Introduction to Matching Methods for Causal Inference

Zeren Li

Introduction

- Useful for observational studies
- Sometimes better than simply using control variables.
- It's popular! (Since the 1970s, 93,700 articles uses matching method)

Regression v.s. Matching ¶

Both methods assumes **Selection on Observables**

- assume Potential outcomes (y_t, y_c) are independent of treatment after controlling for observed covariates x (Conditional Independence Assumption)
- use a set of covariates to adjust the estimation

Both methods are **weight scheme**. Regression aims to minimize squared errors, so observations on the margins have more weight. Matching puts the emphasis on observations that have similar covariates.

Two methods can work together! (e.g. Regression with Propensity Score Matching)

Why matching

Reduce Model Dependence:

- Does not rely on modeling assumptions of linear dependencies across the entire range of the control variables to do controlled comparisons.
- Enhances apples to apples comparisons: Making sure the treated and untreated observations are balanced.

Matching Reduces Model Dependence

Figure from Ho, Imai, King, Stuart (2007)
alt text

Matching Reduces Model Dependence

Figure from Ho, Imai, King, Stuart (2007)
alt text

Matching Reduces Model Dependence

Figure from Ho, Imai, King, Stuart (2007)
alt text

Matching Reduces Model Dependence

Figure from Ho, Imai, King, Stuart (2007)
alt text

Matching Reduces Model Dependence

Figure from Ho, Imai, King, Stuart (2007)
alt text

Matching & Common Support

- Common support ensures the treated have the control with “close” P(x)
- Lack of common support appears in tails of distributions
- Larger sample of eligible nonparticipants helps matching
- Poor common support can induce bias in matching estimator
  - E.g., if no matches may drop nonrandom subset of participants

Matching & Common Support

Khandler et al(2010)

When Matching

- usually used in observational studies
- treatment and control groups are imbalanced
- a lot of covariates may be correlated with Y and D
- pre-process the data before running regression
- as a robustness check

Steps of Matching

Defining closeness I: Variables to Include

Defining “closeness”: the distance measure used to determine whether an individual is a good match for another

- Be liberal in terms of including variables that may be associated with treatment assignment & outcomes
- In small sample, select variables without using the observed outcomes, but based on the previous research
- DO NOT include post-treatment variables

Defining closeness II: Metric of Closeness

- Exact
- Mahalanobis
- Propensity Score

Matching Methods

Implement a matching method, given that measure of closeness

- Nearest neighbor matching
- Subclassification, Full Matching, and Weighting

Assessing Common Support

- show distributions of propensity scores of treat&control groups
- substantial overlap of the propensity score distributions in the two groups

Matching Diagnostics

Assess the quality of the resulting matched samples

- diagnose the quality of the matching through an assessment of **covariate balance**
- **covariate balance** is defined as similarity of empirical distributions of the full set of covariates in the matched treated and control groups
- **Numerical diagnostics** (e.g. standardized difference in means): difference in means of each covariate, divided by the standard deviation in the full treated group: $\frac{X_t - X_c}{\sigma_t}$
- Graphical diagnostics

Analysis of the outcome

Analyze of the outcome and estimation of the treatment effect

- **Matching** and **Regression** have been shown to work best in combination
- After k:1 matching
  - do same analysis as what did with original data, but using the matched data instead, e.g. regression
  - weight needs to be incorporated
- After subclassification or full matching

Propentiy Score Matching Example

We practise the matching method by replicating the prominent paper on job training. The 'lalonde.dta' consists of the real earnings in the year 1978 (the response), a treatment indicator, and a number of demographic variables (controls). Here are the variable definitions.

- age: age in years.
- educ: years of schooling.
- black: indicator variable for blacks.
- hisp: indicator variable for Hispanics.
- married: indicator variable for martial status.
- nodegr: indicator variable for high school diploma.
- re74: real earnings in 1974.
- re75: real earnings in 1975.
- re78: real earnings in 1978.
- treat: an indicator variable for treatment status.

The original paper is here: Robert Lalonde, "Evaluating the Econometric Evaluations of Training Programs", American Economic Review, Vol. 76, pp. 604-620

In [41]:
```
# load packages
# now pymatch only works with older versions of pandas, please make sure you have a version older than 0.23.4
# If not, try this to downgrade pandas: pip install pandas==0.23.4
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from pymatch.Matcher import Matcher
%matplotlib inline
```

In [42]:
```
lalonde = pd.read_stata("lalonde.dta")
lalonde.head()
```

Out[42]:

	treated	age	education	black	married	nodegree	re74	re75	re78
0	1.0	33	12	0	1	0	0.000000	0.000000	12418.070312
1	1.0	20	12	0	1	0	8644.156250	8644.156250	11656.505859
2	0.0	39	12	1	1	0	19785.320312	6608.137207	499.257202
3	1.0	49	8	0	1	1	9714.596680	7285.947754	16717.121094
4	0.0	26	8	0	1	1	37211.757812	36941.265625	30247.500000

In [43]:
```
lalonde.groupby("treated").mean()
```

Out[43]:

	age	education	black	married	nodegree	re74	re75	r
treated								
0.0	24.447059	10.188235	0.800000	0.157647	0.814118	3672.485151	3026.682738	5090.048
1.0	24.626263	10.380471	0.801347	0.168350	0.730640	3570.998967	3066.098189	5976.352

Data Prepreation

load the `lalonde.dta` , show the raw mean difference in earning between the control group and the treated group.

In [44]:
```
data = lalonde[["treated","age","education", "black", "married","nodegre
e","hispanic","re74","re75","re78"]]
data.head()
```

Out[44]:

	treated	age	education	black	married	nodegree	hispanic	re74	re75	
0	1.0	33	12	0	1	0	0	0.000000	0.000000	124
1	1.0	20	12	0	1	0	0	8644.156250	8644.156250	116
2	0.0	39	12	1	1	0	0	19785.320312	6608.137207	4
3	1.0	49	8	0	1	1	0	9714.596680	7285.947754	167
4	0.0	26	8	0	1	1	0	37211.757812	36941.265625	302

In [45]:
```
treatment = data[data.treated == 1]
control = data[data.treated == 0]
```

Fit Propensity Score Model(s)

First we initialize the Matcher object:

- `Matcher` shows the formula used to fit logistic regression model(s) and the number of records in the majority/minority class.
- we use the covariates on the right side of the equation to estimate the probability of being treated (treated = 1).
- Any covariates passed to the (optional) exclude parameter will be ignored from the model fitting process. This parameter is particularly useful for unique identifiers like a userid.

In [46]:
```
m = Matcher(control, treatment, yvar="treated", exclude=["re78"])
```

```
Formula:
treated ~ age+black+education+hispanic+married+nodegree+re74+re75
n majority: 425
n minority: 297
```

- The model shows an **imbalance** in our data. The majority group (control group) having more observations than the minority group (treatment group). We account for this by setting `balance=True` in `Matcher.fit_scores()` .
- This tells `Matcher()` to sample from the majority group when fitting the logistic regression model(s) so that the groups are of equal size.
- When undersampling this way, it is highly recommended that nmodels is explicitly assigned to a integer much larger than 1. This ensures that more of the majority group is contributing to the generation of propensity scores.
- The value of this integer should depend on the severity of the imbalance: here we use nmodels=100.

In [47]:
```
np.random.seed(2020)
m.fit_scores(balance=True, nmodels=100)
```

```
Fitting Models on Balanced Samples: 100\100
Average Accuracy: 55.53%
```

- The average accuracy of our 100 models is 55.31\%, suggesting that there's separability within our data and justifiying the need for the matching procedure.
- We don't pay **too much** attention to the estimates of logistic models
- We are interested in **the predicted value of the model** (propensity score of each observation).

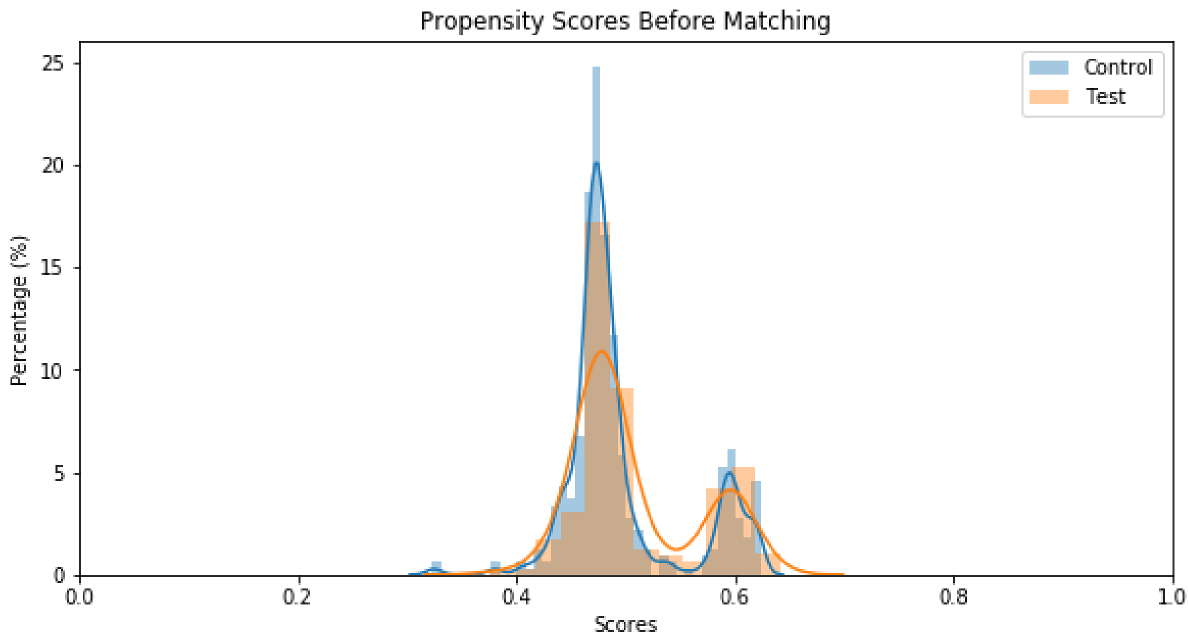
Assign a propensity score to each record in the dataset

In [48]:
```
m.predict_scores()
```

Evaluating Common Support

We can view the distributions using `m.plot_score()` . It shows that the test and control have good common support.

In [49]:
```
m.plot_scores()
```



K:1 Matching

Then we match one observation from the majority group to each record in the minority group with **replacement**. It means a single majority record can be matched to multiple minority records.

`Matcher` assigns a unique record_id to each record in the test and control groups so this can be addressed after matching. If subsequent modeling is planned, one might consider weighting models using a weight vector of 1/f for each record, f being an observation's frequency in the matched dataset. Thankfully Matcher can handle all of this for you :).

In [50]:
```
m.match(method="min", nmatches=1) # you can also tune the threshold
```

The matching result is shown as follows: the bulk of our matched-majority-group observation occur only once, 38 occur twice, ... etc.

In [51]:
```
m.record_frequency()
```

Out[51]:

	freq	n_records
0	1	422
1	2	38
2	3	21
3	4	7
4	5	1

We can preemptively generate a weight vector using `Matcher.assign_weight_vector()`

In [52]:
```
m.assign_weight_vector()
```

Matched Data

Let's take a look at our matched data!

Note that in addition to the weight vector, Matcher has also assigned a match_id to each record indicating our (in this cased) paired matches since we use nmatches=1. We can verify that matched records have scores within 0.0001 of each other.

In [53]:
```
matched = m.matched_data.sort_values("match_id")
matched.head(10)
```

Out[53]:

	record_id	weight	age	black	education	hispanic	married	nodegree	re74	
8	12	0.250000	18	0	12	0	0	0	735.205078	7
297	425	1.000000	33	0	12	0	1	0	0.000000	
9	12	0.250000	18	0	12	0	0	0	735.205078	7
298	426	1.000000	20	0	12	0	1	0	8644.156250	86
106	152	0.250000	19	1	9	0	1	1	0.000000	
299	427	1.000000	49	0	8	0	1	1	9714.596680	72
170	216	0.333333	19	1	12	0	0	0	8417.000000	28
300	428	1.000000	33	1	12	0	1	0	20279.949219	109
93	131	1.000000	25	1	10	0	1	1	0.000000	
301	429	1.000000	35	1	9	0	1	1	13602.429688	138

Assess Matched Data

In [54]:
```
matched.groupby("treated").mean()
```

Out[54]:

	record_id	weight	age	black	education	hispanic	married	nodegree	
treated									
0	203.814815	0.646465	24.292929	0.774411	10.313131	0.111111	0.148148	0.73064	35
1	573.000000	1.000000	24.626263	0.801347	10.380471	0.094276	0.168350	0.73064	35

In [55]:
```
# standardized mean difference
def stmean(x):
    x_t = matched[x][matched.treated == 1].mean()
    x_c = matched[x][matched.treated == 0].mean()
    sd_t = matched[x][matched.treated == 1].std()
    stdmean =  (x_t-x_c)/sd_t
    print(stdmean)

covariates = ["age","education", "black", "married","nodegree","hispani
c","re74","re75"]
for x in covariates:
    stmean(x)
```

```
0.049852505643968814
0.03704661050257367
0.06739739852585085
0.053899582655417604
0.0
-0.05751517950116839
0.010446133396393737
0.005471770201405774
```

Regression with the Matched Data

According to Ho, Imai, King, and Stuart (2007), we can use regression to fit the model.

In [56]:
```python
# load regression package
import statsmodels.formula.api as smf
import statsmodels.api as sm

weight = matched["weight"].values
smf.wls('re78 ~treated',matched, weights = weight ).fit().summary()
```

Out[56]:

WLS Regression Results

Dep. Variable:	re78	**R-squared:**	0.002
Model:	WLS	**Adj. R-squared:**	0.001
Method:	Least Squares	**F-statistic:**	1.419
Date:	Mon, 24 Feb 2020	**Prob (F-statistic):**	0.234
Time:	11:12:02	**Log-Likelihood:**	-6085.8
No. Observations:	594	**AIC:**	1.218e+04
Df Residuals:	592	**BIC:**	1.218e+04
Df Model:	1		
Covariance Type:	nonrobust		

	coef	**std err**	**t**	**P>|t|**	**[0.025**	**0.975]**
Intercept	5323.2134	427.232	12.460	0.000	4484.138	6162.288
treated	653.1386	548.201	1.191	0.234	-423.517	1729.795

Omnibus:	394.296	**Durbin-Watson:**	1.933
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	6572.060
Skew:	2.678	**Prob(JB):**	0.00
Kurtosis:	18.390	**Cond. No.**	2.95

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

In [57]:
```
# regression with unmatached data
smf.ols('re78 ~treated',lalonde ).fit().summary()
```

Out[57]:

OLS Regression Results

Dep. Variable:	re78	**R-squared:**	0.005
Model:	OLS	**Adj. R-squared:**	0.003
Method:	Least Squares	**F-statistic:**	3.525
Date:	Mon, 24 Feb 2020	**Prob (F-statistic):**	0.0609
Time:	11:12:03	**Log-Likelihood:**	-7333.1
No. Observations:	722	**AIC:**	1.467e+04
Df Residuals:	720	**BIC:**	1.468e+04
Df Model:	1		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	5090.0482	302.783	16.811	0.000	4495.606	5684.491
treated	886.3038	472.086	1.877	0.061	-40.526	1813.134

Omnibus:	384.449	**Durbin-Watson:**	2.072
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	3767.288
Skew:	2.195	**Prob(JB):**	0.00
Kurtosis:	13.294	**Cond. No.**	2.46

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

In [58]:

```
# you could also add covariates into the regression
smf.wls('re78 ~treated + age+ education + black+ married + nodegree + hi
spanic + re74 + re75', matched, weights =weight ).fit().summary()
```

Out[58]:

WLS Regression Results

Dep. Variable:	re78	**R-squared:**	0.029
Model:	WLS	**Adj. R-squared:**	0.014
Method:	Least Squares	**F-statistic:**	1.925
Date:	Mon, 24 Feb 2020	**Prob (F-statistic):**	0.0460
Time:	11:12:04	**Log-Likelihood:**	-6077.8
No. Observations:	594	**AIC:**	1.218e+04
Df Residuals:	584	**BIC:**	1.222e+04
Df Model:	9		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	367.5721	2940.969	0.125	0.901	-5408.592	6143.736
treated	606.3762	546.507	1.110	0.268	-466.983	1679.735
age	29.9408	40.656	0.736	0.462	-49.909	109.790
education	398.9888	200.406	1.991	0.047	5.384	792.594
black	-261.3799	864.816	-0.302	0.763	-1959.908	1437.148
married	774.7219	765.501	1.012	0.312	-728.749	2278.193
nodegree	-189.5796	834.935	-0.227	0.820	-1829.420	1450.261
hispanic	858.3063	1164.786	0.737	0.461	-1429.373	3145.986
re74	-0.0135	0.096	-0.140	0.889	-0.202	0.176
re75	0.1005	0.113	0.892	0.373	-0.121	0.322

Omnibus:	397.540	**Durbin-Watson:**	1.931
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	7054.114
Skew:	2.685	**Prob(JB):**	0.00
Kurtosis:	19.006	**Cond. No.**	1.03e+05

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 1.03e+05. This might indicate that there are
strong multicollinearity or other numerical problems.

In [ ]: