# Principle Components and Ridge Regression

## Math 463, Spring 2015, University of Oregon

David A. Levin

University of Oregon

May 11 and 13, 2015

# Principle Component Regression

- Goal: Change to a new set of regressors $Z_1, \ldots, Z_p$ such that $Z_1, \ldots, Z_p$ are orthogonal.
- Only keep some of these regressors which account for most of the variation in response variable $Y$.
- That is, goal is: find a $p \times p$ rotation matrix $\boldsymbol{U}$ such that $\boldsymbol{Z} = \boldsymbol{X}\boldsymbol{U}$, and such that

$$\boldsymbol{Z}'\boldsymbol{Z} = \text{diag}(\lambda_1, \ldots, \lambda_p)$$

  where $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_p$ are the eigenvalues of $\boldsymbol{Z}$. Note that $\lambda_i$ is the variance of $Z_i$.
- Note that $\text{Var}(\boldsymbol{u}_1'\boldsymbol{X})$ is maximized subject to $\boldsymbol{u}_1'\boldsymbol{u}_1 = 1$; Then $\text{Var}(\boldsymbol{u}_2'\boldsymbol{X})$ is maximized subject to $\boldsymbol{u}_1'\boldsymbol{u}_2 = 0$ and $\boldsymbol{u}_2'\boldsymbol{u} = 1$, etc.
- Either include that all 1's vector in $\boldsymbol{X}$ or center all the columns of $\boldsymbol{X}$, in which case each column is orthogonal to the all 1's vector. The latter is conventional. Then the $\boldsymbol{Z}_i$'s are linear combination of the $\boldsymbol{X}_i$'s without a constant term.
- The variables should be on the same scale, so usually the $\boldsymbol{X}_i$'s are scaled by their standard deviations.

# Crime Data

We have seen that the variables in the crime data

```
> crime = read.table("http://pages.uoregon.edu/dlevin/DATA/crime.txt",
+                    header=T)
```

are highly correlated, and noted that this kind of collinearity causes problems.

```
> print(cor(crime[,-1]),digits=2)
```

```
       Age      S     Ed    Ex0    Ex1    LF      M      N     NW     U1     U2
Age  1.000  0.584 -0.530 -0.506 -0.513 -0.16 -0.029 -0.281  0.593 -0.224 -0.245
S    0.584  1.000 -0.703 -0.373 -0.376 -0.51 -0.315 -0.050  0.767 -0.172  0.072
Ed  -0.530 -0.703  1.000  0.483  0.499  0.56  0.437 -0.017 -0.665  0.018 -0.216
Ex0 -0.506 -0.373  0.483  1.000  0.994  0.12  0.034  0.526 -0.214 -0.044  0.185
Ex1 -0.513 -0.376  0.499  0.994  1.000  0.11  0.023  0.514 -0.219 -0.052  0.169
LF  -0.161 -0.505  0.561  0.121  0.106  1.00  0.514 -0.124 -0.341 -0.229 -0.421
M   -0.029 -0.315  0.437  0.034  0.023  0.51  1.000 -0.411 -0.327  0.352 -0.019
N   -0.281 -0.050 -0.017  0.526  0.514 -0.12 -0.411  1.000  0.095 -0.038  0.270
NW   0.593  0.767 -0.665 -0.214 -0.219 -0.34 -0.327  0.095  1.000 -0.156  0.081
U1  -0.224 -0.172  0.018 -0.044 -0.052 -0.23  0.352 -0.038 -0.156  1.000  0.746
U2  -0.245  0.072 -0.216  0.185  0.169 -0.42 -0.019  0.270  0.081  0.746  1.000
W   -0.670 -0.637  0.736  0.787  0.794  0.29  0.180  0.308 -0.590  0.045  0.092
X    0.639  0.737 -0.769 -0.631 -0.648 -0.27 -0.167 -0.126  0.677 -0.064  0.016
         W      X
Age -0.670  0.639
S   -0.637  0.737
Ed   0.736 -0.769
Ex0  0.787 -0.631
Ex1  0.794 -0.648
LF   0.295 -0.270
M    0.180 -0.167
N    0.308 -0.126
NW  -0.590  0.677
U1   0.045 -0.064
U2   0.092  0.016
W    1.000 -0.884
X   -0.884  1.000
```

- There are various functions that perform principle component analysis. The function **prcomp** works, but **pcr** in the package **pls** has some nice feature.
- Below we show **pcr**

```
> library(pls)
> pcf.crime = pcr(R~., data=crime ,scale=TRUE)
> print(summary(pcf.crime),digits=2)

Data:  X dimension: 47 13
Y dimension: 47 1
Fit method: svdpc
Number of components considered: 13
TRAINING: % variance explained
   1 comps  2 comps  3 comps  4 comps  5 comps  6 comps  7 comps  8 comps
X    43.48    62.86    77.52    85.03    89.91    92.62    94.63    96.45
R    15.47    23.26    26.17    62.13    62.17    62.20    62.75    63.24
   9 comps  10 comps  11 comps  12 comps  13 comps
X    97.92     98.94     99.49     99.96    100.00
R    63.50     72.66     73.27     76.16     76.92
NULL
```

We note that if we regress on just the first four principle components, we obtain a $R$-squared of 0.62. Including all 13 principle components gives a $R$-squared of 0.76.

# Cross-validation

- Evaluating the predictive capability of a model on the same data which was used to fit the model understates the predictive error: the model fit to the data specifically minimized the errors *for that data*.
- On independent data, the errors will be larger.
- Often split the data into two pieces, the *training* piece and the *validation* (or *testing*) piece. Model is fit on the training piece and then tested on the validation piece.
- Cross validation takes this a step further. In this case, the data is divided into $k$ equal portions, at random. Call these subsets $D_1, \ldots, D_k$.
- There are then $k$ performance measures: If $j$ ranges in $1, 2, \ldots, k$, the $j$-th measure fits the model on all the data but $D_j$, and then tests the model on $D_j$. Performance is measured by, say, mean squared error between predicted and observed values on $D_j$. Call this $\text{MSE}_j$.
- The $k$-fold cross validation measure is then defined as

$$\text{CV}_{(k)} = \frac{1}{k} \sum \text{MSE}_j$$

# Using cross-validation to pick the number of PCs

- One method to pick $M$, the number of PC regressors, is to use cross-validation to measure prediction error for a given choice of $M$. Then pick $M$ to minimize.

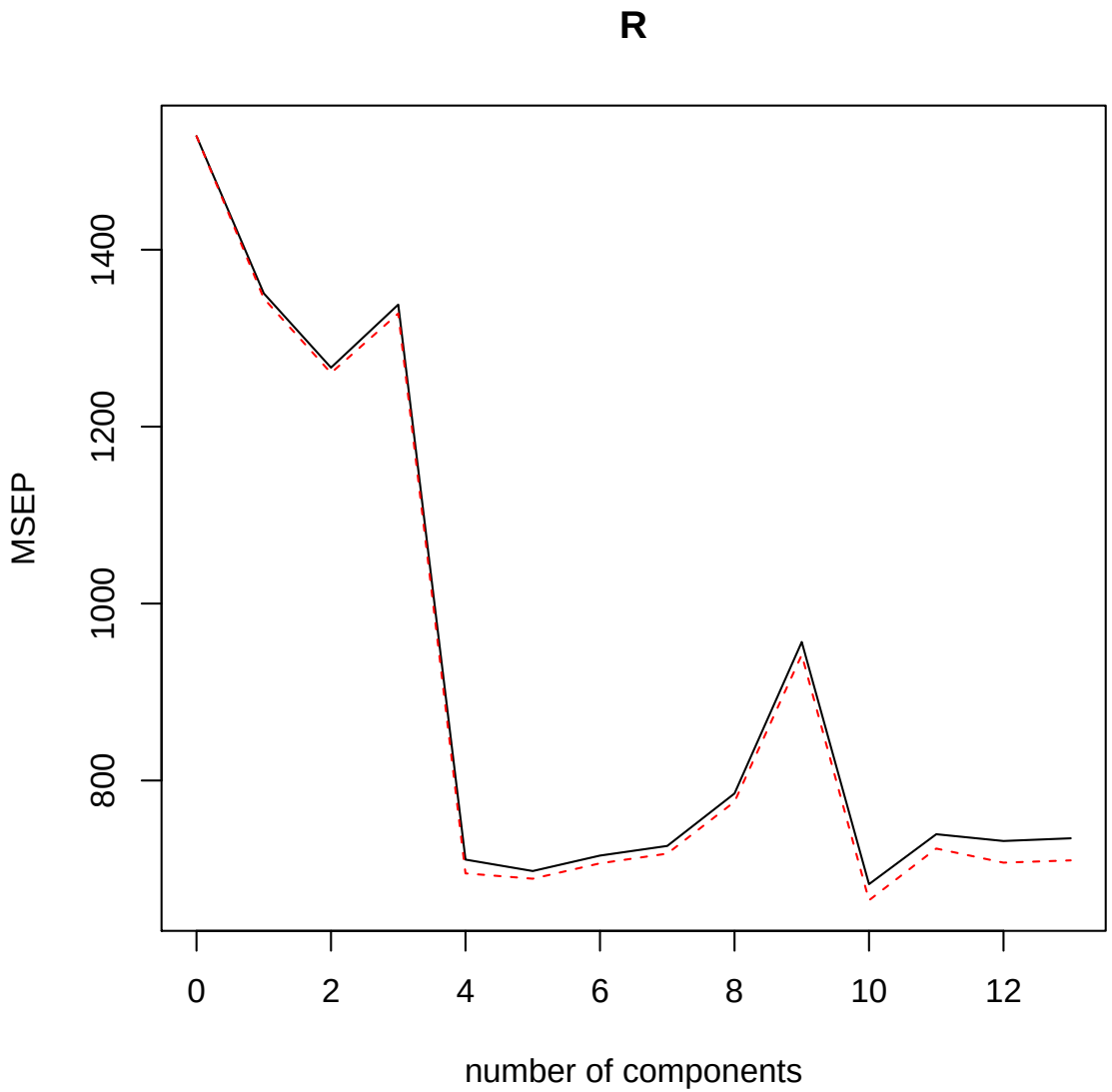
```
> pcr.fit=pcr(R~., data=crime ,scale=TRUE, validation ="CV")
> validationplot(pcr.fit,val.type="MSEP")
> print(summary(pcr.fit),digits=2)

Data:  X dimension: 47 13
Y dimension: 47 1
Fit method: svdpc
Number of components considered: 13

VALIDATION: RMSEP
Cross-validated using 10 random segments.
       (Intercept)  1 comps  2 comps  3 comps  4 comps  5 comps  6 comps
CV           39.09    36.75    35.59    36.58    26.66    26.41    26.74
adjCV        39.09    36.67    35.50    36.44    26.36    26.25    26.58
       7 comps  8 comps  9 comps  10 comps  11 comps  12 comps  13 comps
CV       26.95    28.02    30.93     26.13     27.19     27.05     27.10
adjCV    26.78    27.86    30.69     25.78     26.89     26.59     26.64

TRAINING: % variance explained
   1 comps  2 comps  3 comps  4 comps  5 comps  6 comps  7 comps  8 comps
X    43.48    62.86    77.52    85.03    89.91    92.62    94.63    96.45
R    15.47    23.26    26.17    62.13    62.17    62.20    62.75    63.24
   9 comps  10 comps  11 comps  12 comps  13 comps
X    97.92     98.94     99.49     99.96    100.00
R    63.50     72.66     73.27     76.16     76.92
NULL
```

MSE of prediction falls off at 4 PCs:



# Ridge Regression

- Instead of minimizing RSS, minimixing

$$\text{RSS} + \lambda \sum_{j=1}^{p} \beta_j^2$$

- The solution is

$$\hat{\beta}^R = (\boldsymbol{X}'\boldsymbol{X} + \lambda \boldsymbol{I})^{-1} \boldsymbol{X}'\boldsymbol{Y}$$

- How to choose $\lambda$? One aproach is too choose so that estimates $\hat{\beta}^R$ stabilize as $\lambda$ varies.
- Another approach is to pick $\lambda$ to minimize prediction error via cross-validation.

```
> library(glmnet)
> x = model.matrix(R~., crime)
> y = crime$R
> grid=10^seq(3,-1,length=100)
> ridge.mod = glmnet(x,y,alpha=0,lambda=grid)
> set.seed(1)
> train=sample(1:nrow(x), nrow(x)/2)
> test=(-train)
> y.test=y[test]
> ridge.mod=glmnet(x[train,],y[train],alpha=0,lambda=grid, thresh =1e-12)
> matplot(log(grid),t(coef(ridge.mod)),type="l",lty=1)
> #ridge.pred=predict(ridge.mod,s=4,newx=x[test,])
```