Instrumental Variables using a Neural Network

Bastiaan Quast

18th April 2016

This paper discussed the possibility of using a feed-forward neural network to implement an Instrumental Variables approach. I use two types of data, the `CitarettesSW` dataset from the `AER` (Applied Econometric Regressions), which is the standard method for instrumental variables in R. The `ivreg()` function in the `AER` package uses the Two-Stage Least Squares (TSLS) approach. The advantage of the simulated dataset is that we can be certain that the conditions for Instrumental Variables (IV) hold.

1 Data

Table 1: AER Package & Data

```
library(AER)
data("CigarettesSW")
rprice  <- with(CigarettesSW, price/cpi)
tdiff   <- with(CigarettesSW, (taxs - tax)/cpi)
packs   <- CigarettesSW$packs
```

A manual Two-Stage Least Squares analysis can be performed as follows (using the built in `lm()` function).

Table 2: Manual TSLS

```
# first stage
s1 <- lm(rprice ~ tdiff)

# estimate second stage using fitted values (Xhat)
lm(packs ~ s1$fitted.values)
```

	Estimate	Std. Error	t value	Pr(>|t|)
(Intercept)	219.576384	20.8630993	10.524629	0e+00
s1$fitted.values	-1.019485	0.1914682	-5.324565	7e-07

We can verity these results using `AER`'s built in function `ivreg()`.

Table 3: TSLS using `ivreg()`

```
ivreg(packs ~ rprice | tdiff)

##
## Call:
## ivreg(formula = packs ~ rprice | tdiff)
##
## Coefficients:
## (Intercept)       rprice
##     219.576       -1.019
```

A good way to prove this theoretically is using simulated data.

Table 4: Simulated Data

```
# library for generation of multivariate distributions
library(MASS)

# always use the same random numbers
set.seed(123)

# the means and errors for the multivariate distribution
MUs    <- c(10,15)
SIGMAs <- matrix(c(1,   0.5,
                   0.5, 2   ),
                 nrow=2,
                 ncol=2       )

# the multivariate distribution
mdist <- mvrnorm(n     = 1000,
                 mu    = MUs,
                 Sigma = SIGMAs)

# create unobserved covariate
c <- mdist[ , 2]

# create the instrumental variable
z <- rnorm(1000)

# create observed variable
x <- mdist[ , 1] + z

# constuct the dependent variable
y <- 1 + x + c + rnorm(1000, 0, 0.5)
```

Check if the variables behave as expected.

Table 5: Simulated Variables

```
cor(x, c)

## [1] 0.1986307

cor(z, c)

## [1] -0.0120011
```

Let's look at the true model.

Table 6: True Model (OLS)

```r
lm(y ~ x + c)
```

	Estimate	Std. Error	t value	Pr(>|t|)
(Intercept)	0.9078664	0.1860081	4.880789	1.2e-06
x	1.0155658	0.0118758	85.515517	0.0e+00
c	0.9955182	0.0111828	89.021974	0.0e+00

Now, we assume that we do not have access to `c`.

Table 7: OLS without c

```r
lm(y ~ x)
```

	Estimate	Std. Error	t value	Pr(>|t|)
(Intercept)	13.78678	0.3495719	39.43904	0
x	1.22556	0.0348005	35.21669	0

We now assume to have access to the Instrumental Variable `z`, and estimate it using Two-Stage Least Squares.

Table 8: Manual IV

```r
# first stage
lms1 <- lm(x ~ z)

# manually obtain fitted values
lmXhat <- lms1$coefficients[2]*z + lms1$coefficients[1]

# estimate second stage using Xhat
(lms2 <- lm(y ~ lmXhat) )
```

	Estimate	Std. Error	t value	Pr(>|t|)
(Intercept)	15.948900	0.6715031	23.75105	0
lmXhat	1.008351	0.0671579	15.01464	0

Or equivalently, using `ivreg()`.

Table 9: IV using `ivreg()`

```
ivreg(y ~ x | z)

##
## Call:
## ivreg(formula = y ~ x | z)
##
## Coefficients:
## (Intercept)            x
##      15.949        1.008
```

The Instrumental Variables analysis can also be performed using feed-forward neural networks. Neural networks in their most basic form find weights that correspond to the coefficients of an Ordinary Least Squares (OLS) estimation. However, as the weights are obtained without OLS, this is Instrumental Variables without using Two-Stage Least Squares.

The following table demonstrates how the weights obtained by a neural network are virtually identical to the coefficients obtained by Ordinary Least Squares estimation.

Table 10: Neural Network

```
library(nnet)

# first stage with neural network
nns1 <- nnet(x ~ z, size=0, skip=TRUE, linout=TRUE)

## # weights:  2
## initial  value 98765.653982
## final  value 924.804075
## converged
```

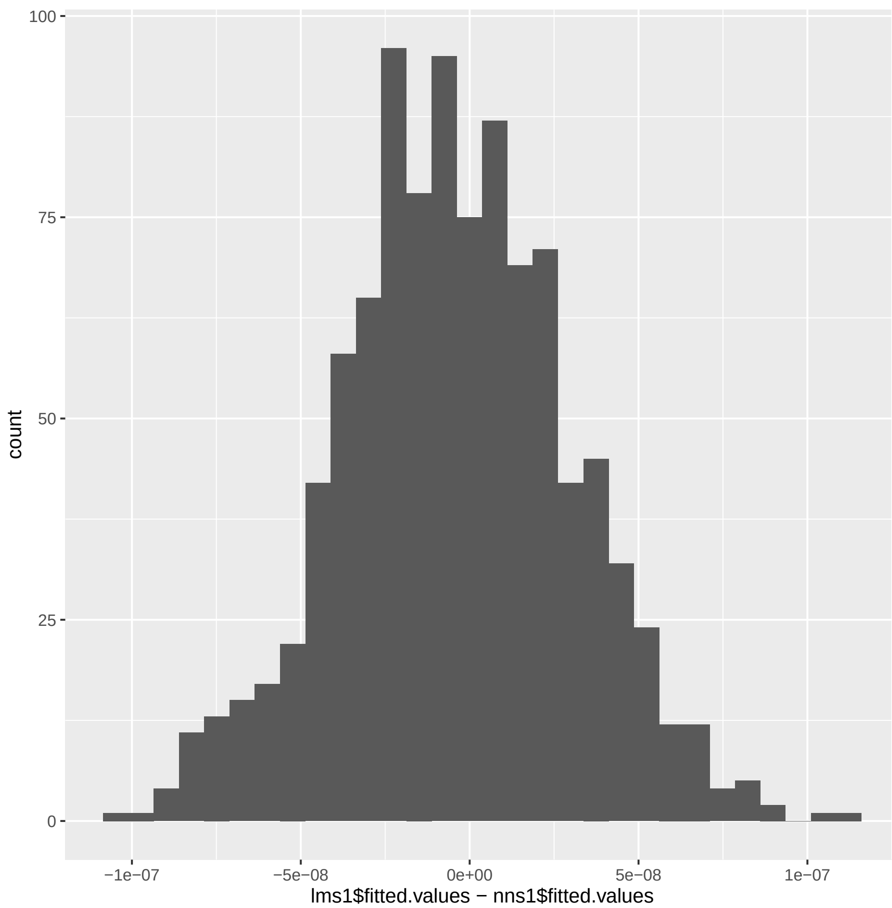
The results obtained by nns1 are virtually identical to those in lms1.

Table 11: Neural Network and OLS

```
lms1$coefficients - nns1$wts

##   (Intercept)             z
## -3.365223e-09  3.461435e-08

# graphically
library(ggplot2)
qplot(lms1$fitted.values - nns1$fitted.values)
```



We can now also perform the second stage using neural networks.

Table 12: IV using Neural Network

```r
# manually obtain fitted values
nnXhat <- nns1$fitted.values

# estimate second stage using Xhat
nns2 <- nnet(y ~ nnXhat, size=0, skip=TRUE, linout=TRUE)

## # weights:  2
## initial  value 874286.766246
## final  value 4019.409973
## converged

# evaluate outcome
summary(nns2)

## a 1-0-1 network with 2 weights
## options were - skip-layer connections  linear output units
##  b->o i1->o
## 15.95  1.01
```

Now compare the final estimates.

Table 13: Neural Network and TSLS

```r
lms2$coefficients - nns2$wts

## (Intercept)      lmXhat
##  1.0366e-06 -1.1273e-07

# graphically
library(ggplot2)
qplot(lms2$fitted.values - nns2$fitted.values)
```