Kernel PLS

Erik Kuitunen

October 12, 2024

Introduction

Partial Least Squares (PLS) is an useful tool for regression, but has limited capacity when there exist non-linearities in the data. Kernel PLS (k-PLS) is a modification to the original PLS method, which can better take into account these non-linearities [1].

In this task, liquid ultrasonic flow meter diagnostics data is studied. The data consists of 87 observations of 36 features, with two classes: ”healthy” and ”installation effects”, labeled as ”1” and ”2”. A k-PLS model is trained to the data, and a test set is then used to measure the model's ability to predict these classes.

Data partitioning and pre-treatment

The data is partitioned into training and test sets using train-test split functionality found in Python package sklearn, with the size of test set being 20% of the data set. As a pre-treatment, the data is scaled using StandardScaler, also found in sklearn. After scaling, it can be seen from Figure 1 that most of the data is quite free from possible outliers, except for features 11-19, which contain multiple possible outliers. In this task, these are however left untreated.

No cross-validation is performed due the small size of observation in the dataset.

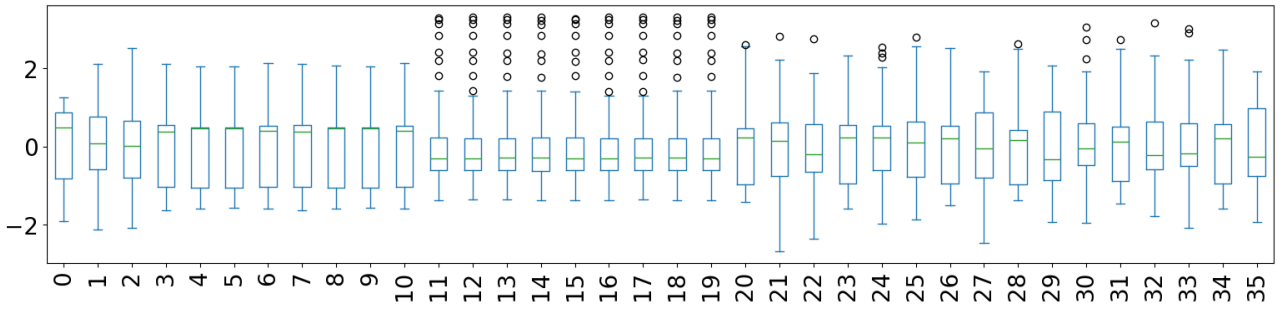

Figure 1: Boxplot of the scaled features.

k-PLS method

In k-PLS, the input data is projected into Reproducing Kernel Hilbert Space (RKHS) using a kernel function, where the actual PLS is then performed.[1] In this task, SIMPLS algorithm is used with three different kernel functions $k(\cdot, \cdot)$. The kernels are then tested and compared against standard PLS. The kernels used are linear, Gaussian and polynomial kernels, which are described in equations 1, 2 and :

$$k_{\text{linear}}(\boldsymbol{x}, \boldsymbol{x'}) = \boldsymbol{x}\boldsymbol{x'}^{\top} \tag{1}$$

$$k_{\text{G}}(\boldsymbol{x}, \boldsymbol{x'}) = \exp(-\frac{||\boldsymbol{x} - \boldsymbol{x'}||^2}{2\sigma^2}) \tag{2}$$

$$k_{\text{poly}}(\boldsymbol{x}, \boldsymbol{x'}) = (\boldsymbol{x'}^{\top}\boldsymbol{x} + 1)^d$$

A kernel matrix K is then calculated such that each element $K_{i,j} = k(\boldsymbol{x_i}, \boldsymbol{x_j})$, where $\boldsymbol{x_i}$ and $\boldsymbol{x_j}$ denote ith and jth observation of the original data matrix X. Before PLS, the kernel matrix needs to be centered:

$$\widetilde{K} = (I - \frac{1}{n}1_n 1_n^{\top})K(I - \frac{1}{n}1_n 1_n^{\top}),$$

where n is the number of samples and 1_n is a matrix of ones with size $n \times n$. As output from the NIPALS algorithm, score matrices T and U of the input data X and Y are obtained. Finally, using the score matrices and centered kernel matrix, the weights B of the features and predictions for data y_{pred} can be calculated:

$$B = U(T^{\top}\widetilde{K}U)^{-1}T^{\top}Y \quad \text{and} \tag{3}$$

$$y_{\text{pred}} = \widetilde{K}B. \tag{4}$$

The complete algorithm is presented in more detail in [1].

Results

The performance of k-PLS with different kernels is evaluated using R^2, Q^2 and MSE metrics. In Figure 2, the metrics of standard PLS are represented. It can be seen that even standard PLS is able to achieve good accuracy, provided enough latent variables are used in the model. With 23 LVs, Q^2 is a bit over 0.9 and R^2 is only a small amount below Q^2. Using more LV's would provide smaller MSE, but then the risk of overfitting would increase as Q^2 and R^2 quickly approach unity. Another choice for number of LV's could be 10, since there appears to be a local maximum for the Q^2 score, while MSE has a local minimum there which is not that far away from the MSE score of 23 LVs. Interestingly, experiment with linear kernel provides identical results with standard PLS as can be seen from Figure 3.

Gaussian kernel results are shown in Figure 4. Gaussian kernel achieves high R^2 and Q^2 with very low amount of LVs; only three are needed for good scores and lowest MSE. The seemingly good result is shadowed by the fact that MSE stays somewhat higher than with linear kernel, even when it is at its minimum. However, the model complexity would not be an issue here, and there is definitely no risk of overfitting.

Polynomial kernel of degree $d = 2$ was used in the last test. As can be seen in Figure 5, the scores and MSE blow up when number LVs go higher than 5. Investigation of the actual

numbers reveals, that the results actually are quite poor: Q^2 has maximum value of 0.68 at 5 LV's, and at that point R^2 is only 0.31. Minimum MSE occurs at only one LV, indicating that this model completely unusable.

Model accuracies can be seen in Table 1.

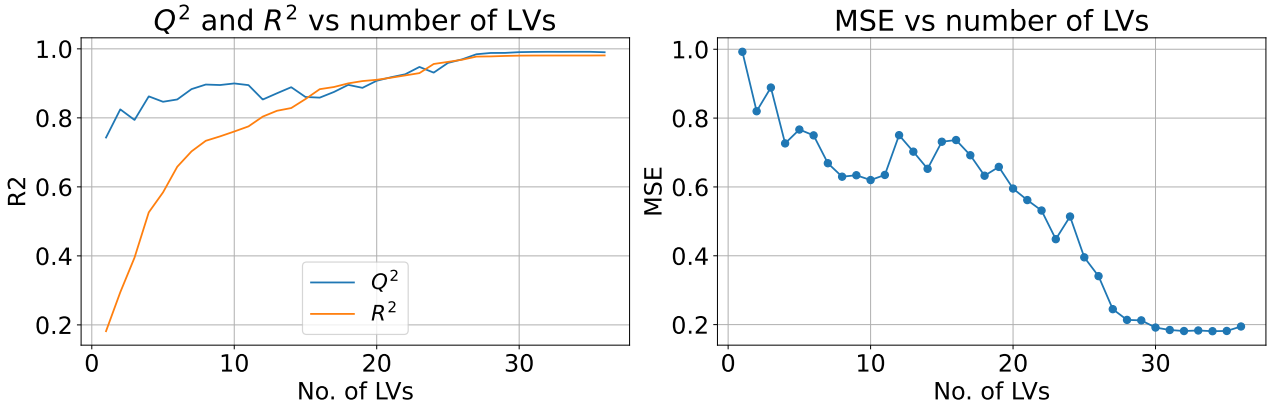

Figure 2: Q^2, R^2 and MSE as function of latent variables using standard PLS

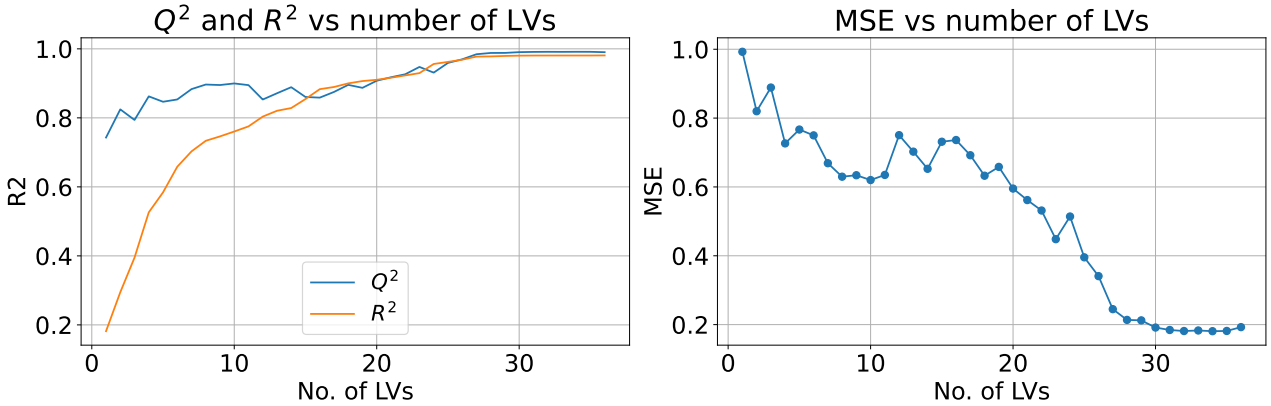

Figure 3: Q^2, R^2 and MSE as function of latent variables using linear kernel

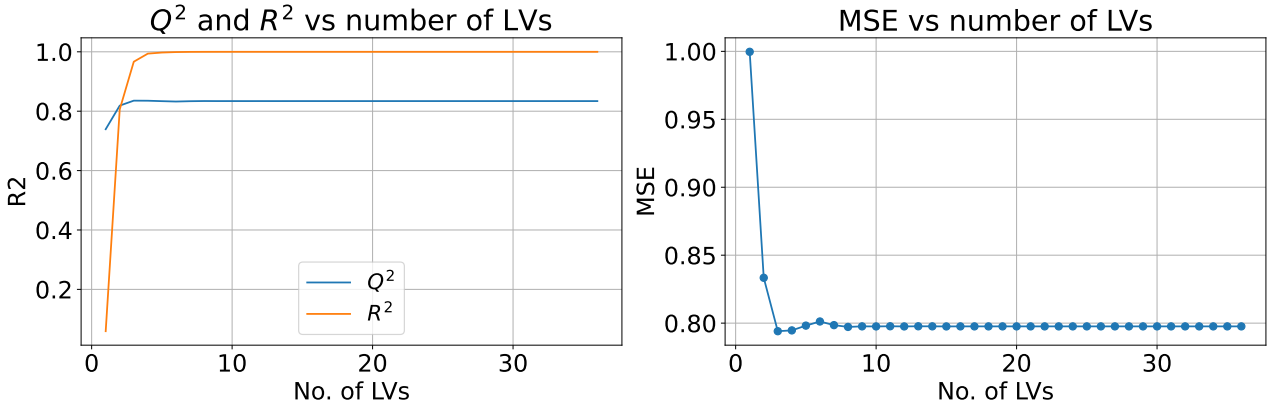

Figure 4: Q^2, R^2 and MSE as function of latent variables using Gaussian kernel

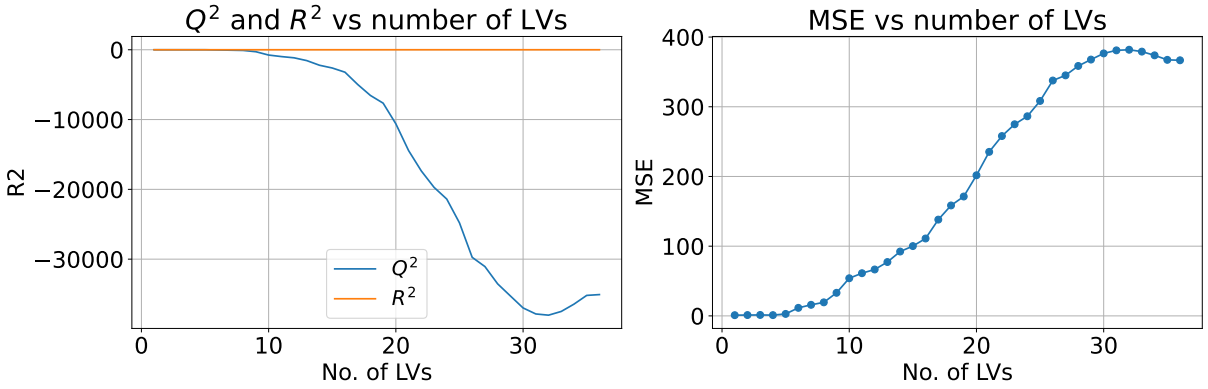

Figure 5: Q^2, R^2 and MSE as function of latent variables using polynomial kernel

Table 1: Accuracies of different models

	PLS	Linear	Gaussian	Polynomial
No. of LVs	23	10	3	5
Accuracy	0.94	0.94	0.67	0.61

Conclusion

The best results were achieved using linear kernel, although it seems that standard PLS would be as good choice as linear kernel PLS as well. Even though it is not shown in Table 1, standard PLS also achieved accuracy of 0.94 using only 10 LVs.

Originally, I tried to complete this task using dataset "Meter B", to which Gaussian kernel seemed to perform significantly better than any of the other three models discussed here.