STA 663 - Final Project Report

Application and Optimization of
Partial Least Square Regression

Renjian Pan, Xinyue Ding
(GitHub: https://github.com/xynx59/PyPLS)

April 2019

Contents

Abstract

Partial least squares (PLS) is an extension of PCA in which both the **X** and **Y** data are considered. In PCA, only the **X** data is considered. The goal of the PLS analysis is to build an equation that predicts **Y** values based on **X** data.

For this project, we will go through the original Partial Least Square Regression method as well as the optimized version. The algorithm will be tested on both real-life data sets and one simulated data set. The goal of our optimized algorithm is to speed up the original algorithm, and also keep the performance accurate and robust.

From the comparison of results, we can tell that the performance of our algorithm is quite good when the dataset is small. Besides, the PLSR has better performance when **Y** has more than one variables. It gives us intuition of matrix decomposition and the inner relationship within variables.

Here is the outline of this report. Section 2 presents a basic introduction to the original PLSR algorithm as well as a table of notations. In Section 3 we describe an improved PLSR algorithm. Next, in Section 4 we will implement both of the algorithms on real life data and simulated data. Finally, Section 5 covers the conclusion and further discussion of PLSR algorithm in our project.

Keywords: Principal Component; Partial Least Square; Kernel Method; Dimensionality Reduction

1 Background

Partial least-squares regression (PLSR) is an important method in many industry fields such as chemistry and analytical, physical and clinical chemistry, as well as other industrial process control. It has been shown that PLSR is a robust and powerful method compared to some more classical method like multiple linear regression (MLR) and principal component regression (PCR). [1]

The PLS algorithm was originally been put forward in 1980s. In fact, the origin of PLSR lies in chemistry, and is designed to confront the situation that there are many, possibly correlated, predictor variables, and relatively few samples - a situation that is common, especially in chemistry where development in spectroscopy since the seventies have revolutionised chemical analysis. [2] There has been many improved algorithms based on the original PLSR algorithm aimed to improve the robustness or speed. We will discuss some of them and try to develop an optimized PLSR algorithm.

2 Algorithm Description

2.1 Notation

The table below lists the notation used.

Symbol	Meaning
$\mathbf{X}$	Independent variable matrix with size $n \times m$
$\mathbf{Y}$	Dependent variable matrix with size $n \times p$
$\mathbf{E_h}$	Residual of $\mathbf{X}$ after h iterations
$\mathbf{F_h}$	Residual of $\mathbf{Y}$ after h iterations
$\mathbf{B}$	Regression coefficients
$\mathbf{T}$	The matrix of $\mathbf{X}$ scores
$\mathbf{P}$	The matrix of $\mathbf{X}$ loadings
r	Rank of $\mathbf{X}$
m	Column numbers of $\mathbf{X}$
p	Column numbers of $\mathbf{Y}$
n	Row numbers of $\mathbf{X}$

Table 1: Notations used in this report

2.2 Principal Component Regression (PCR)

Principal component regression (PCR) is based on the method of Principal Component Analysis (PCA). The basic idea of PCA is dimensionality reduction, that is decomposing a $m \times n$ matrix $\mathbf{X}$ into

$$\mathbf{X} = \mathbf{T}\mathbf{P}^T$$

where $\mathbf{T}$ is called as score matrix and $\mathbf{P}$ as loading matrix. The dimensionality of $\mathbf{T}$ and $\mathbf{P}$ is $m \times r$ and $n \times r$ respectively, where r is the fixed number of principal components and satisfying $1 \leq r \leq rank(\mathbf{X})$.

PCA can be conducted by either matrix decomposition (like spectral decomposition or singular value decomposition) or nonlinear numerical computation. Nonlinear iterative partial least square (NIPALS) is one of the most popular nonlinear approaches. Unlike matrix decomposition, it does not calculate all the principal components at once. Instead, it calculates $\mathbf{t_1}$ and $\mathbf{p_1}$ from the $\mathbf{X} := \mathbf{E_0}$ matrix, and then subtract the outer product $\mathbf{t_1 p_1^T}$ from $\mathbf{E_0}$. Then the residual $\mathbf{E_1}$ is calculated and can be used in the next round of iteration to derive $\mathbf{t_2}$, $\mathbf{p_2}$.

The NIPALS algorithm is as follows:

Algorithm 1: NIPALS

Input : $\mathbf{X}$, r(number of principal components), eps(convergence checking)
Output: $\mathbf{T}, \mathbf{P}$
$\mathbf{E_0} = \mathbf{X}$
for h *in* $1 : r$ **do**
  (1) Take a vector $\mathbf{x_j}$ from $\mathbf{X}$ and denote it as $\mathbf{t_h}$: $\mathbf{t_h} = \mathbf{x_j}$
  (2) Calculate $\mathbf{p_h}$: $\mathbf{p_h} = \frac{\mathbf{X^T t_h}}{\mathbf{t_h^T t_h}}$
  (3) Normalize $\mathbf{p_h}$: $\mathbf{p_h^{new}} = \frac{\mathbf{p_h^{old}}}{\|\mathbf{p_h^{old}}\|}$
  (4) Calculate $\mathbf{t_h}$: $\mathbf{t_h} = \frac{\mathbf{X p_h}}{\mathbf{p_h^T p_h}}$
  (5) Check convergence:
  if $\|\mathbf{t_h^{old}} - \mathbf{t_h^{new}}\| > eps$ **then**
    $\mathbf{E_h} = \mathbf{E_{h-1}} - \mathbf{t_h p_h^T}$
    $h := h + 1$
  else
    stop and break out the iteration

It has been proven that on convergence, the NIPALS algorithm is the same as that calculated by eigenvector formulae. The advantages of NIPALS method are the convenience for computation as well as a good understanding of PLS.[1]

The PCR method is built on PCA we described above. After decomposing $\mathbf{X}$ into $\mathbf{TP^T}$, we have

$$\mathbf{T} = \mathbf{XP}$$

Then the original MLR formula

$$\mathbf{Y} = \mathbf{Xb} + \mathbf{e}$$

can be rewritten as

$$\mathbf{Y} = \mathbf{TB} + \mathbf{E}$$

with the solution

$$\hat{\mathbf{B}} = (\mathbf{T^T T})^{-1} \mathbf{T^T Y}$$

PCR solves the collinearity problem by guaranteeing an invertible matrix in the calculation of $\hat{\mathbf{B}}$ and is well conditioned. Therefore it is reasonable for us to claim that PCR is more robust than MLR.

2.3 Partial Least-Square Regression (PLSR)

The PLSR model is based on the NIPALS algorithm we described above. As mentioned in the PCR section, it is possible to let score matrix represent the data matrix. A simplified model would consist of a regression between the scores for both $\mathbf{X}$ and $\mathbf{Y}$ blocks. The PLS model can be considered a consisting of outer relations($\mathbf{X}$ and $\mathbf{Y}$ block individually) as well as inner relations (linking both blocks). [1]

The PLSR algorithm is given as below.

Algorithm 2: PLSR

Input : $\mathbf{X}, \mathbf{Y}$, r(number of principal components), eps(convergence checking)
Output: $\mathbf{T}, \mathbf{P}$
Standarlize $\mathbf{X}, \mathbf{Y}$
$\mathbf{E_0} = \mathbf{X}, \mathbf{F_0} = \mathbf{Y}$
for h *in* $1 : r$ **do**
- (1) Take a vector $\mathbf{y_j}$ from $\mathbf{Y}$ and denote it as $\mathbf{u_h}$: $\mathbf{u_h} = \mathbf{y_j}$
- (2) $\mathbf{w_h} = \frac{\mathbf{X^T u_h}}{\mathbf{u_h^T u_h}}$
- (3) $\mathbf{w_h^{new}} = \frac{\mathbf{w_h^{old}}}{\|\mathbf{w_h^{old}}\|}$
- (4) $\mathbf{t_h} = \frac{\mathbf{X w_h}}{\mathbf{w_h^T w_h}}$
- (5) $\mathbf{q_h} = \frac{\mathbf{X^T t}}{\mathbf{t^T t}}$
- (6) $\mathbf{q_h^{new}} = \frac{\mathbf{q_h^{old}}}{\|\mathbf{q_h^{old}}\|}$
- (7) $\mathbf{u_h} = \frac{\mathbf{Y q_h}}{\mathbf{q_h^T q_h}}$
- (8) **if** $\|\mathbf{t_h^{old}} - \mathbf{t_h^{new}}\| > eps$ **then**
  - go to (2)
- **else**
  - go to (9)
- (9) $\mathbf{p_h} = \mathbf{X_h^t}$
- (10) $\mathbf{p_h^{new}} = \mathbf{p_h^{pld}} / \|\mathbf{p_h^{old}}\|$
- (11) $\mathbf{t_h^{new}} = \mathbf{t_h^{old}} \|\mathbf{p_h^{old}}\|$
- (12) $\mathbf{w_h^{new}} = \mathbf{w_h^{old}} \|\mathbf{p_h^{old}}\|$
- (13) Calculate regression cefficient:
  $b_h = \mathbf{u^T t / t^T t}$
- (14) Calculate and update residuals:
  $\mathbf{E_h} = \mathbf{E_{h-1}} - \mathbf{t_h p_h^T}, \mathbf{F_h} = \mathbf{F_{h-1}} - \mathbf{b_h t_h q_h^T}$

Note:
(1) After the first component, $\mathbf{X}$ in steps (2), (4) and (9) and $\mathbf{Y}$ in steps (5) and (7) are replaced by $\mathbf{E_h}$ and $\mathbf{F_h}$.

(2) If $\mathbf{Y}$ block has only one variable, steps (5) - (8) can be omitted by simply putting $q = 1$ and no more iteration is necessary.

3 Optimized Algorithm

Kernel PLS algorithm is one of the improved PLS algorithms. It is first proposed by Lindgren in 1994 [3] and further developed by De Jong and Ter Braak[4]. It has been proved that these kernel algorithms are much faster than the original PLS algorithms. [5]

3.1 Kernel Algorithm

Algorithm 3: Kernel RPLS

Input : $\mathbf{X}$, $\mathbf{Y}$, r(number of principal components), eps(convergence checking)
Output: $\mathbf{T}, \mathbf{P}$
Standarlize $\mathbf{X}, \mathbf{Y}$
(1) Compute the covariance matrices $(\mathbf{X^T X})_h$ and $(\mathbf{X^T Y})_h$
(2) $\mathbf{w_h}$ is computed as the eigenvector corresponding to the largest eigenvalue
(3) Compute $\mathbf{p_h}, \mathbf{q_h}$
$\mathbf{p_h^T} = \frac{\mathbf{w_h^T X^T X_h}}{\mathbf{w_h^T X^T X_h w_h}}$
$\mathbf{q_h^T} = \frac{\mathbf{w_h^T Y^T X_h}}{\mathbf{w_h^T X^T X_h w_h}}$
(4) After each latent vector computation, update the covariance matrices
$(\mathbf{X^T X})_{\mathbf{h+1}} = (\mathbf{I} - \mathbf{w_h p_h^T})^{\mathbf{T}} (\mathbf{X^T X})_{\mathbf{h}} (\mathbf{I} - \mathbf{w_h p_h^T})$
$(\mathbf{X^T Y})_{\mathbf{h+1}} = (\mathbf{I} - \mathbf{w_h p_h^T})^{\mathbf{T}} (\mathbf{X^T Y})_{\mathbf{h}}$

There is also another modified version of kernel PLSR algorithm. The algorithm is given as below.

Algorithm 4: Modified Kernel RPLS

Input : $\mathbf{X}$, $\mathbf{Y}$, r(number of principal components), eps(convergence checking)
Output: $\mathbf{T}, \mathbf{P}$
Standarlize $\mathbf{X}, \mathbf{Y}$
(1) Compute the covariance matrices $(\mathbf{X^T X})_h$ and $(\mathbf{X^T Y})_h$
(2) $\mathbf{w_h}$ is computed as the eigenvector corresponding to the largest eigenvalue
(3) Compute $\mathbf{p_h}, \mathbf{q_h}$
$\mathbf{p_h^T} = \frac{\mathbf{w_h^T X^T X_h}}{\mathbf{w_h^T X^T X_h w_h}}$
$\mathbf{q_h^T} = \frac{\mathbf{w_h^T Y^T X_h}}{\mathbf{w_h^T X^T X_h w_h}}$
(4) After each latent vector computation, update the covariance matrices
$(\mathbf{X^T X})_{\mathbf{h+1}} = (\mathbf{X^T X})_{\mathbf{h}} - \mathbf{p_h p_h^T} (\mathbf{t_h^T t_h})$
$(\mathbf{X^T Y})_{\mathbf{h+1}} = (\mathbf{X^T Y})_{\mathbf{h}} - \mathbf{p_h q_h^T} (\mathbf{t_h^T t_h})$

This reduces the computational effort during the deflation step by avoiding the multiplication of X^TX and X^TY by $(I - w_h p_h^T)$. This modified kernel algorithm was proven to be much faster than the original kernel algorithm. [5]

3.2 Improvements

In our python package, we also tried to improve the current PLSR from other approaches.

First, the PLSR and PCR algorithms currently implemented in some packages (like **pls** in R) do not handle missing values intrinsically, so observations with missing values must be removed before processing. This can be resolved by filling the NAs with some aggregation values, such as the median/mean for each variable. We can also handle the missing data problem by filling any other reasonable values by imputation. This improvement is included in our optimized PLSR algorithm.

Apart from handling missing data, we also introduce Cross-Validation into the optimized algorithm. Most of the current packages dealing with PLSR separate the cross validation part from the main PLSR fitting part, thus do not have the ability of automatically choosing the best number of principal components and cause many inconvenience. Therefore, we would like to add the cross validation process into our PLSR python package.

4 Application

4.1 Real-life Data

In this section, we will go through several real-life data sets to test our algorithm and code. Here we considered four methods: (1)our optimized PLSR code, (2)PLSR function in 'sklearn' package, (3) PLSR with fixed number of principal components, (4) LR from 'sklearn' package. For (3) we choose 30% of the features in $\mathbf{X}$.

We want to accelerate the algorithm without loss of accuracy. Here we consider two characteristics for evaluating the performance of algorithms: (1) The Frobenius norm of residuals of $\mathbf{Y}$, that is $\mathbf{F}_F$, (2) running time. For reference, we also include the dimension of data sets (the number of pricipal components we used).

1. The music-geography data
The dataset was built from a personal collection of 1059 tracks covering 33 countries/area and is made up of audio features extracted from 1059 wave files. The music used is traditional, ethnic or 'world' only, as classified by the publishers of the product on which it appears. The geographical location of origin was manually collected the information from the CD sleeve notes, and when this information was inadequate we searched other information sources. The task associated with the data is to predict the geographical origin of music. [7]

The comparison of performance and run-time is given in the table below.

	PLSR (optimized)	PLSR (sklearn)	PLSR (fixed PCs)	LR (sklearn)
$\|F\|_F$	42.0650	42.0652	42.3578	42.4799
RunTime	38.8760	7.3553	0.1307	0.0409
Dimension	6	6	20	None

Table 2: Results comparison on music-geography data

2. The residential building data
The residential building data set is one that includes construction cost, sale prices, project variables, and economic variables corresponding to real estate single-family residential apartments in Tehran, Iran. There are totally 105: 8 project physical and financial variables, 19 economic variables and indices in 5 time lag numbers (5*19 = 95), and two output variables that are construction costs and sale prices in this data set. [8]

The comparison of performance and run-time is given in the table below.

	PLSR (optimized)	PLSR (sklearn)	PLSR (fixed PCs)	LR (sklearn)
$\|F\|_F$	6.3753	6.3752	6.3944	8.0123
RunTime	40.9186	8.0973	0.1117	0.0359
Dimension	21	21	32	None

Table 3: Results comparison on residential building data

3. The gasoline data
The 'gasoline' data set is given by the R package 'pls'. The data set was originally provided by John H. Kalivas in his paper *Two Data sets of Near Infrared Spectra* and is made up of two parts: near infrared (NIR) spectra and octane numbers of 60 gasoline samples. The NIR spectra were measured using diffuse reflectance as log(1/R) from 900 nm to 1700 nm in 2 nm intervals, giving 401 wavelengths.[6] Below is a summary table of the data set.

Variable	Dimension	Description
octane	60×1	A numeric vector. The octane number.
NIR	60×401	a matrix with 401 columns. The NIR spectrum.

Table 4: Description of 'gasoline' data set

The comparison of performance and run-time is given in the table below.

	PLSR (optimized)	PLSR (sklearn)	PLSR (fixed PCs)	LR (sklearn)
$\|F\|_F$	35.1000	35.1000	35.8538	2.1915
RunTime	1.0233	0.866	0.0179	0.0539
Dimension	4	4	120	None

Table 5: Results comparison on gasoline data

4.2 Simulated Data

Practical examples with simulated data are described to illustrate the material described in the preceding tutorial on partial least-squares regression. [9]

It is possible for us to simulate a data set consisting $\mathbf{X}$ and $\mathbf{Y}$ from the MLR formula

$$\mathbf{Y} = \mathbf{X}\beta + \epsilon$$

as well as the matrix decomposition formula of $\mathbf{X}$

$$\mathbf{X} = \sum_{i=1}^{r} \mathbf{t_i}\mathbf{p_i^T} = \mathbf{T}\mathbf{P^T}$$

where r is the rank of $\mathbf{X}$.

Here we choose $m = 100, n = 200, r = 10$, hence $\mathbf{X}$ is a 1000×100 matrix and $\mathbf{Y}$ is a 100×5 matrix. We fix $\beta, \mathbf{t_i}, \mathbf{p_i}$ to be some fixed vectors.The python code for generating the simulated data is attached in the appendix.

The comparison of performance and run-time is given in the table below.

	PLSR (optimized)	PLSR (sklearn)	PLSR (fixed PCs)	LR (sklearn)
$\|F\|_F$	1.2607	1.2607	1.2607	1.8102
RunTime	10.4471	2.0256	0.0908	0.0269
Dimension	10	10	30	None

Table 6: Results comparison on simulated data

5 Conclusion

The PLS equation is based on decomposing both the X and Y data into a set of scores and loadings, similar to PCA. However, the scores for both the $\mathbf{X}$ and $\mathbf{Y}$ data are not selected based on the direction of maximum variation but are selected in order to maximize the correlation between the scores for both the $\mathbf{X}$ and $\mathbf{Y}$ variables. [10]

As with PCA, in the PLS regression development the number of components or factors is an important practical consideration. Therefore, we introduced cross validation for choosing the best number of principal components, as well as imputation for missing data to make the algorithm more robust.

We tested our code on three real-life data sets and also a simulated data set. We also compare the efficiency of our code with the 'sklearn' package in Python as well as the multivariate linear regression. The performance of our algorithm is quite good for small scale data sets; However, when the data set scale becomes huge, the speed of our algorithm could become slow. Further techniques in speeding up would be preferable for improving the performance of the current algorithm.

To be noted, when there is only one variable in $\mathbf{Y}$, the MLR(Multivariate Linear Regression) has much better performance than PLSR. This gives us some intuition on the idea of PLSR: this algorithm is mainly designed to solve the problem of co-linearity. When there is no inner relationship in $\mathbf{Y}$ block, there is no need to perform matrix decomposition on it.

Appendix

Attached is the python code for generating a simulated data.

```
%matplotlib inline
import numpy as np
import scipy.linalg as la
import matplotlib.pyplot as plt
import pandas as pd

# simulation
np.random.seed(58)
m = 100
n = 200
r = 10
T = np.zeros((n, r))
P = np.zeros((m, r))
for i in range(r):
    t = np.random.normal(0., 1., n)
    p = np.random.normal(0., 1., m)
    T[:,i] = t
    P[:,i] = p

X = T @ P.T
beta = np.random.normal(0., 1., (m,5))
Y = X @ beta + np.random.normal(0., 1., (n,1))

# convert to a dataframe
data = np.c_[Y,X]
cols = ['Y'+str(i) for i in range(1, 6)] + ['X'+str(i) for i in range(1, m-
df = pd.DataFrame(data = data, columns=cols)

# save data to .csv file
df.to_csv('simulated_data.csv')
```

References

[1] Abdi H. *Partial least square regression (PLS regression)*[J]. Encyclopedia for research methods for the social sciences, 2003, 6(4): 792-795.

[2] Wehrens R, Mevik B H. *The pls package: principal component and partial least squares regression in R*[J]. 2007.

[3] Rännar S, Lindgren F, Geladi P, et al. *A PLS kernel algorithm for data sets with many variables and fewer objects. Part 1: Theory and algorithm*[J]. Journal of Chemometrics, 1994, 8(2): 111-125.

[4] De Vries S, Ter Braak C J F. *Prediction error in partial least squares regression: a critique on the deviation used in The Unscrambler*[J]. Chemometrics and intelligent laboratory systems, 1995, 30(2): 239-245.

[5] Dayal B S, MacGregor J F. *Improved PLS algorithms*[J]. Journal of Chemometrics: A Journal of the Chemometrics Society, 1997, 11(1): 73-85.

[6] Kalivas J H. *Two data sets of near infrared spectra*[J]. Chemometrics and Intelligent Laboratory Systems, 1997, 37(2): 255-259.

[7] Zhou F, Claire Q, King R D. *Predicting the geographical origin of music*[C]//2014 IEEE International Conference on Data Mining. IEEE, 2014: 1115-1120.

[8] Rafiei M H, Adeli H. *A novel machine learning model for estimation of sale prices of real estate units*[J]. Journal of Construction Engineering and Management, 2015, 142(2): 04015066.

[9] Geladi P, Kowalski B R. *An example of 2-block predictive partial least-squares regression with simulated data*[J]. Analytica Chimica Acta, 1986, 185: 19-32.

[10] Issaq, Haleem J., and Timothy D. Veenstra, eds. *Proteomic and metabolomic approaches to biomarker discovery*. Academic Press, 2013.