ISYE 6501 Week 6 Homework

2023-09-26

Question 9.1

Using the same crime data set uscrime.txt as in Question 8.2, apply Principal Component Analysis and then create a regression model using the first few principal components. Specify your new model in terms of the original variables (not the principal components), and compare its quality to that of your solution to Question 8.2. You can use the R function prcomp for PCA. (Note that to first scale the data, you can include scale. = TRUE to scale as part of the PCA function. Don't forget that, to make a prediction for the new city, you'll need to unscale the coefficients (i.e., do the scaling calculation in reverse)!)

As a first step to answering this question, I performed Principal Component Analysis (PCA) using R's prcomp() function.

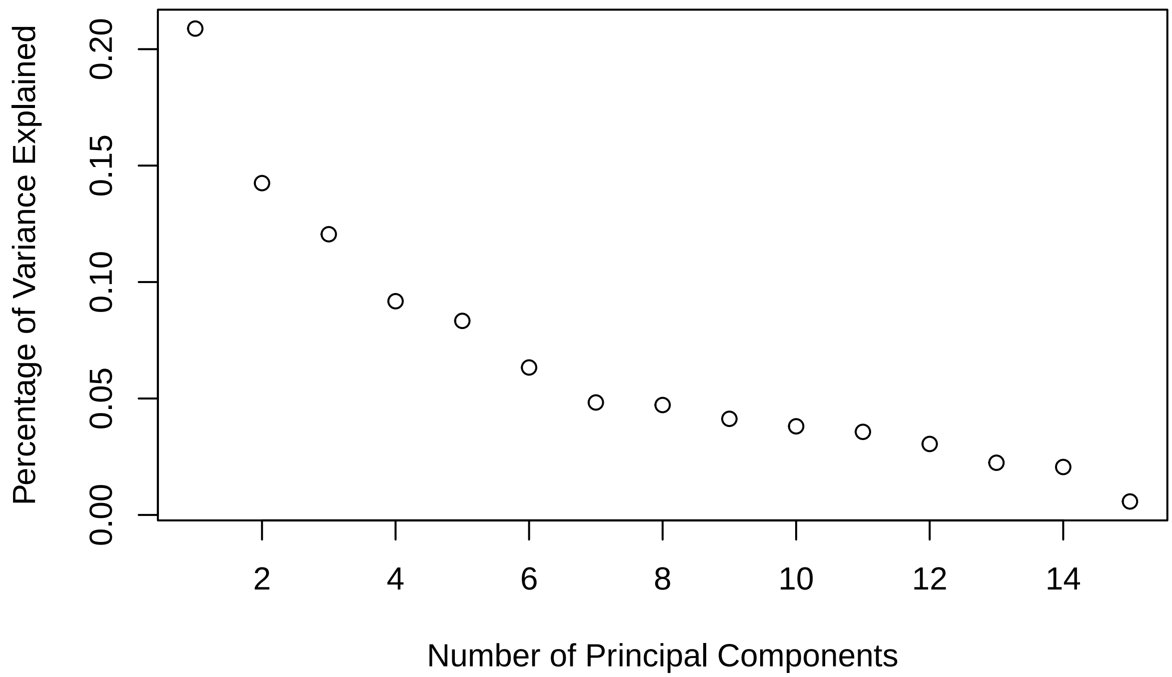
```r
#load the data
crime_data <- read.table("http://www.statsci.org/data/general/uscrime.txt", header=TRUE)

#subset predictors
predictors <- subset(crime_data, select=-Crime)

#perform PCA
pca_result <- prcomp(x=predictors, scale.=TRUE)
summary(pca_result)
```

```
## Importance of components:
##                           PC1    PC2    PC3     PC4     PC5     PC6     PC7
## Standard deviation     2.4534 1.6739 1.4160 1.07806 0.97893 0.74377 0.56729
## Proportion of Variance 0.4013 0.1868 0.1337 0.07748 0.06389 0.03688 0.02145
## Cumulative Proportion  0.4013 0.5880 0.7217 0.79920 0.86308 0.89996 0.92142
##                            PC8     PC9    PC10    PC11    PC12    PC13   PC14
## Standard deviation     0.55444 0.48493 0.44708 0.41915 0.35804 0.26333 0.2418
## Proportion of Variance 0.02049 0.01568 0.01333 0.01171 0.00855 0.00462 0.0039
## Cumulative Proportion  0.94191 0.95759 0.97091 0.98263 0.99117 0.99579 0.9997
##                           PC15
## Standard deviation     0.06793
## Proportion of Variance 0.00031
## Cumulative Proportion  1.00000
```

Next, I looked at the standard deviation output (the square roots of the eigenvalues). These values tell me how much variance (or information) is captured within each principal component. Larger standard deviations indicate more important components. As shown below, I created an elbow diagram to visualize how adding more components impacts the the cumulative proportion of variance explained. It looks like the diagram's "elbow" is around X=7, meaning that the marginal benefit of adding more principal components beyond the first 7 is small.

Removing principal components will result in a loss of information. Thus, it is important to consider the trade offs between information loss and having a simpler model. In this case, I can still retain ~75% of the initial information by only using 7 principal components.

```
#create elbow diagram to determine how many principal components to keep
sum_sd <- sum(pca_result$sdev)
elbow_plot <- plot(x=1:length(pca_result$sd),
                   y=pca_result$sd/sum_sd,
                   xlab="Number of Principal Components",
                   ylab="Percentage of Variance Explained"
                   )
```




```
elbow_plot
```

```
## NULL
```

```
#calculate total variance explained in first 7 PCs
sum(pca_result$sd[1:7]/sum_sd)
```

```
## [1] 0.758537
```

Then, I transformed my data using the selected number of principal components (X=7). The code block below reorients the crime_data data set from its original axes to ones represented by the principal components.

```r
#transform data using selected number of components
num_components <- 7
crime_data_transformed <- as.data.frame(predict(pca_result, newdata=predictors)[,1:num_components])
crime_data_transformed$Crime <- crime_data$Crime
```

Then, I performed linear regression on the reduced data set and observed the model's output. Compared to the model in my homework 5 submission, the adjusted R-squared value is slightly lower (0.63 vs. 0.73). This means that my model from last week explains more variance in the data.

```r
#perform linear regression with transformed data
lm_model <- lm(Crime~., data=crime_data_transformed)

#observe model output
summary(lm_model)
```

```
##
## Call:
## lm(formula = Crime ~ ., data = crime_data_transformed)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -475.41 -141.65   34.73  137.25  412.32
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   905.09      34.21  26.454  < 2e-16 ***
## PC1            65.22      14.10   4.626 4.04e-05 ***
## PC2           -70.08      20.66  -3.392   0.0016 **
## PC3            25.19      24.42   1.032   0.3086
## PC4            69.45      32.08   2.165   0.0366 *
## PC5          -229.04      35.33  -6.483 1.11e-07 ***
## PC6           -60.21      46.50  -1.295   0.2029
## PC7           117.26      60.96   1.923   0.0617 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 234.6 on 39 degrees of freedom
## Multiple R-squared:  0.6882, Adjusted R-squared:  0.6322
## F-statistic:  12.3 on 7 and 39 DF,  p-value: 3.513e-08
```

```r
lm_model$coefficients
```

```
## (Intercept)         PC1         PC2         PC3         PC4         PC5
##   905.08511    65.21593   -70.08312    25.19408    69.44603  -229.04282
##         PC6         PC7
##   -60.21329   117.25590
```

I tried two different approaches to make a prediction on new data.

Approach 1

In the first approach, I transformed the PCA model coefficients back into their original factors. By converting back to the original variables, I can assess the contribution of each of my original predictors to the model, just like I would in a typical linear regression. This approach yielded a crime rate prediction of 1230, which is close to my prediction of 1304 in the Week 5 Homework submission.

```
#create prediction data frame
new_data <- data.frame(M=14, So=0, Ed=10, Po1=12, Po2=15.5, LF=0.640, M.F=94, Pop=150,
                       NW=1.1, U1=0.12, U2=3.6, Wealth=3200, Ineq=20.1, Prob=0.04, Time=39, Crime=NA)

#unscale coefficients
coefficients_converted <- (pca_result$rotation[,1:7] %*% lm_model$coefficients[2:8])/pca_result$scale
coefficients_converted
```

```
##                 [,1]
## M        5.523735e+01
## So       1.397571e+02
## Ed      -6.803836e+00
## Po1      4.458638e+01
## Po2      4.642432e+01
## LF       6.733809e+02
## M.F      4.440293e+01
## Pop      9.599076e-01
## NW       5.684940e+00
## U1      -1.027735e+03
## U2       2.441589e+01
## Wealth   2.883565e-02
## Ineq     1.245113e+01
## Prob    -5.170569e+03
## Time    -2.215095e+00
```

```
#adjust intercept based on pca$center
intercept <- lm_model$coefficients[1] - sum(coefficients_converted * pca_result$center)

#make prediction on new data
prediction <- sum(coefficients_converted * new_data[1:15]) + intercept
prediction
```

```
## (Intercept)
##    1230.418
```

Approach 2

In the second approach, I reoriented my new data from its original axes to ones represented by the principal components. This approach yielded the same crime rate prediction of 1230. However, it is more difficult to explain my model when I convert everything into the PCA variables, which don't have a clear real-life meaning.

```
#create prediction data frame
new_data <- data.frame(M=14, So=0, Ed=10, Po1=12, Po2=15.5, LF=0.640, M.F=94, Pop=150,
                       NW=1.1, U1=0.12, U2=3.6, Wealth=3200, Ineq=20.1, Prob=0.04, Time=39, Crime=NA)

#transform new data to same feature space as lm_model
new_data_transformed <- as.data.frame(predict(pca_result, newdata=new_data)[,1:num_components])
new_data_transformed <- as.data.frame(t(new_data_transformed))
```

```r
#make prediction on new data
prediction <- predict(lm_model, newdata=new_data_transformed)
prediction
```

```
## predict(pca_result, newdata = new_data)[, 1:num_components]
##                                                    1230.418
```