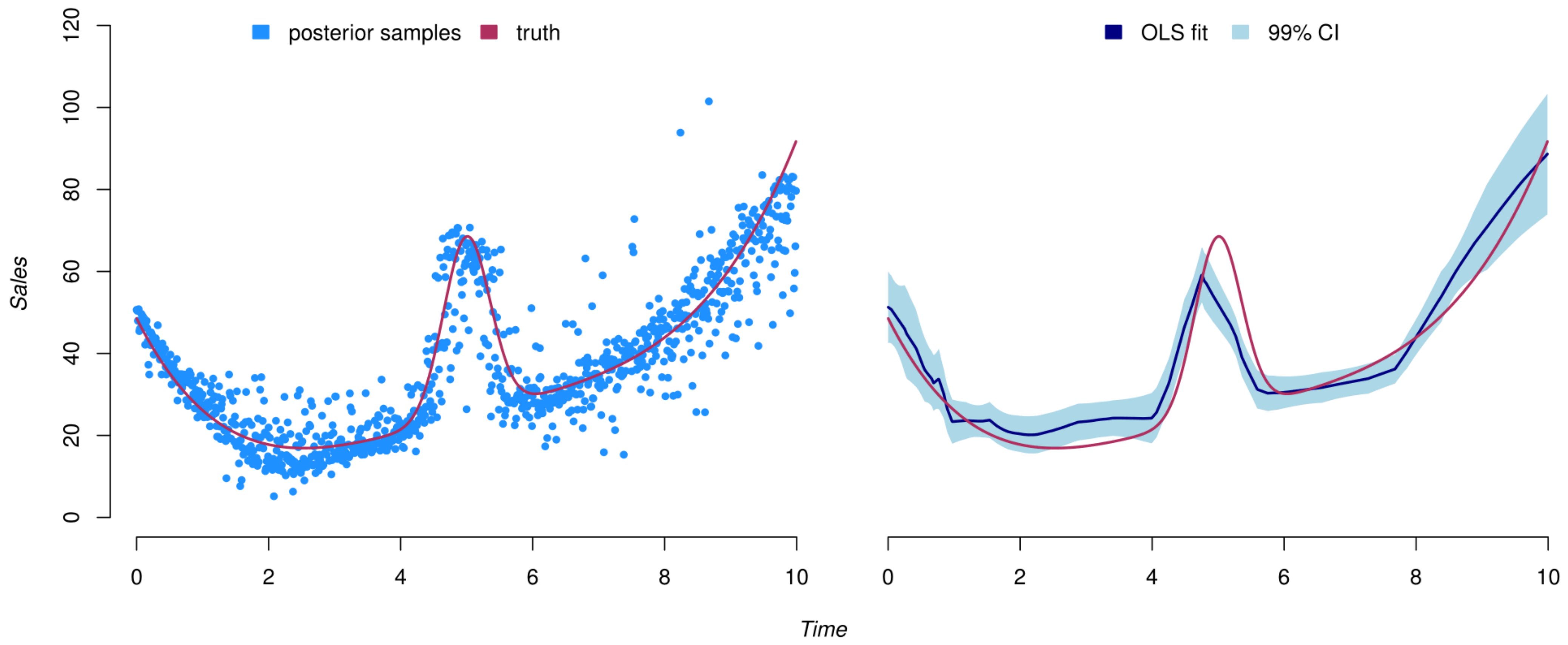
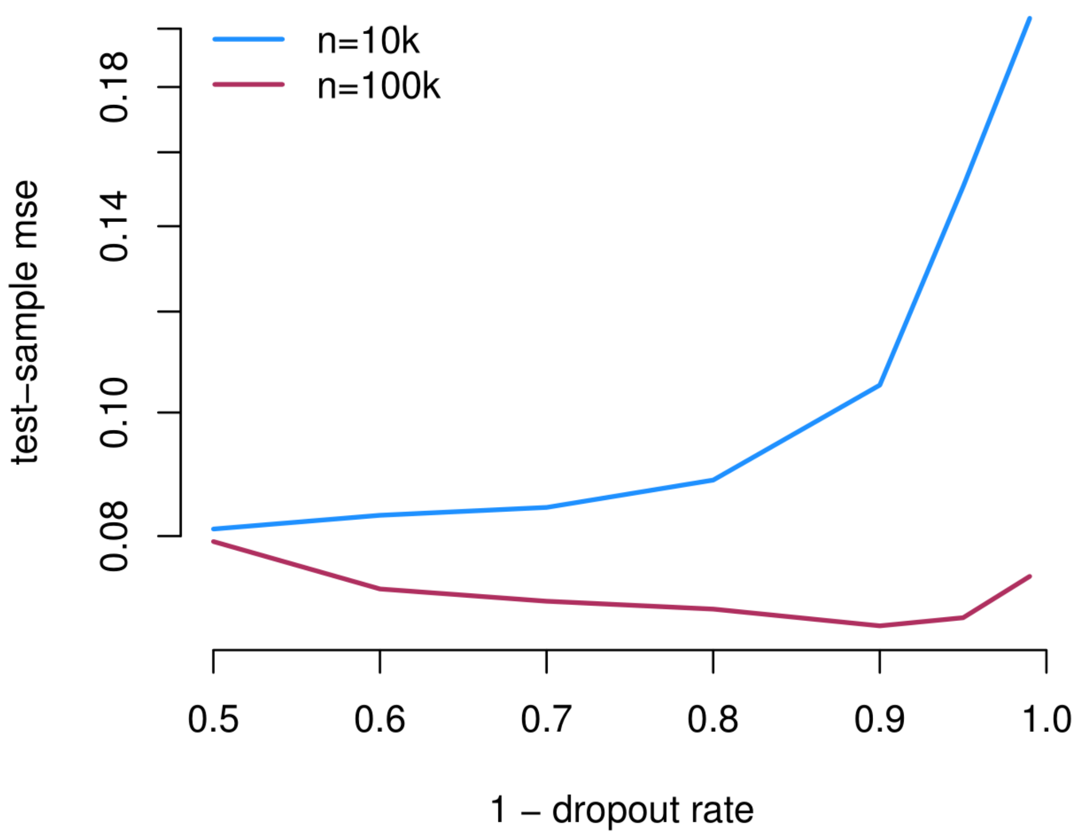
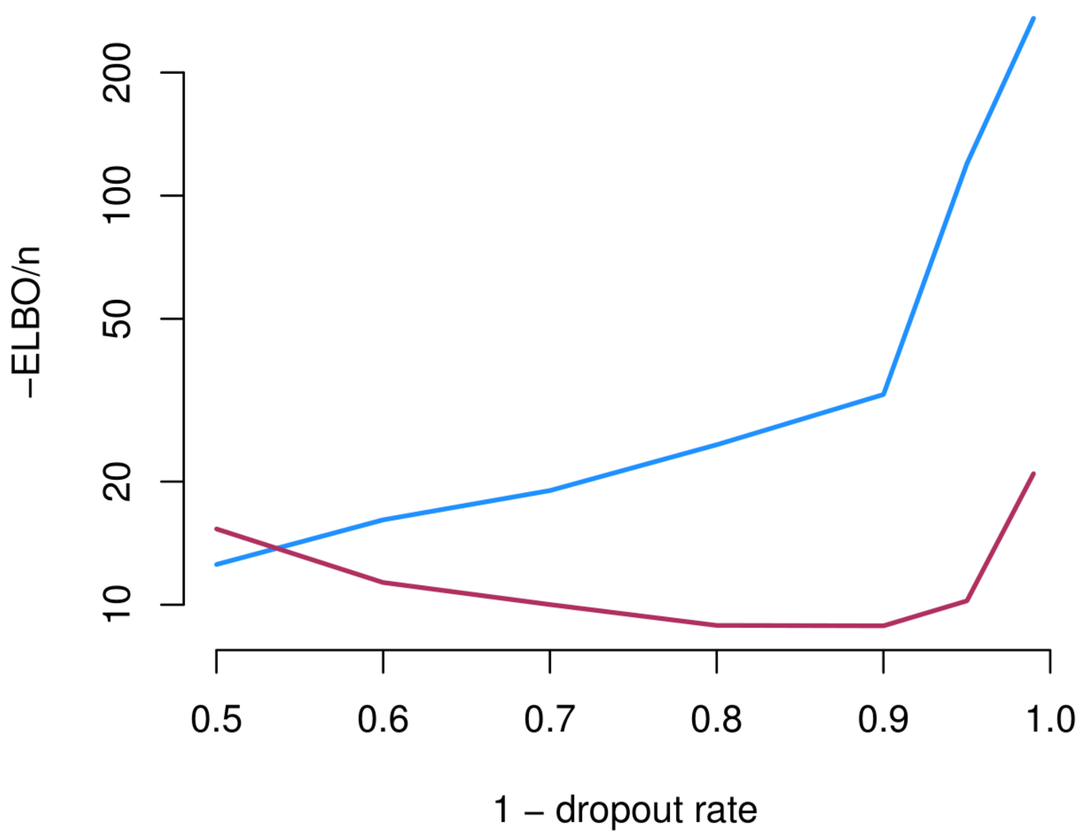


Figure 3: Bayesian (left) and Frequentist (right) inference for a central slice of the counterfactual function, taken at the average price and in our 4^{th} customer category. Since the price effect for a given customer at a specific time is constant in (27), the curves here are a rescaling of the customer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

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





Ads Application

Taken from Goldman and Rao (2014)

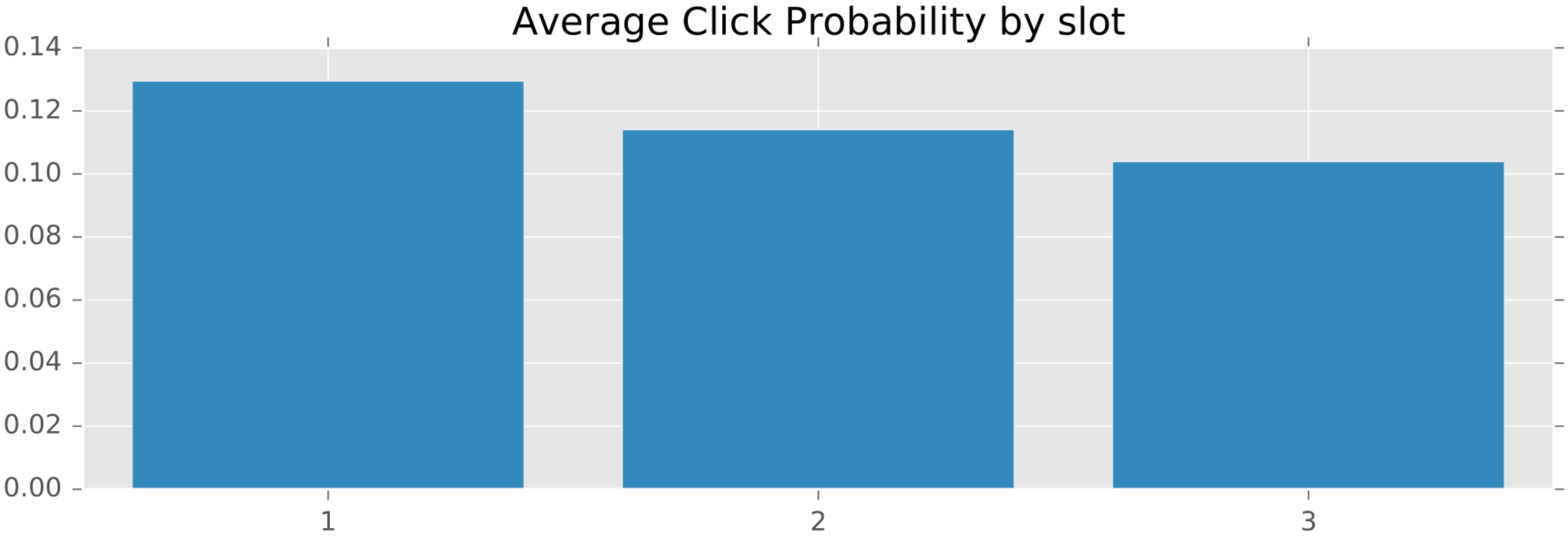
We have 74 mil click-rates over 4 hour increments for 10k search terms

Treatment: ad position 1-3

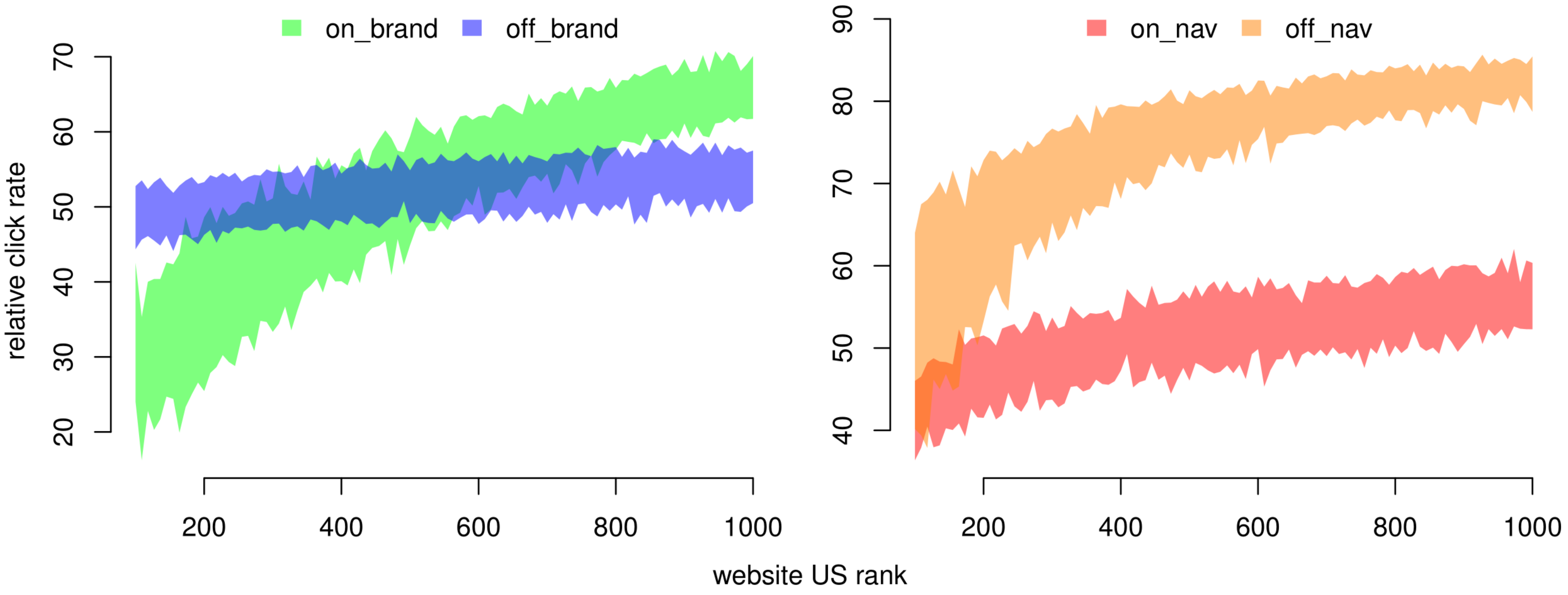
Instrument: background AB testing (bench of ~ 100 tests)

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