Matching and Weighting

Drew Dimmery drewd@nyu.edu

February 28, 2014

Structure

- IPW and Sampling
- Matching
  - Nearest Neighbor
  - Mahalanobis distance
  - Genetic Matching
  - CEM
- Beyond Matching
  - Entropy balancing, etc

Big Picture

- ahem -
- MATCHING IS NOT AN IDENTIFICATION STRATEGY.
- Heckman, Ichimura, Smith and Todd (1998) provide a nice decomposition:
  - $B = \int_{S_{1X}} E[Y_0|X, D = 1]dF(X|D = 1) - \int_{S_{0X}} E[Y_0|X, D = 0]dF(X|D = 0)$
  - $B = B_1 + B_2 + B_3$
  - $B_1 = \int_{S_{1X} \setminus S_X} E[Y_0|X, D = 1]dF(X|D = 1) - \int_{S_{0X} \setminus S_X} E[Y_0|X, D = 0]dF(X|D = 0)$
  - $B_2 = \int_{S_X} E[Y_0|X, D = 0](dF(X|D = 1) - dF(X|D = 0))$
  - $B_3 = P_X \bar{B}_{S_X}$
  - Matching addresses B_1 and B_2. CIA requires an assumptions to control B_3.
  - Relative magnitudes are unknown.
- This gets to the question Cyrus has been repeating a lot: How could two seemingly identical units receive *different* treatments?

Slightly Smaller Picture

- Okay, we have some random mechanism that exists after controlling for covariates.
- Why don't we just put them in a regression?
  - There's an intuitive appeal to be able to do all of this controlling while keeping the outcome in a lockbox.
  - Separating the procedures mean that you can address two types of confounding separately.
    1. Different treatment groups may have different chances of getting treated.
    2. Different treatment groups may have different baseline (control) potential outcomes.
  - A design which addresses both of these options separately is called "doubly robust".
  - Double robustness means that we only have to get ONE of these right for consistent estimation.
  - (What's the probability of getting a one out of two independent bernoulli trials with $\pi = 0$?)
- I'm going to do most matching by hand to show you what's under the hood. You should use `MatchIt` for the homework.
- There's an extensive manual – use it.

Setup dataset

- Today, because we're doing matching, we're going to be looking at the Lalonde data.
- If you ever read any paper about matching, you'll probably see this data again. (I've heard this called the Lalonde Fallacy)

. . .

```
require(MatchIt)
data(lalonde, package = "MatchIt")
trt <- lalonde$treat == 1
means <- apply(lalonde[, -1], 2, function(x) tapply(x, trt, mean))
sds <- apply(lalonde[, -1], 2, function(x) tapply(x, trt, sd))
rownames(means) <- rownames(sds) <- c("Treated", "Control")
varratio <- sds[1, ]^2/sds[2, ]^2
ks.p <- apply(lalonde[, -1], 2, function(x) ks.test(x[trt], x[!trt])$p.value)
```

```
## Warning: p-value will be approximate in the presence of ties
## Warning: p-value will be approximate in the presence of ties
## Warning: p-value will be approximate in the presence of ties
## Warning: p-value will be approximate in the presence of ties
## Warning: p-value will be approximate in the presence of ties
## Warning: p-value will be approximate in the presence of ties
## Warning: p-value will be approximate in the presence of ties
## Warning: p-value will be approximate in the presence of ties
## Warning: p-value will be approximate in the presence of ties
```

```
t.p <- apply(lalonde[, -1], 2, function(x) t.test(x[trt], x[!trt])$p.value)
```

View Initial Balance

```
round(t(rbind(means, sds, varratio, ks.p, t.p)), 3)
```

```
##             Treated  Control  Treated  Control varratio  ks.p   t.p
## age          28.030   25.816   10.787    7.155    2.273 0.003 0.003
## educ         10.235   10.346    2.855    2.011    2.017 0.081 0.585
## black         0.203    0.843    0.403    0.365    1.219 0.000 0.000
## hispan        0.142    0.059    0.350    0.237    2.174 0.339 0.001
## married       0.513    0.189    0.500    0.393    1.624 0.000 0.000
## nodegree      0.597    0.708    0.491    0.456    1.161 0.081 0.007
## re74       5619.237 2095.574 6788.751 4886.620    1.930 0.000 0.000
## re75       2466.484 1532.055 3291.996 3219.251    1.046 0.000 0.001
## re78       6984.170 6349.144 7294.162 7867.402    0.860 0.162 0.349
```

Propensity Score

- The propensity score is based on a sort of Horvitz-Thompson estimator.
- Dividing by the probability of sampling means that we weight higher for units with low inclusion probabilities.
- In our case, we can imagine having a sample of units (each with Y_0 and Y_1). We then randomly assign them to treatment.
- This is equivalent to randomly sampling potential outcomes.
- So if we believe that treatment(/sampling) probabilities are assigned according to some covariates, then we just need to know what those probabilities are.
- Call the propensity score $e(X)$. Then $e(X)$ tells us the probability of sampling Y_1 (treating out sample as the population, because we're interested in a SATE).
- This suggests that we can just use $\frac{1}{n_1}\sum_{i=1}^{n_1}\frac{(Y_i\backslash N)}{e(X_i)}$ to estimate $E[Y_1]$.

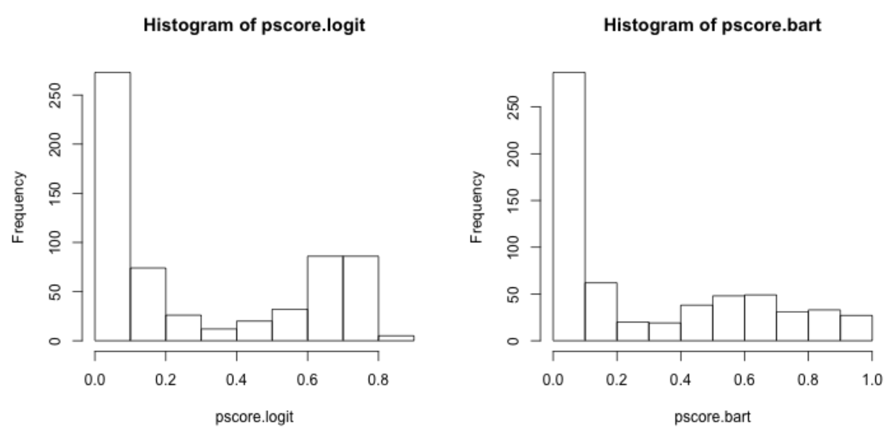
- This embeds the logic of IPW.

Fitting the Propensity Score

- First, estimate a model of the propensity score.
- (Typically just some logit)

. . .

```
p.model <- glm(treat ~ age + educ + black + hispan + married + nodegree + re74 +
    re75, lalonde, family = "binomial")
require(BayesTree)
# p.bart <- bart(lalonde[,-c(1,ncol(lalonde))],lalonde$treat,verbose=FALSE)
pscore.logit <- predict(p.model, type = "response")
pscore.bart <- pnorm(colMeans(tttt$yhat.train))
par(mfrow = c(1, 2))
hist(pscore.logit)
hist(pscore.bart)
```



Estimate Model

- What do you want to estimate? This will change the appropriate weights.
- For ATT, sampling probability for treated units is 1.

. . .

```r
base.mod <- lm(re78 ~ treat + age + educ + black + hispan + married + nodegree +
    re74 + re75, lalonde)
ipw.logit <- trt + (1 - trt)/(1 - pscore.logit)
ipw.logit.mod <- lm(re78 ~ treat + age + educ + black + hispan + married + nodegree +
    re74 + re75, lalonde, weights = ipw.logit)
ipw.bart <- trt + (1 - trt)/(1 - pscore.bart)
ipw.bart.mod <- lm(re78 ~ treat + age + educ + black + hispan + married + nodegree +
    re74 + re75, lalonde, weights = ipw.bart)
coefs <- c(base = coef(base.mod)[2], ipw.logit = coef(ipw.logit.mod)[2], ipw.bart = coef(ipw
coefs
```

```
##      base.treat ipw.logit.treat  ipw.bart.treat
##            1548            1332            1304
```

Propensity Score matching

- We don't have to weight, though. We might match, instead.

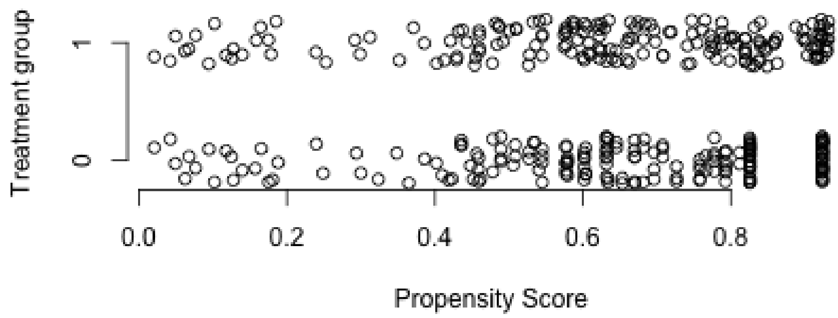
. . .

```r
ctl.data <- subset(lalonde, treat == 0)
pscore.logit.ctl <- pscore.logit[!trt]
pscore.logit.trt <- pscore.logit[trt]
pscore.bart.ctl <- pscore.bart[!trt]
pscore.bart.trt <- pscore.bart[trt]
match.data <- subset(lalonde, treat == 1)
matches <- sapply(pscore.logit.trt, function(x) which.min(abs(pscore.logit.ctl -
    x)))
match.data <- rbind(match.data, ctl.data[matches, ])
pm.logit.mod <- lm(re78 ~ treat + age + educ + black + hispan + married + nodegree +
    re74 + re75, match.data)
match.data <- subset(lalonde, treat == 1)
matches <- sapply(pscore.bart.trt, function(x) which.min(abs(pscore.bart.ctl -
    x)))
match.data <- rbind(match.data, ctl.data[matches, ])
pm.bart.mod <- lm(re78 ~ treat + age + educ + black + hispan + married + nodegree +
    re74 + re75, match.data)
```

Estimation and such

```r
plot(c(pscore.bart.trt, pscore.bart.ctl[matches]), jitter(rep(c(1, 0), c(N,
    N))), axes = F, ylab = "Treatment group", xlab = "Propensity Score")
```

```r
axis(1)
axis(2, c(0, 1))
```




```r
coefs <- c(coefs, pmat.logit = coef(pm.logit.mod)[2], pmat.bart = coef(pm.bart.mod)[2])
coefs
```

```
##       base.treat  ipw.logit.treat   ipw.bart.treat pmat.logit.treat
##             1548             1332             1304             1964
##  pmat.bart.treat
##             1352
```

Conditional Treatment effects

- You can also think about using the local linear regression we talked about last week.
- Weight according to the propensity score.
- This allows you to see how the treatment effect varies along the propensity score.
- Does the treatment only seem to have an effect on people who were very unlikely to be exposed? etc

Mahalanobis Distance

- $(x - \mu)' V^{-1} (x - \mu)$
- In our case, μ corresponds to a given treated unit.

- Mahalanobis distance is a very common distance "metric".
- You can think about it as simple Euclidean distance in a warped feature space (warped according the the inverse variance-covariance matrix)

. . .

```
V <- cov(lalonde[, -c(1, ncol(lalonde))])
match.data <- subset(lalonde, treat == 1)
mahal.dist <- apply(match.data[, -c(1, ncol(match.data))], 1, function(x) mahalanobis(ctl.da
    -c(1, ncol(ctl.data))], x, V))
matches <- apply(mahal.dist, 2, which.min)
N <- length(matches)
match.data <- rbind(match.data, ctl.data[matches, ])
table(apply(mahal.dist, 2, which.min))
```

```
##
##   1   6  17  23  59  72  95  96  97  99 110 112 118 127 134 140 150 158
##   1   2   1   1   1   1   1   1   1   3   2   1   9   1   4   6   1   1
## 159 168 177 179 199 202 218 220 224 226 228 235 237 238 247 253 265 266
##   2   1   1   2   1   1   2   1   1   5   2   1   1   1   1   4   1   2
## 269 278 290 291 308 322 326 327 330 331 333 335 339 341 345 352 353 354
##   3   1   1   1   3   1   1   1   1   2   2   2   1   1   1   8   2   1
## 355 361 366 367 368 372 373 374 376 380 381 383 388 391 392 393 399 400
##   2   1   1   4  13   2   7   3   4   1   1   1   6   1   4   1   3   3
## 407 412 416 419 423 428
##   1   2   3   1  18   1
```

Evaluate Balance

```
trt.factor <- rep(c("Treat","Control"),c(N,N))
means <- apply(match.data[,-1],2,function(x) tapply(x,trt.factor,mean))
sds <- apply(match.data[,-1],2,function(x) tapply(x,trt.factor,sd))
varratio <- sds[1,]^2/sds[2,]^2
ks.p <- apply(match.data[,-1],2,function(x) ks.test(x[1:N],x[{N+1}:{2*N}])$p.value)
```

```
## Warning: p-value will be approximate in the presence of ties
## Warning: p-value will be approximate in the presence of ties
## Warning: p-value will be approximate in the presence of ties
## Warning: p-value will be approximate in the presence of ties
## Warning: p-value will be approximate in the presence of ties
## Warning: p-value will be approximate in the presence of ties
## Warning: p-value will be approximate in the presence of ties
## Warning: p-value will be approximate in the presence of ties
## Warning: p-value will be approximate in the presence of ties
```

```r
t.p <- apply(match.data[,-1],2,function(x) t.test(x[1:N],x[{N+1}:{2*N}])$p.value)
```

View Matched Balance

```r
round(t(rbind(means, sds, varratio, ks.p, t.p)), 3)[-9, ]
```

```
##            Control    Treat  Control    Treat varratio  ks.p   t.p
## age         25.546   25.816    8.745    7.155    1.494 0.003 0.745
## educ        10.443   10.346    1.841    2.011    0.838 0.999 0.628
## black        0.832    0.843    0.374    0.365    1.055 1.000 0.779
## hispan       0.059    0.059    0.237    0.237    1.000 1.000 1.000
## married      0.184    0.189    0.388    0.393    0.978 1.000 0.894
## nodegree     0.703    0.708    0.458    0.456    1.011 1.000 0.910
## re74      1871.365 2095.574 4213.141 4886.620    0.743 0.008 0.637
## re75      1141.974 1532.055 2428.479 3219.251    0.569 0.577 0.189
```

And Estimate ATT

```r
mahal.match.mod <- lm(re78 ~ treat + age + educ + black + hispan + married +
    nodegree + re74 + re75, match.data)
coefs <- c(coefs, mahal.match = coef(mahal.match.mod)[2])
coefs
```

```
##        base.treat   ipw.logit.treat    ipw.bart.treat  pmat.logit.treat
##            1548.2            1332.0            1303.7            1963.9
##   pmat.bart.treat mahal.match.treat
##            1351.9             417.8
```

Genetic Matching

- This is a fancy and very effective algorithm developed by Jas Sekhon.
- The basic logic is as follows:
    - Start with the mahalanobis distance solution.
    - Evaluate balance (by default, by paired t-tests and KS tests on covariates)
    - Tweak the covariance matrix.
    - New matching solution
    - See if balance improved
    - Iterate

- It uses a genetic algorithm to tweak the covariance matrix.
- It is NOT fast. And you should use a large value of `pop.size`, which will make it even slower (10 is WAY too low. The default is 100, and even that is too low). Also, you should use the available wrapper functions via MatchIt (or even just in the Matching package)

. . .

```
require(Matching)
require(rgenoud)
# gmatch <- GenMatch(lalonde$treat,lalonde[,-c(1,ncol(lalonde))],pop.size =
# 1000,ties=FALSE,print.level=0)
matches <- gmatch$matches[, 2]
match.data <- subset(lalonde, treat == 1)
match.data <- rbind(match.data, lalonde[matches, ])
```

Balance Tests for genMatch

```
trt.factor <- rep(c("Treat","Control"),c(N,N))
means <- apply(match.data[,-1],2,function(x) tapply(x,trt.factor,mean))
sds <- apply(match.data[,-1],2,function(x) tapply(x,trt.factor,sd))
varratio <- sds[1,]^2/sds[2,]^2
ks.p <- apply(match.data[,-1],2,function(x) ks.test(x[1:N],x[{N+1}:{2*N}])$p.value)
```

```
## Warning: p-value will be approximate in the presence of ties
## Warning: p-value will be approximate in the presence of ties
## Warning: p-value will be approximate in the presence of ties
## Warning: p-value will be approximate in the presence of ties
## Warning: p-value will be approximate in the presence of ties
## Warning: p-value will be approximate in the presence of ties
## Warning: p-value will be approximate in the presence of ties
## Warning: p-value will be approximate in the presence of ties
## Warning: p-value will be approximate in the presence of ties
```

```
t.p <- apply(match.data[,-1],2,function(x) t.test(x[1:N],x[{N+1}:{2*N}])$p.value)
```

View Matches Balance

- You won't find better results for these metrics (doesn't necessarily make it "best", though)

. . .

```r
round(t(rbind(means, sds, varratio, ks.p, t.p)), 3)[-9, ]
```

```
##            Control    Treat  Control    Treat varratio  ks.p   t.p
## age         25.724   25.816    7.729    7.155    1.167 0.416 0.906
## educ        10.362   10.346    1.949    2.011    0.939 0.493 0.937
## black        0.838    0.843    0.370    0.365    1.028 1.000 0.887
## hispan       0.059    0.059    0.237    0.237    1.000 1.000 1.000
## married      0.222    0.189    0.416    0.393    1.125 1.000 0.441
## nodegree     0.708    0.708    0.456    0.456    1.000 1.000 1.000
## re74      2062.483 2095.574 4446.327 4886.620    0.828 0.665 0.946
## re75      1354.161 1532.055 2795.864 3219.251    0.754 0.899 0.571
```

And Estimate ATT

```r
gen.match.mod <- lm(re78 ~ treat + age + educ + black + hispan + married + nodegree +
    re74 + re75, match.data)
coefs <- c(coefs, gen.match = coef(gen.match.mod)[2])
coefs
```

```
##        base.treat   ipw.logit.treat    ipw.bart.treat  pmat.logit.treat
##            1548.2            1332.0            1303.7            1963.9
##   pmat.bart.treat mahal.match.treat   gen.match.treat
##            1351.9             417.8            1003.4
```

CEM

- CEM just creates bins along each covariate dimension (either pre-specified or automatic)
- Units lying in the same strata are then matched together
- Curse of dimensionality means that with lots of covariates, we'll only rarely have units in the same strata.
- What does that mean we're estimating? Is it the ATT?

. . .

```r
# install.packages('cem',repos='http://r.iq.harvard.edu', type='source')
require(cem)
cem.match <- cem(treatment = "treat", data = lalonde, drop = "re78")
cem.match
```

```
##           G0  G1
## All      429 185
## Matched   78  68
## Unmatched 351 117
```

```
cem.mod <- lm(re78 ~ treat + age + educ + black + hispan + married + nodegree +
    re74 + re75, lalonde, weights = cem.match$w)
coefs <- c(coefs, coef(cem.mod)[2])
coefs
```

```
##        base.treat   ipw.logit.treat    ipw.bart.treat  pmat.logit.treat
##            1548.2            1332.0            1303.7            1963.9
##   pmat.bart.treat mahal.match.treat   gen.match.treat             treat
##            1351.9             417.8            1003.4             744.2
```

Tweaking CEM

```
cutpoints <- list(age = c(25, 35), educ = c(6, 12), re74 = c(100, 5000), re75 = c(100,
    5000))
cem.tweak.match <- cem(treatment = "treat", data = lalonde, drop = "re78", cutpoints = cutpo
cem.tweak.match
```

```
##           G0  G1
## All      429 185
## Matched  168 160
## Unmatched 261  25
```

```
cem.tweak.mod <- lm(re78 ~ treat + age + educ + black + hispan + married + nodegree +
    re74 + re75, lalonde, weights = cem.tweak.match$w)
coefs <- c(coefs, coef(cem.tweak.mod)[2])
coefs
```

```
##        base.treat   ipw.logit.treat    ipw.bart.treat  pmat.logit.treat
##            1548.2            1332.0            1303.7            1963.9
##   pmat.bart.treat mahal.match.treat   gen.match.treat             treat
##            1351.9             417.8            1003.4             744.2
##             treat
##            -451.8
```

Entropy Balance

- What if we framed preprocessing explicitly as an optimization problem?
- We want to minimize difference between empirical moments of treatment and control by varying the weights accorded to individual observations in our dataset.
- All while keeping weights relatively stable.
- This is "entropy balancing" created by Jens Hainmueller.
- We optimize the following problem: $\min_{\boldsymbol{W},\lambda_0,\boldsymbol{\lambda}} L^p = \sum_{D=0} w_i \log(w_i/q_i) + \sum_{r=1}^R \lambda_r \left(\sum_{D=0} w_i c_{ri}(X_i) - m_r\right) + (\lambda_0 - 1)\left(\sum_{D=0} w_i - 1\right)$

. . .

```
require(ebal, quietly = TRUE)
ebal.match <- ebalance(lalonde$treat, lalonde[, -c(1, ncol(lalonde))])

## Converged within tolerance

ebal.w <- c(rep(1, N), ebal.match$w)
ebal.mod <- lm(re78 ~ treat + age + educ + black + hispan + married + nodegree +
    re74 + re75, lalonde, weights = ebal.w)
```

Final Estimates

```
coefs <- c(coefs, ebal = coef(ebal.mod)[2])
coefs

##        base.treat   ipw.logit.treat    ipw.bart.treat  pmat.logit.treat
##            1548.2            1332.0            1303.7            1963.9
##   pmat.bart.treat mahal.match.treat    gen.match.treat             treat
##            1351.9             417.8            1003.4             744.2
##             treat        ebal.treat
##            -451.8            1273.3
```