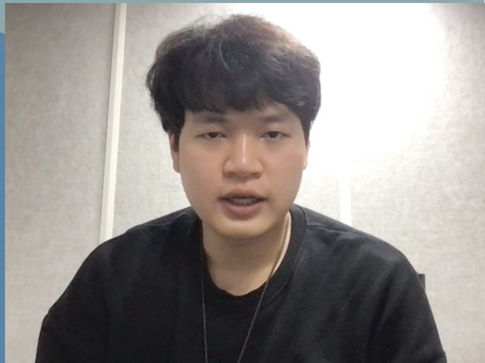
Week 14 – Dimensionality reduction

L14-07. Partial least square (PLS) and Canonical correlation analyisis (CCA) (How)

Donghee and Jungwoo



PLS and CCA

are supervised dimension reduction! which allows multivariate dependent variables!

Use information of dependent variable to reduce dimension!

CCA cost function was…

$$\max_{W_x, W_y} \quad \operatorname{tr}\left(W_x^T X Y^T W_y\right)$$
$$\text{subject to} \quad W_x^T X X^T W_x = I, \;\; W_y^T Y Y^T W_y = I$$

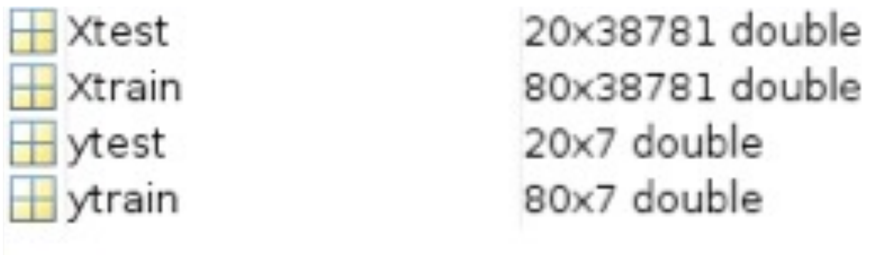
PLS cost function was…

$$\max cov({w_x}^T X, YTWy)$$

Partial least Square regression

Resting state data of 100 participants…

We're going to use resting state functional connectivity (RSFC) data
to see if it can explain depression related variables.



Xtest	20x38781 double
Xtrain	80x38781 double
ytest	20x7 double
ytrain	80x7 double

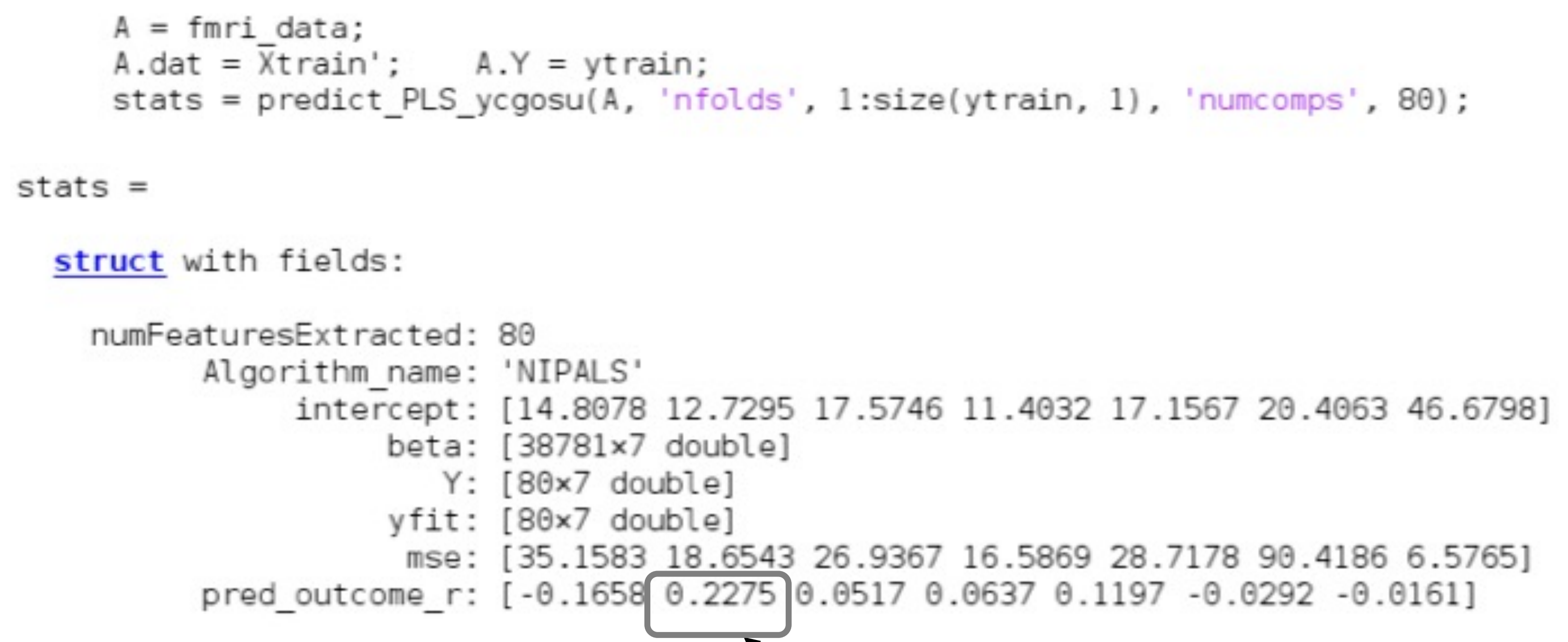
80: number of used training datasets
20: number of used testing datasets
38781: used 279 parcellation to calculated RSFC so it results in 279 * 278 / 2
7: number of depression related variables. (CES-D, RRS 3 subscales, MASQ 3 subscales)

More info about behavioral measures

http://www.chcr.brown.edu/pcoc/cesdscale.pdf
http://www-personal.umich.edu/~gonzo/RuminationScale.pdf
https://arc.psych.wisc.edu/self-report/mood-and-anxiety-symptom-questionnaire-masq/

Partial least Square regression

PLS in one simple block of code!



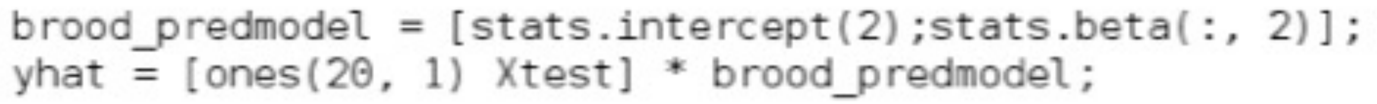
```
    A = fmri_data;
    A.dat = Xtrain';    A.Y = ytrain;
    stats = predict_PLS_ycgosu(A, 'nfolds', 1:size(ytrain, 1), 'numcomps', 80);

stats =

  struct with fields:

    numFeaturesExtracted: 80
          Algorithm_name: 'NIPALS'
               intercept: [14.8078 12.7295 17.5746 11.4032 17.1567 20.4063 46.6798]
                    beta: [38781×7 double]
                       Y: [80×7 double]
                    yfit: [80×7 double]
                     mse: [35.1583 18.6543 26.9367 16.5869 28.7178 90.4186 6.5765]
         pred_outcome_r: [-0.1658 0.2275 0.0517 0.0637 0.1197 -0.0292 -0.0161]
```

Only the brooding subscale of RRS survived!

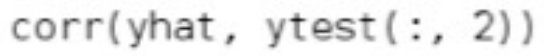
You can find the 'predict_PLS_ycgosu' function in the github below!
https://github.com/didch1789/yanchogosu_toolbox

Partial least Square regression

We might say that we found a model that predicts brooding by PLS…!
However, since it is susceptible to overfitting…let's try it in an independent dataset!

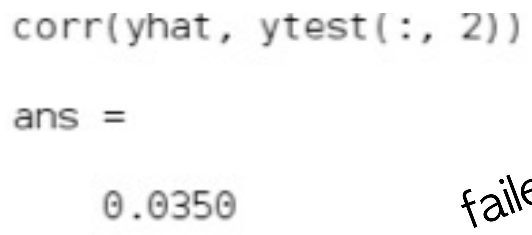


```
brood_predmodel = [stats.intercept(2);stats.beta(:, 2)];
yhat = [ones(20, 1) Xtest] * brood_predmodel;
```



```
corr(yhat, ytest(:, 2))
```

results show…



```
corr(yhat, ytest(:, 2))

ans =

    0.0350
```

failed…!

You might ask…

Is it different from using principal component regression(PCR) for each dependent variables?

Thoretically, yes.

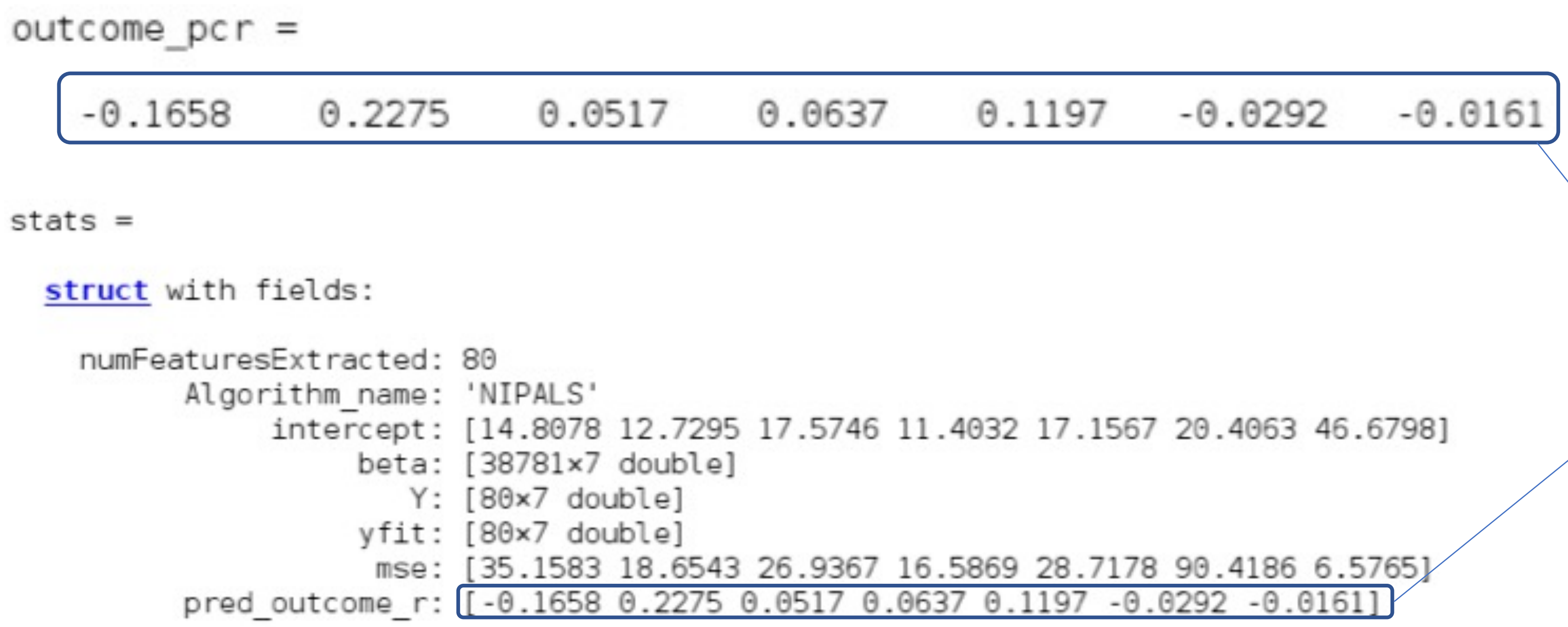
Since PCR uses dimensions independent from Y,
while PLS uses demensions that are more fitted (aligned) to Y.

How about in the actual result?

Partial least Square regression

Using same cross-validation (leave-one out) …

I ran PCR code and the results…!



```
outcome_pcr =

   -0.1658    0.2275    0.0517    0.0637    0.1197   -0.0292   -0.0161


stats =

  struct with fields:

    numFeaturesExtracted: 80
          Algorithm_name: 'NIPALS'
               intercept: [14.8078 12.7295 17.5746 11.4032 17.1567 20.4063 46.6798]
                    beta: [38781×7 double]
                       Y: [80×7 double]
                    yfit: [80×7 double]
                     mse: [35.1583 18.6543 26.9367 16.5869 28.7178 90.4186 6.5765]
           pred_outcome_r: [-0.1658 0.2275 0.0517 0.0637 0.1197 -0.0292 -0.0161]
```

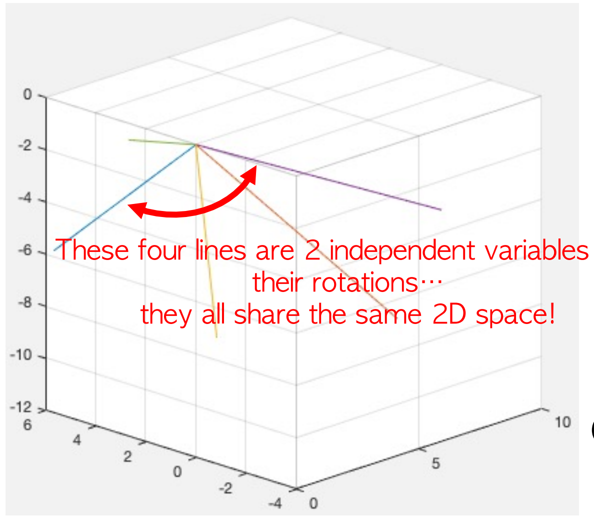
EXACTLY SAME?

Partial least Square regression

How is this possible…?

The reason is that…I used full components for both algorithms.

It means reduced dimension in both PLS and PCR retains all the variance of X which would results in **SAME** amount of variance left in reduced dimensions.



If you want to grasp more about the idea…
think about 2D data and one dependent variables…
rotation of that 2D matrix (which is same as retaining their variance)
will not change the explained variance of the dependent variables.

If you are a linear algebra lover, you can think it in a geometrical way and see why the explained variance does not change!

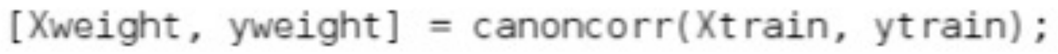
Of course, results will change if you use different numbers of dimensions for each algorithm.

Canoncial correlation analysis

Let's do CCA with the same dataset…

To figure out which features of dependent variables are best explained by which features of RSFC.

As in PLS, simple one block of code will do!

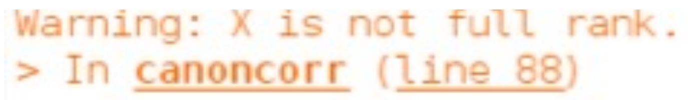


```
[Xweight, yweight] = canoncorr(Xtrain, ytrain);
```

Canoncial correlation analysis

But···

If you run this code, it will show out the warning···



```
Warning: X is not full rank.
> In canoncorr (line 88)
```

This is due to our Xtrain (80 x 38781)··· since it has lot more features than observations···
It means that it does not have a single solution (one linear combination) to yield maximum correlation!
And this is the usual case in fMRI!

Then···

There are several ways to deal with the problem···

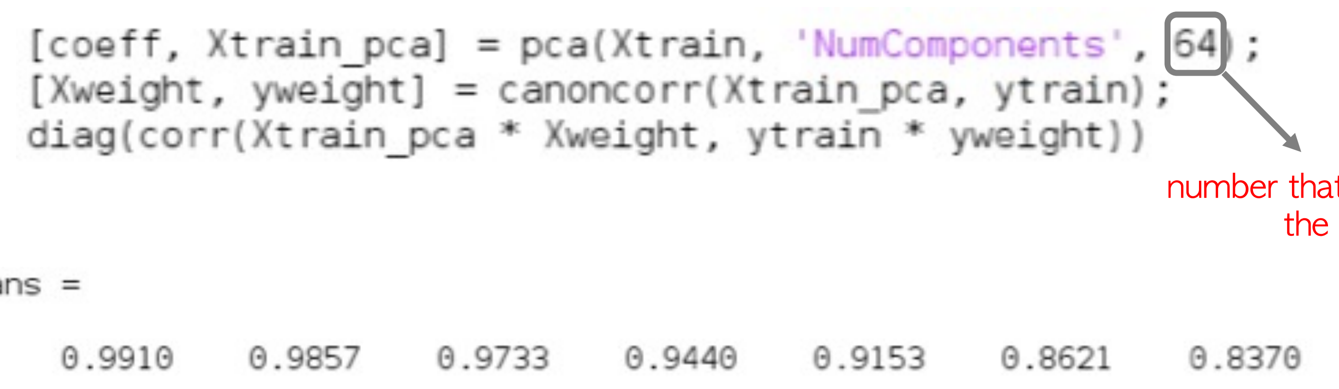
Usual cases are···

1) Dimension reduction by PCA and then do CCA.
2) Add contraints (giving the term with L1 or L2 norm penalty. called sparse CCA)
3) or do both ···!

CCA itself is for finding latent space···
so PCA and then do CCA might sounds weird but these kinds of methods are quite prevalent in the field as far as it makes sense.
(e.g. LASSO-PCR)

Canoncial correlation analysis

Let's try PCA and then CCA



```
[coeff, Xtrain_pca] = pca(Xtrain, 'NumComponents', 64);
[Xweight, yweight] = canoncorr(Xtrain_pca, ytrain);
diag(corr(Xtrain_pca * Xweight, ytrain * yweight))
```

```
ans =

    0.9910    0.9857    0.9733    0.9440    0.9153    0.8621    0.8370
```

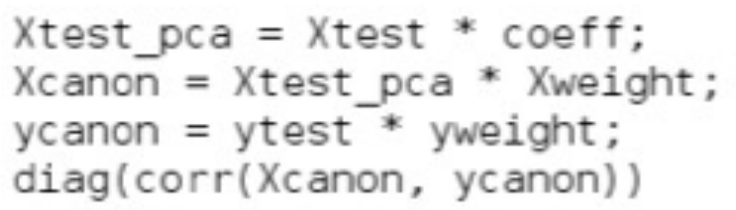
As expected, linear combination weights chosen by CCA shows high correlation as expected…

and linear combinations of each X and Y are often called 'canonical variates'

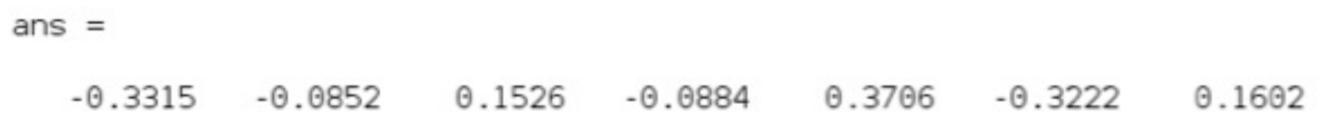
Canoncial correlation analysis

By looking each X and Y weights …
We could see which behavioral features are matched with connectivity features…

unless it is not overfitted…!



```
Xtest_pca = Xtest * coeff;
Xcanon = Xtest_pca * Xweight;
ycanon = ytest * yweight;
diag(corr(Xcanon, ycanon))
```



```
ans =

   -0.3315   -0.0852    0.1526   -0.0884    0.3706   -0.3222    0.1602
```

maybe needs
further testing…!

failed…!

Last comments…

Important to note that failure of generalization doesn't mean that there are no relationship between independent and dependent variables…!
(Actually this applies to all other machine learning algorithms!)

Even though your algorithm succeeds, it is important to test it on an independent dataset!
(Especially important for CCA and PLS!)

1) Sensitivity
2) Specificity
3) Generalizability

See how previous works have used the algorithm and interpreted their results!

Cocoan 101

https://cocoanlab.github.io