Economic AI

ABFER 2017 – Singapore

Masterclass Part I

Matt Taddy

ABFER Econometrics Masterclass

Part 1: Economic AI

How ML can be useful in Economics and Finance

Part 2: An ML Primer

Fast and flexible modeling without overfit

Economic AI breaks complex systemic questions into structures of ML tasks

Economists Study Systems

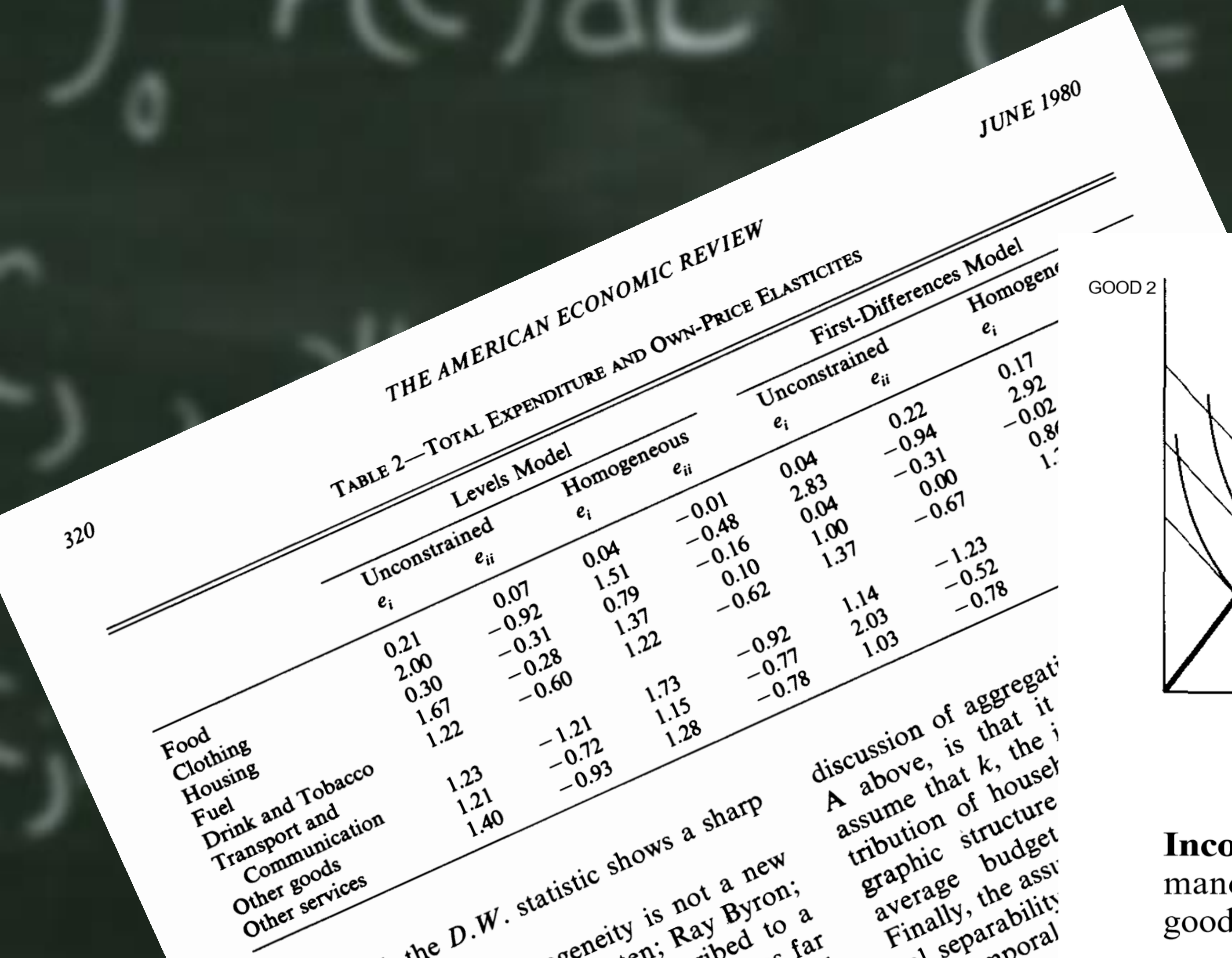
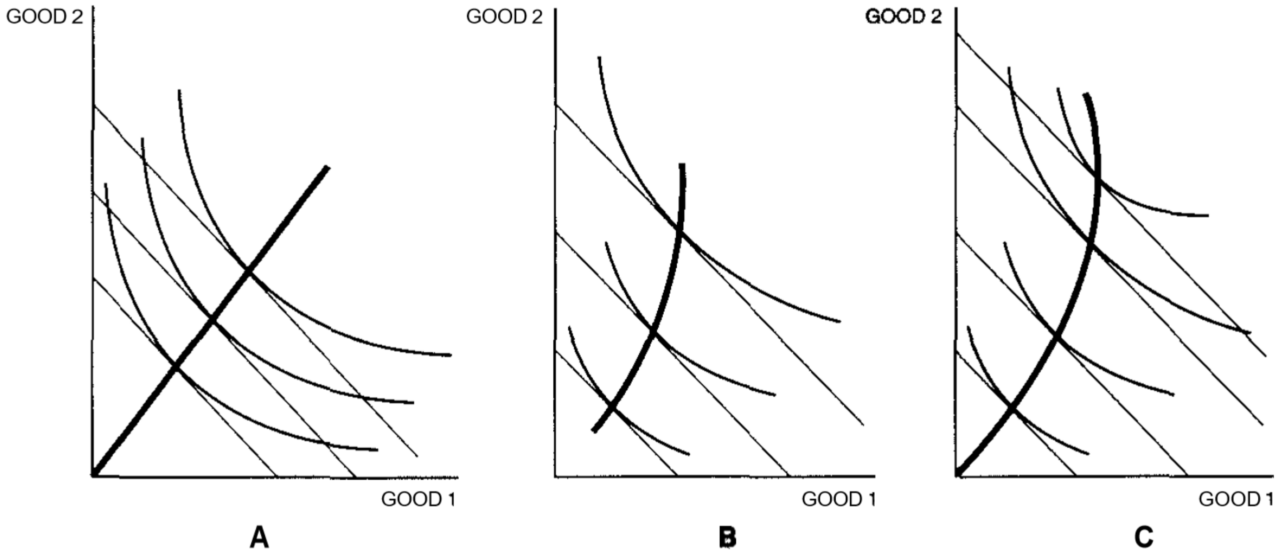


TABLE 2—TOTAL EXPENDITURE AND OWN-PRICE ELASTICITES

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

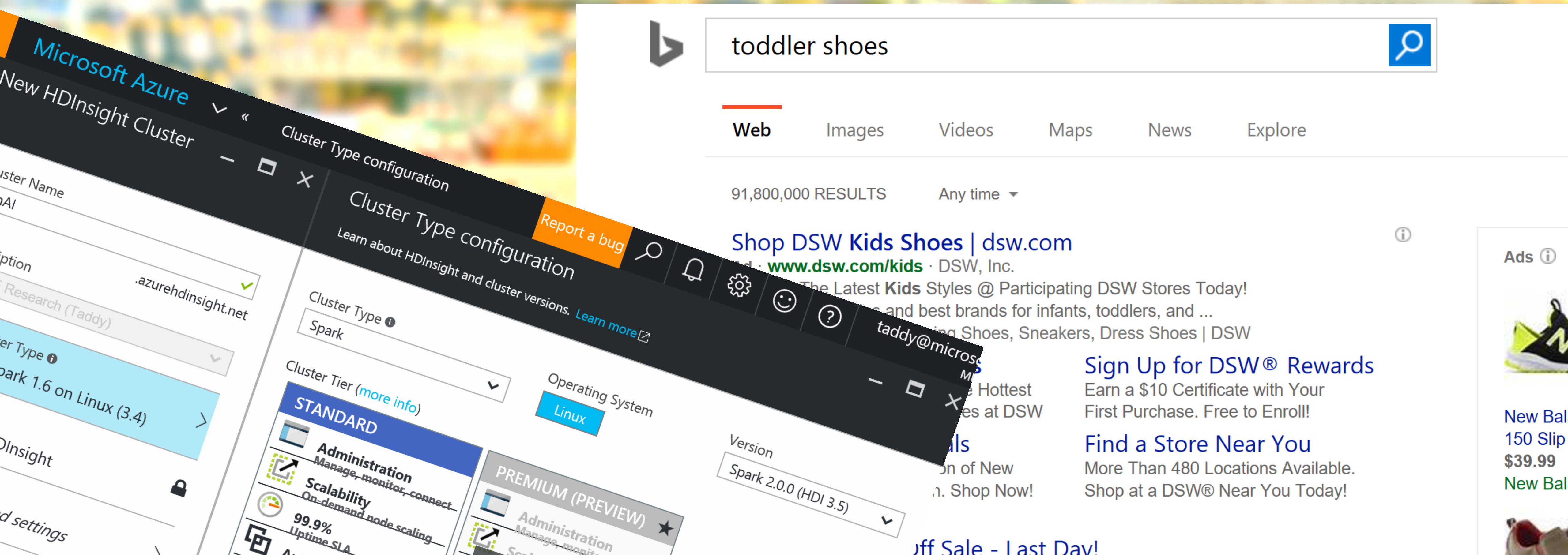
the D.W. statistic shows a sharp
geneity is not a new
ten; Ray Byron;
ribed to a
f far

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.

And they are in high demand



Inference about systems is 'causal'

Good decisions are a result of causal understanding (or luck!)

Pricing: Ex. How much will sales rise *if I lower prices?*

Policy: Ex. Do people work less *because* of disability insurance?

Marketing: Ex. What is the *causal ROI* from this ad campaign?

Causal reasoning is absent from most AI Systems.
Why? Because it is notoriously difficult…

Applied econometrics (via experimentation)

Example

- Question: what is the impact of sponsored search ads on revenue?
- Confound: revenue changes in time with other known and unknown factors
- Experiment: do an 'AB test', randomly turning off ads for certain users/markets

Limitations

- Very expensive and politically difficult to run [big/long] experiments
- Design and analysis still requires high level of sophistication

Applied econometrics (mostly harmless version)

Example

- Question: what is the impact of going to charter school on college success?
- Confound: students who seek charter schools are different *to begin with*
- Experiment: compare students with high and low scores in enrollment lottery

Limitations

- Requires a high level of sophistication and a lot of luck
- Too cute: these natural experiments occur in special settings

Applied econometrics (may be hazardous)

Example

- Question: what is the impact of going to charter school on college success?
- Confound: students who seek charter schools are different *to begin with*
- Experiment: compare kids who are similar on observables (income, race, …)

Limitations

- Results are very sensitive to the model specification
- Selection of the control variables is subjective and hugely labor intensive

Economist as applied econometrician

Measurement in [micro]econometrics today has a mix of

a. A small number of designed experiments (often from tech)
b. Natural experiments via fortunate `upstream randomization'
c. Natural pseudo-experiments via instruments of varying quality
d. Analyses that control for a set of hand-picked observables

(c+d overlap: many IV arguments rely on conditional exclusion)

[L]ATEs are in forefront, with some hand-picked heterogeneity

This is enough work to be a full-time job!

Machine Learning can automate and accelerate tasks in these applied econometric workflows

Example: Heterogeneous Treatment Effects

A typical A/B trial breaks users 'i' into

- Control group who sees existing website, say $d_i = 0$
- Treatment group who sees an altered website, say $d_i = 1$

e.g., eBay might change the size of pictures

$i \in \text{control} : d_i = 0$

$i \in \text{treatment} : d_i = 1$

and want to know the change in revenue $y_i(d_i = 1) - y_i(d_i = 0)$

What is HTE?

Different units [people, devices] respond differently to some treatment you apply [change to website, marketing, policy].

It exists.

We know $\boldsymbol{x}_i$ about user i.

- Their previous spend, items bought, items sold...
- Page view counts, items watched, searches, ...
- All of the above, broken out by product, fixed v. auction, ...

Can we accurately measure heterogeneity: index it on $\boldsymbol{x}_i$?

The usual A/B analysis workflow:

1. Calculate the ATE as $\bar{y}_1 - \bar{y}_0$
2. Repeat for subgroups defined by covariate regions r

$\mathrm{ATE}(r) = \bar{y}_1(r) - \bar{y}_0(r)$ where $\bar{y}_d(r) = \hat{\mathrm{E}}[y_i \mid d_i = d,\ \boldsymbol{x}_i \in r]$

In #2, task of selecting 'interesting' r is subjective and laborious.
It is also a 'pure' prediction problem. ML can automate this task.

What is Machine Learning?

ML combines flexible semiparametric models with fast estimation algorithms and tools that ensure out-of-sample validity.

Supervised ML is trained to minimize loss on a small set of outcomes ‘y’. Unsupervised ML loss is defined over all variables.

ML discovers patterns in the DGP.

It is *backwards looking:* predicts a future that behaves like the past.

This is what I call a pure prediction problem.

For HTE analysis we want to predict $\mathrm{E}[y_i(1) - y_i(0) \mid \boldsymbol{x}_i]$. This is a pure prediction problem if the marginal DGP for $\boldsymbol{x}$ is unchanging.

Say q is the probability of being in the treatment group ($d = 1$).

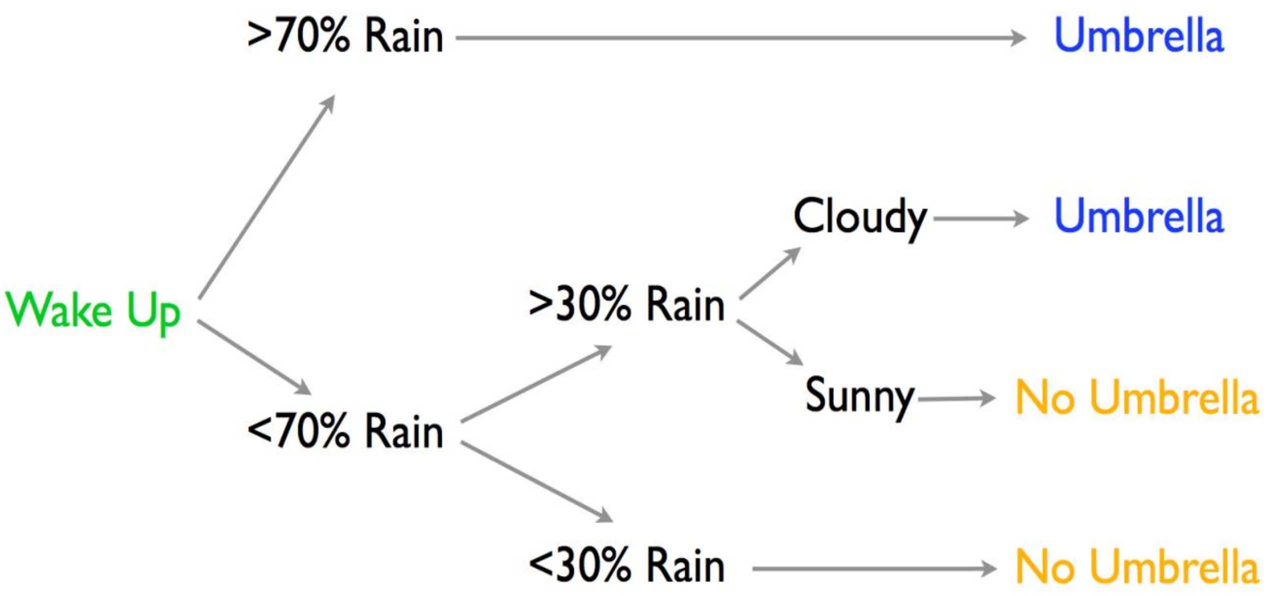
Following Athey + Imbens 'causal trees', define

$$y_i^\star = y_i \frac{d_i - q}{q(1-q)} = \begin{cases} y_i/(1-q) & \text{if } d_i = 0 \\ y_i/q & \text{if } d_i = 1 \end{cases}$$

So that $\mathrm{E}[y_i^\star \mid \boldsymbol{x}_i] = \mathrm{E}[y_i(1) - y_i(0) \mid \boldsymbol{x}_i]$.

(usual propensity scoring argument; perhaps not an efficient estimator)

Regression Trees and Forests



Trees work well for modeling interaction and nonlinear effects for a low dimensional set of covariates ($\lesssim \sqrt{n}$). They are perfect for automating the usual `let's look at some bins' workflow of HTE.

Regression Trees and Forests

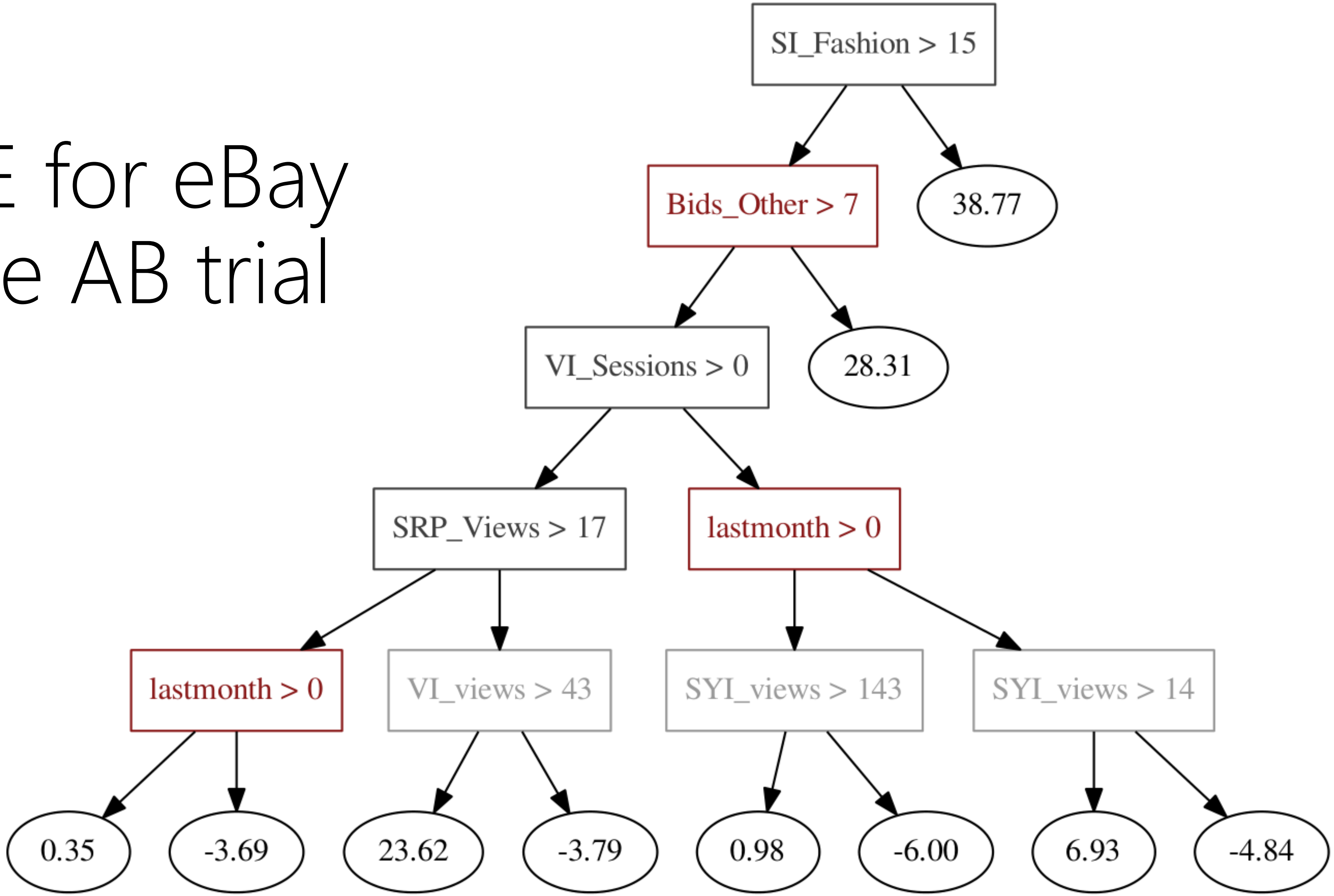
CART: greedy growing with optimal splits

Given node $\{\mathbf{x}_i, y_i\}_{i=1}^n$ and DGP weights $\boldsymbol{\theta}$, find x to minimize

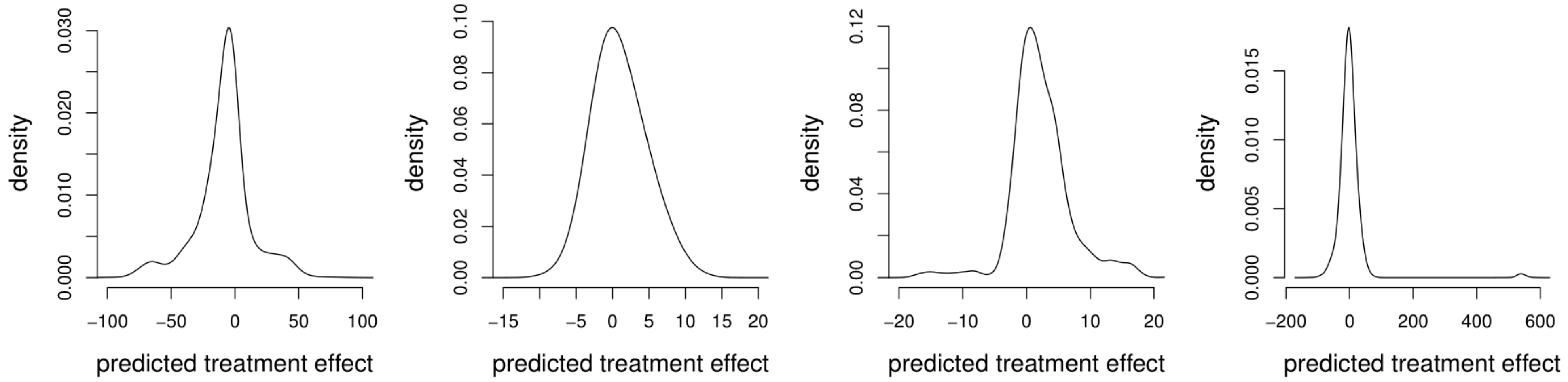
$$|\boldsymbol{\theta}|\sigma^2(x, \boldsymbol{\theta}) = \sum_{k \in \mathrm{left}(x)} \theta_k (y_k - \mu_{\mathrm{left}(x)})^2 + \sum_{k \in \mathrm{right}(x)} \theta_k (y_k - \mu_{\mathrm{right}(x)})^2$$

CART fits are *unstable*, so we prefer to bootstrap and average (a forest)

CART HTE for eBay image-size AB trial



Better: fit many trees to data resamples and look at the *distribution* of predicted treatment effects at individual $\boldsymbol{x}_i$



Better still: a `gradient forest' (Athey, Wager, Tibshirani) where CART on $y^\star$ is replaced with splitting to maximize TE heterogeneity across nodes.

See also Taddy, Gardner, Chen, Draper for eBay eg and these Bayesian forests, and Wager+Athey for honest trees and frequentist properties of forests

Why ML HTE?

The practice of sub-group and HTE analysis was ripe for ML automation, and trees/forests were the perfect tools for the job.

The results are not structural – we don't know why $\mathrm{ATE}(r)$ is higher/lower than $\mathrm{ATE}(r')$, but this happens for the observed $p(\boldsymbol{x})$

This is consistent with common practice around CATEs [Imbens] and it works for e-commerce apps: setting d_i won't change $p(\boldsymbol{x}_i)$

Bonus: ML does more than save time: it adds objectivity (avoids p-hacks).

Why not ML?

Many problems are not pure prediction problems.

e.g., 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 pure prediction. It doesn't work for counterfactuals

What happens if I change p independent of e?

Example: estimating short-term elasticities

Quantities y_i sold at prices p_i in scenarios indexed by $\boldsymbol{x}_i$

If I offer a Δ% discount at $\boldsymbol{x}_i$, what will be my expected sales?

This is a counterfactual question.

A [very] simple reduced form has, for products 'j',

$$\log y_{ij} \;=\; \alpha_{ij}(\boldsymbol{x}_i) \;+\; \gamma_j p_{ij} + e_{ij}$$

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

But it's a system!

Where does price come from?

Demand system might have

$$\log p_{ij} \;=\; \varphi_{ij}(\boldsymbol{x}_i) + \psi_j \log\, q_{ij}^{\star} \;+\; \nu_{ij}$$

and be in equilibrium when $q_{ij}^{\star} = q_{ij}$

Both prices and sales are responding to underlying demand

Also an issue: demand for j depends on substitutes and complements

Beer Elasticity

`smallbeer.csv` is a small dataset of beer transactions from a single grocery store. We have 400 different SKUs over 35 Weeks

```
> head(beer)
  item                      description week price units
1  242 10 BARREL APOCALYPSE IPA 6PK          00  9.39     4
2  244 10 BARREL BREWING PRAY FOR SNOW WINT  00  9.39     1
3  250 2 TOWNS CRISP APPLE CIDER             00  4.69     1
4  298 21ST AMENDMNT BREW FREE IPA 6PK       00  8.49     1
5  270 AMERICAN BLONDE                       00  3.79     2
6  272 AMERICAN CABOOSE STOUT 22             00  3.79     2
> nrow(beer)
[1] 7969
>
```

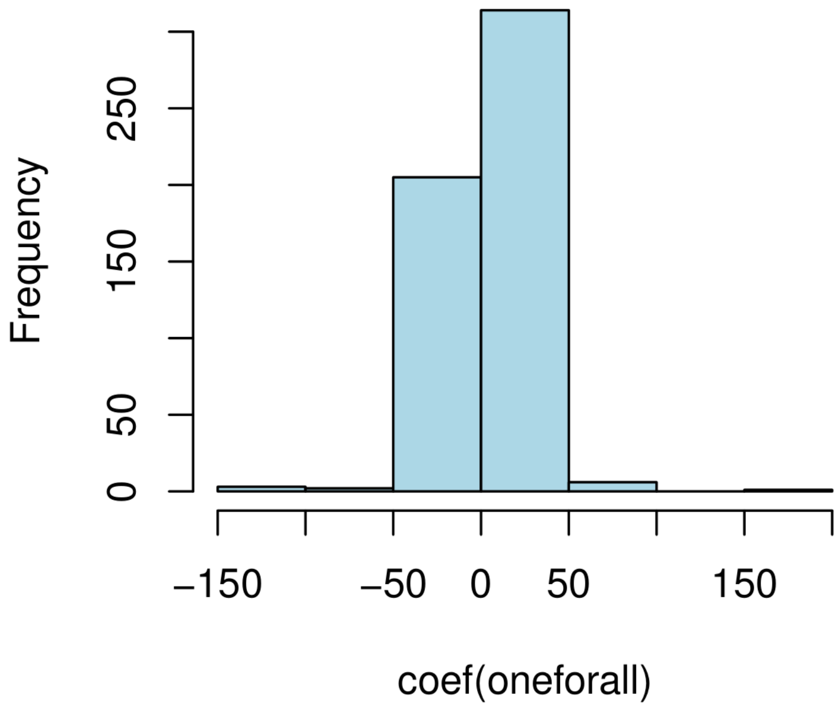
Beer Elasticity

```
> ( allforone <- lm(log(units) ~ log(price), data=beer) )

Call:
lm(formula = log(units) ~ log(price), data = beer)

Coefficients:
(Intercept)   log(price)
     1.3499      -0.2346

> oneforall <- lm(log(units) ~ log(price)*item, data=beer)
>
```



A single shared elasticity gives tiny -0.23

Separate elasticity for each gives wildly noisy zeros

Beer Elasticity

Not enough price variation to estimate individual elasticities.

We could group the products together using brand, pack, etc.

That seems like a lot of boring work.

Instead, we can featurize the products from their text description.

Say $w_{ik} = 1$ if word k is in description for beer i.

Then we could have something like

$$\log y_i = \alpha_i + \delta_t + \boldsymbol{w}_i' \boldsymbol{\tau} + (\rho_t + \boldsymbol{w}_i' \boldsymbol{\gamma}) \log p_i$$

Beer Elasticity

By the way, `tokenizing' text like this is really easy

```
> # parse the item description text
> library(tm)
> descr <- Corpus(VectorSource(as.character(beer$description)))
> xtext <- DocumentTermMatrix(descr)
> xtext <- sparseMatrix(i=xtext$i,j=xtext$j,x=as.numeric(xtext$v>0),
+                  dims=dim(xtext),dimnames=dimnames(xtext))
> |
```

Then, for example, `American IPA` becomes a row in sparse matrix

American	Canadian	…	IPA	Light	…
1	0	…	1	0	…

See Gentzkow, Kelly, Taddy (2017) for a *text-as-data* review article

Beer Elasticity

$$\log y_{it} = \alpha_i + \delta_t \ + \boldsymbol{w}_i'\boldsymbol{\tau} + (\rho_t + \boldsymbol{w}_i'\boldsymbol{\gamma})\log p_{it} + \varepsilon_{it}$$

Now we've got a stack of parameters. MLE gives garbage.

No problem for ML, right? Just throw everything in a lasso, right?

(The lasso minimizes deviance plus an L1 penalty on coefficients)

Not so fast…

Beer Elasticity

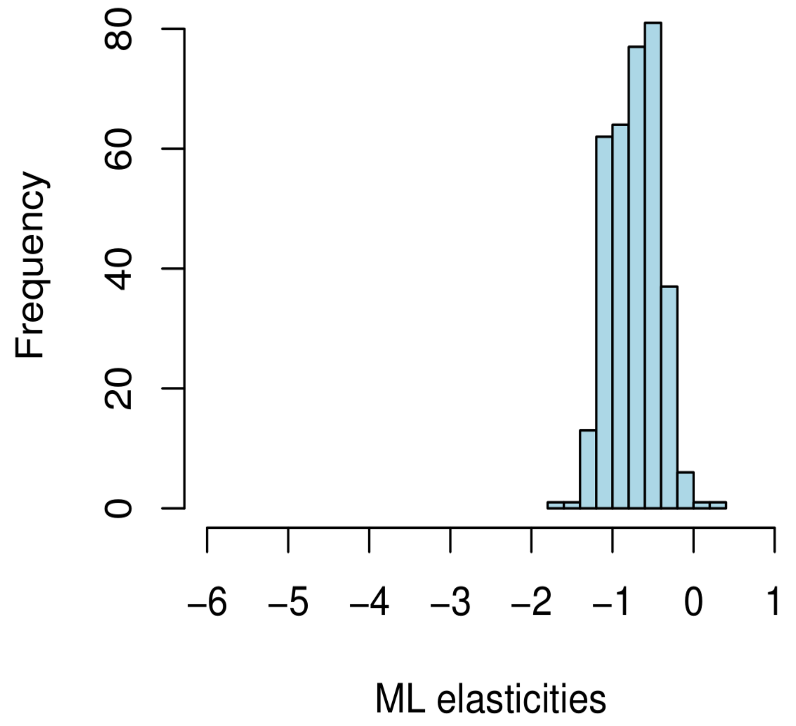
```
> xx <- cBind(xweek, xitem, xtext)
> xtreat <- cBind(1,xtext,xweek)
> dim(xx)
[1] 7969  827
> dim(xtreat)
[1] 7969  484
> naiveml <- gamlr(x=cBind(xtreat*log(beer$price), xx), free=1,
+                y=log(beer$units), lmr=1e-4, standardize=FALSE)
>
```

We'll unpack `gamlr` and lassos tomorrow.

Think of it as a generic 'ML prediction machine'

The elasticities are not as crazy as before, but they are too small (we know from experiments that they usually live between -5 to -1).

Our issue: this is not a pure prediction problem



Orthogonal Machine Learning

The naïve ML conflates two problems:

- selecting controls and predicting the response conditional upon controls.

Chernozhukov+ 2016 Double ML is a nice synthesis of ideas on this issue.
It builds on BCH 2013/14, Newey 1994, and even Neyman 1979.

Basic Idea:
Estimate nuisance functions that are orthogonal to γ in its conditional score.
Then estimation for γ is robust to slow learning on these nuisance functions.

Orthogonal Machine Learning

A simple partially linear formulation

1. $y_i = p_i \gamma + g(\boldsymbol{x}_i) + v_i, \quad \mathrm{E}[v_i | \boldsymbol{x}_i, p_i] = 0$
2. $p_i = h(\boldsymbol{x}_i) + \nu_i, \quad \mathrm{E}[\nu_i | \boldsymbol{x}_i] = 0$

Estimating #1 directly solves conditional $\sum_i \psi_{naive}(\hat{\gamma}; y_i, \boldsymbol{x}_i, p_i, \hat{g}) = 0$ where

$$\psi_{naive} = [y - p\,\hat{\gamma} - \hat{g}(\boldsymbol{x})]p$$

The problem:

$$\mathrm{E}[\,\partial_g \psi_{naive}\,]\Big|_{g=g_0} \neq 0$$

⇒ you need to do a really good job on $\hat{g}$, which is unrealistic for HD g

Orthogonal Machine Learning

Instead, estimate two *nuisance* functions

$$f(\boldsymbol{x}) = \mathrm{E}[y|\boldsymbol{x}] = \mathrm{E}[p|\boldsymbol{x}]\gamma + g(\boldsymbol{x})$$

$$h(\boldsymbol{x}) = \mathrm{E}[p|\boldsymbol{x}]$$

Then γ can be estimated to solve a conditional score that sums over

$$\psi_{\perp} = \left[y \; - \; \hat{f}(\boldsymbol{x}) \; - \left(p - \hat{h}(\boldsymbol{x})\right)\right]\left(p - \hat{h}(\boldsymbol{x})\right)$$

Which has the property that $\mathrm{E}\left[\;\partial_{f,h}\psi_{\perp}\right]$ vanishes at $f \; = f_0, h \; = h_0$.

Orthogonal ML for Pricing

Price sensitivity estimation breaks into two ML tasks:

1. Predict prices from the demand variables: $\boldsymbol{p} \sim \boldsymbol{x}$
2. Predict sales from the demand variables: $\boldsymbol{y} \sim \boldsymbol{x}$

Plus a final regression:

$$\left(\boldsymbol{y} - \widehat{\boldsymbol{y}}(\boldsymbol{x})\right) \sim \left(\boldsymbol{p} - \widehat{\boldsymbol{p}}(\boldsymbol{x})\right)$$

Estimated relationship is causal if $\boldsymbol{x}$ contains all demand info known to pricer

The final stage is just OLS for low-D γ, but we replace with 3rd ML step.

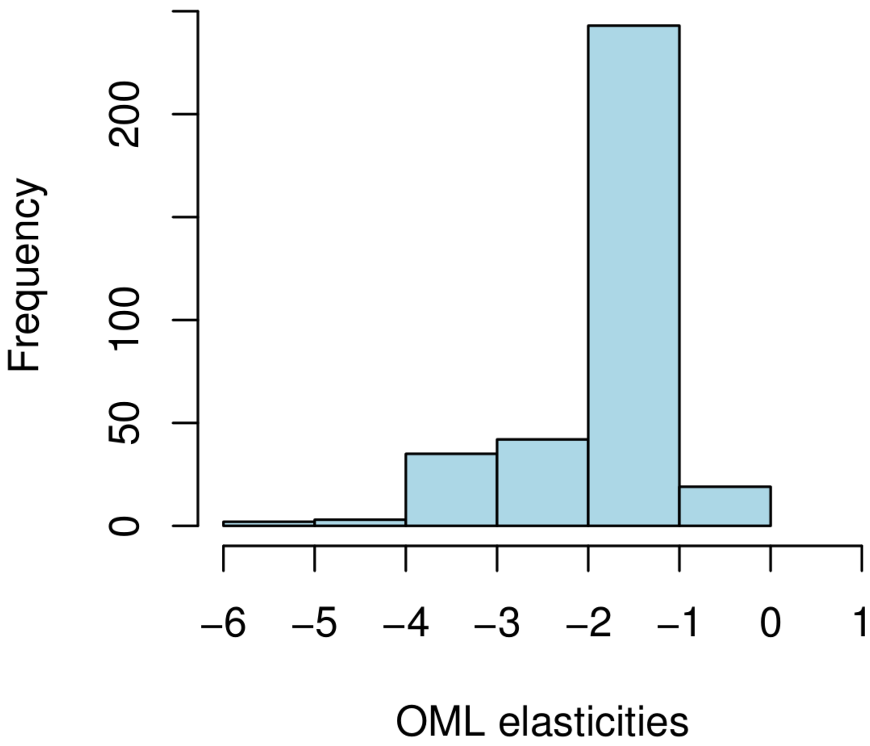
(For inference you can data split: use one sample for 1-2, another for step 3)

Orthogonal ML for Beer

For our beer data, $\boldsymbol{x}$ includes *item id*, *text tokens*, and *week*.

For the final regression, interact price residuals with *text tokens* and *week*.

```
# OML steps 1-2
pfit <- gamlr(x=xx, y=log(beer$price), lmr=1e-5, standardize=FALSE)
qfit <- gamlr(x=xx, y=log(beer$units), lmr=1e-5, standardize=FALSE)
# Calculate residuals
lpr <- drop(log(beer$price) - predict(pfit, xx))
lqr <- drop(log(beer$units) - predict(qfit, xx))
# Run 3rd ML step to get gammas
ofit <- gamlr(x=(lpr*xtreat), y=lqr, standardize=FALSE, free=1)
gams <- coef(ofit)[-1,]
```



There's no ground truth,

but these elasticities are in the expected range

Orthogonal ML for Beer

The text encodes a natural hierarchy

- Many beers are `IPA` or `Cider` or `Draught`
- But individual brands also load; e.g., `Pyramid` or `Elysian`
- And we find technical terms: `4pk` `6pk` `12pk` `24pk`

4pk	6pk	12pk	24pk
-0.2	-0.4	0.0	0.3

Most price sensitive

```
> names(sort(el)[1:5])
[1] "GUINNESSS DRAUGHT 6PK BTL                      "
[2] "GUINNESS DRAUGHT 4PK CAN                       "
[3] "PYRAMID OUTBURST IMP IPA 6PK                   "
[4] "ELYSIAN IMPORTAL IPA 6PK                       "
[5] "PYRAMID OUTBURST IMP IPA 12PK                  "
```

Least price sensitive

```
> names(sort(-el)[1:5])
[1] "2 TOWNS CRISP APPLE CIDER                      "
[2] "2 TOWNS BAD APPLE CIDER                        "
[3] "ATLAS BLKBRY APPLE CIDER                       "
[4] "D'S WICKED BAKED APPLE CIDER                   "
[5] "D'S WICKED GREEN APPLE CIDER                   "
```

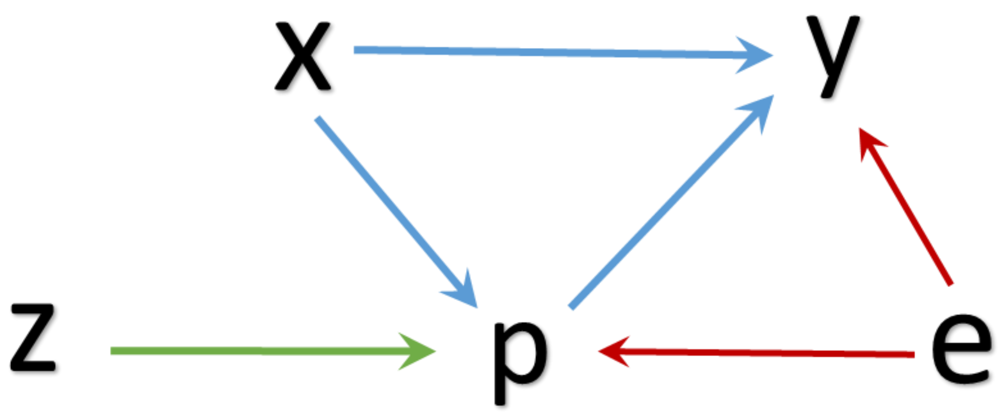
This is what econometricians do!

They break complex systems into measurable pieces.

Another common example: Instrumental Variables and 2SLS

Regress $p \approx z\tau$ then $y \approx \gamma(z\hat{\tau})$

Inference breaks into two regressions



The Econ AI distinction is that we're expanding beyond OLS

(you need to be careful and can't simply swap OLS for ML)

Deep IV

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

cf Newey+Powell 2003,

Use arbitrary ML to learn $\hat{F}$, then solve

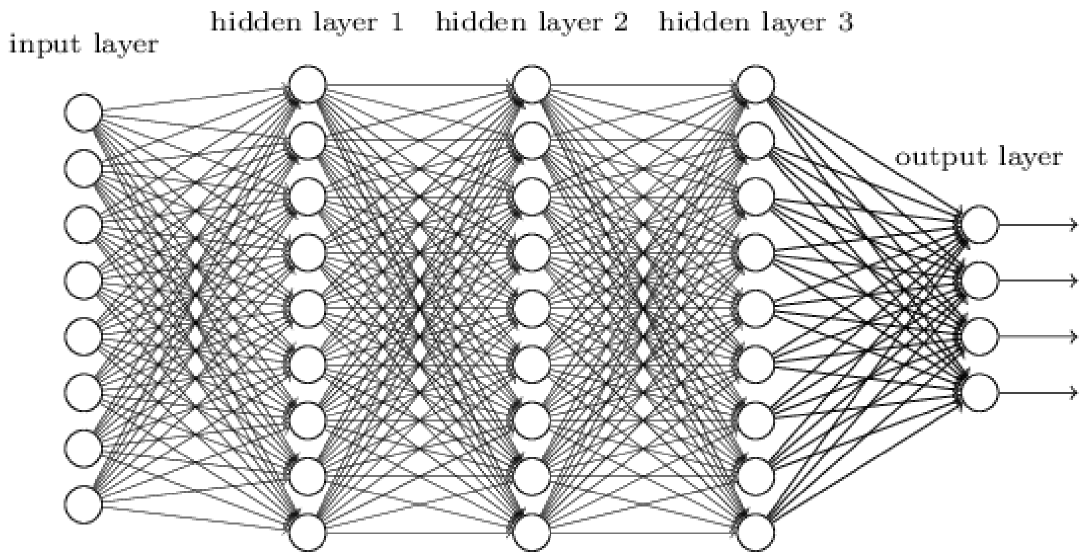
$$\min_{g \in G} \sum \left(y_i - \int g(p, x_i) d\hat{F}(p|x_i, z_i) \right)^2$$

Stochastic Gradient Descent is great for integral loss. And lots of IVs inside firms.
See Hartford/Lewis/Leyton-Brown/Taddy 2017

Deep Neural Networks

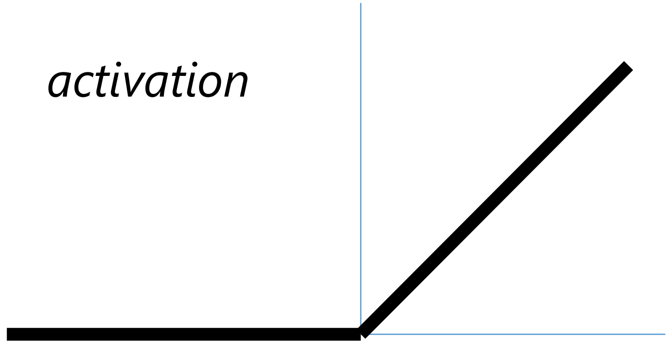
Massive number of parameters, mapping output of each layer to each node activation in the next

$$z_i^L \to h_k\left(\langle W_k^{L+1}, z_i^L \rangle\right)$$



Regularize

- deviance penalties $\lambda||W||$
- dropout training (zeros in grad)
- Stochastic gradient descent



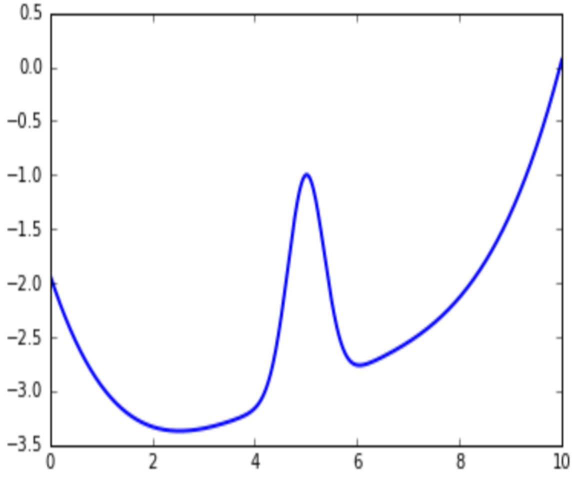
A pricing simulation

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

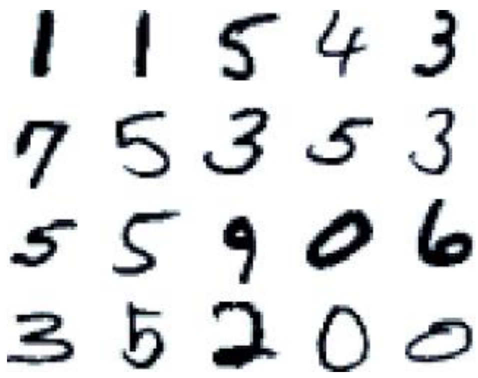
$$p = 25 + (z+3)\psi_t + v$$

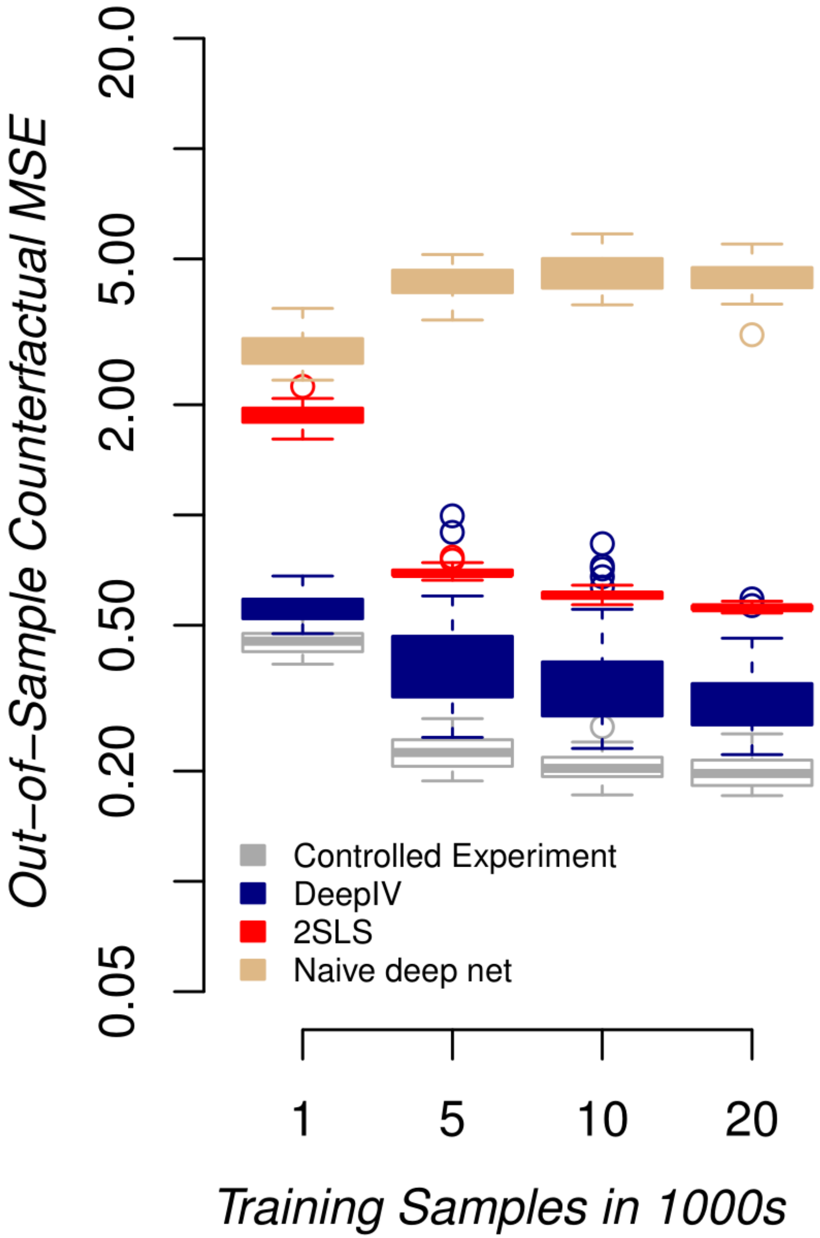
$$z,\ v \sim \mathrm{N}(0,1) \quad \text{and} \quad e \sim \mathrm{N}(\rho v, 1-\rho^2),$$

Time-dependent ψ_t



Customer type 's'





Reinforcement Learning (RL)

We've covered conditional ignorability and natural experiments

The remaining domain is designed experimentation

This is also due for acceleration and automation

Much of what we're doing is optimize under uncertainty

And A/B trials are a very inefficient way to optimize

An RL algorithm chooses what data to collect, and tries to minimize expected regret: $\sum_t (r_t - r_t^{best})$ where r_t is the *reward* at time t

RL and Bandits

One way of phrasing this has a bunch of action options (arms) a_k

Predict a_k at time t with probability

$$p_{tk} \approx \mathrm{pr}\big(r_t(a_k) > r_t(a_j)\forall j \neq i\big)$$

Get p_{tk} by featurizing scenario(t) with $\boldsymbol{x}_t$ and fitting $\hat{r}(a_k, \boldsymbol{x}_t)$
where you need to train $\hat{r}(a_k, \boldsymbol{x}_t)$ on $r_t^\star = \frac{r_t}{p_{tk_t}}$ (propensities again!)

This is an essential piece of any AI system.

Economic AI

Use economic structure to break questions into ML problems

Don't try to re-learn things you already know with an AI baby

Deep learning revolution: good low-dev-cost off-the-shelf ML

As the tools become plug-n-play, teams get interdisciplinary

The next big gains in AI are coming from domain context

A little understanding of ML can go a long way…