Prinicipal Component Regression (PCR)- Boston Data

Tanweer Shapla

2023-12-05

## Principal Component Regression

Principal Component Regression (PCR) is a statistical technique that is based on the dimension reduction and combines the principles of principal component analysis (PCA) and multiple linear regression analysis. It is used in situations where there is multicollinearity among the predictor variables, meaning that the predictor variables are correlated with each other. This multicollinearity can cause issues in traditional multiple linear regression models by inflating the standard error of parameter estimates making the estimation less precise.

Here are some key features of Principal Component Regression (PCR):

PCR reduces the dimensionality of a dataset by projecting it onto a lower-dimensional subspace, using a set of orthogonal linear combinations of the original variables called principal components.

PCR is often used as an alternative to multiple linear regression, especially when the number of variables is large or when the variables are correlated.

By using PCR, we can reduce the number of variables in the model and improve the interpretability and stability of the regression results.

To perform PCR, we first need to standardize the original variables and then compute the principal components using singular value decomposition (SVD) or eigen decomposition of the covariance matrix of the standardized data.

The principal components are then used as predictors in a linear regression model, whose coefficients can be estimated using least squares regression or maximum likelihood estimation.

Below are couple of notes:

**PCR vs. PCA**: PCR is similar to PCA, in that both techniques use principal components to reduce the dimensionality of the data. However, PCR differs from PCA in that it uses the principal components as predictors in a linear regression model, whereas PCA is an unsupervised technique that only analyzes the structure of the data itself, without using a response variable.

**PCR vs. PLSR**: A possible drawback of PCR is that we have no guarantee that the selected principal components are associated with the outcome. Thus, the selection of the PCs to incorporate in the model is not supervised by the outcome variable.

An alternate to PCR is Partial least squares regression (PLSR), which identifies new principal components that not only summarizes the original predictors, but also are related to the outcome. These components are then used to fit the regression model. So, compared to PCR, PLS uses a dimension reduction technique that is supervised by the outcome.

Overall, PCR is a useful technique for regression analysis that can be compared to multiple linear regression, PCA, and PLSR, depending on the specific characteristics of the data and the goals of the analysis.

Some of the most notable advantages of performing PCR are the following:

**Dimension reduction**

By using PCR you can easily perform dimensionality reduction on a high dimensional dataset and then fit a linear regression model to a smaller set of variables, while at the same time keep most of the variability of the original predictors. Since the use of only some of the principal components reduces the number of variables in the model, this can help in reducing the model complexity, which is always a plus. In case you need a lot of principal components to explain most of the variability in your data, say roughly as many principal components as the number of variables in your dataset, then PCR might not perform that well in that scenario.

**Addressing multicollinearity**

A significant benefit of PCR is that by using the principal components, if there is some degree of multicollinearity between the variables in your dataset, this procedure should be able to avoid this problem since performing PCA on the raw data produces linear combinations of the predictors that are uncorrelated.

**Potential drawbacks and warnings**

As always with potential benefits come potential risks and drawbacks.

Note that each of the calculated principal components is a linear combination of the original variables. Using principal components instead of the actual variables make it harder to explain what is affecting what.

Another major drawback of PCR is principal components are obtained in an unsupervised way. Therefore, there is no guarantee that the selected principal components are associated with the outcome.

**Steps in PCR**

Principal Component Analysis (PCA): The first step in PCR is to perform PCA on the original set of independent variables. PCA is a dimension reduction technique that transforms the original variables into a new set of uncorrelated variables called principal components. These principal components are linear combinations of the original variables, and they capture the maximum variance in the data.

Selection of Principal Components: In PCA, you typically select a subset of the principal components rather than using all of them. The number of principal components chosen is often determined based on the amount of variance they explain.

Regression Modeling: After selecting the principal components, a multiple linear regression model is built using these components as predictor variables. The regression coefficients obtained from this model are then used to make predictions on the outcome or response variable.

Before performing PCR, it is preferable to standardize your data. This step is not necessary but strongly suggested since PCA is not scale invariant. Variables with higher variance will influence more the calculation of the principal components and overall have a larger effect on the final results of the algorithm.

## Example : Boston data

We use Boston data from MASS package. Read the description of the data from Help page.

#install.packages("MASS")  
data("Boston", package="MASS")  
names(Boston)

[1] "crim" "zn" "indus" "chas" "nox" "rm" "age"   
 [8] "dis" "rad" "tax" "ptratio" "black" "lstat" "medv"

dim(Boston)

[1] 506 14

## Building PCR

Splitting the data: We assign 80% of the data to training set and 20% data to the testing set. We build the model based on the training set and evaluate its performance using the testing set.

set.seed(123)#for reproducing results  
id=sample(nrow(Boston),ceiling(0.80\*nrow(Boston)), replace=F)  
train.Boston=Boston[id,]  
test.Boston=Boston[-id,]

Below, we use train() function from caret package to build the principal component regression model. We consider the median value of houses as the response variable. The option method can take value pcr or pls. The argument scale=T standardizes the variables.

We use k-fold cross-validation which is specified by the argument trControl=trainControl. For this example, we use k=10 specified by number = 10.

caret uses cross-validation to automatically identify the optimal number of principal components (ncomp) to be incorporated in the model. We will test 10 different values of tuning parameter ncomp. This is specified using the option tuneLength. The optimal number of principal components is selected so that the cross-validation error (RMSE) is minimized. Make sure to use set seed for reproducibility.

set.seed(123) #for cross-validation reproducibility  
library(caret) #needed for train() function  
  
pcr.model=train(medv~., data=train.Boston, method="pcr", scale=T, trControl=trainControl(method="cv", number = 10), tuneLength=10)  
  
#method="repeatedcv", repeats=3 for repeated k=fold cross-validation  
  
pcr.model

Principal Component Analysis   
  
405 samples  
 13 predictor  
  
No pre-processing  
Resampling: Cross-Validated (10 fold)   
Summary of sample sizes: 364, 365, 364, 364, 364, 365, ...   
Resampling results across tuning parameters:  
  
 ncomp RMSE Rsquared MAE   
 1 7.038736 0.4208042 4.996474  
 2 6.590128 0.4969077 4.759402  
 3 5.440278 0.6520666 3.909461  
 4 5.060007 0.6981832 3.521990  
 5 4.888088 0.7279121 3.366218  
 6 4.888686 0.7283287 3.368996  
 7 4.905693 0.7266457 3.380026  
 8 4.897631 0.7294699 3.401646  
 9 4.932080 0.7250324 3.423196  
 10 4.952640 0.7268383 3.478901  
  
RMSE was used to select the optimal model using the smallest value.  
The final value used for the model was ncomp = 5.

summary(pcr.model)

Data: X dimension: 405 13   
 Y dimension: 405 1  
Fit method: svdpc  
Number of components considered: 5  
TRAINING: % variance explained  
 1 comps 2 comps 3 comps 4 comps 5 comps  
X 47.43 58.62 67.77 74.55 80.76  
.outcome 36.86 47.44 62.01 67.02 69.60

pcr.model$bestTune# print the best tuning parameter ncomp for which RMSE is minimum

ncomp  
5 5

Taken together, cross-validation identifies ncomp=5 as the optimal number of PCs that minimize the prediction error (RMSE) and explains enough variation in the predictors and in the outcome.

The summary() function provides the percentage of variance explained in the predictors (X) and in the outcome (medv) using different number of components in the training data.

For example 80.76% of the variation (or information) contained in the predictors are captured by 5 principal components. Also, 69.60% of the variation in the response variable medv is explained by these 5 principal components.

We can print number of principal components, and associated RMSE and MAE as given below:

ncomp=pcr.model$results$ncomp  
ncomp

[1] 1 2 3 4 5 6 7 8 9 10

RMSE=pcr.model$results$RMSE  
RMSE

[1] 7.038736 6.590128 5.440278 5.060007 4.888088 4.888686 4.905693 4.897631  
 [9] 4.932080 4.952640

Rsquared=pcr.model$results$Rsquared  
Rsquared

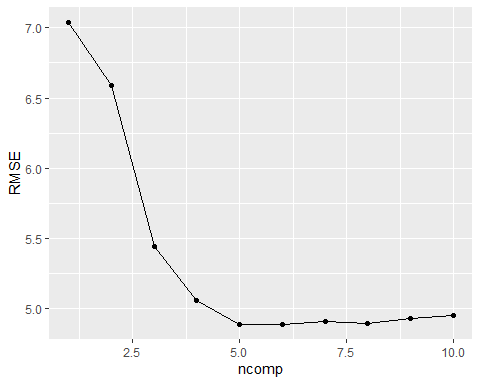
[1] 0.4208042 0.4969077 0.6520666 0.6981832 0.7279121 0.7283287 0.7266457  
 [8] 0.7294699 0.7250324 0.7268383

MAE=pcr.model$results$MAE  
MAE

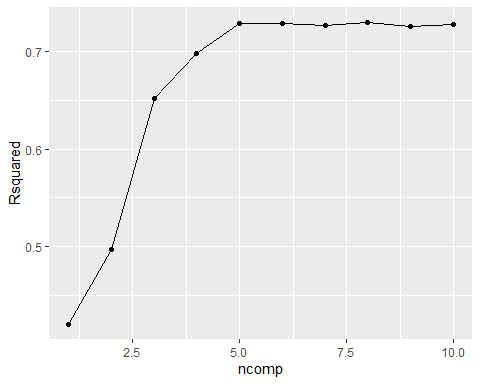
[1] 4.996474 4.759402 3.909461 3.521990 3.366218 3.368996 3.380026 3.401646  
 [9] 3.423196 3.478901

The plot of RMSE vs ncomp and R squared value vs ncomp can be drawn as follows:

library (ggplot2)  
  
RMSE.ncomp=data.frame(cbind(ncomp, RMSE))  
ggplot(RMSE.ncomp, mapping=aes(x=ncomp,y=RMSE))+geom\_point()+geom\_line()



R2.ncomp=data.frame(cbind(ncomp,Rsquared))  
ggplot(R2.ncomp, mapping =aes(x=ncomp, y=Rsquared))+geom\_point()+geom\_line()



Finally, we assess the performance of the pcr model on test data by computing the predicted outcome values of medv from the test data and then calculating the RMSE, Rsquared and MAE values.

predictions=predict(pcr.model, test.Boston)  
(RMSE.p=RMSE(predictions, test.Boston$medv))

[1] 4.932123

(R.squared.p=R2(predictions, test.Boston$medv))

[1] 0.7163319

(MAE.p=MAE(predictions, test.Boston$medv))

[1] 3.487195

The estimated prediction error rate is 0.2178538.

error.rate=RMSE.p/mean(test.Boston$medv)  
error.rate

[1] 0.2178538

## Building PLS model

set.seed(123) #for cross-validation reproducibility  
library(caret) #needed for train() function  
  
pls.model=train(medv~., data=train.Boston, method="pls", scale=T, trControl=trainControl(method="cv", number = 10), tuneLength=10)  
  
#method="repeatedcv", repeats=3 for repeated k=fold cross-validation  
  
pls.model

Partial Least Squares   
  
405 samples  
 13 predictor  
  
No pre-processing  
Resampling: Cross-Validated (10 fold)   
Summary of sample sizes: 364, 365, 364, 364, 364, 365, ...   
Resampling results across tuning parameters:  
  
 ncomp RMSE Rsquared MAE   
 1 6.296761 0.5353332 4.443994  
 2 4.837827 0.7321052 3.364633  
 3 4.779023 0.7405499 3.348271  
 4 4.764914 0.7458244 3.390243  
 5 4.729299 0.7457032 3.414703  
 6 4.698864 0.7476776 3.376781  
 7 4.692161 0.7485488 3.381486  
 8 4.682189 0.7496978 3.369427  
 9 4.682832 0.7498402 3.366663  
 10 4.684145 0.7498746 3.366456  
  
RMSE was used to select the optimal model using the smallest value.  
The final value used for the model was ncomp = 8.

summary(pls.model)

Data: X dimension: 405 13   
 Y dimension: 405 1  
Fit method: oscorespls  
Number of components considered: 8  
TRAINING: % variance explained  
 1 comps 2 comps 3 comps 4 comps 5 comps 6 comps 7 comps  
X 46.17 57.06 64.86 69.90 75.71 78.98 83.23  
.outcome 49.31 70.41 72.32 73.65 74.12 74.26 74.34  
 8 comps  
X 85.82  
.outcome 74.40

pls.model$bestTune# print the best tuning parameter ncomp for which RMSE is minimum

ncomp  
8 8

predictions.pls=predict(pls.model, test.Boston)  
(RMSE.pls=RMSE(predictions.pls, test.Boston$medv))

[1] 4.882126

(R.squared.pls=R2(predictions.pls, test.Boston$medv))

[1] 0.7225743

(MAE.pls=MAE(predictions.pls, test.Boston$medv))

[1] 3.635244

In our example, the cross-validation error RMSE obtained with the PLS model is lower than the RMSE obtained using the PCR model. So, the PLS model is the best model for explaining our data, compared to the PCR model.