Principal Component Regression. PLS.

Mathematical Tools for ITS (11MAI)

Mathematical tools, 2022

Jan Přikryl

(based on the book "Introduction to Statistical Learning", `https://www.statlearning.com/`)

11MAI, lecture 8

Monday, November 21, 2022

version: 2022-11-21 13:12

Department of Applied Mathematics, CTU FTS

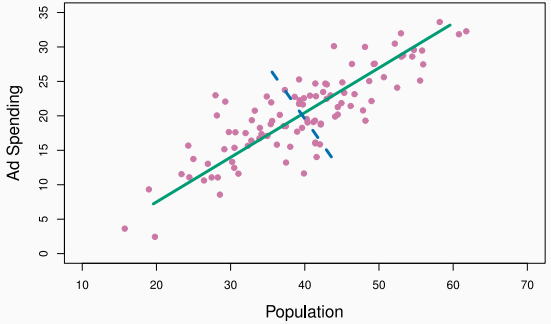
Lectue Contents

Dimensionality reduction: Principal Components Regression

We said that a clever linear combination of original predictors may result in a set of synthetic predictors with lower variance.

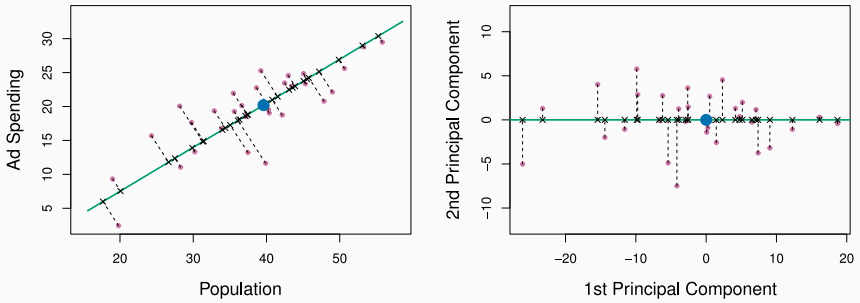
- Let's apply *principal components analysis (PCA)* to define the linear combinations of the predictors that we will use in our regression.
- The first principal component is the (normalized) linear combination of the predictors with the largest variance. The second principal component has the largest variance, subject to being uncorrelated with the first. And so on ...
- Hence with many correlated original variables, we replace them with a small set of principal components that capture their joint variation.

Pictures of PCA



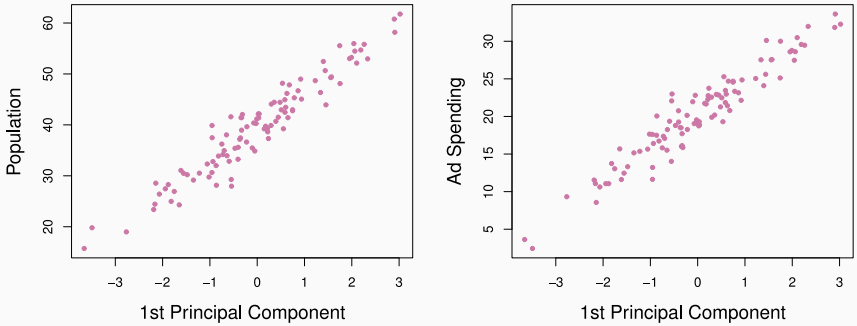
Advertising data: The population size (pop) and ad spending (ad) for 100 different cities are shown as purple circles. The green solid line indicates the first principal component, and the blue dashed line indicates the second principal component.

Pictures of PCA: continued



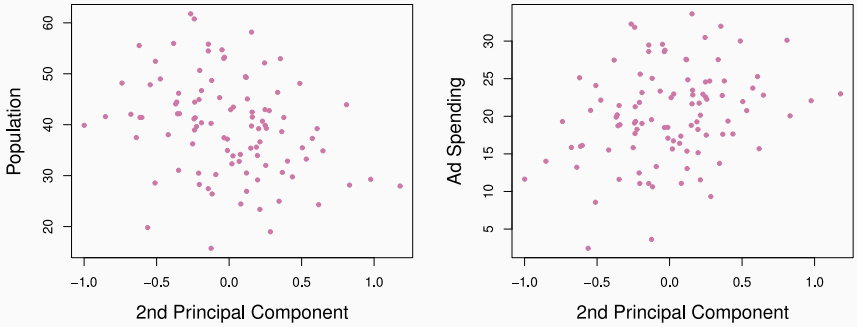
A subset of the advertising data. Left: The first principal component, minimizing the sum of the squared perpendicular distances to each point, is shown in green. These distances are represented using the black dashed line segments. Right: The left-hand panel has been rotated so that the first principal component lies on the x-axis.

Pictures of PCA: continued



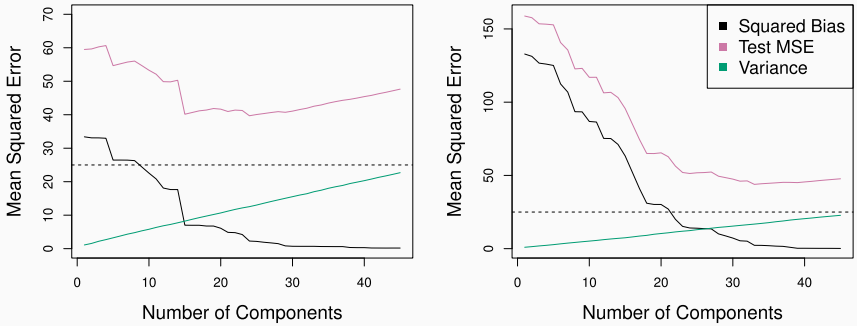
Plots of the first principal component scores z_{i1} versus pop *and* ad*. The relationships are strong.*

Pictures of PCA: continued



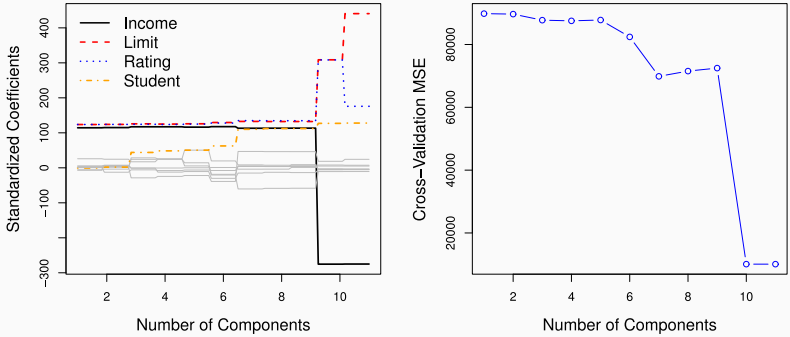
Plots of the first principal component scores z_{i2} versus pop *and* ad*. The relationships are weak.*

Application to Principal Components Regression



PCR was applied to two simulated data sets. The black, green, and purple lines correspond to squared bias, variance, and test mean squared error, respectively. Left: Simulated data from slide ??. Right: Simulated data from slide ??.

Choosing the number of directions M



Left: *PCR standardized coefficient estimates on the* `Credit` *data set for different values of* M. Right: *The 10-fold cross-validation MSE obtained using PCR, as a function of* M.

Important drawback of PCR

PCR reduces the number of "important" predictors.

- It identifies linear combinations, or *directions*, that best represent the predictors $X_1, \ldots, X_p$.
- These directions are identified in an *unsupervised* way, since the response Y is not used to help determine the principal component directions.
- That is, the response *does not supervise* the identification of the principal components.
- Consequently, PCR suffers from a potentially serious drawback:
  There is no guarantee that the directions that best explain the predictors will also be the best directions to use for predicting the response.

Matlab Session 8.1

Matlab Session 8.1 — PCR on Synthetic data

Matlab Session 8.2

Matlab Session 8.2 — PCR on Hitters dataset

Let us now use our the function `islr_hitters_split()` to read in and process the ISLR "Hitters" dataset using PCA regression.

```
% Make the random number sequence always the same.
rng(3);
% Read in the data
[train,test,data,hitters] = islr_hitters_split();
```

We will perform the standard PCA on the dataset, which will yield a matrix of PCA coeeficients.

```
% The tilde ~ ignores an output variable.
[coeffs,scores,~,~,varexpl,muest] = pca(train.X);
```

Now we have to select the optimal minimal number of coefficients for regression. This will be accomplished using cross-validation.

To use the CV to find the most appropriate number of PCA components we need to define a new function `mai_82_pcareg()` that will perform one round of model identification on training data and return the test data prediction using this particular model:

```
function y_test = mai_82_pcareg(X_train,y_train,X_test)
    mdl = fitlm(X_train, y_train);
    y_test = mdl.predict(X_test);
end
```

We will use the library function `crossval()` for the cross-validation.

The default is a 10-fold CV with a predictor function specified by `Predfun`.

```
num_components = size(scores, 2);
% Store the CV MSE for regression using 1,2,...,M components
cv_mse = zeros(num_components, 1);
% Compute the CV MSE for regression M=1, M=2, ... components
for M=1:num_components
    X = scores(:,1:M); % Select the first M of already PCA mapped inputs
    cv_mse(M) = crossval('mse', X, train.y, 'Predfun', @mai_82_pcareg);
end
```

Plot the estimated MSE as a function of the number of components:

```
figure(1);
plot(cv_mse);
ylim([0,1.05*max(cv_mse)]); % To have some margins
xlabel('Number␣of␣PCA␣components␣used␣(M)');
ylabel('10-fold␣CV␣MSE');
```

Plot the total explained variance as a function of the number of components:

```
figure(3);
plot(cumsum(varexpl));
xlabel('Number␣of␣PCA␣components␣used␣(M)');
ylabel('Variance␣explained␣by␣using␣M␣components');
```

Matlab Session 8.2 — PCR on Hitters dataset

Let us use the CVMSE-based guess that 7 components are optimal:

```
comp_range = 1:7;
test_X_pca = (test.X - muest) * coeffs(:,comp_range);
```

Fit a linear regression model on the testing data transformed into PCA scores and provide us with the predicted test response:

```
y_test = mai_82_pcareg(scores(:,comp_range), train.y, test_X_pca);
y_mse = mean((y_test - test.y).^2);
```

Partial Least Squares (PLS)

PLS is a dimension reduction method, which (like PCR) first identifies a new set of features $Z_1, \ldots, Z_M$ that are linear combinations of the original features $X_1, \ldots, X_p$, and then fits a linear model via OLS using these M new features.

- But unlike PCR, PLS identifies these new features in a *supervised* way – that is, it makes use of the response Y in order to identify new features that not only approximate the old features well, but also that are related to the response.
- Roughly speaking, the PLS approach attempts to find directions that help explain both the response and the predictors.

Details of Partial Least Squares

After standardizing the p predictors, PLS computes the first direction Z_1 by setting each ϕ_{1j} in

$$Z_m = \sum_{j=1}^{p} \phi_{mj} X_j$$

equal to the coefficient from the simple linear regression of Y onto X_j.

- One can show that this coefficient is proportional to the correlation between Y and X_j.
- Hence, in computing $Z_1 = \sum_{j=1}^{p} \phi_{1j} X_j$ PLS places the highest weight on the variables that are most strongly related to the response.
- Subsequent directions are found by taking residuals and then repeating the above prescription.

Matlab Session 8.3

Matlab implements partial least squares (PLS) using the `plsregress()` function from the Statistics and Machine Learning Toolbox.

```
num_components = size(data_x, 2);
num_cv_folds = 10;
[XL,yl,XS,ys,beta,varexpl,cv_mse] = plsregress(...
    data_x, data_y, num_components, 'cv', num_cv_folds);
```

Matlab Session 8.4

Matlab implements partial least squares (PLS) using the `plsregress()` function from the Statistics and Machine Learning Toolbox.

```
num_components = size(data_x, 2);
num_cv_folds = 10;
[XL,yl,XS,ys,beta,varexpl,cv_mse] = plsregress(...
    data_x, data_y, num_components, 'cv', num_cv_folds);
```