Dimension Reduction
Part 2: Principal Component Analysis

Aram Balagyozyan

Department of Operations and Information Management
Kania School of Management
The
University of Scranton

March 6, 2022

Outline

1. Principal Component Analysis

Outline

1. Principal Component Analysis

Principal Component Analysis

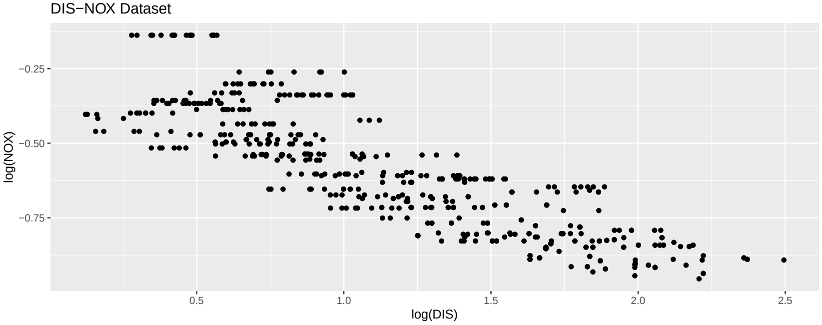
- Principal component analysis (PCA) is a very powerful and useful method for data dimension reduction.
- It takes in `n` variables and produces up to `n` composite variables (principle components) each being a weighted linear combination of two or more explanatory variables.
- The goal is that this conversion results in only minimal loss of information carried by the entire set of original variables.
- If the first few principal components encompass most of the variation in the original data then we may keep and use only the first few principal components and discard the rest.
- To see what all of these means, consider two highly correlated (explanatory) variables in the Boston Housing dataset:`DIS` and `NOX`

Principal Component Analysis

```
pca_data<-log(housing.df[,c(5,8)])
cor(pca_data$NOX,pca_data$DIS)# correlation is -0.86
```

```
## [1] -0.8600183
```

```
library(ggplot2)
ggplot(pca_data, aes(x=DIS, y=NOX))+
  geom_point()+labs(x="log(DIS)",y="log(NOX)",
                    title = "DIS-NOX Dataset")
```



Principal Component Analysis

- Each point on the scatter plot abode has two coordinates (dimensions), one for `DIS` and one for `NOX`.
- Roughly speaking, 89% of the total variation in both variables is actually "covariation," or variation in one variables that is duplicated by similar variation in the other variable.
- Can we use this fact to reduce the number of dimensions, while making maximum use of their unique contributions to the overall variation? It might be possible to reduce the two dimensions to a single dimension without losing too much information.
- The idea of PCA is to find a linear combination of the two variables (dimensions) that contains most, even if not all, of the information in such a way that this new variable can replace the two original variables and still account for most variability in the original data.

Principal Component Analysis

- Let's see PCA in action:

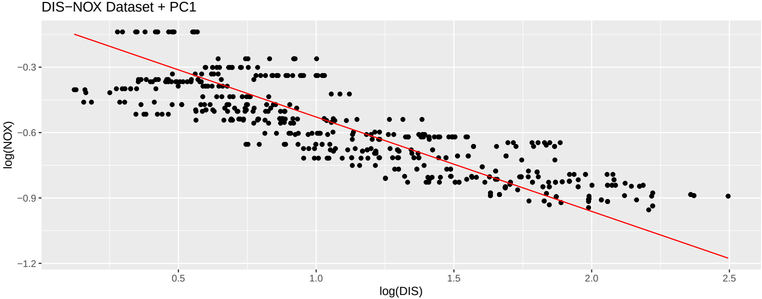
```
pca<-prcomp(pca_data)
summary(pca)
```

```
## Importance of components:
##                           PC1     PC2
## Standard deviation     0.5676 0.09773
## Proportion of Variance 0.9712 0.02879
## Cumulative Proportion  0.9712 1.00000
```

- Above, we used function `prcomp()` to obtain PC's of our two-dimensional dataset. As you can see, the first principal component (`PC1`) alone can explain 97.12% of cumulative variance in the original two variables. Obviously, with two principal components, we can explain 100% of the variation of the original, two dimensional data.
- PCA in effect creates a new set of axis with lower dimensionality (or more precisely, rotates the existing axes) and projects the existing data points onto those axis.

Principal Component Analysis

- The graph below, simply overlays the first principal component of the `DIS-NOX` dataset on the graph above.

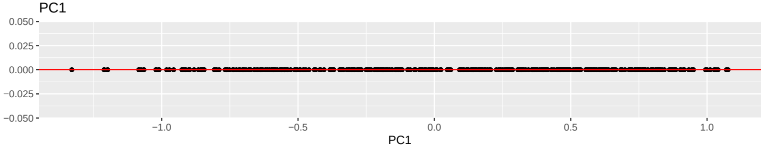


- This PC can be viewed as one of the axes rotated in such a way that it encompasses most of the variation in the original data

Principal Component Analysis

- The graph below, illustrates the first principal component with the data points projected onto it.

```
library(dplyr)
pca_data<-mutate(pca_data,pca$x[,1],pca$x[,2])
names(pca_data)[3:4]<-c("PC1","PC2")
ggplot(pca_data)+
  geom_point(aes(x=PC1,y=0))+labs(x="PC1",y="")+
  geom_abline(slope = 0,intercept = 0,color="red")+
  labs(title="PC1")
```



- Note that the transformed data has only one dimension that can explain 97.12% of the variation in the original two-dimensional data.

Principal Component Analysis

- In sum, the PCA method searches for a set of "new" variables, each being a linear combination of the original variables.
- Consider the following code:

```
pca$rotation
```

```
##              PC1        PC2
## NOX  0.3151324 -0.9490477
## DIS -0.9490477 -0.3151324
```

- `prcomp()` produces the `rotation` attribute that exhibits the found (rotated) axes.
- The above output implies that the 1st principal component is calculates as:

$$PC1 = (-0.9490477) \times (DIS - \overline{DIS}) + 0.3151324 \times (NOX - \overline{NOX})$$

Principal Component Analysis

- The two coordinates of each data point of the DIS-NOX dataset (two graphs above) as well as the averages of `DIS` and `NOX` are fed into the formula above to obtain the (only one) coordinate of the corresponding point of the 1st principal component (the above graph).
- As you may suspect, we don't need to do these calculations by hand, the `prcomp()` function will handle it for us.
- We can obtain the values of the new features (i.e. the principal components) for all points in the DIS-NOX dataset as follows (here I display the first 5 records only):

```
head(pca$x,5)
```

```
##             PC1          PC2
## [1,] -0.2123876 -0.060122618
## [2,] -0.4400330  0.008912222
## [3,] -0.4400330  0.008912222
## [4,] -0.6365973 -0.031349386
## [5,] -0.6365973 -0.031349386
```

Principal Component Analysis

- Suppose we are happy with the proportion of variance explained by the 1st principal component. We could proceed with the posterior modeling stages substituting this `PC1` for the the two original variables `DIS` and `NOX`.
- One bid advantage of the PCA method is that it can help us to avoid the problem with multicollinearity. When we have more than 2 original variables, X_1, X_2,. . . , X_n, there will typically be up to $n-1$ potentially usable principal components PC_1, PC_2,. . . , PC_{n-1} each being a weighted average of the original variables X_1, X_2,. . . , X_n:

$$PC_i = a_{i,1} \times (X_1 - \overline{X_1}) + a_{i,2} \times (X_2 - \overline{X_2}) + ... + a_{i,n} \times (X_n - \overline{X_n})$$

- By design, all the produced principle components will be uncorrelated. If we construct regression models using these PCs as predictors, we will not encounter the problem of multicollinearity.

Principal Component Analysis

- While the PCA method is extremely useful and powerful, there are at least a couple of drawbacks of the method that it is worth mentioning:
  1. It requires the original variables to be numeric.
  2. The resulting principal components are less comprehensible and harder to interpret for the end user as the they are combinations of the original data set.

Normalizing Principal Components

- Since all 12 original predictor variables in `housing.df` are numerical, why not run PCA on all of them?
- The code below, applies PCA to all predictor variables and then displays the importance of the top 5 PCs.

```
pci_all<-prcomp(housing.df[,1:12])
summary(pci_all)$importance[,1:5]
```

	PC1	PC2	PC3	PC4	PC5
Standard deviation	169.70529	28.66383	16.33371	7.195238	5.299771
Proportion of Variance	0.96004	0.02739	0.00889	0.001730	0.000940
Cumulative Proportion	0.96004	0.98743	0.99632	0.998050	0.998980

Normalizing Principal Components

- It seems that the first 2 principal components explain about 98.5% of the total variation in the data.
- Let's examine the weights (loadings) of these two PCs.

```
pci_all$rotation[,1:2]*100 # *100 for readability
```

```
##                 PC1         PC2
## CRIM     2.964858667 -1.43114827
## ZN      -4.489793743 63.19239729
## INDUS    2.939699213 -8.84690934
## CHAS    -0.005039603 -0.09492233
## NOX      0.046164098 -0.18227156
## RM      -0.122264531  0.47078384
## AGE      8.630572282 -75.59828082
## DIS     -0.676253943  4.51610479
## RAD      4.672423488  0.21428748
## TAX     99.296965059  9.98186109
## PTRATIO  0.590768880 -1.09200722
## LSTAT    2.319593108 -9.47469522
```

- Why is `PC1` dominated by `AGE` and `TAX`?

Normalizing Principal Components

- When compared to other variables, `AGE` and `TAX` have distinctively higher standard deviation (review the descriptive statistics above). Thus, the PCs that are more heavily dependent on high-variance variables will naturally explain a higher proportion of total data variability.
- This property of PCA is undesirable since it is the variables with greater *absolute* variance that will shape the first principle component rather than the ones with greater *relative* variance.
- One common approach to dealing with this issue is scaling variables (prior to the principal component analysis) dividing the values of each variable by their standard deviation).
- Scaling is easily accomplished from within the `prcomp()` function:

```
pci_all_scaled<-prcomp(housing.df[,1:12],
                       scale=TRUE)
summary(pci_all_scaled)$importance[,1:5]
```

Normalizing Principal Components

	PC1	PC2	PC3	PC4	PC5
Standard deviation	2.43045	1.183242	1.086599	0.9243602	0.8957392
Proportion of Variance	0.49226	0.116670	0.098390	0.0712000	0.0668600
Cumulative Proportion	0.49226	0.608930	0.707320	0.7785200	0.8453900

- The first two PCs are no longer dominating in terms of their sheer ability to explain the large proportion of total data variance.
- Still, by using only the first 5 PCs (instead of the 12 original variables), we are able to explain about 85% of total variation in the data.

Normalizing Principal Components

- Next let's examine the loadings of the top two new PCs.

```
pci_all_scaled$rotation[,1:2]*100
```

```
##                PC1        PC2
## CRIM     25.1296229 -27.355692
## ZN      -26.6381728 -25.012639
## INDUS    35.4873085   9.314825
## CHAS      0.7588523  50.285462
## NOX      34.9948498  23.250734
## RM      -19.6439268  27.304276
## AGE      32.3177006  29.297560
## DIS     -33.0882488 -34.262077
## RAD      32.2249962 -23.092958
## TAX      34.2255662 -21.304270
## PTRATIO  21.0942091 -39.255695
## LSTAT    31.5339517 -12.810818
```

```
# *100 for readability
```

Normalizing Principal Components

- The top 3 positive contributors to PC1 are `INDUS`, `NOX`, and `TAX`.
- Interestingly, the variables that have an obviously "negative" connotation (such as `INDUS` and `NOX`) impact the 1^{st} PC in the same direction as `TAX`. Thus, higher crime rate has the same qualitative impact on the 1^{st} PC as higher `TAX`.
- The second PC is obviously dominated by the proximity to the Charles river, `CHAS`.

Should I Normalize or Not?

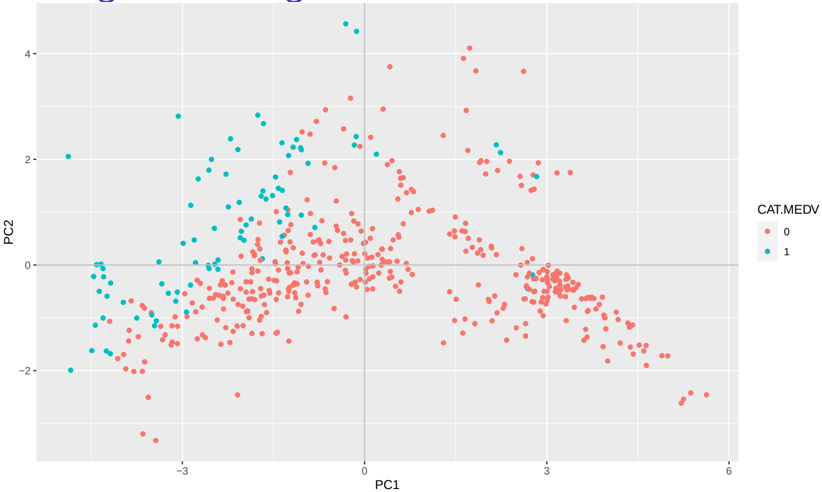
- Whether we should normalize or not depends on the nature of the data.
- It's a good idea to normalize when
  1. the variables have strikingly different scales and
  2. the variables are measured in different units so that it is unclear how to compare the variability of different variables (e.g., dollars for some, parts per million for others)
- Even if the units of measurement are common (e.g., dollars) for variables, scaling may be appropriate when variable scale does not reflect importance (earnings per share, gross revenues).
- However, when the units of measurement are common for the variables, and when their scale reflects their importance (sales of jet fuel, sales of heating oil), it is probably best not to normalize.

Visualizing Data Using PCs

- Suppose we are interested in predicting whether a neighborhood is a high- or low-value neighborhood (whether `CAT.MEDV=1` or 0).
- If we limit our attention to two principal components, a useful plot is a scatter plot of the first vs. second PC scores with colors (or labels) determined by `CAT.MEDV`.

```
housing.df_w2PC<-cbind(housing.df,
                  pci_all_scaled$x[,1:2])
ggplot(housing.df_w2PC,aes(x=PC1,y=PC2,
                 color=as.factor(CAT.MEDV)))+
          geom_point()+labs(color="CAT.MEDV")+
     geom_vline(xintercept = 0, color="gray")+
     geom_hline(yintercept = 0, color="gray")
```

Visualizing Data Using PCs



- With few exceptions, there are no high-value neighborhoods with `PC1 > 0`.
- When `PC1 < 0`, the greater the value of `PC2` the more likely it is for a neighborhood to be in the high-value category.

Using PCA for Classification and Prediction

- When the goal of the data reduction is either classification or prediction, we can proceed as follows:
  1. Apply PCA to the predictors using the training data.
  2. Use the output to determine the number of principal components to be retained.
  3. The predictors in the model will now use the reduced number of principal scores' columns.
  4. For the validation set, we can use the weights computed from the training data to obtain a set of principal scores by applying the weights to the variables in the validation set.
  5. These new variables are then treated as the predictors.