Principal Component Regression and Partial Least Squares

UBCO MDS — DATA 572

Motivation

- Recall: PCA is an unsupervised technique. If you look back, there is no inherent response variable (Y) that we are optimizing.
- We simply rotated our original predictors in a ‘clever’ manner, and toss out the rotations that do not appear to contain important information.
- We then discussed some interpretations of these rotations, etc. Rarely would this be the end of the story...

Motivation

- PCA (once you've removed some components) is generally viewed as a dimensionality reduction technique.
- Most commonly, you'll be interested in taking that reduced space, and using it for more machine learning! (Be it supervised, or unsupervised)
- Let's get into some details...we'll start with the simplest setup...

Principal Components Regression

- Principal Components Regression, or PCReg for short, is the most natural (though not necessarily the most useful) form for making use of PCA in a supervised context.

- Suppose we perform PCA on X, which is an $n \times p$ matrix of observations (on p predictors), and decide to retain h components. AKA
$$\Sigma \approx \mathrm{P}\Lambda\mathrm{P}'$$
where Σ is a $p \times p$ covariance/correlation matrix, P is a $p \times h$ matrix of eigenvectors (loadings) and Λ is a $h \times h$ diagonal matrix with the h largest eigenvalues.

- We then map our $n \times p$ matrix X to an $n \times h$ matrix S via $\mathrm{S} = \mathrm{XP}$, these are the scores.

Principal Components Regression

- Suppose that a continuous response variable Y is on hand, then we can easily assume the following model for linear regression

$$Y = \beta_0 + \beta_1 S_1 + \beta_2 S_2 + \cdots + \beta_h S_h + \epsilon$$

with all the usual linear regression assumptions (for inferential purposes).

the S here are the principal components

- And that's it! That is PCReg: using your retained principal components as predictors for a continuous response.

- So for practical purposes, it's straightforward. It's worth thinking a tad more deeply about what's being achieved though...

PCReg Deeper

- Firstly, if $h = p$ (aka, you retain all PCs), then the model is equivalent to ordinary least squares modelling in terms of $\hat{Y}$. Obviously the estimated coefficients differ, etc...but you do not lose any predictive power as you are not losing any information.

- Next, consider a pro...

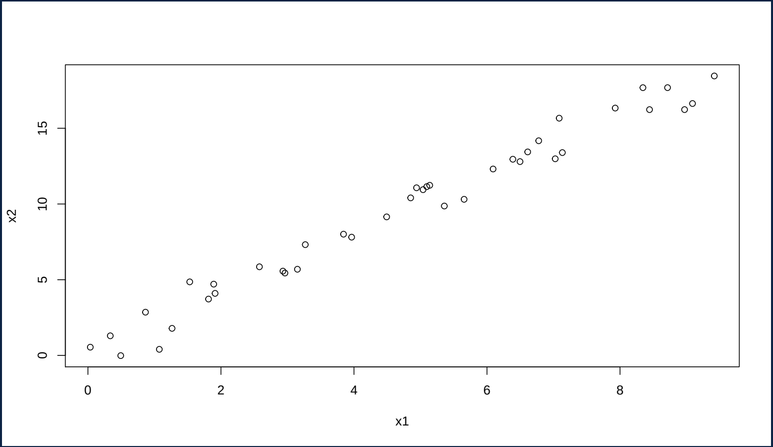
- Congratulations, you've just taken care of the problem of multicollinearity. How?

PCReg Deeper

- Finally, some cons...
- PCReg can be viewed as a discretized regularizer, removing low variance components (but not entire predictors). So it has some commonality with ridge regression, but due to ridge regression's smooth regularization, ridgereg is often a better option than PCReg.
- There is absolutely NO guarantee that higher variance components — which is with respect to the predictor space — contain more predictive power towards Y!

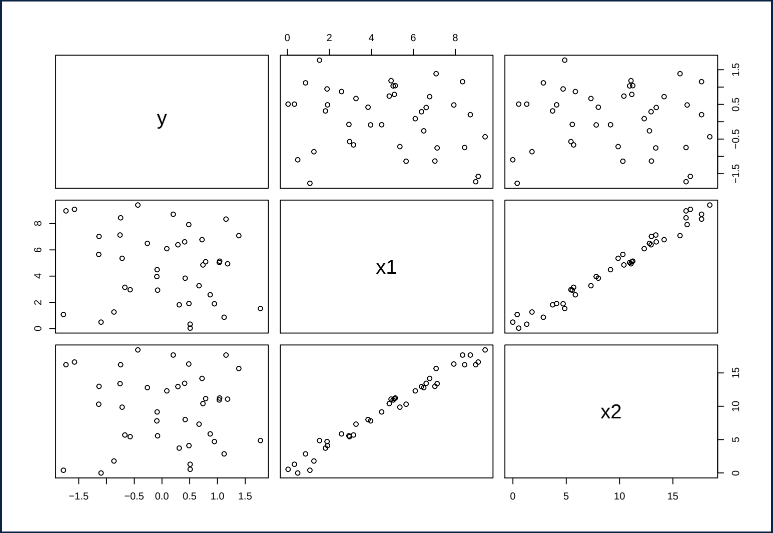
A cautionary tale...

- Recall our motivating simulation from the PCA lecture



A cautionary tale...

- I've added a response variable



A cautionary tale...

```
> linmod <- lm(y~x1+x2)
> summary(linmod)

Call:
lm(formula = y ~ x1 + x2)

Residuals:
     Min       1Q   Median       3Q      Max 
-0.26489 -0.07545 -0.01078  0.08809  0.21513 

Coefficients:
             Estimate Std. Error t value Pr(>|t|)    
(Intercept)  0.007547   0.038730   0.195    0.847    
x1          -2.040311   0.043519 -46.883   <2e-16 ***
x2           1.020113   0.022127  46.103   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1 

Residual standard error: 0.1199 on 37 degrees of freedom
Multiple R-squared:  0.9835,Adjusted R-squared:  0.9826 
F-statistic:  1100 on 2 and 37 DF,  p-value: < 2.2e-16
```

A cautionary tale...

```
> pcs <- prcomp(cbind(x1, x2))
> summary(pcs)
Importance of components:
                          PC1     PC2
Standard deviation     6.0766 0.39399
Proportion of Variance 0.9958 0.00419
Cumulative Proportion  0.9958 1.00000
```

A cautionary tale...

```
> linpc <- lm(y~pcs$x[,1])
> summary(linpc)

Call:
lm(formula = y ~ pcs$x[, 1])

Residuals:
    Min      1Q  Median      3Q     Max
-1.9419 -0.7712  0.2144  0.6891  1.6439

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  0.08484    0.14517   0.584    0.562
pcs$x[, 1]   0.00834    0.02419   0.345    0.732

Residual standard error: 0.9181 on 38 degrees of freedom
Multiple R-squared:  0.003117,Adjusted R-squared:  -0.02312
F-statistic: 0.1188 on 1 and 38 DF,  p-value: 0.7322
```

A cautionary tale...

```
> linpc2 <- lm(y~pcs$x[,2])
> summary(linpc2)

Call:
lm(formula = y ~ pcs$x[, 2])

Residuals:
      Min        1Q    Median        3Q       Max
-0.261207 -0.080209 -0.007321  0.081190  0.261425

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  0.08484    0.02039   4.161 0.000174 ***
pcs$x[, 2]   2.28110    0.05241  43.527  < 2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.1289 on 38 degrees of freedom
Multiple R-squared:  0.9803,Adjusted R-squared:  0.9798
F-statistic:  1895 on 1 and 38 DF,  p-value: < 2.2e-16
```

A cautionary tale...

“But you didn’t standardize!” ... okay then ...

```
> pcss <- prcomp(cbind(x1, x2), scale.=TRUE)
> summary(pcss)
Importance of components:
                          PC1     PC2
Standard deviation     1.4097 0.11337
Proportion of Variance 0.9936 0.00643
Cumulative Proportion  0.9936 1.00000
```

A cautionary tale...

```
> linpcs <- lm(y~pcss$x[,1])
> summary(linpcs)

Call:
lm(formula = y ~ pcss$x[, 1])

Residuals:
    Min      1Q  Median      3Q     Max
-2.0009 -0.7343  0.2586  0.7136  1.5962

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  0.08484    0.14463   0.587    0.561
pcss$x[, 1]  0.06621    0.10390   0.637    0.528

Residual standard error: 0.9147 on 38 degrees of freedom
Multiple R-squared:  0.01057,Adjusted R-squared:  -0.01546
F-statistic: 0.4061 on 1 and 38 DF,  p-value: 0.5278
```

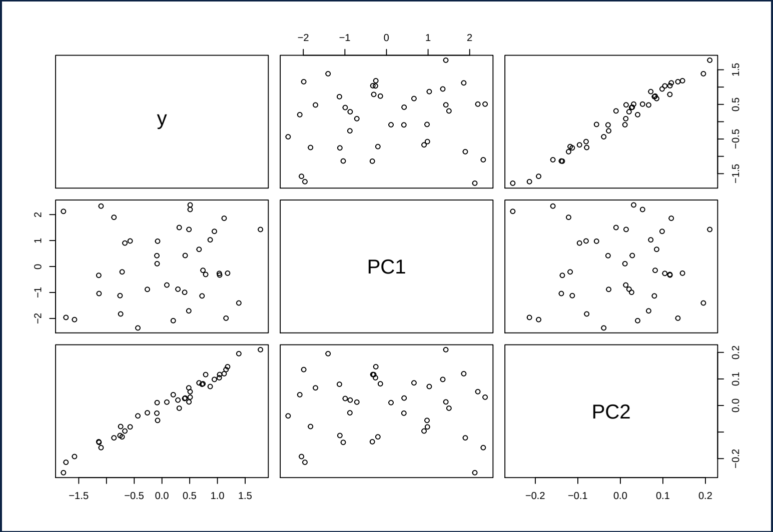
A cautionary tale...

Going from an unsupervised approach to supervised approach, we don't know which component will explain the variation in Y, it isn't always the first principal component as shown in this example

```
> linpcs2 <- lm(y~pcss$x[,2])
> summary(linpcs2)

Call:
lm(formula = y ~ pcss$x[, 2])

Residuals:
      Min        1Q    Median        3Q       Max
-0.257674 -0.126315  0.009366  0.095724  0.306733

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  0.08484    0.02394   3.543  0.00107 **
pcss$x[, 2]  7.89705    0.21389  36.921  < 2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.1514 on 38 degrees of freedom
Multiple R-squared:  0.9729,Adjusted R-squared:  0.9722
F-statistic:  1363 on 1 and 38 DF,  p-value: < 2.2e-16
```

A cautionary tale...

- Here's the visual...



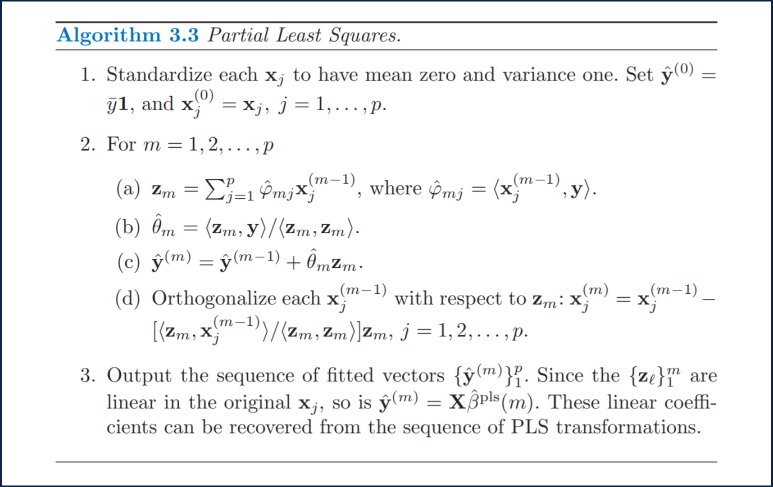
Comments

- Clearly PCReg has some issues that need fixing...
- One simple approach is to disregard the standard PCA approaches for tossing out components, and instead take all p components, then perform standard stepwise regression procedures.
- This would fix the issue with our simulation, but would not cover all potential issues.
- Leaving the problems aside for a moment, note that of course PCs can be used as predictors in any supervised context (random forests, neural nets, discriminant analysis, etc).

Partial Least Squares

- A more proper approach is to somehow consider the response variable Y while performing the PCA in the first place!
- Partial Least Squares (PLS) does exactly this.
- It seeks linear combos of predictors that have high variance AND high correlation with Y .
- Just like PCA, it is not scale invariant. Therefore, the predictors are generally scaled to have mean 0 variance 1.

PLS Algorithm



Algorithm 3.3 *Partial Least Squares.*

1. Standardize each $\mathbf{x}_j$ to have mean zero and variance one. Set $\hat{\mathbf{y}}^{(0)} = \bar{y}\mathbf{1}$, and $\mathbf{x}_j^{(0)} = \mathbf{x}_j$, $j = 1, \ldots, p$.
2. For $m = 1, 2, \ldots, p$
   - (a) $\mathbf{z}_m = \sum_{j=1}^{p} \hat{\varphi}_{mj}\mathbf{x}_j^{(m-1)}$, where $\hat{\varphi}_{mj} = \langle \mathbf{x}_j^{(m-1)}, \mathbf{y} \rangle$.
   - (b) $\hat{\theta}_m = \langle \mathbf{z}_m, \mathbf{y} \rangle / \langle \mathbf{z}_m, \mathbf{z}_m \rangle$.
   - (c) $\hat{\mathbf{y}}^{(m)} = \hat{\mathbf{y}}^{(m-1)} + \hat{\theta}_m \mathbf{z}_m$.
   - (d) Orthogonalize each $\mathbf{x}_j^{(m-1)}$ with respect to $\mathbf{z}_m$: $\mathbf{x}_j^{(m)} = \mathbf{x}_j^{(m-1)} - [\langle \mathbf{z}_m, \mathbf{x}_j^{(m-1)} \rangle / \langle \mathbf{z}_m, \mathbf{z}_m \rangle]\mathbf{z}_m$, $j = 1, 2, \ldots, p$.
3. Output the sequence of fitted vectors $\{\hat{\mathbf{y}}^{(m)}\}_1^p$. Since the $\{\mathbf{z}_\ell\}_1^m$ are linear in the original $\mathbf{x}_j$, so is $\hat{\mathbf{y}}^{(m)} = \mathbf{X}\hat{\beta}^{\text{pls}}(m)$. These linear coefficients can be recovered from the sequence of PLS transformations.

From Elements of Statistical Learning. This is the classical algorithm, others have been put forth.

A cautionary tale...

```
> plsmod <- plsr(y~x1+x2, method="oscorespls")
> summary(plsmod)
Data:  X dimension: 40 2
Y dimension: 40 1
Fit method: oscorespls
Number of components considered: 2
TRAINING: % variance explained
   1 comps  2 comps
X   99.033   100.00
y    1.672    98.35
```

Note that the extremely high variance that I manufactured within the predictors still drives the first component. However, the output at least makes this clear to us!

PLS Comments

- There are PLS versions for classification tasks as well (PLS-DA for example), and plenty of extensions/refinement.
- PLS originated in the 80's after all.
- It is the predominant model in several applied fields (some chemistry fields, sensory sciences, marketing, and more).
- This is likely due to its inherent interpretability (linear combos), in addition to being an early solution to the problem of moderate-to-high dimensionality.

THE UNIVERSITY OF BRITISH COLUMBIA