Evaluation of Dynamical-Inner Partial Least Squared Projections to Latent Structures Method

Project Learning Report

Authored by Vicky R. Zhu

Advised by Hao Wei and Shyam Panjwani

Introduction

Real word data has rich information. In genetic study, there may be over hundreds of features associated with a single cell. In chemical industry, the property descriptions of a material such as the resilience, density, tear strength, etc. of a polymer form can also have lots of component descriptors involved. For any given observed sample, effectively extract the information while maintain high accuracy has many benefits since not all features give the same significant contributions for the prediction. Filtering out the unnecessary variables also can bring some financial benefits such as the cooperation can avoid using those unnecessary materials in the product line or perhaps eliminate a particular measurement during data collection step. In multivariate data analysis, information can be also understood in a comprehensive way such as the data itself has its internal structure rearrangement and provide a different representation of the variable. The projection approach in finding quantitative relationship among the variables has been widely used in both multi- and mega-variate data analysis regardless of sample observations exceed the variables or vice versa. In addition, projection method can handle data collinearity very well by the nature of reconstruction.

Transformation and data representation

Although data are not synonymous with information, proper analytical tools are needed to process the information and transfer into useful data. As features are measured in different scales, one example of data-preprocessing is to do zero mean and unit variance transformation (**Figure 1**). Putting all variables into the same standard can provide a fair feature selection to the variables.

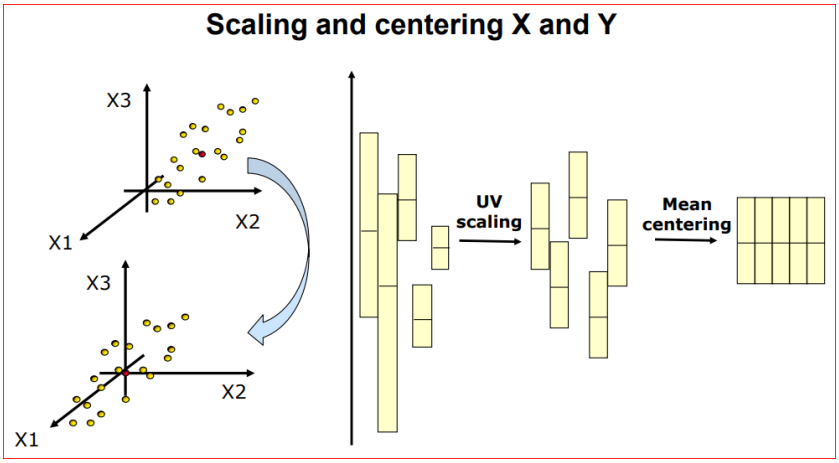


Figure 1: Transferred data input and output:

For supervised regression task where the inputs X and outputs Y matrices are given, the partial least square (PLS) projections to the latent variable modeling can accomplish our prediction goal. One may view PLS as the regression extension of principal component analysis (PCA). However, unlike PCA, each point refers to two spaces, one in X input space and also corresponds to its Y output space (**Figure 2**).

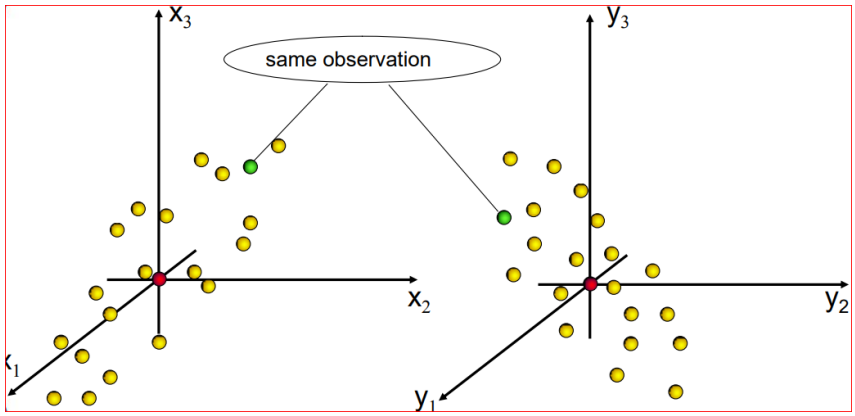


Figure 2: Transferred data input and output. Red is the center (not from the observations) and sample observations are refereed as yellow points, and each has two-space representation (green).

PLS construction

The construction of latent variables in projection method starts from a sequential order. The first PLS component t1 and u1 are extracted in the X space and Y space such that the covariance between t1 and u1 is maximize (**Figure 3**). The second PLS components, t2 and u2 can be obtained in a similar way to maximize the covariance of t2 and u2, except t2 is orthogonal to t1, but this does not necessary apply for u2 and u1. We can extract the components the number of components up to the numbers of variables in X and Y spaces. All the PLS components together provided a good approximation of both X and Y space.

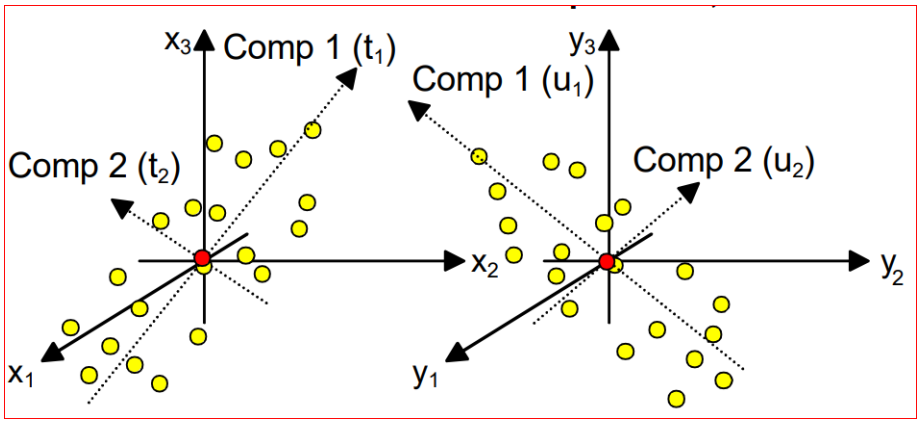


Figure 3: Construction of PLS components. Dash lines (t1, t2, u1, and u2) are the components extracted from the original X and Y space such that the covariance between them is maximized.

The process of extraction the latent components can be viewed as outer modeling, and inner model is to understand the relationship between latent variables after each time obtaining the corresponding u and t components (**Figure 4**). For example, PLS showed a linear static relationship between the first component.

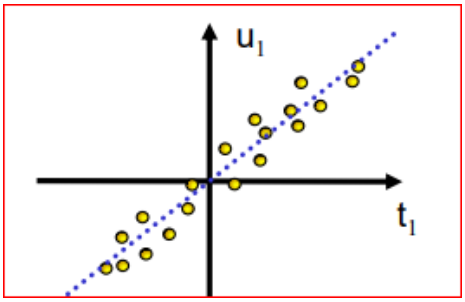


Figure 4: **linear static relationship between latent variables**. u1 can be expressed as a simple linear regression with t1.

Dynamical PLS

In pharmaceutical industry, data are often collected with time components. The traditional PLS approach cannot handle this situation, my summer project mainly focusses on dynamical PLS that has time series inputs and outputs. Several methods are found to tackle this problem. They are in some degree the extension of the PLS with additional tricks to handle the dynamical parts. Qin and McAvoy proposed an integration of neural networks approach to handle a nonlinear process with input collinearity data (Qin and McAvoy, 1995). This is a straightforward method such that the inner modeling can capture nonlinear relationship between latent variables. The downside is that this approach does not give explicit representation of the dynamic relationship, so it is difficult to interpret (**Figure 5**).

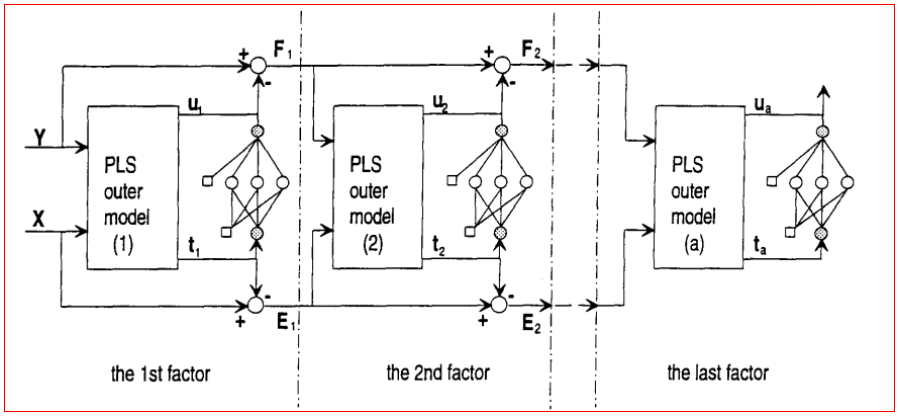


Figure 5: Nonlinear finite impulse response modeling via neural net PLS (Qin and McAvoy, 1995). The outer modeling is the same as the traditional PLS and the inner modeling use neural network approach to handle the nonlinearity.

While Qin and McAvoy's NNPLS is powerful to handle nonlinearity, it might not directly relate to the dynamical components of the process and over killed the task. Kaspar and Ray took a transformation approach by first choose a dynamic filtering on X time series input, this step is to wash away the

dynamical part of the input data and left the remaining part like the regular PLS input, that is to keep the static nature of the data. After transformation step achieved, then apply an algebraic relationship with Y output. This method has compact representation since no lagged variables appear in the outer modeling so long as the dynamical filter applied. One way to pick such filter is to consider taking "average" on X, a diagram is illustrated below (**Figure 6**). The disadvantage of this method is that we need prior knowledge to design the filter, so the dynamic components in the inputs can be removed. In addition, there is an inconsistent issue between the outer model and the inner modeling as the inner model has dynamics, but not the outer model.

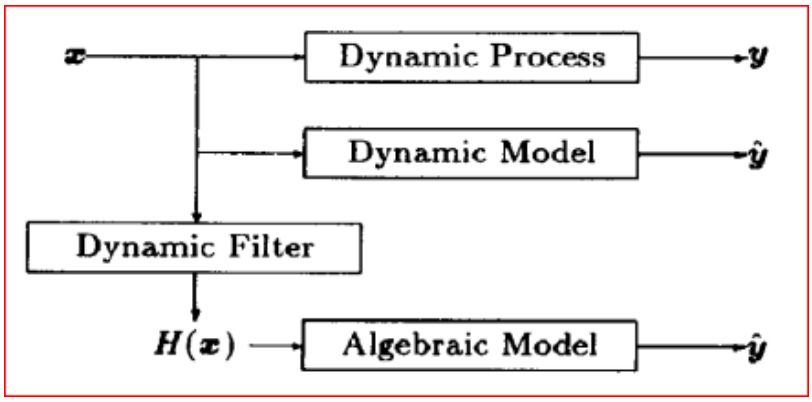


Figure 6: Dynamic PLS for process control. One can take the average of the input as a dynamical filter to get rid of lag variables.

Another approach is to use Hammerstein structure (**Figure 8**) and decompose the multi-input, multi-output by employing many single-input and single output (**Figure 9**).

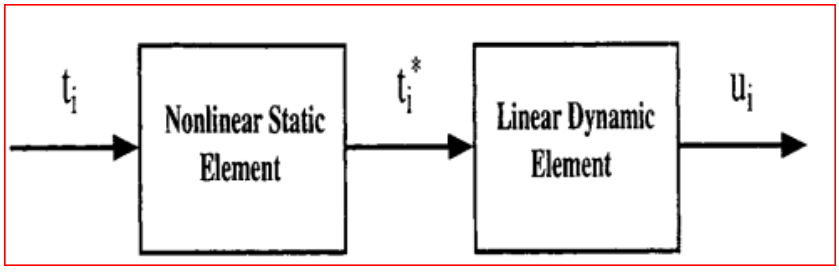


Figure9: Hammerstein structure. Two stages are needed, the nonlinear static element can be obtained by a polynomial with some degree greater than 2. The following step involved of using a linear dynamic method to handle the inner product part such as using ARX modeling.

This approach has a direct modification of inner relationship. It breaks into many univariate Hammerstein models to obtain an overall model. However, since all the information is used in model building step, so it does not reduce the dimensionality of the redundancy. Furthermore, Hammerstein structure cannot model every type of nonlinearity.

$$Y = H_1(t_1)q_1' + H_2(t_2)q_2' + \cdots + H_n(t_n)q_n' + F$$
$$= Y_1^{\exp} + Y_2^{\exp} + \cdots + Y_n^{\exp} + F.$$

Figure 9: the constructure of dynamical PLS (Lakshminaryaana, etc, 1997). Multi-input and multi-output data can be rewritten as multiple single-input and single-output form.

These methods are all straightforward and direct modification of traditional PLS method for regression problems. Each focus on a specific task such as the NNPLS method is used to handle nonlinear data input, and the transformation of dynamical filter method can be applied for the process control. Like NNPLS, the Hammerstein structure approach can handle certain nonlinear data with multi-input and multi-output type of problems. In addition, the transformation and Hammerstein modeling can bring some dynamical inconsistent issue. To overcome that, Doing & Qin came up Dynamical-Inner PLS algorithm (DiPLS). This projection method not only gives an explicit description between inner and outer model, but also achieves the consistency challenge. By specifying a time k at each time modeling, one can find the corresponding latent variables at that time k. The algorithm aims to extract a dynamic inner relation through auto regression analysis between latent variable u and its counterpart t together with a history of t up to s step, where s is the time lag. The outer modeling looks like equation 1.

$$u_k = \mathbf{y}_k^T \mathbf{q}$$
$$t_k = \mathbf{x}_k^T \mathbf{w} \quad (1)$$

Whereas the inner modeling part is

$$\mathbf{u}_s = \alpha_0 \mathbf{t}_s + \alpha_1 \mathbf{t}_{s-1} + \cdots + \alpha_s \mathbf{t_0} + \mathbf{r}_s \quad (2)$$

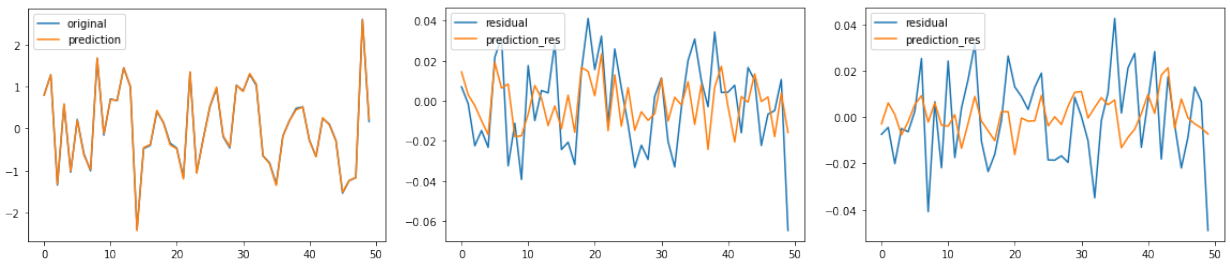
Now putting both equation (1) and (2) together, we can connect inner and outer modeling together and have the explicit expression take directly take the dynamical description of the X inputs in equation (3).

$$\begin{aligned} \hat{u}_k &= \mathbf{x}_k^T \mathbf{w} \beta_0 + \mathbf{x}_{k-1}^T \mathbf{w} \beta_1 + \cdots + \mathbf{x}_{k-s}^T \mathbf{w} \beta_s \\ &= [\mathbf{x}_k^T \quad \mathbf{x}_{k-1}^T \cdots \mathbf{x}_{k-s}^T](\boldsymbol{\beta} \otimes \mathbf{w}) \end{aligned} \quad (3)$$

Results

To demonstrate the idea of DiPLS method, we follow the footprint from Dong and Qin's paper, and generate some case studies from 1000 synthetic data, and break them into 500 training, 400 validation, and 100 testing samples. unlike the real data, synthetic data gives a better control as there is no outlier and the underline parameters are known. For case 1 study, both X inputs and Y output are static process, meaning no lag here, so DiPLS should recover PLS.

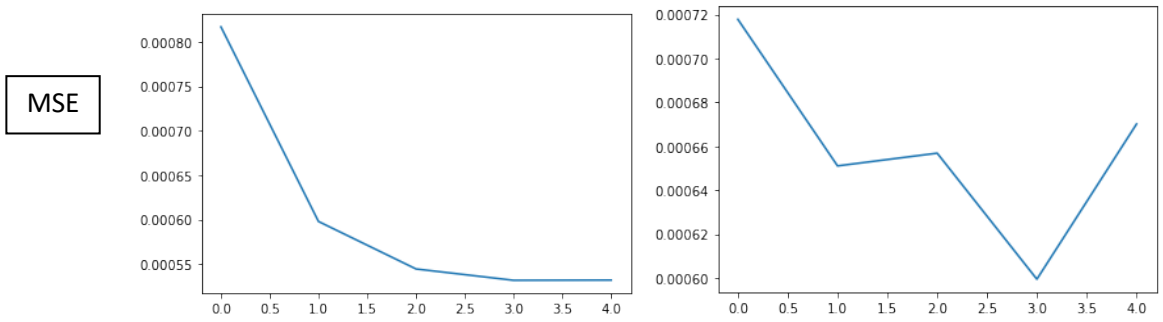
We first trained 500 samples and showed an evolution of each time the for a specific component, the prediction of Y value compares with the remaining unexplained residuals (**Figure 10**). Notice that as more components are extracted, the remaining part become just white noise, so the prediction does not perfectly align with noise.



(a) (b) (c)

Figure 10: Evolution of the DiPLS training process. (a) plot on the left is the prediction after first latent component is extracted, the prediction Y explained by the first component (in orange) comparison with original Y (skyblue). (b) plot in the middle is the prediction after second PLS component to explain what is left from the oritinal Y that has not yet been explained by the first PLS comonent. (c) plot on the right the prediction Y compared with remaining white noise after thrid component extraction.

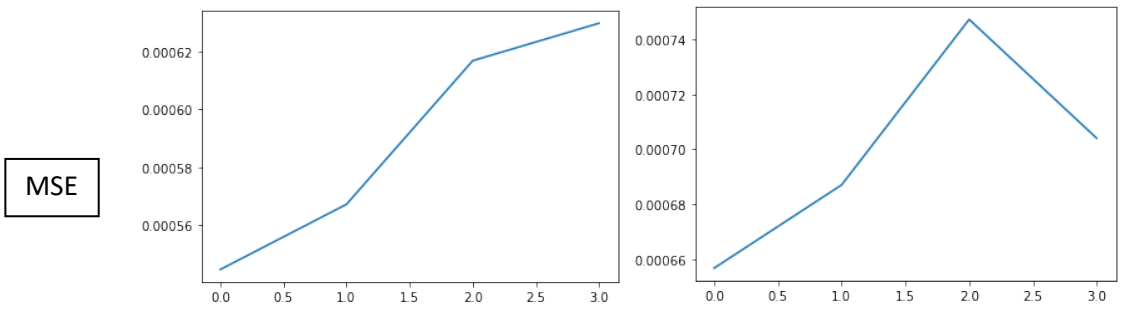
To determine how many latent variables lags are needed? We use our validation set to calculate mean square errors (MSE). Notice that adding more latent variables, meaning the more we we could explaination the original data, hence MSE will always decreass. We compared the smallest five errors and stop when we see some relative flat surface. This results also verified with 5 fold cross-validation approach (**Figure 11**).



(a) (b)

Figure 11: mean square error for adding additional components when lag is fixed at 0. (a) plot is to directly use validation set. (b) plot uses cross-validation method.

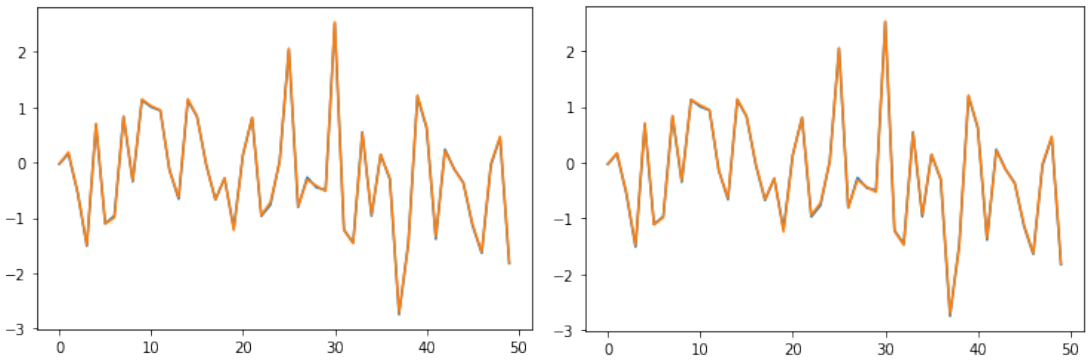
We can also use our validation set to calculate MSE and determine the number of lags are needed (**Figure 12**). This could also verify from cross validation methods.



(a) (b)

Figure 12: MES for adding additional lag when number of components is fixed at 3. (a) plot is to directly use validation set. (b) plot uses cross-validation method.

After finding the suitable parameter, we also checking the performance with some unseen data. We use the already developed latent variables t and u, and together with some weights from training part. To verify our results, we further compared already well established PLS regression from sklearn package (**Figure 13**). Note that PLS regression only need one parameter input, namely the number of component, whereas DiPLS takes additional parameter by specifying the lag number. In our case, number of the latent components is 3 and vlag number is 0. DiPLS has a similar high accuracy with PLS up to 5 decimal digit (DiPLS has R2 = 0.99948 versus PLS has R2 = 0.99941). In compare with errors, DiPLS also obtain similar low MSE as PLSup to the 4th decimal place (DiPLS has MSE = 0.00046 versus MSE = 0.00053).



(a) (b)

Figure 13: Predictions on testing sets. (a) plot is generated from DiPLS(n_component = 3, lag = 0). (b) Plot is the standard PLSRegression(n_component = 3) from SKlearn package.

If we increased our lag from 0 to 1, we can see the insignificant coefficents appear both from outer and inner modeling. Note that the second row of both tables are much smaller than the first row.

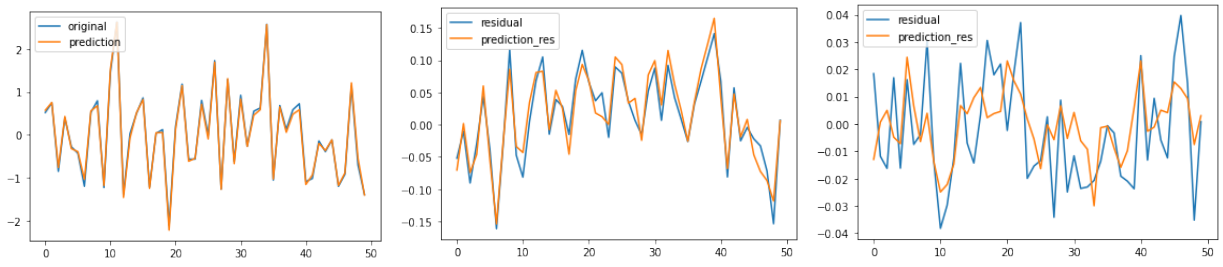
Table 1:

Outer-modeling coefficients	Feature 1	Feature 2	Feature 3
beta0	0.99948	0.96563	0.99889
beta1	-0.00319	0.25990	-0.04704

Table 2:

Inner modeling coefficients	Feature 1	Feature 2	Feature 3
alpha0	0.4927	0.0063	0.1418
alpha1	0.4362	0.0002	-0.0046

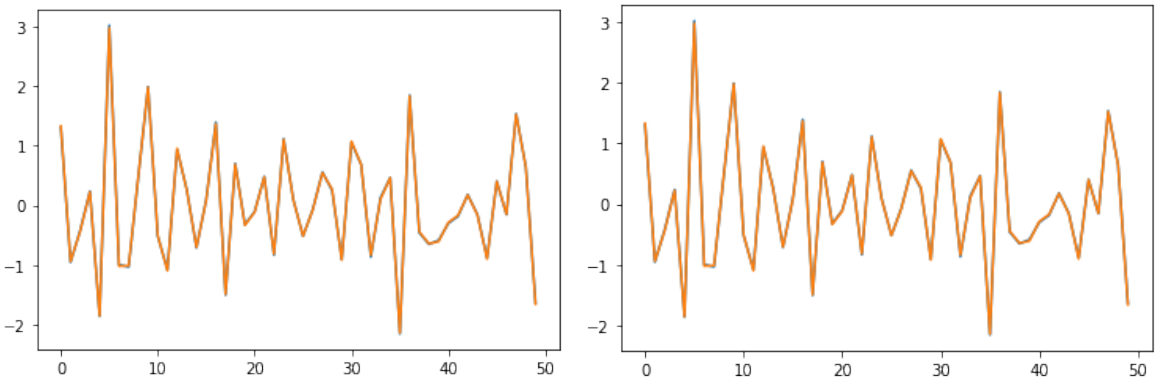
We see that the DiPLS completely recover PLS when having static input X and output Y, and this result is confirmed with PLS regression from sklearn library. Next, we evaluate a second case such that the input X is generated from dynamical process that depends on t_k-1 and t_k-2, whereas the output Y is still static. Notice that, the lag parameter is still 0 as the covariance between input X and Y is 0. The dynamical part appears only in describing the covariance between X itself is not zero. Like case 1 study, we presented a training process in **Figure 14**.



(a) (b) (c)

Figure 14: Same as in Figure 10, except the X input is generated from dynamical process.

We follow the same process to check the testing performance and showing in **Figure 15**. We obtain a better accuracy R2 (0.999776 versus 0.999772) and lower MSE (0.000241 versus 0.000243) up to the 6^{th} decimal place.



(a) (b)

Figure 15: Same as in Figure 13, except the X input is generated from dynamical process.

If checking the coefficients of the outer and inner modeling, we see a similar results as in case 1 such that the first-row coefficients are much larger than the second row, so setting lag = 1 is insufficient. See the tables below.

Table 3

Outer modeling coefficient	Feature 1	Feature 2	Feature 3

beta0	0.97219	0.94773	0.91380
beta1	-0.23415	0.31938	0.40615

Table 4

Inner modeling coefficient	**Feature 1**	**Feature 2**	**Feature 3**
alpha0	**0.54828**	**0.0039**	**0.0131**
alpha1	**-0.00347**	**0.0009**	**0.0020**

Moving to the third case study, now we keep X input static and change Y output as dynamical process. Specifically at each time k, y_k is generated from the current, x_k and previous one, x_k-1. The lag parameter is now become 1. The performance on testing set in **Figure 16** showed us PLS algorithm is useless in this case. However, the DiPLS can really show its power. We have R2 = 0.9998, meaning DiPLS can explained the original Y values very well. MSE is 0.00015 in compared with 1.199 MSE from PLS regression, this suggested that the DiPLS algorithm has achieved its goal of handling dynamical part.

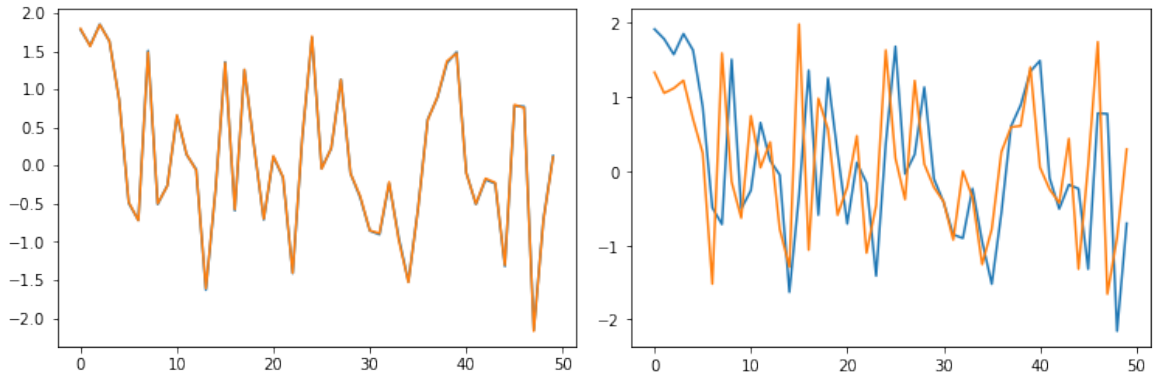


Figure 16: Same as in Figure 13, except the Y output is generated from dynamical process.

Now if we check the coefficient of outer and inner modeling, we notice some significant appeared in the second row of the tables below. Meaning that it is sufficient to include the dynamical parts in the prediction modeling.

Table 5

Outer coeff	Feature 1	Feature 2	Feature 3	Feature 4
beta0	0.4681	0.0143	-0.0825	-0.8276
beta1	0.8836	-0.9998	-0.9965	-0.5613

Table 6:

Inner- coeff	Feature 1	Feature 2	Feature 3	Feature 4
alpha0	0.2272	0.0011	-0.0340	-0.3457
alpha1	0.4399	-0.0913	-0.4185	-0.2343

Although Dong and Qi stopped right here, for curiosity we also investigate on additional 4th case study such that generating both X input and Y output as dynamical process. Specifically, generate X as an auto regression from t_k-1, t_k-2 like our case 2 study and generate Y from x_k and x_k-1 like our case 3 study (**Figure 17**).

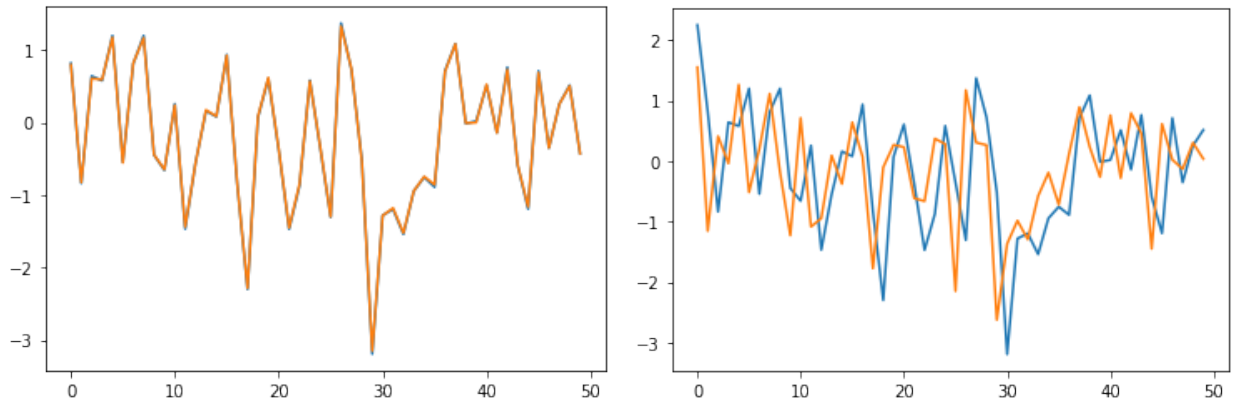


Figure 17: Same as in Figure 13, except both X inputs and Y output are generated from dynamical process.

As expected, PLS failed to predict the dynamical process, whereas the DiPLS can capture 99.96% of the trend. The MSE can be as low as 0.00038 in comparison with a large MSE that PLS produced. Furthermore, table below also showed a similar result as in case 3 that the some of the second row coefficients from both outer and inner modeling are significant. We concluded that the lag parameter in DiPLS depends on the intrinsic output data since this is determined by the nonzero covariance between X inputs and Y outputs, and regardless how we generate X inputs, it could be a static or dynamical process.

Table 7

Outer modeling coefficient	Feature 1	Feature 2	Feature 3	Feature 4
beta0	0.3654	0.7393	0.4862	-0.6493
beta1	0.9308	-0.6733	0.8738	-0.7604

Table 8

Inner- coeff	Feature 1	Feature 2	Feature 3	Feature 4
alpha0	0.2797	0.0434	0.0285	-0.2198
alpha1	0.5363	-0.0418	0.0967	-0.2651

Discussion & New Findings

This summer project, we started with multi-and mega-variable analysis to distinguish a basic concept of PCA and PLS. We further identify the popular use of PLS for prediction tasks in chemical and pharmaceutical realm and realize that data often come with dynamical components. We search methods such as developing an integration of neural network, applying transformation filter, and embedding multiple Hammerstein structures to handle the dynamical and possible nonlinear parts. Those methods are the extension of traditional PLS, some are computationally expensive and overkilled the problems, and some required prior knowledge of dynamical filtering. While some of these methods also have inconsistent problems between inner and outer modeling, Dong and Qin proposed a new DiPLS algorithm provides explicit description at each time step, hence resolved the issue of inconsistency. Furthermore, in comparison with already established and efficient algorithms in sklearn package, DiPLS can achieve the efficiency and accuracy up to 5 decimal places above and produce a lower MSE in general. DiPLS takes number of components and number of lags parameters as two arguments, when the lag parameter is zero, it can also completely recover the traditional PLS method.

For parameter tunning, we developed a method uses validation set directly and together with 5-fold cross-validation to confirm the best number of components and number of lags parameter pair. We extended 3 cases presented in Dong and Qin's paper by consider additional case study such that both X inputs and Y outputs are dynamical process. We found that that the lag parameter in DiPLS algorithm is only related to the nature of Y outputs regardless of if the X inputs are static or dynamical processes.

The lag parameter is determined from the nonzero covariance of inputs and outputs instead of the covariance of inputs themselves.

Future direction

To check the robustness, we can apply DiPLS algorithm to some real data and see if we can get some consistent result as the synthetic data analysis. Within PLS scheme, algorithms contain neural network structure and transformation tricks usually handle nonlinear data structure could also worth a try to handle dynamical process type. Other methods such as batch statistical process such as batch evolution modeling and batch level modeling also can be considered for data contains time components.

References

1. Dong Y, Qin SJ. Dynamic-Inner Partial Least Squares for Dynamic Data Modeling. *IFAC-PapersOnLine*. 2015;48(8):117-122. doi:10.1016/j.ifacol.2015.08.167

2. KASPAR MH, RAY WH. Dynamic PLS modelling for process control. *Chemical engineering science*. 1993;48(20):3447-3461. Accessed September 3, 2022. https://search.ebscohost.com/login.aspx?direct=true&db=edscal&AN=edscal.4873926&site=eds-live

3. LAKSHMINARAYANAN S, SHAH SL, NANDAKUMAR K. Modeling and control of multivariable processes : Dynamic PLS approach. *AIChE journal*. 1997;43(9):2307-2322. Accessed September 3, 2022. https://search.ebscohost.com/login.aspx?direct=true&db=edscal&AN=edscal.2815058&site=eds-live

4.QIN SJ, MCAVOY TJ. Nonlinear FIR modeling via a neural net PLS approach. *Computers & chemical engineering*. 1996;20(2):147-159. Accessed September 3, 2022. https://search.ebscohost.com/login.aspx?direct=true&db=edscal&AN=edscal.2948707&site=eds-live