Using machine learning to infer and benchmark firm-firm trading networks

Jesse Tweedle

November 17, 2016

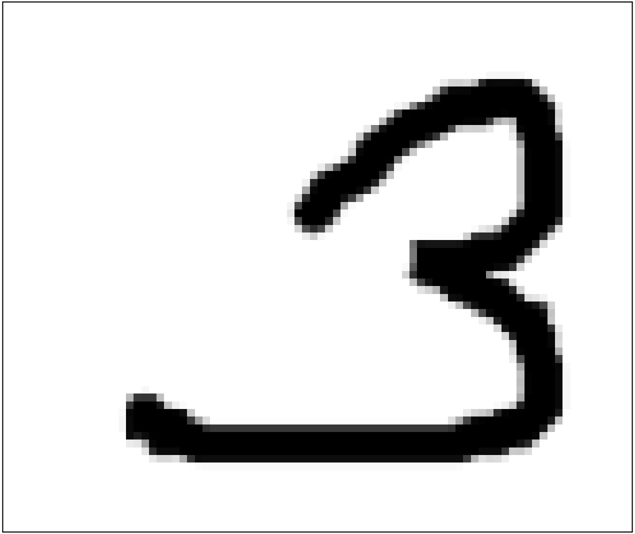
Broad introduction to machine learning

Algorithms that do statistics

- Like regular econometrics, but focused on getting $\hat{y}$ right instead of estimating $\hat{\beta}$
- Mainly classification; e.g., optical character recognition ("what letter is this?")
- Converted to regression-style question:
  - One observation = one image of a character
  - Dependent variables = each pixel in the image
  - Predict what the letter is (e.g., $\hat{y} =$ 'a')

Example: image recognition

Figure: A three. A prediction algorithm should produce $\hat{y} = 3$.



Why “learning”?

Easy reason—it does variable selection

Problem: you get firm data from CRA, you want to assign a NAICS code (y).

- There are tons of possible variables you could use to predict a firm’s industry.
- Possibly more variables than observations (which means can’t solve OLS problem $(X^TX)^{-1}X^Ty$

ML approach

Stick all the variables in a blender

- Algorithm starts with no predictors, adds them one-by-one
- “Learns” which ones are most important
- Use those coefficients to predict industries for out-of-sample firms

Some links / resources

Google's Open Source AI: TensorFlow

- `https://github.com/tensorflow/tensorflow`
- `https://www.tensorflow.org/`
- `http://playground.tensorflow.org/`

Facebook / Columbia / World Bank datasets

Analyze satellite imagery to create dataset on building density
`https://code.facebook.com/posts/596471193873876/open-population-datasets-and-open-challenges/`

Text: Elements of Statistical Learning

Hastie, Tibshirani, Friedman (2009)
`http://statweb.stanford.edu/~tibs/ElemStatLearn/`

Problem

Goal: study firm-firm trade

To understand

- supply chains and vertical integration
- intra-firm trade
- and more

Problem: we don't have firm-firm transaction data

But:

- we have lots of useful data
- and an idea of how to use it

Facts / needs

Facts: start with manufacturing

- $\approx$ 30k plants.
- $\approx$ 900m possible connections.
- $\approx$ 70 industries (IOIC)
- $\approx$ 200+ goods (detailed confidential IOCC)

Needs

- Data that indicates relationship between plants
- Method to identify / predict relationships: typical application of machine learning
- Method to benchmark to make sure it all adds up: unusual application of machine learning

Normal approach: idea

To achieve this:

- Data that indicates relationship between plants
- Method to identify relationships

Do this:

- Use supply-use/input-output tables
- Assume every firm-firm relationship is the same as the industry IO relationship

Normal approach: problems

Implies way too many plant-plant relationships

- Use goods-only, square IO table, DC level
- $\approx 70^2 = 4900$ possible connections
- $\approx$ actual connections
- 50% density—half of the possible connections are given
- Gets worse using full table: 90% density of 50000+ possible connections

Normal approach: problems

Plant and IO data may not be consistent

- Relationship may not be consistent with plant data; some plants may export more or less, some plants may import more or less
- Plant-plant relationships may not be consistent with industry IO

Problems: summary

- Don't correctly identify the links
- And still need to benchmark anyway

Machine learning approach

Step 1. Data

Use lots, from many sources.

Step 2. Identify / predict connections

- Typical, statistical, “classification” application of machine learning.
- For a possible firm-firm pair (i, j), predict whether there is a link or not.

Step 3. Benchmark (solving a huge linear programming problem)

- Unusual application of machine learning.
- Quick, cheap, accurate way to solve *sparse* system of equations.

Step 1. Data

For now, stick to one year, 2010. Easier on classifications / concordances / consistency across datasets.

- Surface Transportation File (STF): database of goods shipments; commodity, value, distance, postal code origin and destination
- ASM: industry, postal codes, provincial shipment breakdown, commodity inputs and outputs
- Input-output tables (IO): industry, commodity inputs and outputs; can calculate industry pair expenditures and direct requirements
- Inter-provincial trade flows (IPTF): trade by province origin and destination, and commodity

Step 2. Identify / predict connections

Want regular (logistic) regression form.

Write $y_{ij} = \mathbf{1}\{\text{firm } i \text{ buys from firm } j\}$, Estimate:

$$P(y_{ij} = 1 | X_{ij})$$

What is in X_{ij}?

- Entry in industry IO table or IPTF
- STF shipment from j's postal code to i's postal code
- j exports to i's province
- Entry in plant IO table defined by ASM commodity data
- Is there another possible producer of that good in the area of j, or another consumer in the area of i?
- Combinations of all of the above, et cetera

Step 2. Identify / predict connections

Solve problem

- This is classification problem: does the link exist (classify ‘yes’), or not (classify ‘no’)
- Many algorithms for this; I broadly divide into “things I understand” (all very similar to econometrics) and “deep learning” (e.g., neural networks).
- Called “supervised learning” because we know the things we want to predict, so we “supervise” the algorithm until it does what we want (predict firm-firm links).

Predict links $\hat{y}_{ij}$

And use them as input into next step. (Still a work in progress. For now, I’ll focus on benchmarking bit.)

Step 3. Benchmark

Have links, but need to benchmark:

To make internally consistent

- E.g., a plant that produces \$1m in output and is the only supplier to a plant that buys \$10m in intermediates—firm data not consistent with each other.

To make externally consistent

- Want the implied expenditure between industries to match the industry totals from the IO tables

Benchmark application

Benchmarking is a big linear program

We want to make sure everything adds up to national accounts by solving a bunch of equations. But! We have an underdetermined system, too many parameters and not enough equations.

Another but!: Recall we want firm-firm network to be sparse

Donoho and Tsaig (2008): if the solution to linear program is sparse, we can use ℓ-1 minimization (LASSO) to solve the program quickly

LASSO

Definition

- Least absolute shrinkage and selection operator (Tibshirani 1996)
- Just like OLS, plus an extra parameter that defines the “penalty”
- Can solve problems with more coefficients than observations (or in our case, more unknowns than equations)

Penalty

- Penalty helps to select important parameters by shrinking less important ones to zero

OLS / LASSO: very similar problems

OLS

$$\min_{\beta} \sum_{i} (y_i - X_i\beta)^2 \tag{1}$$

LASSO

$$\min_{\beta} \sum_{i} (y_i - X_i\beta)^2 + \overbrace{\lambda \sum_{k} |\beta_k|}^{\text{Penalty}} \tag{2}$$

Next, get the equations we need to benchmark, and then translate them into a form we can use.

Benchmark application

Firm sales to all customers add up to total sales; N equations.

$$\sum_r \underset{\substack{\downarrow\\ r\text{'s share of}\\ \text{expenditure on } i}}{a_{ri}} \overset{\substack{\text{Region } r\text{'s}\\ \text{total expenditure}\\ \uparrow}}{M_r} + \sum_j (1 - \overset{\substack{i\text{'s value}\\ \text{added share}\\ \uparrow}}{\beta_i}) \underset{\substack{\downarrow\\ j\text{'s share of}\\ \text{expenditure on } i}}{g_{ji}} s_j = \overset{\substack{\text{sales of firm } i\\ \uparrow}}{s_i}, \forall i = 1, \ldots, N \quad (3)$$

Benchmark application

Firm expenditures on all other firms add up to total expenditures; same for regions. $R + N$ equations.

$$\sum_j (1-\beta_i)\overset{\substack{i\text{'s share of}\\ \text{expenditure on } j\\ \uparrow}}{\boxed{g_{ij}}} s_i = (1-\beta_i)s_i, \forall i = 1, \ldots, N \tag{4}$$

$$\sum_i a_{ri} M_r = \underset{\substack{\downarrow\\ \text{Region } r\text{'s}\\ \text{total expenditure}}}{\boxed{M_r}}, \forall r = 1, \ldots, R \tag{5}$$

Benchmark application

Firm expenditures on all other firms within industry pairs add up to industry pair expenditure from IO table; K^2 equations.

$$\underbrace{\sum_{i \in k} \sum_{j \in k'}}_{\text{all firm pairs in } (k, k')} \overbrace{(1 - \beta_i) g_{ij} s_j}^{j\text{'s expenditure on } i} = \underset{\substack{\big\downarrow \\ \text{Total expenditure of} \\ \text{industry } k \text{ on industry } k'}}{\boxed{S_{kk'}}}, \forall \text{ industry pairs } (k, k') \tag{6}$$

Benchmark application

Size of problem

- Unknowns: RN region-firm shares $(a_{11}, \ldots, a_{RN})$ and N^2 firm-firm shares $(g_{11}, \ldots, g_{NN})$
- Equations: N firm sales, N firm expenditure, R region expenditure, K^2 industry pair expenditure
- Number of unknowns much bigger than number of equations
$$(R + N)N >> 2N + R + K^2$$
- E.g., for $N \approx 30,000$ plants, $R \approx 70$ ERs, $K \approx 200$ industries, unknowns almost 900 million, equations around 100,000

Benchmark application

Rewrite these equations into matrix form for input into algorithm

$$\underset{\substack{\downarrow \\ \text{known parameters on} \\ \text{LHS of equations}}}{X}\ \overset{\substack{\text{unknown vectors } a \text{ and } g \text{ all stuck together} \\ \uparrow}}{y} = \underset{\substack{\downarrow \\ \text{known parameters} \\ \text{on RHS of equations}}}{c} \tag{7}$$

Algorithm solves for parameters y

Many details. To translate to regression notation, X is X (characteristics), y is β (parameters), c is y (outcomes).

Results

To start, fake results to make sure it works

Generate economy with regions, firms and industries. See if the algorithm can solve the equations.

Code available

For generating fake data, converting equations into the correct form, and analysis:

`https://github.com/tweed1e/firm-network-lasso`

Results—fake data

Timing

$N = 15000$, $R = 70$, $K = 70$, with 1% firm-firm density; around 200m parameters, 2m of which are nonzero.

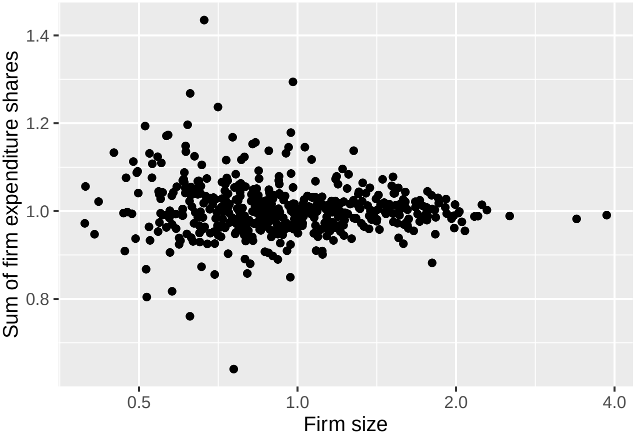
- $\approx$ 5 minutes on laptop

Table: % difference between predicted and actual values

	Obs	Mean	25th	Median	75th
Firm sales	15000	−0.003	−0.004	−0.003	−0.002
Firm exp.	15000	0.200	−2.660	−0.023	2.825
Industry pair exp.	4900	0.000	−0.930	−0.006	0.841
Region exp.	70	0.000	0.000	0.000	0.000

Results—fake data

Figure: Predicted expenditure shares vs. firm size (random sample of 500 firms). All should be equal to 1. Note increase in accuracy as size increases.



Results—real data

Fig: firm size, and maybe expenditure. And next frame: industry.

Conclusion—benchmarking algorithm

On fake data

- Works well, and fast

On real data

- Works fast, but so-so accuracy
- Not surprising, given lack of trade data
- More importantly: means input data and prediction algorithm needs work; not getting the right industry connections / values