Appendix for Deep IV: A Flexible Approach for Counterfactual Prediction

Jason Hartford   Greg Lewis   Kevin Leyton-Brown   Matt Taddy

1. Model architectures and training details

1.1. Optimization and regularization

All models optimized with Adam (Kingma & Ba, 2014) with learning rate = 0.001, $\beta_1 = 0.9, \beta_2 = 0.999$ and $\epsilon = 1e-08$.

Unless otherwise specified, we trained our models for $(1.5 \times 10^6/n)$ epochs with a batch size of 100, weight decay (L2 regularization) set to 0.001 and a dropout rate of $\min(1000/(1000+n); 0.5)$ where n is the number of training examples. These heuristics were chosen because models trained on larger datasets larger datasets typically need fewer epochs (because they do more mini-batch updates per an epoch), and models trained on larger datasets need less dropout regularization (because the larger datasets reduce the risk of overfitting).

1.2. Architectures

The low dimensional domain experiment used multi-layer perceptrons for both the treatment and response networks. Both networks had three hidden layers with 128, 64 and 32 units respectively. The treatment networks used tanh activation functions and a mixture of gaussian output with 10 mixture components, while the response networks used relu activation functions.

The high dimensional experiment used multiple convolution layers to construct an image embedding which was merged with the additional features before applying further dense layers to produce the output (See Figure 1). Again, both the treatment and response networks used the same architecture. The image embedding was constructed by applying two convolution layers, each with 64 3×3 filters and relu activation layers, followed by a 2×2 max pooling layer and a single fully connected layer that mapped to a 64 dimensional output embedding. The *time of year* feature, x, and *fuel price* instrument, z, (or *price*, p, treatment variable in the case of the response network) were concatenated with this embedding layer before applying one hidden layer and mapping to the output. The treatment network used the same mixture of gaussian output with 10 mixture components while the response network mapped to a scalar output.

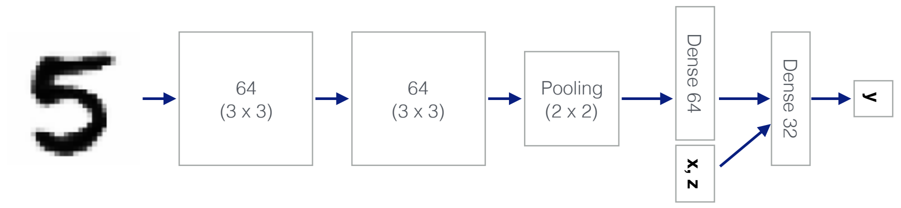

Figure 1. The convolutional architecture used for the high dimensional simulation experiments.

2. Baselines models

Two stage least squares Standard two-stage least squares was used for the low dimensional experiment. For the high dimensional experiment, it was necessary to add some $L2$ regularization (i.e. ridge regression) to avoid overfitting because the model had a large number of parameters (MNIST digits are 28*28 = 784 pixel). The pixels were flattened into a vector and treated as features.

Two stage least squares with polynomial basis Again we followed the standard two-stage least square procedure, but applied a polynomial basis expansion at each stage, where the basis expansion with interaction terms. The order of the polynomial and the ridge regression hyper-parameter were chosen using 5-fold cross-validation at each stage. Note: this is *not* a causally valid procedure because the polynomial basis expansion invalidates the closed form solutions to the expectations. We include it simply to compare to a more scalable flexible model.

Nonparametric Instrumental Kernel Regression We used the `npregiv` function in the `np` package (Hayfield et al., 2008) in R. To reproduce the method described in Darolles et al. (2011), we followed the `np` package documentation's directives to set `method = 'Tikhonov'` and local constant kernel weighting. We experimented with alternative parameter settings, but we unable to improve on these defaults in a reasonable amount of time. These methods tend to be extremely slow in practice.

References

Darolles, Serge, Fan, Yanqin, Florens, Jean-Pierre, and Renault, Eric. Nonparametric instrumental regression. *Econometrica*, 79:1541–1565, 2011.

Hayfield, Tristen, Racine, Jeffrey S, et al. Nonparametric econometrics: The np package. *Journal of statistical software*, 27 (5):1–32, 2008.

Kingma, Diederik and Ba, Jimmy. ADAM: A method for stochastic optimization. *International Conference on Learning Representations (ICLR)*, 2014.