Partial Least Square Regression

May 1, 2017

STA 663: Final Project
Yao Song, Yi Zhao

1 Abstract

Partial least-squares (PLS) regression method is a popular method in various disciplines such as chemistry, economics, medicine, psychology, and pharmaceutical science where the number of explanatory variables is high, and where it is likely that the explanatory variables are correlated. This paper introduces the basic concepts of PLS regression and implements the PLS method in python.

Keywords:*Partial Least-Squares Regression, Laten Structures, Correlation, Variance-Covariance Matrix, Decomposition, Multi-colinearity, Multidimensional.*

2 Introduction

2.1 Background

In many multivariate regression analysis, we often start with a large number of possible dependent/ independent variables. Usually, many varibles that measuring the same characteristics used in a regression are highly correlated. For instance, demographic variables measuring population density characteristics or health status variables measuring BMI, height and weight are often display a high degree of correlation. A high degree of correlation among the independent variables increases the variance in estimates of the regression parameters, and causes the problem of multicolinearity. Consequently, the estimated parameters in a regression equation may affected by data source and hence are not stable for future prediction. Therefore, dimension reduction is a major task for multivariate analysis[1].

Partial least-squares (PLS) regression method, a dimension reduction methodology, is applied with the consideration of the correlation between the dependent variable and the independent variables. PLS method was first introduced by the Swedish statistician Herman Wold in 1966 in the field of econometrics. This project implements the Partial least-squares regression method described by Geladi and Kowalski (1985) in their paper 'Partial Least-Squares Regression: A Tutorial'[2].

2.2 Application

The idea behind the PLS is based on linear transition from a large number of original descriptors to a new variable space based on small number of orthogonal factors (latent variables). The

latent variables are chosen in a way such that contains maximum correlation with dependent variable. Therefore, the PLS model would resulting in smallest necessary number of factors, which is different from other similar approaches such as standard regression and principal component regression (PCR). Thus, PLS is particularly useful when we need to predict a set of dependent variables from a large set of independent variables, and when there is multicollinearity among independent variables.

2.3 Comparision

The goal of PLS regression is to predict dependent variables from independent variables and to describe their common structure. When we got single dependent variable and independent variables matrix is full rank, the problem can be solved by using Ordinary Multiple Regression. However, when the number of predictors (independent variables) is large, X is likely to be singular and we no longer can use the regression approach due to multi-colinearity.

Various type of methodologies, such as Principle Component Analysis (PCA), Stepwise approach, and Partial least-squares (PLS) can be used to address the issue of multi-colinearity.

A well-known method in regression analysis that eliminate some predictors is called Stepwise Regression algorithem. One of its major limitation is that when some predictors are highly correlated, the tests of statistical significance in the Stepwise method are not reliable, since independence is one of the important underlying assumptions of these tests. Principal component regression(PCR) is another approach that performs PCA of the X matrix and then use only the principal components of X to predict Y. The orthogonality of the principal components eliminates the multi-colinearity problem, but it's hard to choose an optimum subset of predictors. Since the principal components are chosen to explain variabilities in X rather than Y, it's hard to guarantee that the principal components that explained X are relevant for Y. Comparing to PCR, PLS regression method finds components from X that are also relevant for Y. That is, PLS regression extract a set of components (named laten vectors) that simultaneously performs the decomposition of X and Y with the constraint that the extracted components explain as much as possible of the covariance between the dependent and independent variables. By doing this, PLS generalized the PCA method, and followed by a regression step that using the decomposition of X to predict Y. Therefore the main advantage of the PLS mehod is that it preserves the asymmetry of the relationship between independent and dependent variables, wherase the other methods treat them symmetrically [3].

Flow of this project:

1. Implement the Partial least-squares regression method in python.
2. Code optimization
3. Compare results with existing libraries

4. Apply the python code to simulated datasets
5. Apply the python code to real datasets

3 Description of Algorithm

3.1 Variable Notation

- n the number of samples in the training set
- m the number of independent variables

- p the number of dependent vairables
- a the number of factors used
- X a matrix of features for the independent variables ($n \times m$)
- Y a matrix of features for the dependent variables ($n \times p$)
- B a matrix of sensitivities for the MLR metod ($m \times p$)
- T the matrix of X scores ($n \times a$)
- P′ the matrix of X loadings ($a \times p$)
- t_h a column vector of scores of the X block, factor h ($n \times 1$)
- p'_h a row vector of loadings of the X block, factor h ($1 \times p$)
- w'_h a row vector of weights for the X block, factor h ($1 \times m$)
- U the matrix of Y scores ($n \times a$)
- Q′ the matrix of Y loadings ($a \times p$)
- u_h a column vector of scores of the Y block, factor h ($n \times 1$)
- q'_h a row vector of loadings of the Y block, factor h ($1 \times p$)
- E_h the residual of X after subtraction of h components ($n \times m$)
- F_h the residual of Y after subtraction of h components ($n \times p$)
- ***Note:*** in this paper, we assume that X (independent variables) and Y (dependent variables) are mean-centered and scaled.

3.2 Multiple Linear Regression (MLR)

The Multiple Linear Regression (MLR) attempts to model the relationship between m independent variables $x_j (j = 1 - m)$ and a dependent variable y by fitting a linear (or first-order) equation to observed data. In general, when we got n samples and m independent variables, the problem can be mathematically expressed as:

$$Y = X\beta + \epsilon$$

where Y is a $n \times 1$ column vector, X is a $n \times m$ matrix which the vectors x'_j form the rows of it ϵ. β is a $m \times 1$ column vector and ϵ, the 'noise'(residual or error) variable, is a Normally distributed random variable with mean equal to zero and standard deviation σ.

Three cases can be observed from a MLR problem:

1. When m > n (more variables than samples): there is no unique solution for β unless we delet independent variables
2. When m = n (equal number of samples and variables): there is unique solution for β provided that X has full rank. This would allows us to write $\epsilon = Y - X\beta = 0$ where ϵ is a vector of zeroes
3. When m < n (less variables than samples): we can use the least-squares method to minimize the length of the residual vector $\epsilon = Y - X\beta$ to get a solution for β. The least-squares solution is $\beta = (X'X)^{-1}X'Y$.

- Note: problem of collinearity/ zero determinant/ singularity would happen when $(X'X)^{-1}$ does not exist

MLR can also be used with more than one dependent variable bu putting Y, X and ϵ as matrices rather than vectors.

3.3 Principal Component Analysis (PCA): Nonlinear Iterative Partial Least Squares (NIPALS)

3.3.1 PCA

Principal Component Analysis (PCA) is a commonly used method to reduce the number of independent variables and solve the multi-colinearity problem. It looks for a few linear combinations of the variables that can be used to summarize the data without losing too much information.

First we need to know the concept of 'rank'. Rank of a matrix denotes the maximum number of lineraly independent rows or columns of a matrix (i.e. rank reveals the true underlying dimensionality of a matrix). Statistically, correlation is a measure of linear dependence among variables and presence of highly correlated variables indicate a linear dependence among the varibles. Knowing this, we can states the Principal Component Analysis method as follows:

- Decomposing a data matrix X of rank r to a sum of r rank 1 matrices:

$$X = M_1 + M_2 + M_3 + \cdots + M_r$$

- Writting these rank 1 matrices as outer products of two vectors: a score vector t_h and a loading vector p'_h:

$$X = t_1 p'_1 + t_2 p'_2 + \cdots + t_a p'_a = TP'$$

  where:

- p'_h(loadings): The wights of the variables in X on the scores. Of the loadings we can see which variables that are respondible for patterns found in scores. The elements are the direction cosines, or the projections of a unit vector along the principal component on the axes

- t_h(scores): Summary of the original variables in X that describe how the different rows in X relate to each other. The elements are the coordinates of the respective points on the principal component line.

3.3.2 NIPALS

The properties of PLS model can be analyzed from the original algorithm (named NIPALS). The NIPALS method calculate the scores and loadings pair-by-pair by an iterative procedure: it first calculates t_i and p'_i ($i = 1, 2, \dots$), then comput residual $E_i = X - t_1 p'_1$ which can then be used to calculate $E_{i+1} = E_i - t_{i+1} p'_{i+1}$.

The NIPALS algorithm is stated as follows:

- *Step 1:* Taking a vector x_J from the feature matrix X and call it t_h

$$t_h = x_J$$

- *Step 2:* Calculating a row vectorof loadings of the X block p'_h

$$p'_h = \frac{t'_h X}{t'_h t_h}$$

- *Step 3:* Normalizing the calculated p'_h to length 1 (i.e. transformed into Z-scores)

$$p'_{hnew} = \frac{p'_{hold}}{\|p'_{hold}\|}$$

- *Step 4:* Calculating t_h

$$t_h = \frac{Xp_h}{p_h' p_h}$$

- *Step 5:* Comparing t_hs, if t_h has not converged, then go to step 2, otherwise stop the iteration.

3.4 Principal Component Regression (PCR)

Principal Component Regression (PCR) is a regression analysis technique that is baed on PCA. PCR regress the dependent variables on a set of independent variables with the help of PCA for estimating the unknown regression coeffieics. The PCR algorithm can be states as follows:

- Representing a data matrix X by its score matrix T:

$$T = XP = TP'P = TI_n$$

  then the MLR formula can be written as

$$Y = TB + E$$

  with solution $\hat{B} = (T'T)^{-1}T'Y$

- Replacing the variables of X by new ones that have better properties (i.e., orthogonality) and also span the multidimensional space of X.

PCR solves the collinearity problem, since it gives no matrix inversion problems in the calculation of $\hat{B}$ and PCR has the ability to eliminate the lesser principal components allows some noise reduction.

3.5 Partial Least-Squares Regression (PLS)

3.5.1 Model Building

Based on the NIPALS algorithm, the Partial least-squares regreesion model can be build. We can use the score matrix to represent the data matrix, then the general underlying model is consists of outer relations and an inner relation: - The outer relation for the X block:

$$X = TP' + E = \sum t_h p_h' + E$$

- The outer relation for the Y block:

$$Y = UQ' + F^* = \sum u_h q_h' + F^*$$

- The inner relation that links X and Y blocks:

$$\hat{u_h} = b_h t_h$$

If we do the decompositions of X and Y independently using NIPALS, we get these two sets of update rules:

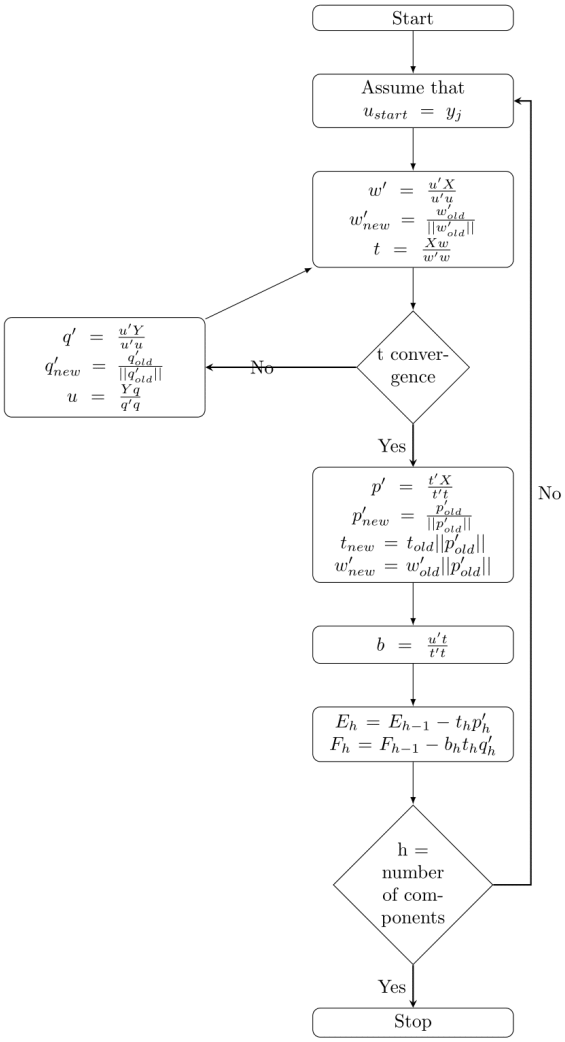
X block	Y block
$t = x_j$ for some j	$u = y_j$ for some j
Loop	Loop
$p' = \frac{t'X}{t't}$	$q' = \frac{u'Y}{u'u}$
$p'_{new} = \frac{p'_{old}}{\|\|p'_{old}\|\|}$	$q'_{new} = \frac{q'_{old}}{\|\|q'_{old}\|\|}$
$t = \frac{Xp}{p'p}$	$u = \frac{Yq}{q'q}$
Until t stop changing	Until u stop changing

In order to improve the inner relation, one intuitive way is to exchange the scores and then interleave the two sets of update rules inside the loop, resulting in the following procedure:[4]

- ***Note:*** $X = E_0$ and $Y = F_0$
- ***Note:*** If Y block has only one variable, q = 1 and no more iteration is necessary

```
In [1]: from IPython.display import Image
        Image(filename = 'test-1.png')
```

```
Out[1]:
```


Start
Assume that
$u_{start} = y_j$
$w' = \frac{u'X}{u'u}$
$w'_{new} = \frac{w'_{old}}{||w'_{old}||}$
$t = \frac{Xw}{w'w}$
t conver-
gence
No
$q' = \frac{u'Y}{u'u}$
$q'_{new} = \frac{q'_{old}}{||q'_{old}||}$
$u = \frac{Yq}{q'q}$
Yes
$p' = \frac{t'X}{t't}$
$p'_{new} = \frac{p'_{old}}{||p'_{old}||}$
$t_{new} = t_{old}||p'_{old}||$
$w'_{new} = w'_{old}||p'_{old}||$
No
$b = \frac{u't}{t't}$
$E_h = E_{h-1} - t_h p'_h$
$F_h = F_{h-1} - b_h t_h q'_h$
h =
number
of com-
ponents
Yes
Stop

3.5.2 Properties of the PLS Factor

For the loading matrice P and Q of X and Y blocks:

- $||p'_h|| = 1$ or $\sum p^2_{hj} = 1$ for j = 1 to m
- $||q'_h|| = 1$ or $\sum q^2_{hj} = 1$ for j = 1 to k

For the score matrice T and U of X and Y blocks:

- $\sum t_{hj} = 1$ for j = 1 to n
- $\sum u_{hj} = 1$ for j = 1 to n

For the weight matrix W and score matrix T of X block:

- $w'_i w_i = \delta_{ij}||w'_i||^2$ where δ_{ij} is the Kronecker delta
- $t'_j t_j = \delta_{ij}||t'_j||^2$ where δ_{ij} is the Kronecker delta

All of the properties should be checked in the PLS regression.

3.5.3 Prediction of the PLS Regression

From the algorithm of the Partial Least Square Regression, we obtain the vectors p', q', w' and b for every component.

- *Step 1:* $\hat{t_h} = E_{h-1} w_h$
- *Step 2:* $E_h = E_{h-1} - \hat{t_h} p'_h$
- *Step 3:* $Y = F_h = \sum b_h \hat{t_h} q'_h$

3.5.4 Determine the Number of Components

In the multiple linear regression model between X and Y, the number of components should be the model dimensionality of X variables. However, in the partial least square regression, we decompose the X and Y by using the PCA method, we can set up the number of components to predict the dependent variables, where the number of components should be less than or equal to the number of variables in X block. Although it is possible to calculate as many PLS components as the rank of X block matrix, some smaller components will only describe noise. Therefore, it is important to use appropriate value of number of components.

In this paper, we use the prediction residual sum of squares (PRESS) to determine the proper number of components. The number of components giving a minimum PRESS is the right number for the model that gives optimal prediction. When the PRESS is less than some thresholds, we can obtain the minimun number of components to predict the dependent values.

$$PRESS = \sum (Y - Y_{prediction})^2$$

4 Code Optimization

Using a simulated data sets where both independent and dependent variables follow standard normal distribution, we compare the computing time of PLS function with the time of python PLSRegression function under 10^{-6} convergence bound. Note that they use different algorithm, one is the Partial Least Square and the other is Nonlinear Iterative Partial Least Squares (NIPALS).

Because many matrix multiplications are used in the PLS function, it is important to compare the time of using different matrix multiplication methods in Python. After we use np.dot() function to do the matrix production and delete the unneccesary storation of matrices, the time of running function does not change too much. The results are shown in the table.

Function	Time
PLS Function	10000 loops, best of 3: 1.9 ms per loop
PLS_modified Function	10000 loops, best of 3: 1.86 ms per loop
NIPALS Function	10000 loops, best of 3: 2.46 ms per loop
NIPALS Prediction Function	10000 loops, best of 3: 37.1 μs per loop

We also checked the function calls and total time of each function call. There are 1304 function calls in 0.004 seconds. The results show that function f1 (calculate w and t) is called 39 times which takes 0.001 seconds and function pls_fit takes 0.003 seconds per call. Therefore, we will improve these parts in the future work.

5 Comparation between PLS and NIPALS

In this paper, we use the same simulated data set to test the prediction from both PLS function and python PLSRegression function. Let the dependent variables Y block be a matrix with 2 vairables and 10 observations. Let the dependent variables X block be a matrix with 3 vairables and 10 observations. All the observations are from standard normal distribution with mean 0 and standard deviation 1.

$$X_j \sim N(0,1) \quad j = 1...3$$
$$Y_j \sim N(0,1) \quad j = 1...2$$

Because we assume the X and Y block are mean-centered and scaled in the Partial Least Square Algorithm, we standardize the X and Y blocks and use the matrices in thoes two functions. First, we checked the properties of PLS factors in our function, and all the properties are satisfied under 10^{-6} convergency bound with 3 components. Second, we compared the prediction of X values and the prediction residual sum of squares in two functions. The results are shown below.

PLS Function	NIPALS Function
-0.05685202, -0.34046818	-0.05393456 -0.32299648
0.22479798, -0.15284406	0.21326209 -0.1450006
0.48826246, -0.06764669	0.46320644 -0.06417529

PLS Function	NIPALS Function
0.39923264, 0.63048359	0.37874534 0.59812925
-0.1501486 , 0.76762229	-0.14244347 0.72823045
-0.88229469, -1.72104717	-0.83701823 -1.6327287
-0.50815355, -0.22609268	-0.48207679 -0.21449035
-0.35954595, 0.23233151	-0.34109524 0.22040903
0.30865549, 0.18892197	0.29281631 0.17922712
0.53604624, 0.68873941	0.50853811 0.65339558

As we can see, the prediction results from two methods are similar. By comparing the prediction residual sum of squares, we find that the PLS method is better than the NIPALS method in python package when the number of components is 3,

$$PRESS_{PLS} = \sum (Y - Y_{prediction})^2 = 13.2696$$

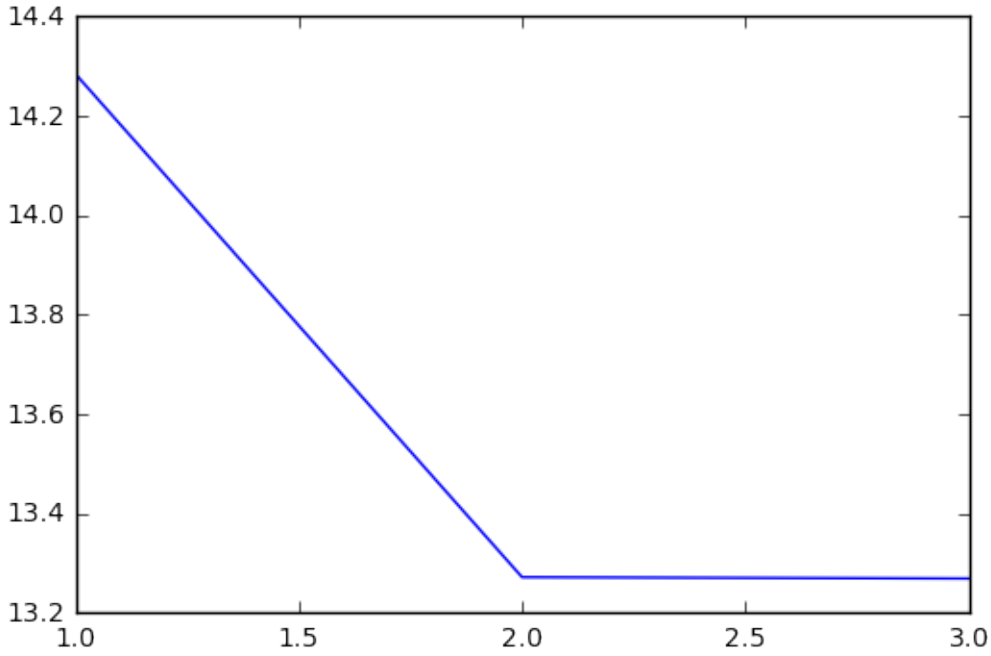
$$PRESS_{NIPALS} = \sum (Y - Y_{prediction})^2 = 13.2873$$

That is, the partial least square method will provide more precise prediction for dependent variables.

Then we can determine the number of components using the simulated data set, in this example, the PRESS against the number of components are {14.28474676, 13.27230472, 13.26958899}. Although the prediction residual sum of square is the smallest when there is 3 components, it is similar as the PRESS with 2 components. Therefore, the model with 2 or 3 components would be acceptable. Note that when the number of X variables increases, it is possible to obtain an appropriate number of components that is less than the rank of X block.

```
In [8]: from IPython.display import Image
        Image(filename = 'number of component_simu.png')
```

```
Out[8]:
```



6 Application to Simulated Data Set

In order to check the precision of the prediction for PLS function, we used a simulated data set with 100 observations, where the first 70 observation is considered as the training data set. We built the PLS Regression and NIPALS model with convergence bound 10×-6 and 3 components to obtain weight and loading matrix of X, Y blocks.

$$X_j \sim N(0,1) \quad j = 1...6$$
$$Y_j \sim N(0,1) \quad j = 1...2$$

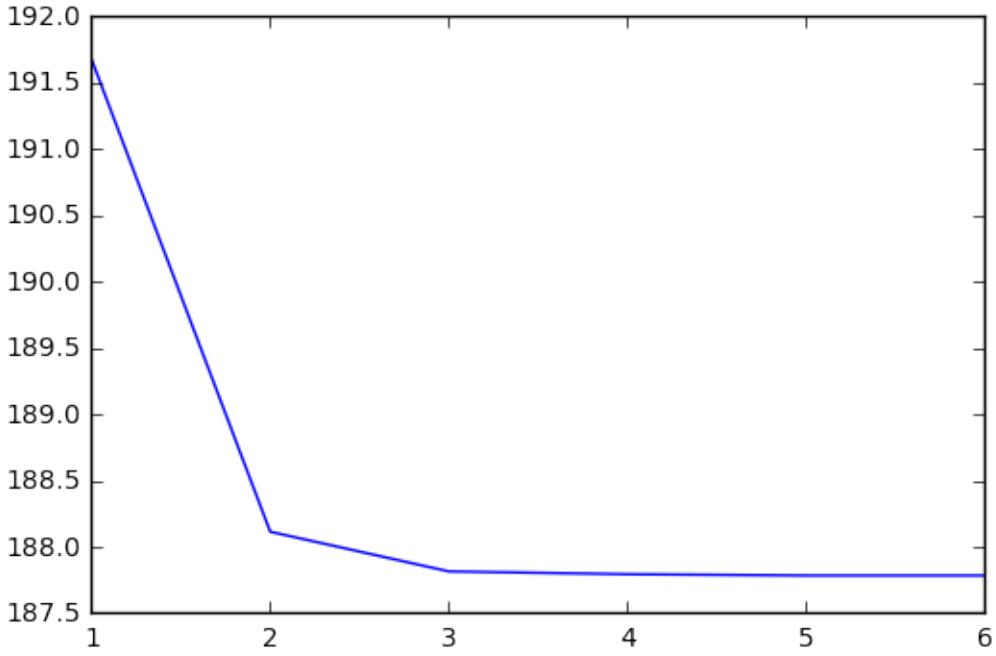
The difference between the predictions from two methods with 3 components under 10^{-6} convergence bound is less than 0.01, which indicates that the results are similar based on the fit model of 70 observations in training data set.By comparing the PRESS of the 30 observations in the test data set, we find that the PLS also provide better results than the NIPALS method in Python,

$$PRESS_{PLS} = \sum (Y - Y_{prediction})^2 = 70.9846$$
$$PRESS_{NIPALS} = \sum (Y - Y_{prediction})^2 = 73.7727$$

The PRESS sequence for all 6 components is {191.67935349, 188.11414837, 187.81467454, 187.79346654, 187.78380184, 187.78367464}. The differences become smaller when the number of components is greater than 3. The plot of PRESS against the number of components in this case shows that models with 3-6 components would be acceptable, which provides the relatively minimum PRESS and would give optimal prediction.

```
In [9]: from IPython.display import Image
        Image(filename = 'number of component_simu_pre.png')
```

Out[9]:



7 Application to Real Data Set

We also tested the algorithm on a real-world data. The dataset we used is wine data. We want to predict the subjective evaluation of a set of 5 wines. The variables in this dataset are defined as follows:

Dependent Variables:

- Likeability for each wine (5 wines in total)
- How well it goes with meat (rated by a panel of experts)
- How well it goes with dessert (rated by a panel of experts)

Independent Variables:

- Price of each wine
- Sugar of each wine
- Alcohol of each wine
- Acidity content of each wine

```
In [10]: import pandas as pd
         wine = pd.read_excel('wine.xlsx')
         Y=wine[wine.columns[0:3]]
         Y
```

```
Out[10]:    Hedonic  Goes with meat  Goes with dessert
          0       14               7                  8
          1       10               7                  6
          2        8               5                  5
          3        2               4                  7
          4        6               2                  4
```

```
In [13]: X=wine[wine.columns[3:7]]
         X
```

```
Out[13]:    Price  Sugar  Alcohol  Acidity
          0      7      7       13        7
          1      4      3       14        7
          2     10      5       12        5
          3     16      7       11        3
          4     13      3       10        3
```

We want to predict the subjective evaluation of a set of 5 wines. The dependent variables that we want to predict for each wine are its likeability, and how well it goes with meat, or dessert (as rated by a panel of experts) (Y block). The predictors are the price, the sugar, alcohol, and acidity content of each wine (X block). In this paper, we used the standardized observations to compare the prediction performances of two functions, PLS function versus NIPALS function, under the convergence bound 10^{-6} and 3 components.

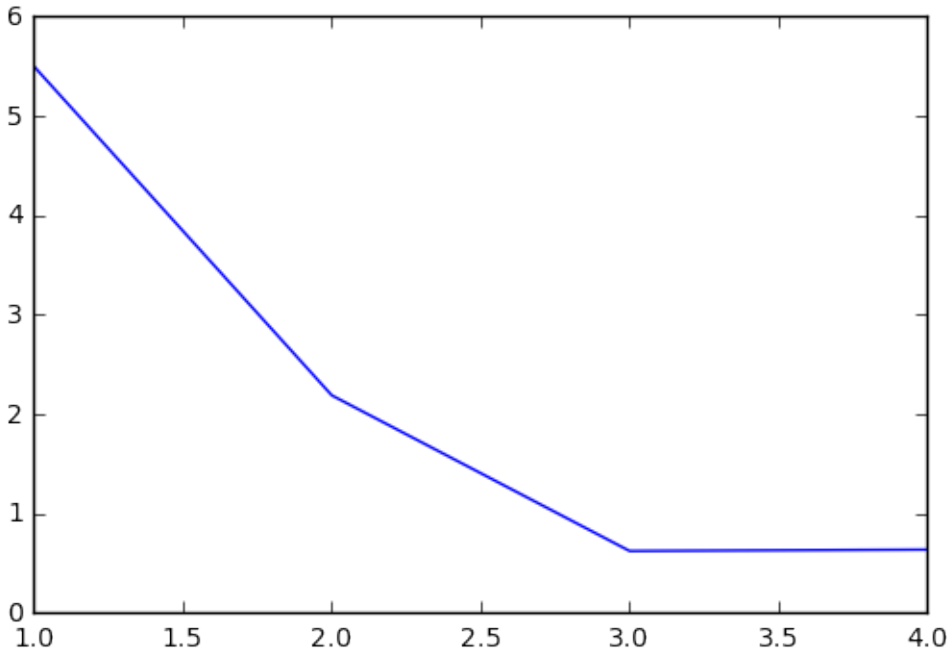
$$PRESS_{PLS} = \sum (Y - Y_{prediction})^2 = 0.6249$$

$$PRESS_{NIPALS} = \sum (Y - Y_{prediction})^2 = 0.7852$$

As we can see, the Partial Least Square method would provide a less PRESS with more precised prediction. The PRESS sequence for all 4 components is {5.50014623, 2.19057571, 0.625, 0.63863096} and the plot of PRESS against the number of components illustrates that the right number for the model that gives optimal prediction is 3, where the prediction residual sum of square for 4 components is slightly larger than the model with 3 components.

```
In [14]: from IPython.display import Image
         Image(filename = 'number of component_real.png')
```

```
Out[14]:
```



8 Discussion

Partial Least-Squares (PLS) is a widely used approach in many fields, it especially useful in the situation that the number of independent variables is larger than the number of data points. In this paper, we were able to implement the PLS method using Python based on the concepts introduced by Geladi and Kowalski (1985) in their paper 'Partial Least-Squares Regression: A Tutorial'.

We first compared the user defined PLS function with the build-in PLSRegression function in python under 10^{-6} convergence bound with a simulated dataset. We found that the running time didn't change a lot, but functions that calculating w and t as well as the pls_fit function can be improved. We then compared the PLS method with the NIPALS method using the same simulated dataset, and found that PLS would resulting in better prediction. After that, we checked the precision of the prediction for the PLS function using a sample size 100 (70% training and 30% testing) simulated dataset, and found that the model with 3 to 6 components would be acceptable since they provided a smaller PRESS which led to the optimal prediction. Finally, we applied our function on a real word dataset. The results shown that the PLS method would provide a less PRESS with more precised prediction.

For oprimization the code, we profiled the code but didn't get a much faster result. Several other methods may help the PLS implementation code gain significant improvement. For instance, do matrix multiplication in a less computationally-expensive way, remove unnecessary calculations and Cythonizing Python code. Therefore, more work can be done in the future to boost up the performance of the algorithm.

Many different adaptions, applications and algorithm have been left for the future due to lack of time. Besides continuous outcomes that we have looked at in this paper, categorical outcomes (especially two class problem) is also one of the most common issue we would encounter in real

life. Therefore, one extension of this PLS method is on the clustering problems(e.g., two equal class sizes, unequal class sizes and more than two classes). Also, in this paper, we assumed that X (independent variables) and Y (dependent variables) are all standardized. In the future, we could apply the method on the original scaled dataset to see the performance of the algorithm. In sum, futur works would concern deeper analysis of particular mechanisms, new proposals to try different methods, or simply curiosity.

9 Github Repsository

Please use the code in the following linkage to reproduce the results in the paper.
https://github.com/yaopoppysong/Partial-Least-Square-Regression

10 Reference

- [1] Saikat Maitra and Jun Yan. Principle Component Analysis and Partial Least Squares: Two Dimension Reduction Techniques for Regression, Casualty Actuarial Society, Discussion Paper Program, 2008.
- [2] Paul Geladi and Bruce R. Kowalski. Partial least squares regression: A tutorial. Analytica Chimica Acta, 185:1–17, 1986.
- [3] Herv´e Abdi1. Partial Least Squares (PLS) Regression. The University of Texas at Dallas, 2007.
- [4] Kee Siong Ng. A Simple Explanation of Partial Least Squares. 2013
- [5] Tenenhaus, M., Pagès, J., Ambroisine L. and & Guinot, C. PLS methodology for studying relationships between hedonic judgements and product characteristics; Food Quality an Preference. 16, 4, pp 315-325, 2005