Large-Scale Causal Learning

Tony Jebara, Benoit Rostykus

{*tjebara,brostykus*}*@netflix.com*

October 31, 2017

Abstract

We augment causal learning under instrumental variables (IVs) with computational improvements to allow it to scale to large datasets that are typical in modern machine learning. While traditional IV implementations involve linear algebraic operations that have cubic scaling in the dimensions of interest, we achieve linear scalability across 1) the number of samples, 2) the number of non-zero elements of the features, and 3) the number of non-zero elements of the instruments. This is achieved by reformulating IV projections as a single joint optimization over model parameters and performing what we call *pairwise stochastic gradient descent*. We note that standard stochastic gradient descent fails to achieve meaningful speedup and scalability. We instead propose a pairwise stochastic gradient descent scheme endowed with importance sampling methods and implementation subtleties that provide significantly better computational efficiency (in time and memory). The method is also straightforward to extend to non-linear settings (through neuronal models or kernel methods). We demonstrate dramatic computational improvements on a large-scale synthetic dataset as well as on a real-world dataset.

1 Introduction

Causal learning has a long history with techniques such as instrumental variables (IVs) and two-stage least squares (2SLS) dating back to the early 20th century (Wright 1928). While such methods enjoy widespread use in the Economics literature (Gujarati and Porter 2009), the computational scalability of traditional causal IV methods is limited and is an impediment to their adoption in machine learning.

Standard supervised machine learning recovers a function f that predicts an output y from an input x after learning from *iid* samples. However, this approach can fail when the system under consideration contains endogeneity. Endogeneity emerges due to confounders, missing variable bias, simultaneity (e.g. a variable y depends on x but x also depends on y), non-random non-compliance and so on. For instance, consider an application in marketing where a health insurance company is attempting to attract subscribers by showing display ads on the internet. The company sees if a user (cookie) i has been shown an ad, $x_i = 1$ or not $x_i = 0$. The company also sees that if user i signed up with $y_i = 1$ or has not signed up with $y_i = 0$. By doing a simple linear

regression of y on x, the company infers that users who see their ad are twice as likely to sign up for health insurance than users who signup organically (e.g. by hearing about the opportunity from their friends or through other channels). In other words, by sending ads, the company is causing people to sign up for health insurance and that x is causing y. However, in reality this estimate is severely biased because of a confounder: the time of day. During the daytime, users are more actively browsing on the internet, are more likely to see ads and, therefore, more likely to visit the company's website. Thus, time of day is another variable e which is a parent of both x and y. It is quite possible that there is no causal effect of advertising and that the entire behavior can simply be explained by e causing increases in x and y simultaneously.

One way to resolve this problem is to consider an instrumental variable z. If z causes a change in x but is independent of e and somehow this change in x also triggers a change in y then we 2-stage estimation methods can isolate the causal effect of x on y. For instance, the company may have a random variable z which suppresses some ads on some users half of the time. Then, it is possible to change x in a manner that is independent of e and then infer the causal effect from x to y. One framework for this is two-stage least squares (James and Singh 1978).

Two-stage least squares (2SLS) first tries to predict x from z, for instance by learning a linear regression. Call the prediction $\hat{x}$. Second, it learns how to predict y from $\hat{x}$ using another linear regression. Unfortunately, in large-scale machine learning settings, the number of samples and the dimensionalities at hand are both large. The feature space where x resides may be extremely high-dimensional, e.g. with dimension d_X. We may also have a high dimensional vector for each observation z which resides in d_Z dimensional space. Finally, the number of samples we need to consider n may also be large. Therefore, we must learn d_X scalar regressions from a d_Z-dimensional input space. If exact linear-algebraic methods are used to perform 2SLS, we expect a runtime of $\mathcal{O}(d_X \min(n^3, d_Z^3))$ for the first stage and $\mathcal{O}(\min(n^3, d_X^3))$ for the second stage. Essentially, this scales the computation required by d_X relative to a standard ordinary least squares (OLS) problem.

In this article, we propose a method to circumvent this scaling and to learn 2SLS with stochastic gradient techniques. We are therefore able to scale linearly and handle problems that are both high-dimensional and involve a large number of samples. In fact, we will scale linearly with the number of non-sparse elements of the high dimensional feature vectors and instrumental variable vectors.

2 The 2SLS problem as a single optimization

In the 2SLS problem setup, we are given a data-set consisting of feature vectors $x_i \in \mathbb{R}^{d_X}$, instrumental variable vectors $z_i \in \mathbb{R}^{d_Z}$ and output scalars y_i for $i = 1, \ldots, n$. We may also be given scalar weights $w_1, \ldots, w_n$ that indicate the relative important of some samples over others in the data-set (such scalars can capture Bayesian bootstrap weighting, negative class subsampling, etc.).

The goal of 2SLS is to learn a linear causal relationship between the input feature vectors x_i and the output y_i of the form $y = \theta^\top x + noise$ (we omit the bias here for simplicity since it can be obtained simply by concatenating a 1 to each feature vector). The approach first learns to reconstruct each dimension $d =$

$1, \ldots, d_X$ of x_i from the feature vector z_i by learning projection vectors $\pi_d \in \mathbb{R}^{d_Z}$ such that $x(d) = \pi_d^\top z + noise$. By performing linear least-squares, we learn the projection vectors $\hat{\pi}_1, \ldots, \hat{\pi}_{d_X}$. Subsequently, given the projections, we obtain reconstructed feature vectors $\hat{x}_1, \ldots, \hat{x}_n$ by computing $\hat{x}_i(d) = \pi_d^\top z_i$. Finally, given the reconstructions, we recover the desired relationship $\hat{\theta}$ by performing linear least squares to predict y_i from $\hat{x}_i$.

Consider matrix representations of the 2SLS data-set:

- a matrix of features X of size (n, d_X) and of sparsity ratio s_X (i.e each row has an average of $s_X d_X$ non-zeros)
- a matrix of instruments Z of size (n, d_Z) and of sparsity ratio s_Z
- a response matrix Y of size $(n, 1)$
- a diagonal matrix W of size (n, n) where w_i is an arbitrary weight associated with row i.

The 2SLS problem (also known as IV regression) can be then written in the following closed-form formula (here we assume $W = I$ for compactness):

$$\hat{\theta} \quad = \quad \left(X^\top Z (Z^\top Z)^{-1} Z^\top X\right)^{-1} X^\top Z (Z^\top Z)^{-1} Z Y. \tag{1}$$

It is straightforward to show that the above condition is equivalent to

$$\hat{\theta} = \arg\min_\theta \mathcal{E}(\theta)^\top Z Z^\top \mathcal{E}(\theta) \tag{2}$$

where $\theta \in \mathbb{R}^{d_X}$ is the vector of parameters to be estimated and $\mathcal{E}(\theta)_i = y_i - \theta^\top x_i$ are the residuals. In the case of $W \neq I$, we have the following slightly more general formula

$$\hat{\theta} = \arg\min_\theta \mathcal{E}(\theta)^\top W Z Z^\top W \mathcal{E}(\theta) \tag{3}$$

Note that one can view this problem as a standard weighted Ordinary Least Squares (OLS) problem where one would minimize $\mathcal{E}^\top \mathcal{E}$ except that we're using a different norm than the Euclidian norm: we're using the instruments Z to build a positive symmetric matrix ZZ^top. Of course, the matrix $ZZ^\top$ is of size $n \times n$ which makes its evaluation (and inversion in Equation 1) completely impractical.

We are interested in solving (3) in an efficient manner when n, d_X and d_Z are very large (potentially multiple millions) and s_X and s_Z are small (very sparse matrices). If we want perfect scalability, we need the computational solution of (3) to have a complexity linear in n, linear in $s_X d_X$ and linear in $s_Z d_Z$. We will next explore a stochastic gradient descent approach to this problem.

3 Stochastic gradient descent approach

We now present a solution achieving this linear complexity. (3) can be explicitly written as finding θ minimizing the convex loss $L(\theta)$:

$$L(\theta) = \sum_{m=1}^{d_Z} \left(\sum_{i=1}^{n} w_i z_{i,m} \left(y_i - \theta^\top x_i\right) \right)^2 \tag{4}$$

$$L(\theta) = \sum_{i=1}^{n}\sum_{j=1}^{n}\left[w_i w_j \left(y_i - \theta^\top x_i\right)\left(y_j - \theta^\top x_j\right)\sum_{m=1}^{d_Z} z_{i,m} z_{j,m}\right] \tag{5}$$

$$L(\theta) = \sum_{i=1}^{n}\sum_{j=1}^{n} \ell_{i,j}(\theta) \tag{6}$$

We see that we can sample (i, j) pairs and take a stochastic step in the direction of $\nabla \ell_{i,j}(\theta)$ as this is an unbiased estimate of the true gradient:

$$\mathbb{E}\left[\nabla \ell_{i,j}(\theta)\right] = \nabla L(\theta) \tag{7}$$

Notice that $\nabla \ell_{i,j}(\theta)$ is sparse with respect to both the X and Z matrices, so we achieve the linear scaling we wanted with respect to X and Z sparsity patterns.

3.1 Faster sampling

In stochastic SGD, we should not be sampling (i, j) pairs uniformly since it will take $O(n^2)$ samples in the worst case to approximate this sum. Now, view the problem as a weighted version of the form

$$L(\theta) = \sum_{i=1}^{n}\sum_{j=1}^{n}(y_i - \theta^\top x_i)(y_j - \theta^\top x_j)K_{ij}$$

where $K_{ij} = w_i w_j \sum_{m=1}^{d_Z} z_{i,m} z_{j,m}$. Since all the $z_{i,m}$ are non-negative instrumental variables (or can we made as such without any loss of generality), we can view this as an expectation over some distribution

$$L(\theta) \propto \sum_{i=1}^{n}\sum_{j=1}^{n}(y_i - \theta^\top x_i)(y_j - \theta^\top x_j)p(i, j)$$

where $p(i, j) \propto K_{ij}$. We can then decompose this distribution as

$$p(i, j) = \sum_{m=1}^{d_Z} p(i|m)p(j|m)p(m)$$

where

$$p(i|m) = \frac{w_i z_{i,m}}{\sum_{k=1}^{n} w_k z_{k,m}}$$

and equivalently

$$p(j|m) = \frac{w_j z_{j,m}}{\sum_{k=1}^{n} w_k z_{k,m}}$$

and

$$p(m) = \frac{(\sum_{i=1}^{n} w_i z_{i,m})^2}{\sum_{n=1}^{Z_d}(\sum_{i=1}^{n} w_i z_{i,n})^2}$$

It is easy to verify that this is proportional to K_{ij}. Therefore, to sample from $p(i, j)$, we should just sample an m_t from $p(m)$, then sample an i_t from $p(i|m_t)$ and sample a j_t from $p(j|m_t)$ for samples $t = 1, \ldots, T$. The pairs (i_t, j_t) are a

much more efficient way (and exact way) of sampling from $p(i, j)$. Then, the objective function is simply

$$L(\theta) \approx \frac{1}{T} \sum_{t=1}^{\top} (y_{i_t} - \theta^\top x_{i_t})(y_{j_t} - \theta^\top x_{j_t})$$

Thus, for SGD, we should update with a single sample t at a time. This will be more efficient than sampling (i, j) uniformly.

3.2 A further speedup

We offered a pairwise SGD scheme for the IV regression problem. We now look at row-wise variant. Rewrite the loss function using the trick above as

$$L(\theta) \propto \sum_{i=1}^{n} \sum_{j=1}^{n} (y_i - \theta^\top x_i)(y_j - \theta^\top x_j) \sum_{m=1}^{d_Z} p(i|m)p(j|m)p(m).$$

Rearrange

$$L(\theta) \propto \sum_{m=1}^{d_Z} p(m) \sum_{i=1}^{n} \sum_{j=1}^{n} (y_i - \theta^\top x_i)p(i|m)(y_j - \theta^\top x_j)p(j|m).$$

Defining the scalar error terms $\epsilon_m(\theta) = \sum_i p(i|m)y_i - \theta^\top \sum_i p(i|m)x_i$ for $m = 1, \ldots, d_Z$ one can show that the gradient of the above is:

$$\nabla L(\theta) \propto \sum_{m=1}^{d_Z} p(m)\epsilon_m(\theta) \left(- \sum_{i=1}^{n} p(i|m)x_i \right).$$

To get a stochastic version of the gradient, sample m_t according to $p(m)$ and sample i_t according to $p(i|m_t)$. Then perform the following update

$$\theta \leftarrow \theta + \eta \epsilon_{m_t}(\theta) x_{i_t}. \tag{8}$$

This update will cost $O(s_X d_X)$.
We also need to update $\epsilon_m(\theta)$ $\forall m \in [0, d_Z]$. We write:

$$\epsilon_m(\theta) = K_m - \theta^\top T_m$$

With $K_m = \sum_i p(i|m)y_i$ which can be stored in $O(d_Z)$ and $T_m = \sum_i p(i|m)x_i$ which can be stored in $O(d_X \times d_Z)$.
Note that (8) means that we need to update **all** the ϵ_m with:

$$\epsilon_m(\theta_{t+1}) = \epsilon_m(\theta_t) \left[1 - \eta_t x_{i_t}^\top T_m\right]$$

This is expensive in a sparse setting, because it implies a scaling with d_Z with every stochastic step, which we want to avoid. Instead, we propose the following approximation scheme: we only update the single ϵ_m with $m = m_t$ with the following formula:

$$\epsilon_{m_t}(\theta_{t+1}) = \epsilon_{m_t}(\theta_t) |1 - \eta_t x_{i_t}^\top T_m|^{p(m_t)^{-1}} \operatorname{sgn}\left(1 - \eta_t x_{i_t}^\top T_m\right)$$

The idea being that we keep the expectation of the multiplicative update of all the ϵ_m, but only update the single current ϵ_{m_t}.

3.3 Nonlinear and deep models

There is nothing preventing us from generalizing the framework so that we have non-linear and deep mappings rather than just $\theta^\top x$. We can simply do gradient descent of the loss

$$L(\theta) = \sum_{i=1}^{n}\sum_{j=1}^{n}(y_i - f(x_i;\theta))(y_j - f(x_j;\theta))K_{ij}$$

for any nonlinear function $f(x_i;\theta)$ we can parametrize by θ (e.g. a deep neural network). We still would do stochastic gradient descent by sampling as in the above framework and minimizing

$$L(\theta) \approx \frac{1}{T}\sum_{t=1}^{\top}(y_{i_t} - f(x_{i_t};\theta))(y_{j_t} - f(x_{j_t};\theta)).$$

Contrary to the first-stage procedure of the recently proposed DeepIV algorithm (Hartford et al. 2017) which scales with the total dimensionality d_X of the treatment variables irrespective of their sparsity, our single-step SGD solution leverages this sparsity to offer a tractable solution for neural-network based causal regression in a large-dimensional sparse setting.

3.4 Memory complexity of efficient sampling

In order to sample according to $p(m)$, we can just store in memory the CDF of $p(m)$, which requires to store d_Z variables in memory and can be efficiently computed with a single pass over the data.
Efficiently sampling according to $p(i|m)$ is more challenging. The naive approach is to order the training set by $w_i z_{i,m}$ for each instrumental variable m so that one can build the $CDF(i|m)$. This scales with the sparsity of Z, but requires to be able to store Z in memory to efficiently draw from $P(i|m)$.
Overall, the naive solution requires to store in memory something in the order of Z which can be prohibitive.
If one cannot store Z in memory, we can use distribution sketching to store limited information that will give us approximate draws from $P(i|m)$. One idea is to approximate q quantiles of $P(i|m)$ for each m so that one keeps an optimal amount of information per instrument under fixed memory constraint. Keeping this sketch in memory, we can then run out-of-core SGD learning by sequentially reading the data from disk (which is a lot more efficient than random sampling with replacement) and use IPS-reweighting of the original likelihood based on our estimated importance $\hat{P}(i|m)$.

4 L-BFGS solution

Since L is strictly convex in the case of a linear model (provided the addition of an L2 regularization on the parameters), we can use a quasi-Newton solver such as L-BFGS to minimize it.
The full gradient of (4) is:

$$\frac{\partial}{\partial\theta^k}L = -2\sum_{m=1}^{d_Z}\left(\sum_{i=1}^{n}w_i z_{i,m}x_i^k\right)\left(\sum_{i=1}^{n}w_i z_{i,m}(y_i - \theta^\top x_i)\right) \tag{9}$$

$$\frac{\partial}{\partial \theta^k} L = -2 \sum_{m=1}^{d_Z} U(m,k) V_m(\theta) \tag{10}$$

We note that $V_m(\theta)$ can be computed with a single pass over the data and respects the sparsity of X.
$U(m,k)$ is independent of the parameters to be optimized and hence can be pre-computed. It is easy to see that we can pre-compute U with a single pass over the data while respecting the sparsity of both Z and X. However in all generality it requires quadratic storage $d_Z \times d_X$.
So an L-BFGS solution to (4) can be implemented with a CPU complexity proportional to the sparsity of both X and Z.

5 Experiments

We demonstrate the efficiency of our method on a large-scale synthetic problem as well as on a real-world dataset.
For the synthetic data, we will consider the following generative model for Z:

$$Z = (B_1 \otimes N) \times (I_{d_z} + B_2)$$

where:
B_1 is an $n \times d_z$ random Bernoulli $\mathcal{B}(p_1)$ matrix
N is an $n \times d_z$ random Gaussian $\mathcal{N}(0, I)$ matrix
B_2 is an $d_z \times d_z$ random Bernoulli $\mathcal{B}(p_2)$ matrix
$\otimes$ is the Hadamard product

This design allows us to efficiently draw large sparse Z matrices which have a sparsity controlled by p_1 and p_2, while p_2 controls the amount of correlation between the instruments. We choose X to be:

$$X = \gamma_1 \times Z + (1 - \gamma_1) B_3$$

where B_3 is an $n \times d_z$ random Bernoulli $\mathcal{B}(p_3)$ matrix and $\gamma_1 \in [0, 1]$ controls the strength of the instruments. Finally for all $i \in [0, n]$:

$$y_i = \theta_{truth}^\top X_i + \rho \epsilon_i$$

$$\epsilon_i = \left(\gamma_2 Z_i + (1 - \gamma_2) H_i\right)^\top \mathbf{1}$$

with H_i a random Gaussian $\mathcal{N}(0, 1)$ vector of $\mathbb{R}^{d_z}$ and $(\rho, \gamma_2) \in [0, 1]^2$.
With this setup, ρ controls the overall amount of noise while γ_2 controls the level of endogeneity. θ_{truth} is the ground-truth vector of parameters to be recovered.

Since a 2SLS solver is impractical at large scale, we compare our method against the L-BFGS solution described earlier. Because the L-BFGS solution to IV regression requires $d_X \times d_Z$ dense memory storage on top of the rank-k approximation of inverse Hessian, we limit ourself to a relatively small $d_X = d_Z = 1000$ setting, with $n = 1e6$ and $s_X = 1.5\%$, $s_Z = 1\%$. We choose a small level of endogeneity $\gamma_2 = 5\%$ and relatively good instruments $\gamma_1 = 0.5$. All SGD schemes use an AdaGrad schedule. Results are reported in figure (1).

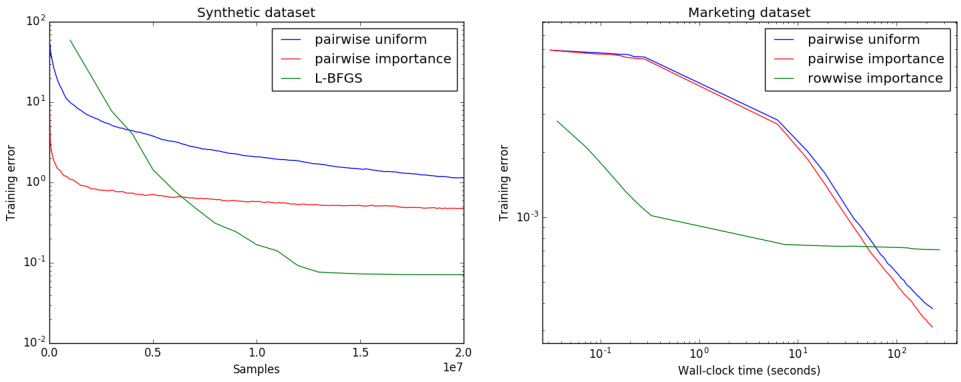

Figure 1: Convergence speed of the proposed methods

We also report in figure (1) the performance of our method on a real-world proprietary marketing dataset, related to causal estimation of ad effectiveness. In this context, we're trying to recover the causal expected ROI of an ad as a function of ad, inventory and user attributes all embedded in a high-dimensional sparse space. This dataset has the following characteristics $n = 1.5 \times 10^7$, $d_X = 1.3 \times 10^5$, $d_Z = 1.6 \times 10^4$, $s_X \approx 10^{-4}$, $s_Z \approx 10^{-3}$.

6 Conclusion

We have offered efficient optimization strategies to the problem of IV regression for causal inference in a large-scale sparse setting. Our stochastic optimization methods make it practically possible to train in the order of minutes causal models when the number data points, covariates and instruments are all in the order of millions.

References

Gujarati, D.N. and D.C. Porter (2009). *Basic Econometrics*. New York: McGraw-Hill Irwin.

Hartford, Jason et al. (2017). "Deep IV: A Flexible Approach for Counterfactual Prediction". In: *Proceedings of the 34th International Conference on Machine Learning*. Vol. 70. PMLR, pp. 1414–1423.

James, L.R. and B.K. Singh (1978). "An introduction to the logic, assumptions, and basic analytic procedures of two-stage least squares". In: *Psychological Bulletin* 85.5, pp. 1104–1122.

Wright, P.G. (1928). *The Tariff on Animal and Vegetable Oils*.