Principal component regression (PCR)

Trevor Hastie and Robert Tibshirani

Here, I am adapting part of the lab associated with Chapter 6 of the textbook.

Data Prep

We wish to predict a baseball player's `Salary` on the basis of various statistics associated with performance in the previous year.

First of all, we note that the `Salary` variable is missing for some of the players. The `is.na()` function can be used to identify the missing observations. It returns a vector of the same length as the input vector, with a `TRUE` for any elements that are missing, and a `FALSE` for non-missing elements. The `sum()` function can then be used to count all of the missing elements.

```
library(ISLR2)
names(Hitters)
```

```
##  [1] "AtBat"     "Hits"      "HmRun"     "Runs"      "RBI"       "Walks"
##  [7] "Years"     "CAtBat"    "CHits"     "CHmRun"    "CRuns"     "CRBI"
## [13] "CWalks"    "League"    "Division"  "PutOuts"   "Assists"   "Errors"
## [19] "Salary"    "NewLeague"
```

```
dim(Hitters)
```

```
## [1] 322  20
```

```
sum(is.na(Hitters$Salary))
```

```
## [1] 59
```

Hence we see that `Salary` is missing for 59 players. The `na.omit()` function removes all of the rows that have missing values in any variable.

```
Hitters <- na.omit(Hitters)
dim(Hitters)
```

```
## [1] 263  20
```

```
sum(is.na(Hitters))
```

```
## [1] 0
```

All is well now, so we `attach` the data for ease of use.

```
attach(Hitters)
```

Principal Components Regression

Principal components regression (PCR) can be performed using the `pcr()` function, which is part of the `pls` library. We now apply PCR to the `Hitters` data, in order to predict `Salary`.

```r
#install.packages("pls")
library(pls)
```

```
##
## Attaching package: 'pls'

## The following object is masked from 'package:stats':
##
##     loadings
```

```r
set.seed(1)
pcr.fit <- pcr(Salary ~ ., data = Hitters, scale = TRUE,
    validation = "CV")
```

The syntax for the `pcr()` function is similar to that for `lm()`, with a few additional options. Setting `scale = TRUE` has the effect of *standardizing* each predictor, prior to generating the principal components, so that the scale on which each variable is measured will not have an effect. Setting `validation = "CV"` causes `pcr()` to compute the ten-fold cross-validation error for each possible value of M, the number of principal components used. The resulting fit can be examined using `summary()`.

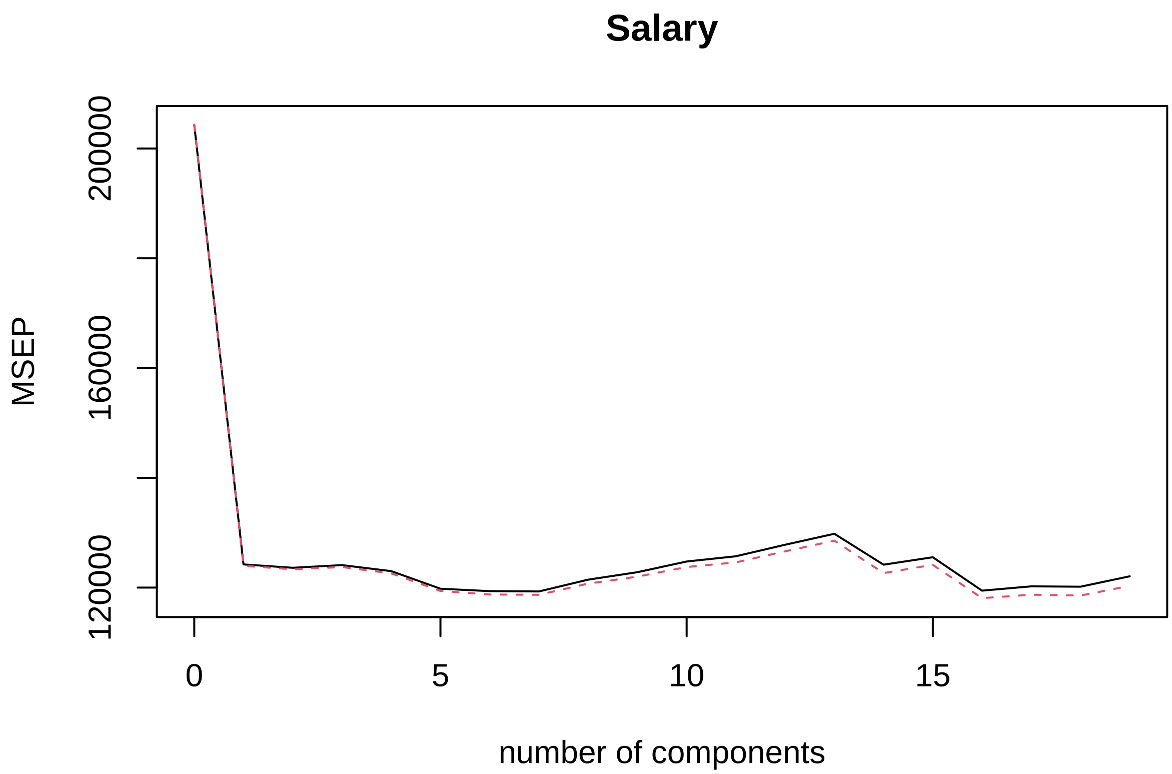
```r
summary(pcr.fit)
```

```
## Data:    X dimension: 263 19
##  Y dimension: 263 1
## Fit method: svdpc
## Number of components considered: 19
##
## VALIDATION: RMSEP
## Cross-validated using 10 random segments.
##        (Intercept)  1 comps  2 comps  3 comps  4 comps  5 comps  6 comps
## CV             452    352.5    351.6    352.3    350.7    346.1    345.5
## adjCV          452    352.1    351.2    351.8    350.1    345.5    344.6
##        7 comps  8 comps  9 comps  10 comps  11 comps  12 comps  13 comps
## CV       345.4    348.5    350.4     353.2     354.5     357.5     360.3
## adjCV    344.5    347.5    349.3     351.8     353.0     355.8     358.5
##        14 comps  15 comps  16 comps  17 comps  18 comps  19 comps
## CV        352.4     354.3     345.6     346.7     346.6     349.4
## adjCV     350.2     352.3     343.6     344.5     344.3     346.9
##
## TRAINING: % variance explained
##         1 comps  2 comps  3 comps  4 comps  5 comps  6 comps  7 comps  8 comps
## X         38.31    60.16    70.84    79.03    84.29    88.63    92.26    94.96
## Salary    40.63    41.58    42.17    43.22    44.90    46.48    46.69    46.75
##         9 comps  10 comps  11 comps  12 comps  13 comps  14 comps  15 comps
## X         96.28     97.26     97.98     98.65     99.15     99.47     99.75
## Salary    46.86     47.76     47.82     47.85     48.10     50.40     50.55
##         16 comps  17 comps  18 comps  19 comps
## X          99.89     99.97     99.99    100.00
## Salary     53.01     53.85     54.61     54.61
```

The CV score is provided for each possible number of components, ranging from $M = 0$ onwards. (We have printed the CV output only up to $M = 4$.) Note that `pcr()` reports the *root mean squared error*; in order to obtain the usual MSE, we must square this quantity. For instance, a root mean squared error of 352.8 corresponds to an MSE of $352.8^2 = 124{,}468$.

One can also plot the cross-validation scores using the `validationplot()` function. Using `val.type = "MSEP"` will cause the cross-validation MSE to be plotted.

```r
validationplot(pcr.fit, val.type = "MSEP")
```



We see that the smallest cross-validation error occurs when $M = 18$ components are used. This is barely fewer than $M = 19$, which amounts to simply performing least squares, because when all of the components are used in PCR no dimension reduction occurs. However, from the plot we also see that the cross-validation error is roughly the same when only one component is included in the model. This suggests that a model that uses just a small number of components might suffice.

The `summary()` function also provides the *percentage of variance explained* in the predictors and in the response using different numbers of components. Briefly, we can think of this as the amount of information about the predictors or the response that is captured using M principal components. For example, setting $M = 1$ only captures 38.31% of all the variance, or information, in the predictors. In contrast, using $M = 5$ increases the value to 84.29%. If we were to use all $M = p = 19$ components, this would increase to 100%.

We now perform PCR on the training data and evaluate its test set performance.

```r
set.seed(1)
#splitting the data set
train <- sample(c(TRUE, FALSE), nrow(Hitters),
    replace = TRUE)
test <- (!train)

pcr.fit <- pcr(Salary ~ ., data = Hitters, subset = train,
    scale = TRUE, validation = "CV")
validationplot(pcr.fit, val.type = "MSEP")
```

Now we find that the lowest cross-validation error occurs when $M = 5$ components are used. We compute the test MSE as follows.

```
x <- model.matrix(Salary ~ ., Hitters)[, -1]
y <- Hitters$Salary
y.test <- y[test]

pcr.pred <- predict(pcr.fit, x[test, ], ncomp = 5)
mean((pcr.pred - y.test)^2)
```

```
## [1] 136435.5
```

Finally, we fit PCR on the full data set, using $M = 5$, the number of components identified by cross-validation.

```
pcr.fit <- pcr(y ~ x, scale = TRUE, ncomp = 5)
summary(pcr.fit)
```

```
## Data:    X dimension: 263 19
##  Y dimension: 263 1
## Fit method: svdpc
## Number of components considered: 5
## TRAINING: % variance explained
##    1 comps  2 comps  3 comps  4 comps  5 comps
## X    38.31    60.16    70.84    79.03    84.29
## y    40.63    41.58    42.17    43.22    44.90
```