Principal Components Analysis

#Abstract This contribution develops an analysis on the dataset 'bad-drivers.csv' (taken from github.com/fivethirtyeight/data) by running the Principal Components Analysis. In particular, the study is focused on the different steps in the computation of the principal componets.
1.How to choose Principal Components 2.Proportion Variance Explained 3.Thanks to this it is now possible to consider just three of the principal components, always keeping in mind that the goal is to lower the cross-validation risk estimate. #Description of the dataset I chose the dataset 'bad-drivers.csv' (taken from github.com/fivethirtyeight/data), that considers data whose sources are: National Highway Traffic Safety Administration and National Association of Insurance Commissioners. This folder contains data behind the story 'Dear Mona, Which State Has The Worst Drivers?', where Mona Chalabi, a British data journalist answer some questions about drivers and car accidents in the USA. She says that historic data that could indicate where America's worst drivers are: the number of car crashes in each state (especially those where the driver was negligent in some way), how much insurance companies pay out, and how much insurance companies charge drivers. The reported variables are: . State (a specific country in the USA); . Number of drivers involved in fatal collisions per billion miles; . Percentage of drivers involved in fatal collisions who were speeding; . Percentage of drivers involved in fatal collisions who were alcohol impaired; . Percentage of drivers involved in fatal collisions who were not distracted; . Percentage of drivers involved in fatal collisions who had not been involved in any previous accidents; . Car insurance premiums; . Losses incurred by insurance companies for collisions per insured driver. I chose to compute the predictions on the variable 'Losses incurred by insurance companies for collisions per insured driver' (variable y).

##Analysis The Principal Component Analysis is one of the unsupervised learning methods and, as all unsupervised methods, it is much more challenging than supervised learning: in fact, it is often performed as an exploratory data analysis, whose goal is to check if there exist a possible dimension of the dataset in which I can study my data and see the relation between the variables. My goal here is to find a low-dimension representation of the dataset by defining the principal components. First of all, I imported the dataset, renamed all the columns and chose not to consider the first column of the dataset in order to make it easier to do the whole analysis. These are the first fifteen rows as example:

Then my analysis starts with the definition of the number of rows and columns, since they are useful for the next steps. In this case the number of rows is 51 and the number of column is 7. Two other elements needed are the mean and the standard deviation for each variable reported in the dataset. I built a table to summarize the means and the standard deviations; the output is the following: | | mean | std | |-|———|——| |a| 15.79 | 4.12| |b| 31.73 | 9.63| |c| 30.69 | 5.13| |d| 85.92 | 15.16| |e| 88.73 | 6.96| |f| 886.96 |178.30| |g| 134.49 | 24.84|

Based on this, I decided to run the analysis by starting from the correlation matrix (the alternative is the covariance matrix) that is the following:

	a	b	c	d	e	f	g
a	1.000	-0.029	0.199	0.010	-0.018	-0.200	-0.036
b	-0.029	1.000	0.286	0.132	0.014	0.043	-0.061
c	0.199	0.286	1.000	0.043	-0.245	-0.017	-0.084
d	0.010	0.132	0.043	1.000	-0.195	0.020	-0.058
e	-0.018	0.014	-0.245	-0.195	1.000	0.076	0.043
f	-0.200	0.043	-0.017	0.020	0.076	1.000	0.623
g	-0.036	-0.061	-0.084	-0.058	0.043	0.623	1.000

The absolute values are between 0 and 1 and it is possible to get the following information: -if the correlation is equal to 1 then the relationship is linear -in case the value is close to 0, it is non-linear -if both values tend to increase or decrease together the coefficient is positive, and the line that represents the correlation slopes upward, otherwise the coefficient is negative. In this case, most of the values have non-linear relationship.

Then, there is a very important step: the definition of the eigenvalues and eigenvectors. The eigenvalues

and eigenvectors of a correlation (or covariance) matrix represent the "core" of a PCA: the eigenvectors (principal components) represent the directions of the new feature space, whereas the eigenvalues determine their magnitude and, consequently explain the variance of the data along the new feature axes. The R function 'eigen()' outputs the eigenvalues and eigenvectors:

principal component analysis

bad_drivers <- read.csv("https://raw.githubusercontent.com/fivethirtyeight/data/master/bad-drivers/bad-drivers.csv", sep = ",") head(bad_drivers) dir.create("data") save(bad_drivers, file=file.path("data","bad_drivers.rda"))

a <- bad_drivers$Number.of.drivers.involved.in.fatal.collisions.per.billion.miles$b <- -bad_drivers$Percentage.Of.Drivers.Inv
c <- bad_drivers$Percentage.Of.Drivers.Involved.In.Fatal.Collisions.Who.Were.Alcohol.Impairedd <- -bad_drivers$Percentage.Of.Dri
-bad_drivers$Car.Insurance.Premiums. . . . g <- bad_drivers$Losses.incurred.by.insurance.companies.for.collisions.per.insured.dr

bad_drivers <- data.frame(a,b,c,d,e,f,g) bad_drivers

n <- nrow(bad_drivers) #number of rows p <- ncol(bad_drivers) #number of columns

mean and std:

mean <- colMeans(bad_drivers) std <- apply(bad_drivers, 2, sd) table <- round(cbind(mean, std),2) table

PCA starting from correlation matrix

calculate matrix R:

corr <- cor(bad_drivers) round(corr,3)

calculate eigenvalues and eigenvectors:

eigen(corr) eigenvalues <- eigen(corr)$values$eigenvectors <- -eigen(corr)vectors

Dall'analisi degli autovalori, le uniche componenti principali di rilevanza

#sono la prima e la seconda (le altre, infatti, hanno autovalori, quindi varianza, #minori di 1). Vediamo cosa accade se si sceglie il numero di componenti principali #sulla base della percentuale di # varianza spiegata: pvarsp = eigenvalues/p pvarspcum = cumsum(pvarsp) tab <- round(cbind(eigenvalues,pvarsp*100,pvarspcum*100),2) colnames(tab)<-c("eigenvelues", "% variance","% cum variance") tab

Use Scree Diagram to select the components:

plot(eigenvalues, type="b", main="Scree Diagram", xlab="Number of Component", ylab="Eigenvalues") abline(h=1, lwd=3, col="red")

We select 4 components

Interpret the principal components selected by their coefficient vectors:

eigen(corr)$vectors[,1:4]

#Matrix of the components, obtained by multiplying the eigenvector by the root of the respective eigenvalue (if necessary we can change the sign for interpretative reasons) comp <-round(cbind(-eigen(rho)$vectors[,1] * sqrt(eigenvalues[1]), -eigen(rho)$vectors[,2] sqrt(eigenvalues[2]),-eigen(rho)$vectors[,3] * sqrt(eigenvalues[3]), -eigen(rho)$vectors[,3] sqrt(eigenvalues[4])),5) rownames(comp)<-row.names(table) colnames(comp)<-c("Comp1","Comp2","Comp3","Comp4") comp

The sum of the squares of the values of each row of the component

#matrix is the respective 'communality', # The communality is the sum of the squared component loadings #up to the number of components you extract. communality<-comp[,1]$^{2+comp[,2]}$2+comp[,3]$^{2+comp[,4]}$2 comp<-cbind(comp,communality) comp

Calculate the scores for the selected components and graph them:

bad_drivers.scale <- scale(bad_drivers, T, T) score <- bad_drivers.scale%*%autovec[,1:4] # normalized scores changed sign (non-normalized scores divided by #square root of the respective eigenvalue) ## score chart scorez<-round(cbind(-score[,1]/sqrt(eigenvalues[1]),-score[,2]/sqrt(eigenvalues[2]),-score[,3]/sqrt(eigenvalues[3]),-score[,4]/sqrt(eigenvalues[4])),4) plot(scorez, main="Scores plot") text(scorez, rownames(bad_drivers)) abline(v=0,h=0,col="red") # Loadings plot plot(comp[,1:4], main="Loadings plot", xlim=range(-1,1)) text(comp, rownames(comp)) abline(v=0,h=0,col="red")