"You really have to do the ground work to have these innovations."

Biao Huang

NSERC Industrial Research Chair video (2012)

University of Alberta

DATA-DRIVEN METHODS FOR NEAR INFRARED SPECTROSCOPY MODELING

by

Mulang Chen

A thesis submitted to the Faculty of Graduate Studies and Research
in partial fulfillment of the requirements for the degree of

Master of Science

in

Process Control

Department of Chemical and Materials Engineering

Fall 2013

Edmonton, Alberta

To my dearest family and friends.

Abstract

Time consuming offline laboratory analysis and high cost hardware measurement techniques render difficulties in obtaining the important quality variables in real time application. Near-infrared (NIR) spectroscopy is widely used as a process analytical tool (PAT) in chemical processes, providing online estimation of the target properties which are often obtained by lab analysis. This thesis focuses on the model building, model structure (wavelength) selection and online model update for NIR applications.

Time varying issue is solved by applying recursive adaptation methods and a novel recursive wavelength selection algorithm is proposed to adapt the model structure during online phase. The Just-in-time (JIT) modeling approach is adopted to model the nonlinear relationships between spectra and properties. A similarity criterion that utilizes input-output information is developed to search for most relevant samples from the database. Finally, the recursive algorithm and locally weighted algorithm are synthesized into the JIT framework in order to deal with both time varying and non-linearity issues of the process.

Acknowledgements

I am so fortunate to work as a member of Computer Process Control Group at the University of Alberta, where I gained numerous help during my efforts toward my MSc degree. Specially, I would like to thank my supervisor Dr. Biao Huang who has guided me through his rigorous research attitude and insightful advice. Without his constant support and guidance, this thesis is impossible to be accomplished. I am grateful to his encouragement and valuable directions during my research work in the past two years.

The Computer Process Control group provided a great working environment for discussing and sharing ideas. I would like to thank Swanand Khare for his instructions and discussions about the mathematical problems I faced during my research. I would also like to thank Shima Khatibisepehr for her help and advice whenever I met obstacle during my MSc study. The help from Zhankun Xi, Tianbo Liu , Xiongtan Yang, Lei Sun, Xing Jin, Da Zheng and Marziyeh Keshavarz are also highly appreciated.

I would like to acknowledge the Department of Chemical and Materials Engineering at University of Alberta for providing me the opportunity to pursue my master degree. Financial support from the NSERC Industrial Research Chair in Control of Oil Sands Processes and Suncor Energy Inc. is greatly acknowledged. It was a valuable opportunity and experience working on the process modeling challenges in the oil sands industry. I would like to thank Dr. Enbo Feng, Dr. Ramesh Kadali, Eric Lau and Haitao Zhang for their guidance and support during the industrial projects.

Contents

List of Figures

List of Tables

List of Symbols

List of commonly used symbols

$\mathbf{X}$	Mean-centred input matrix $(N \times M)$
$\mathbf{y}$	Mean-centred output vector $(N \times 1)$
$\mathbf{x}_q$	The query sample $(1 \times M)$
$\mathbf{x}_i$	The i-th sample in the database $(1 \times M)$
N	Number of samples in database $\mathbf{X}$
M	Number of variables in database $\mathbf{X}$
$\tilde{x}(t)$	Mean-centred new input at time t $(1 \times M)$
$\tilde{y}(t)$	Mean-centred new output at time t (1×1)
$\mathbf{R}_{xx}(t)$	Covariance matrix of $\mathbf{X}$ at time t $(M \times M)$
$\mathbf{R}_{xy}(t)$	Covariance matrix of $\mathbf{X}$ and $\mathbf{y}$ at time t $(M \times 1)$
λ	Forgetting factor for updating $(0 < \lambda \leq 1)$
A	Number of latent variables in PLS calculation
$\hat{y}$	Predicted output value (1×1)
$\mathbf{t}$	Score vector in PLS $(N \times A)$
$\mathbf{p}$	Loading vector in PLS $(M \times A)$
d_i	Distance between query sample and i-th sample
w_i	Weight of i-th sample in the database
$\mathbf{W}$	Diagonal Weighting matrix $(N \times N)$
β	Regression coefficients
ρ	Balancing parameter

List of Abbreviations

List of commonly used abbreviations

NIR	Near-Infrared
VIP	Variable Importance in the Projection
UVE	Uninformative Variable Elimination
PLS	Partial Least Squares
PCR	Principal Component Regression
ANN	Artificial Neural Network
LWR	Locally Weighted Regression
SVM	Support Vector Machines
RPLS	Recursive Partial Least Squares
RLS	Recursive Least Squares
LWPLS	Locally Weighted Partial Least Square
JIT	Just-In-Time approach
OSC	Orthogonal Signal Correction
AKL	Adaptive Kernel Learning
RMSE	Root Mean Square Errors
MSC	Multiplicative Scatter Correction
SNV	Standard Normal Variate
SBC	Slope/Bias Correction
LV	Latent Variable
GA	Genetic Algorithm

Chapter 1

Introduction

1.1 Motivation

Estimation of real-time values of critical process variables is of great importance from the perspectives of both process monitoring and process control. Due to the limitations of measurement techniques, the high installation cost and the time consuming procedures that exist in the traditional offline laboratory analysis, it is desired to develop a measurement mechanism which can provide online fast rate predictions for the important quality variables of the process.

Near-infrared (NIR) spectroscopy has been widely adopted as a process analytical tool (PAT) in various fields; the most important reason is its ability to record spectra in real time to capture process properties [4, 5, 6]. Compared with traditional laboratory analysis, NIR analysis can provide estimation results significantly faster, which results in improved monitoring and control of product quality among petroleum, petrochemical, pharmaceutical, environmental and many other industry sectors [4, 5, 6, 7]. The principle of NIR instrument is shown in Figure 1.1 [1]. The instrument generates light beam and sends it through the optical fiber, at the end of which the probe is immersed into the sample. Part of the light will be absorbed

by the chemical components in the sample. By detecting and calculating the energy loss of the light beam, a spectrum can be generated as shown in Figure 1.2, where the absorbence is calculated by Equation 1.1:

$$a = \log(\frac{I_0}{I}) \tag{1.1}$$

where I is the light intensity after absorption and I_0 is the intensity of the incident light.

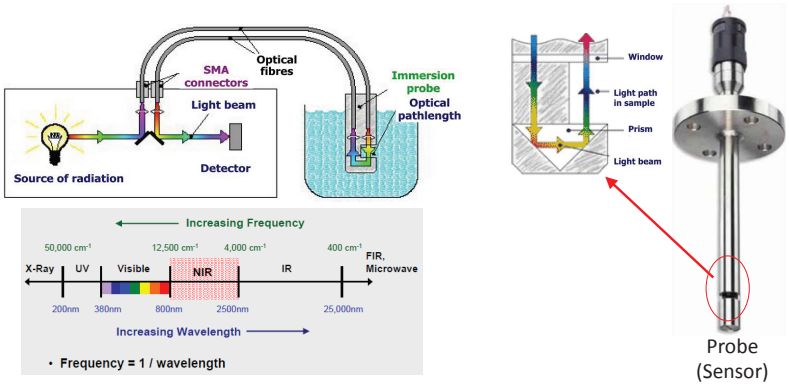

Figure 1.1: NIR instrument and probe [1]

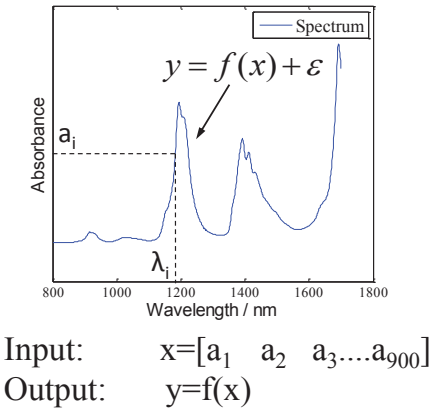

Figure 1.2: A typical NIR Spectrum

In NIR spectroscopy analysis, the main problem of interest is to build a model which relates the spectrum ($\mathbf{x}$) and the target property (y). The

spectrum is usually obtained online and the target property is often obtained by offline lab analysis. So combining these two techniques is of great importance in order to get the fast rate predictions of the quality variables. Traditional data-driven methods such as partial least squares (PLS) regression and principal component regression (PCR) are widely used in NIR modeling; however, when encountered with time varying and/or non-linearity issues, a invariant PLS model is not appropriate in terms of obtaining satisfactory prediction performance. This thesis deals with time varying and non-linearity issues in NIR modeling and three different approaches are proposed and evaluated.

1.2 Contributions

The main contributions of this thesis are the development of data-driven modeling methods to solve the time varying and non-linearity problems in Near-infrared spectroscopy modeling. The proposed methods result in improved model prediction performance. The specific contributions of this thesis can be summarized as follows:

1. Developed a recursive wavelength selection method to update both model structure and model coefficients in each update iteration.

2. Formulated the NIR modeling problem in a Just-in-time framework.

3. Proposed a similarity measurement using orthogonal signal correction (OSC), where both input and output information are utilized to calculate distance.

4. Formulated the recursive adaptation algorithm under the JIT framework and illustrated the relevance measurement from the perspective of both distance and time.

5. Proposed a new weighting scheme for JIT approach, where both non-linearity and time varying issues can be solved.

6. Evaluated the proposed algorithms using NIR data sets obtained from agriculture, petrochemical and pharmaceutical industry.

1.3 Thesis Outline

The rest of this thesis is organized as follows: Chapter 2 deals with time varying issues in NIR modeling using a recursive wavelength selection algorithm. The proposed algorithm can adapt both the model structure (wavelengths in NIR modeling) and the model coefficients by learning from the latest samples. The method is evaluated by case studies for diesel property estimation and wheat kernel protein content estimation. Chapter 3 formulates the NIR modeling problem in a Just-in-time (JIT) framework in order to solve the non-linearity problem that exists in the data. The historical information and the current process input are synthesized in order to search for samples that are similar to the current query sample. By prioritizing and assigning higher weights to samples that are more similar to the query sample, a local model is built by weighted regression. The proposed method utilizes both input and output information to calculate the similarity, and the usefulness of this method is illustrated by a benchmark NIR data set obtained from pharmaceutical industry. Chapter 4 solves both time varying and non-linearity issues in the same time under the JIT framework, where the locally weighted algorithm and recursive algorithm are combined to select important samples. The thesis is concluded in Chapter 5 which summarizes the work that has been done.

The literature review is distributed in each chapter. This thesis is organized in paper format so that there may be some overlap between the chapters for sake of completeness of each chapter.

Chapter 2

Recursive wavelength selection strategy for near infrared spectroscopy model updating

In this chapter, a new model updating approach is proposed which can adjust to process changes by recursively selecting the NIR model structure in terms of wavelength. Wavelength selection is widely accepted as an important step in near-infrared (NIR) spectroscopic model development. In quantitative on-line applications, the robustness of the established NIR model is often jeopardized by instrument response changes, process condition variations or new sources of chemical variation. However, to the best of the authors' knowledge, on-line updating of wavelength selection has not been considered in NIR modeling and property prediction. The advantage of the presented approach is that it can recursively adjust both wavelength selection and model coefficients according to real process variations. The performance of the method has been tested on a spectroscopic dataset

The content of this chapter has been published in M. Chen, S. Khare, B. Huang, H. Zhang, E. Lau, E. Feng, Recursive Wavelength-Selection Strategy to Update Near-Infrared Spectroscopy Model with an Industrial Application, *Industrial & Engineering Chemistry Research*, accepted May 2013.

from a refinery process. Compared with traditional PLS, locally weighted PLS and several updating strategies, it is shown that the proposed method achieves good accuracy in prediction of diesel properties.

2.1 Introduction

During recent years, near-infrared (NIR) spectroscopy has been widely adopted as a process analytical tool (PAT) in various fields; the most important reason is its ability to record spectra in real time to capture process properties [4, 5, 6]. Compared with traditional laboratory analysis, NIR analysis can provide estimation results significantly faster, which results in improved control and product quality among petroleum, petrochemical, pharmaceutical, environmental and many other sectors [4, 5, 6, 7]. In this paper, we estimate the diesel properties (cloud point and flash point) by NIR models, which are important indices for control purpose in a refinery process.

Spectrum measured on modern NIR scanning instruments has hundreds of wavelengths (usually 780-2500nm), which correspond to hundreds of spectral points often termed as variables in system identification literature. To deal with such information-rich data, considerable efforts have been directed towards the following problems: i) preprocessing; ii) wavelength selection; iii) regression analysis; iv) model updating. Spectra preprocessing has become an integral part of NIR modeling [8]. The objective of preprocessing is to remove light scattering phenomena in the spectra in order to improve the performance of the subsequent predictive model building. Due to light path length change, many methods have been proposed to correct for the baseline trend and curvilinearity [9], such as multiplicative scatter correction (MSC), extended multiplicative scatter correction (EMSC) [10], Savitzky-Golay derivative [11], standard normal variate (SNV) [12] and multivariate

method like orthogonal signal correction (OSC) [13].

Wavelength selection in NIR modeling is an important step as the removal of non-informative wavelengths will result in better prediction performances and reduce the model complexity [4, 7]. Other than selecting wavelengths manually based on chemical knowledge, a large amount of efforts have been devoted to the area of statistical methods such as successive projections algorithm (SPA) [14], uninformative variable elimination (UVE) [15], variable importance in the projection (VIP) [16], simulated annealing (SA) [17], Elastic Net Grouping [18], competitive adaptive reweighted sampling (CARS) [19], genetic algorithms (GAs) [20], interval partial least squares (iPLS) [21] and moving windows PLS [22]. Traditionally, the wavelength selection algorithm is performed in the NIR training data set, and then the selected wavelengths are used for modeling and prediction. However, those wavelengths selected from the training data set may not be representative of the future process condition due to process changes, resulting in the deterioration of the prediction performance. Model parameter updating may not be sufficient to capture the change and update of the model structure (i.e., the relevant wavelength in NIR modeling) needs to be considered. This motivates us to explore a wavelength updating method in which wavelengths are reselected according to the latest process information.

Based on Beer's law, most of the past research has used linear regression methods such as partial least squares (PLS) and principal component regression (PCR) to build NIR models [6, 23]. Some properties may have non-linear relationship with NIR absorbance, which results in the use of non-linear regression methods like artificial neural network (ANN) [24], least squares support vector machines (LS-SVM) [25], Gaussian process regression (GPR) [26] and locally weighted regression (LWR) [27].

Besides non-linearity, NIR models also suffer from time-varying issues,

especially when used for on-line prediction. In many process applications, changes in the instrument properties (e.g. spectrometer aging), variations in the measurement conditions (e.g. temperature) and new source of chemical variation may cause the existing models to become invalid [5]. Thus maintaining the model robustness over time is an important task in real time application. The NIR model updating methods can roughly be classified into the following categories: i) predicted value correction; ii) spectral response standardization; iii) recalculation of model coefficients.

One of the most widely used methods in predicted value correction is the slope/bias correction (SBC) [28]. However, since SBC is a univariate approach, it cannot handle complex interactions between wavelength shifts and intensity changes of the instrument (e.g. peak broadening)[29]. Only linear intensity changes can be corrected for by this method. Further, predicted value correction methods may be questionable when the process or measurement conditions undergo some complex changes [30].

The spectral response standardization, also known as calibration transfer, is a popular method in NIR field. In this approach, a transformation matrix is calculated to transform the spectra data obtained from the current condition/instrument into the original one. As a result, the original model can still be used for prediction without the need of updating model coefficients [30, 28]. The transformation matrix is found through some multivariate statistical methods, among which piecewise direct standardization (PDS) [28], spectral space transformation (SST) [31], dynamic orthogonal projection (DOP) [32] and systematic prediction error correction (SPEC) [33] are most widely used. However, most standardization methods require the spectra of a few samples to be measured under both calibration and test condition, which is often not available for on-line applications. Moreover, in the SPEC method, the number of chemical variation sources in the spectra needs to be identified prior to the application, which is a challenging task

for systems that cannot be characterized well (e.g. fuels, minerals).

A more straightforward way for model updating is the model coefficients recalculation. A simple way to do this is to add a few new samples into the original calibration set and identify the model coefficients again, so that the model is extended and adjusted to capture the new variations in the process [34, 35]. When facing a large number of new samples, weights should be assigned to the representative samples [36, 37]. Recently, some advanced methods have been used to accomplish the selection and weighting, such as Tikhonov regularization (TR) [38] and simple interval calculation (SIC) [39]. Another way to update NIR model coefficients is the recursive partial least square (RPLS) [40, 41, 42]. Other than accumulating a certain number of new samples, RPLS expands the database by adding every sample available and continuously recalculates the model coefficients. Since RPLS updates the NIR model frequently (every sample interval), it can capture the new variation without delay, which is preferred for rapidly changing processes (e.g. mining, refining). A successful implementation of RPLS algorithm in mining process has been reported [43, 44, 45].

Among all the model updating methods listed above, on-line wavelength selection has not been considered in the model update. The model structure (wavelength) used in the NIR model is typically selected from the initial calibration data set and remains unchanged in further prediction. However, it is questionable whether the new sources of variation can be captured by these pre-selected wavelengths. As far as author's knowledge, this is the first attempt which integrates the real-time wavelength selection with model update in the NIR applications. In this chapter, a novel algorithm is proposed in which the wavelengths are selected recursively and the model is updated. The proposed algorithm utilizes both the recursive exponentially weighted PLS and the method of variable importance in the projection (VIP). The performance of the proposed algorithm is demonstrated on an NIR data set

from a refinery production line. The prediction results are compared with other modeling techniques such as PLS, locally weighted PLS and recursive PLS. It is shown that the informative wavelengths are updated accordingly through the real-time wavelength selection. The prediction performance is improved by the proposed algorithm compared to predictions without real-time wavelength update.

2.2 Necessity of wavelength updating

In this section, the necessity of wavelength updating is investigated from various aspects.

According to Beer's law, the relationship between output y (property) and input x_i (absorption) can be expressed in a linear form as shown in Equation 2.1.

$$y = b_0 + b_1 x_1 + b_2 x_2 + \cdots + b_m x_m \tag{2.1}$$

where y denotes a property related to concentration (e.g. density, viscosity). Usually, only part of the wavelengths are retained for model training in order to remove the noise from the spectra. However, if there are new chemical resources emerging in the process, the wavelengths have to be updated. This is due to the large number of vibration modes and electronic transitions associated with NIR spectra region. Moreover, neighbor group effects, hydrogen bonding, phase separation etc. can also affect the results of wavelength selection [4, 6]. It is even possible that two samples with very similar element compositions have different spectra [43]. On the other hand, if there is any chemical component disappearing from the process, the informative wavelengths will also change. Thus, a model with flexibility of structure update will be able to capture the real-time variability.

The on-line spectroscopic measurements are almost inevitably subject to fluctuations and variations of physical condition, e.g. temperature, pres-

sure, flow turbulence, sample compactness, particle size and surface topology, which can influence spectra in a non-linear manner and hence lead to a poor predictive performance of linear models [4, 30]. The fluctuations in temperature cause non-linear spectra shift and broaden the spectral bands of chemical constituents [30]. Physical variations such as particle size, sample packing and surface effects change the optical path length and mask the spectral variations related to chemical constituents differences [30], which are commonly observed in heterogeneous mixtures [46, 47]. Air bubbles, flow rate and solid impurities in the process flow also have significant influences on the spectral absorption and baseline [34]. All these factors listed above may vary and cannot be foreseen. This motivates the attempt of adaptively choosing the informative wavelengths according to the latest process and environmental conditions.

Instrument aging or repairing also affects the spectrometer response; for instance, a lamp or probe replacement can usually change the spectra in some unknown manner. The wavelength axis shift, spectral shrink/stretch and non-linear baseline shift are commonly seen among NIR instrument over time [48]. It is then recommended to adjust the wavelengths that are initially chosen from training data set accordingly.

2.3 Theory and algorithm

In this section, the recursive PLS and variance importance in the projection (VIP) selection method are combined to update the model structure (wavelength) and model coefficients recursively. It should be noted that, in NIR modeling, the data need not to be scaled as all the input variables are in the same scale of magnitude. The following section illustrates the details of the algorithm.

2.3.1 Recursive exponentially weighted PLS (rPLS)

The fast recursive exponentially weighted PLS algorithm proposed by Dayal and MacGregor [41] is adopted here for model coefficient update. In this approach, based on the improved PLS kernel algorithm [49], a forgetting factor is introduced to exponentially discount the past data. The updating of covariance matrices is performed through Equation 2.2-2.3.

$$\mathbf{R}_{xx}(t) = \lambda \mathbf{R}_{xx}(t-1) + \tilde{x}^{\mathrm{T}}(t)\tilde{x}(t) \tag{2.2}$$

$$\mathbf{R}_{xy}(t) = \lambda \mathbf{R}_{xy}(t-1) + \tilde{x}^{\mathrm{T}}(t)\tilde{y}(t) \tag{2.3}$$

where the forgetting factor λ, $0 < \lambda \leq 1$, determines how quickly the old data are discounted. Once the covariance matrices are updated, a fast kernel PLS calculation is followed as briefly explained below:

1. Compute the covariance matrices $\mathbf{R}_{xx}$ and $\mathbf{R}_{xy}$.

2. Set $k = 1$ and determine the number of latent variables as A.

3. Compute the weight vector $\mathbf{w}_k$

$$\mathbf{w}_k = \mathbf{R}_{xy} \tag{2.4}$$

$$\mathbf{w}_k = \frac{\mathbf{w}_k}{\|\mathbf{w}_k\|} \tag{2.5}$$

4. Compute $\mathbf{r}_k$ using the following formula:

$$\mathbf{r}_k = \begin{cases} \mathbf{w}_k & \text{if } k = 1 \\ \mathbf{r}_k = \mathbf{w}_k - \mathbf{p}_1^{\mathrm{T}}\mathbf{w}_k\mathbf{r}_1 - \mathbf{p}_2^{\mathrm{T}}\mathbf{w}_k\mathbf{r}_2 \ldots \mathbf{p}_{k-1}^{\mathrm{T}}\mathbf{w}_k\mathbf{r}_{k-1} & \text{if } k > 1 \end{cases} \tag{2.6}$$

5. The kth score vector and loading vector are obtained as in Equation 2.7-2.9.

$$\mathbf{t}_k^{\mathrm{T}}\mathbf{t}_k = \mathbf{r}_k^{\mathrm{T}}\mathbf{R}_{xx}\mathbf{r}_k \tag{2.7}$$

$$\mathbf{p}_k^{\mathrm{T}} = \frac{\mathbf{r}_k^{\mathrm{T}}\mathbf{R}_{xx}}{\mathbf{t}_k^{\mathrm{T}}\mathbf{t}_k} \tag{2.8}$$

$$\mathbf{q}_k^{\mathrm{T}} = \frac{\mathbf{r}_k^{\mathrm{T}}\mathbf{R}_{xy}}{\mathbf{t}_k^{\mathrm{T}}\mathbf{t}_k} \tag{2.9}$$

6. Deflate the covariance matrix $\mathbf{R}_{xy}$ as in Equation 2.10.

$$\mathbf{R}_{xy} = \mathbf{R}_{xy} - \mathbf{p}_k\mathbf{q}_k^{\mathrm{T}}(\mathbf{t}_k^{\mathrm{T}}\mathbf{t}_k) \tag{2.10}$$

7. If $k = A$, go to the next step. Otherwise, set $k = k + 1$ and return to step 3.

8. Compute the regression coefficient by

$$\mathbf{b} = [\mathbf{r}_1 \quad \mathbf{r}_2 \ldots \mathbf{r}_A][\mathbf{q}_1 \quad \mathbf{q}_2 \ldots \mathbf{q}_A]^{\mathrm{T}} \tag{2.11}$$

The regression coefficients $\mathbf{b}$ can be used for future prediction. On the other hand, the score vector $\mathbf{t}_k$, loading vector $\mathbf{q}_k$ and weight vector $\mathbf{w}_k$ are stored and used for VIP wavelength selection as discussed below.

2.3.2 Variable Importance in the Projection (VIP)

There are numerous wavelength selection methods in the literature. Perhaps the simplest method to update wavelength is to retry the corresponding wavelength selection method and reconstruct the PLS in each updating iteration. However, most wavelength selection methods are based on cross validation, where the wavelength updating will not be effective because several new samples are not be able to change the cross validation results too much.

In this study, the variable importance in the projection (VIP) is adopted for its great compatibility with recursive PLS. Recursive PLS can take full advantage of the latest samples by setting a smaller forgetting factor (if the

process is relatively stable, a greater value can be set). In this case, a full spectrum PLS model is retained in memory and updated whenever a new sample is added into the database. Using the score vectors, y-loadings and weight vectors from the updated full spectrum PLS model, the VIP index for each wavelength is recalculated according to Equation 2.12.

$$VIP_j = \sqrt{\frac{M \sum_{k=1}^{A} \mathbf{q}_k^2 \mathbf{t}_k^{\mathrm{T}} \mathbf{t}_k \frac{\mathbf{w}_{jk}}{\|\mathbf{w}_k\|}}{\sum_{k=1}^{A} \mathbf{q}_k^2 \mathbf{t}_k^{\mathrm{T}} \mathbf{t}_k}} \tag{2.12}$$

where j represents the jth wavelength, $1 \leq j \leq M$ and $\mathbf{q}_k$ is 1×1 dimensional. A user defined threshold, α, is set to filter out unimportant wavelengths. As shown in Equation 2.13, wavelengths with a value larger than α will be regarded as informative wavelengths.

$$VIP_j > \alpha \tag{2.13}$$

Since the score vector, loading vector and weight vector are all updated recursively, the wavelengths selection will be adjusted accordingly in every recursion step.

2.3.3 Proposed updating scheme

In this algorithm, a full spectrum PLS model is retained in memory for calculating the VIP indices. However, rather than using the full spectrum model for prediction, another PLS model is built with the wavelengths selected by VIP method. Each time when a new sample is available, the full spectrum model is updated so that the VIP indices can be recalculated. The whole updating scheme is pictorially summarized as in Figure 2.1.

Before updating, an initial model is obtained from the database. The initial covariance matrices are calculated from the mean centred data $(\mathbf{X}, \mathbf{y})$

using Equation 2.14-2.15.

$$\mathbf{R}_{xx}(0) = \mathbf{X}^{\mathrm{T}}\mathbf{X} \tag{2.14}$$

$$\mathbf{R}_{xy}(0) = \mathbf{X}^{\mathrm{T}}\mathbf{y} \tag{2.15}$$

Based on the initial covariance matrices, a full spectrum PLS model is built and the VIP wavelength selection is performed. Then the important variables (wavelengths) are selected to construct the reduced-dimensional covariance matrices $\mathbf{r}_{xx}(0)$ and $\mathbf{r}_{xy}(0)$. Finally, the initial model coefficients $\mathbf{b}_0$ is generated by the fast kernel PLS algorithm mentioned above.

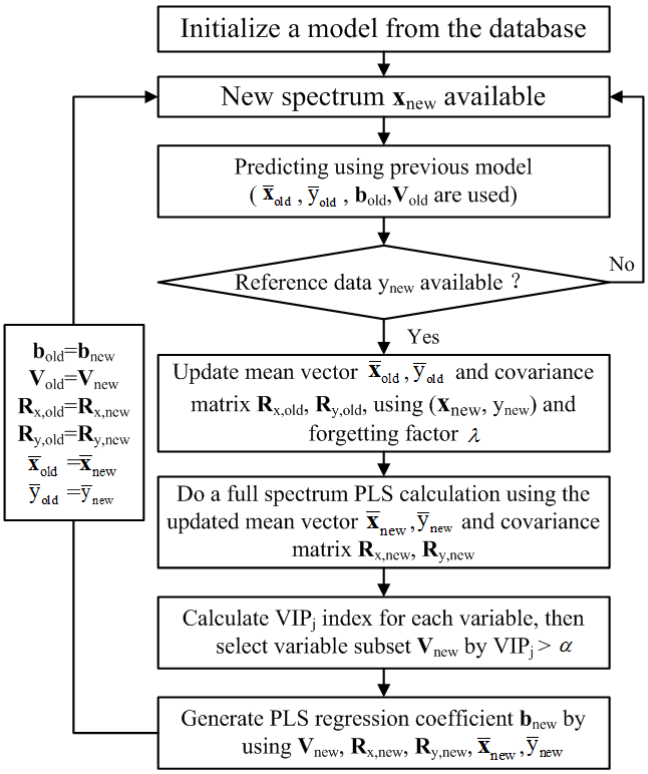

Figure 2.1: Model structure and coefficients update scheme

When the new spectrum $x(t)$ and reference data $y(t)$ are available at time t, the mean vectors are updated according to the method proposed by

Su and Zeng [50], as shown in Equation 2.16-2.17.

$$\overline{x}(t) = \frac{N-1}{N}\overline{x}(t-1) + \frac{1}{N}x(t) \tag{2.16}$$

$$\overline{y}(t) = \frac{N-1}{N}\overline{y}(t-1) + \frac{1}{N}y(t) \tag{2.17}$$

Then the mean-centred new data are obtained as

$$\tilde{x}(t) = x(t) - \overline{x}(t) \tag{2.18}$$

$$\tilde{y}(t) = y(t) - \overline{y}(t) \tag{2.19}$$

which can be used in Equation 2.2-2.3 to update the covariance matrices.

As a result, a full spectrum PLS model is calculated with the updated covariance matrices so that the VIP indices can be recalculated using Equation 2.12. The important wavelengths at time t are selected according to Equation 2.13. At this iteration, let L out of M wavelengths be selected from the full spectrum. The selected wavelengths are stored in a vector denoted by Equation 2.20.

$$\mathbf{v} = [v_1 \quad v_2 \cdots v_L] \tag{2.20}$$

It should be noted that L may differ with each iteration. In order to get the relevant model for prediction purpose, the reduced-dimensional covariance matrices $\mathbf{r}_{xx}(t)$ and $\mathbf{r}_{xy}(t)$ are computed using $\mathbf{v}$. However, they cannot be updated in the same way as $\mathbf{R}_{xx}(t)$ or $\mathbf{R}_{xy}(t)$ since $\mathbf{r}_{xx}(t-1)$ and $\mathbf{r}_{xy}(t-1)$ may be constructed with variables different from $\mathbf{v}$.

The following discussion illustrates a way to extract $\mathbf{r}_{xx}(t)$ directly from the full-dimensional covariance matrice $\mathbf{R}_{xx}(t)$. From Equation 2.2 and

(2.14, we can derive:

$$\begin{aligned}\mathbf{R}_{xx}(t) = \lambda^t \mathbf{X}^{\mathrm{T}}\mathbf{X} + \lambda^{t-1}\tilde{x}^{\mathrm{T}}(1)\tilde{x}(1) + \lambda^{t-2}\tilde{x}^{\mathrm{T}}(2)\tilde{x}(2)\cdots \\ + \lambda\tilde{x}^{\mathrm{T}}(t-1)\tilde{x}(t-1) + \tilde{x}^{\mathrm{T}}(t)\tilde{x}(t)\end{aligned} \tag{2.21}$$

In the meantime, the reduced-dimensional covariance matrix constructed from chosen wavelengths can be expressed as Equation 2.22:

$$\begin{aligned}\mathbf{r}_{xx}(t) = \lambda^t \Phi^{\mathrm{T}}\Phi + \lambda^{t-1}\phi^{\mathrm{T}}(1)\phi(1) + \lambda^{t-2}\phi^{\mathrm{T}}(2)\phi(2)\cdots \\ + \lambda\phi^{\mathrm{T}}(t-1)\phi(t-1) + \phi^{\mathrm{T}}(t)\phi(t)\end{aligned} \tag{2.22}$$

where Φ and $\phi(t)$ are subsets of $\mathbf{X}$ and $\tilde{x}(t)$, respectively, as shown in Equation 2.23.

$$\Phi = [\mathbf{x}_{v_1} \quad \mathbf{x}_{v_2}\cdots\mathbf{x}_{v_L}], \quad \phi(t) = [\tilde{x}_{v_1}(t) \quad \tilde{x}_{v_2}(t)\cdots\tilde{x}_{v_L}(t)] \tag{2.23}$$

where $\mathbf{x}_i$ and $\tilde{x}_i(t)$ denote the ith column of $\mathbf{X}$ and $\tilde{x}(t)$, respectively. To find the relationship of $\mathbf{R}_{xx}(t)$ and $\mathbf{r}_{xx}(t)$, we compare the (i,j)th entry $(i,j = 1\ldots M)$ of $\mathbf{R}_{xx}(t)$ with the (v_p, v_q)th entry $(p,q = 1\ldots L)$ of $\mathbf{r}_{xx}(t)$ as following:

$$\begin{aligned}R_{i,j} = \lambda^t \mathbf{x}_i^{\mathrm{T}}\mathbf{x}_j + \lambda^{t-1}x_i^{\mathrm{T}}(1)x_j(1) + \lambda^{t-2}x_i^{\mathrm{T}}(2)x_j(2)\cdots \\ + \lambda x_i^{\mathrm{T}}(t-1)x_j(t-1) + x_i^{\mathrm{T}}(t)x_j(t)\end{aligned} \tag{2.24}$$

$$\begin{aligned}r_{v_p,v_q} = \lambda^t \mathbf{x}_{v_p}^{\mathrm{T}}\mathbf{x}_{v_q} + \lambda^{t-1}x_{v_p}^{\mathrm{T}}(1)x_{v_q}(1) + \lambda^{t-2}x_{v_p}^{\mathrm{T}}(2)x_{v_q}(2)\cdots \\ + \lambda x_{v_p}^{\mathrm{T}}(t-1)x_{v_q}(t-1) + x_{v_p}^{\mathrm{T}}(t)x_{v_q}(t)\end{aligned} \tag{2.25}$$

From Equation 2.24-2.25, it is easy to get Equation 2.26.

$$r_{v_p,v_q} = R_{v_p,v_q} \tag{2.26}$$

Thus $\mathbf{r}_{xx}(t)$ is nothing but a submatrix of $\mathbf{R}_{xx}(t)$ as shown in Equation 2.27.

$$\mathbf{r}_{xx}(t) = \begin{bmatrix} R_{v_1,v_1} & R_{v_1,v_2} & \cdots & R_{v_1,v_L} \\ R_{v_2,v_1} & R_{v_2,v_2} & \cdots & R_{v_2,v_L} \\ \vdots & \vdots & \ddots & \vdots \\ R_{v_L,v_1} & R_{v_L,v_2} & \cdots & R_{v_L,v_L} \end{bmatrix} \tag{2.27}$$

In the same way, one can show that $\mathbf{r}_{xy}(t)$ can be easily extracted from $\mathbf{R}_{xy}(t)$ as well. Once the updated covariance matrices $\mathbf{r}_{xx}(t)$ and $\mathbf{r}_{xy}(t)$ are obtained, the fast kernel algorithm is applied and the regression coefficients are generated. This PLS model is used for future prediction until next updating instance.

2.4 Results and discussion

2.4.1 Diesel data and benchmark wheat kernels data

The NIR data used here were obtained from a refinery plant, located in Edmonton, Canada. A total of 577 diesel samples are collected from on-line operation between January 2010 and April 2012, the sampling rate of which is one day per sample. The spectra of diesel samples were measured using an NIR spectrometer (Guidewave) having the wavelength range of 800-1700 nm and nominal spectral resolution of 1 nm. Reference diesel properties (flash point and cloud point) were measured using standard ASTM testing methodologies.

There were 11 samples identified as outliers and thus removed from the database. The standard 3-sigma analysis was used to identify the samples with extreme y-values. The samples within the range specified by Equation

2.28 were kept for investigation.

$$\overline{y} - 3\sigma \leq y_i \leq \overline{y} + 3\sigma \tag{2.28}$$

where $\overline{y}$ is the mean value and σ is the standard deviation of outputs. Then, the first 250 samples were used to train the initial model, and the remaining 320 samples were used for testing. The original spectra of diesel samples are shown in Figure 2.2. In order to eliminate the baseline effect, all the spectra were subject to first order derivative preprocessing according to the Savitzky-Golay method. The spectra after preprocessing can be seen in Figure 2.2.

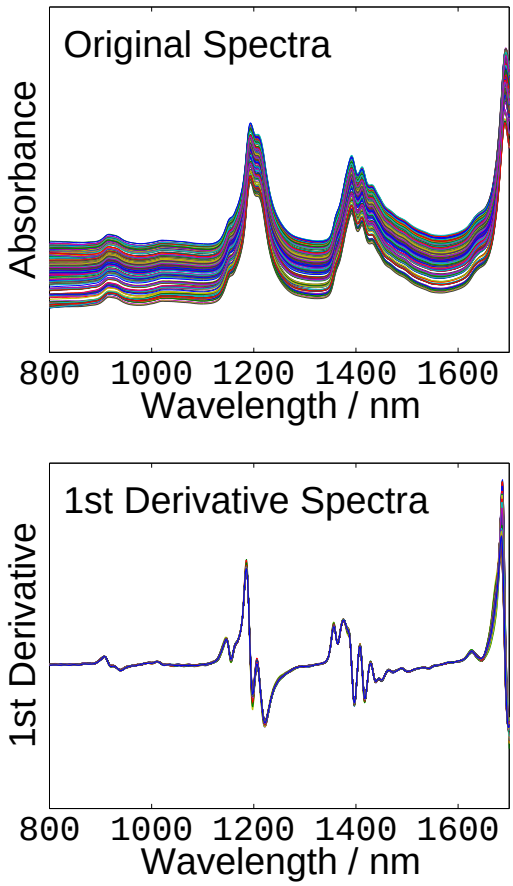

Figure 2.2: Original and first derivative spectra of diesel samples

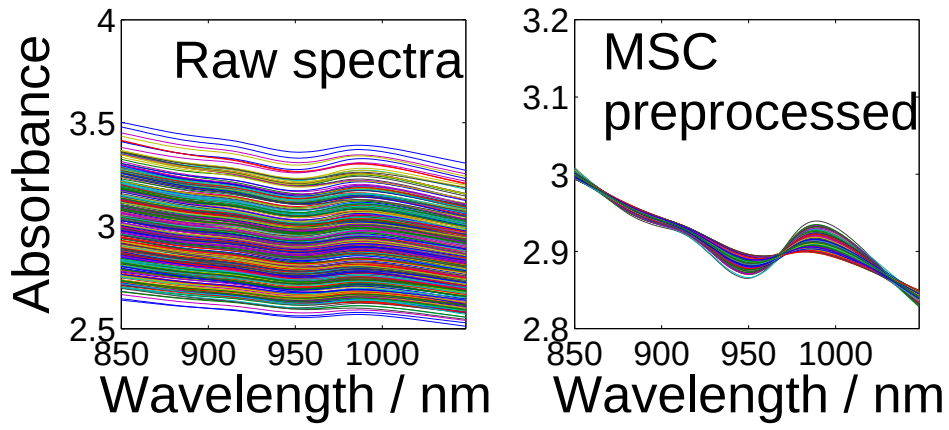

Figure 2.3: Wheat kernels spectra

Wheat kernels data set was downloaded from (http://www.models.kvl.dk/datasets). The calibration set was made up of 415 samples representing 43 different varieties from two different locations. The testing set was made of 108 samples representing 11 different varieties from one location. All kernels were randomly chosen from bulk samples. The test samples were acquired with the calibration samples, but stored for about two additional months before measurement in order to provide a check for temporal drift in the samples and instrumentation. The target property is the protein content (%) in the kernels. The wheat kernels spectra can be seen in Figure 2.3. The spectra have been preprocessed by MSC method before modeling.

2.4.2 Results for diesel data

In this study, two important properties, flash point and cloud point of the diesel from the production line of the refinery plant are to be estimated. Traditionally, the property measurements are obtained from off-line laboratory analysis, which takes one day to have one sample measured. However, no measurements are available for the intermediate points during the sample interval. Thus the estimation of the real time properties is desired. An accurate real-time estimation will help to operate and control the process ef-

ficiently. The NIR analyzer is able to scan one sample in just a few seconds. Once combined with prediction models, it can offer fast on-line estimation of the diesel properties.

Several modeling approaches are attempted and the model performance is evaluated by Root Mean Square Error of Prediction (RMSEP) and the correlation coefficient R. The effects of wavelength updating are also discussed. For proprietary reasons, the property values are normalized between -1 and 1.

The real benefit of recursive wavelength updating scheme (denoted as rwPLS) is that it can adapt both model structure and coefficients according to the latest reference data. To show the advantage of real-time updating wavelength, another five modeling approaches were investigated and compared with the proposed one. It should be noted that, in the initial modeling stage, the same wavelength selection (VIP) was applied to all the six approaches. Only in rwPLS, the VIP calculation was repeated every time when the reference data was available. The threshold α was set as 0.8 for Flash point and 0.7 for Cloud point. A procedure to set optimal value of α has been reported [16]. For the other five approaches, the model structure (wavelength) is fixed all the time. The five approaches are summarized below:

1. PLS: a PLS model with fixed model structure and coefficients was trained by the calibration data set (the first 250 samples). Since the fixed model did not adapt to the process variations, it was expected that its performance would deteriorate quickly.

2. LW-PLS: the locally weighted PLS (LW-PLS) model was investigated since it can deal with changes in process characteristics as well as nonlinearity [51]. In this approach, every query sample had a local model to predict the output, i.e. the local model was only trained after the

query sample was available. Different weights were assigned to the calibration samples based on the distance between the query sample and calibration samples.

3. LW-PLS2: the locally weighted PLS with extended database (LW-PLS2) was attempted to improve the performance of LW-PLS. The most recent samples were added into the calibration data set in order to provide the latest process information. However, in order to avoid the computational load, we kept the number of calibration samples constant by removing the equal number of samples from the distant past.

4. PLS+bias: as illustrated in Equation 2.29, a bias updating mechanism was integrated with the fixed PLS model in order to detect the bias in the predictions. However, the model coefficients remained unchanged.

$$\hat{y}_t = \hat{y}_t^{PLS} + (y_{t-1}^{Lab} - \hat{y}_{t-1}^{PLS}) \tag{2.29}$$

5. rPLS: recursive exponentially weighted PLS (rPLS) was investigated where the model coefficients were updated in every iteration but the model structure was still fixed. In this study, the forgetting factor λ is set as 0.9 by cross validation.

The model performances of fixed PLS and LW-PLS can be easily evaluated by cross-validation. However, because of the adaptive nature of the remaining four approaches, the new responses are predicted with the existing model before using the new sample for updating, and then the predicted responses are compared with the reference value. The RMSEP and correlation coefficient R for each model are shown in Table 2.1.

Table 2.1: Comparison between rwPLS and other updating methods with fixed model structure

	Flash point		Cloud point	
Model	RMSEP	R	RMSEP	R
PLS	0.2998	0.5758	0.3197	0.5854
LW-PLS	0.2511	0.6725	0.2587	0.6081
LW-PLS2	0.2473	0.7448	0.2026	0.6362
PLS+bias	0.1773	0.8238	0.1914	0.7281
rPLS	0.1530	0.8814	0.1488	0.7815
rwPLS	0.1287	0.9009	0.1504	0.8067

The model which gives the lowest RMSEP and the highest R is considered as the best one. For both properties, the two static approaches, fixed PLS and LW-PLS are not able to estimate the validation data effectively and give considerably higher error values than the other four approaches as expected. This clearly shows the advantage of model updating. The LW-PLS approach, which can account for non-linearity, gives a better performance than fixed PLS but is still worse than the models with updating strategies. This indicates that the time-varying issue is more important and more severe than the linearity in the considered process data. LW-PLS2 gives better results than LW-PLS because the new samples are added into the database to account for process variations. However LW-PLS2 cannot reach the performance of simple bias correction. This is because the most recent added samples were not used in every iteration. If the distances between query sample and the recent samples were too large, the recent samples would have been assigned small weights; thus the latest process information was not learned sufficiently. The bias correction seems to be very effective among the first four approaches. But it is still worse than the last two recursive-based methods. Actually, rPLS and rwPLS can detect

the bias term by the mean vector updating in Equation 2.16-2.17, but bias correction cannot cover the model coefficients updating or model structure updating. This is why rPLS and rwPLS outperform the bias correction method. Among all the approaches with fixed model structure, rPLS gives the lowest RMSEP and highest R because it uses the new samples to adapt the model coefficients recursively. Bias updated PLS only adapts the bias term and keeps the model coefficients constant. On the other hand, since rwPLS adapts both the model structure and coefficients recursively, and the process variations are learned by the new model in real time, the lowest prediction error is obtained.

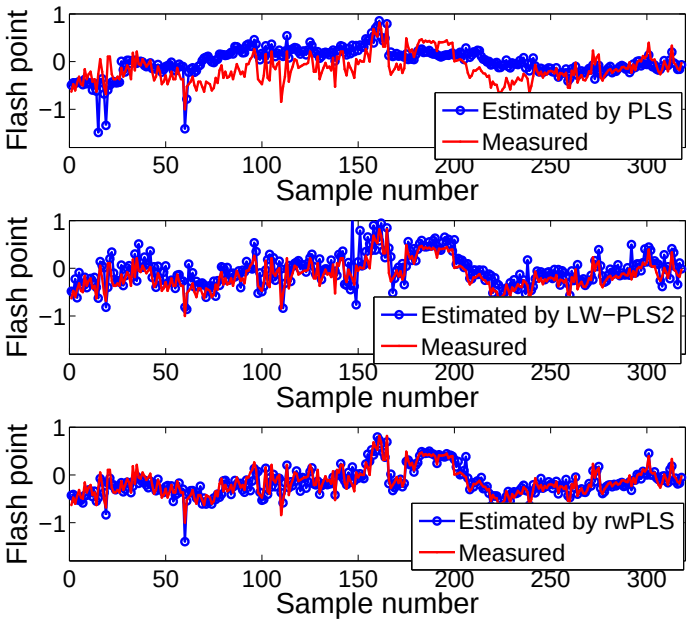

Figure 2.4: Model performance for flash point estimation: a) PLS; b) LW-PLS2; c) rwPLS

The prediction results of flash point are shown in Figure 2.4, and it is clear that the fixed PLS model deteriorates after a first few sampling instances. The higher RMSEP values presented in Table 2.1 confirm this result. LW-PLS2 achieves a better performance since it includes the most

recent data and develops model at every point. However, as seen in Table 2.1, LW-PLS without updated database gets a poorer result than LW-PLS2. From Figure 2.4, it can be seen that the proposed algorithm (rwPLS) provides the best way to prevent the model from gradually losing accuracy. From Figure 2.5, the same conclusion can be drawn for the other property, namely, cloud point.

The comparison in prediction performance is also made by scatter plots as can be seen from Figure 2.6. In summary, the model structure and coefficients updating yields significant improvement in prediction performance.

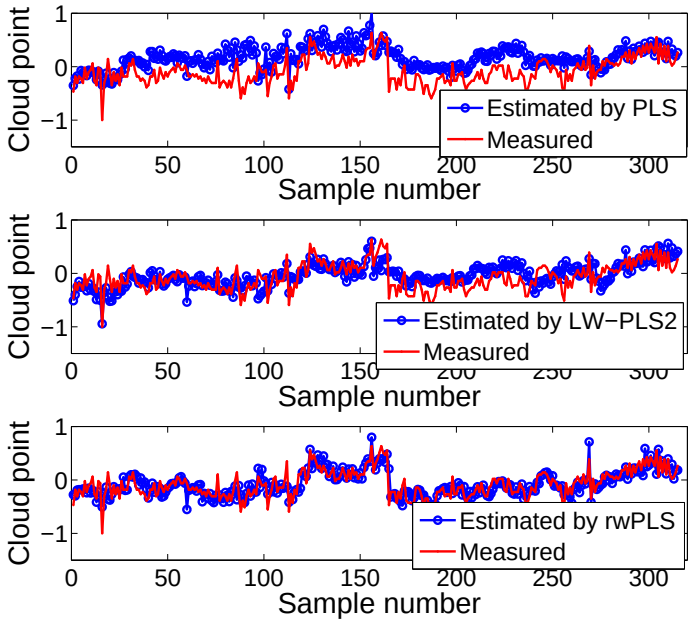

Figure 2.5: Model performance for cloud point estimation: a) PLS; b) LW-PLS2; c) rwPLS

In order to investigate the effects of wavelength updating, the number of the selected wavelengths in each updating iteration was recorded and shown in Figure 2.7 and 2.8. As expected, the number of selected wavelengths indeed changed with time, indicating that the model structure was adapted in every updating interval. It is interesting to see that there exist some

instances where the wavelength number changed suddenly. For example, the shaded region A and D experienced the sharp increase in wavelength numbers. This is probably due to process condition changes reflected in the data. Gradual changes of wavelength number were also seen in the shaded region B and E. In the meantime, the bias terms to correct the fixed PLS models were plotted in Figure 2.7 and 2.8.

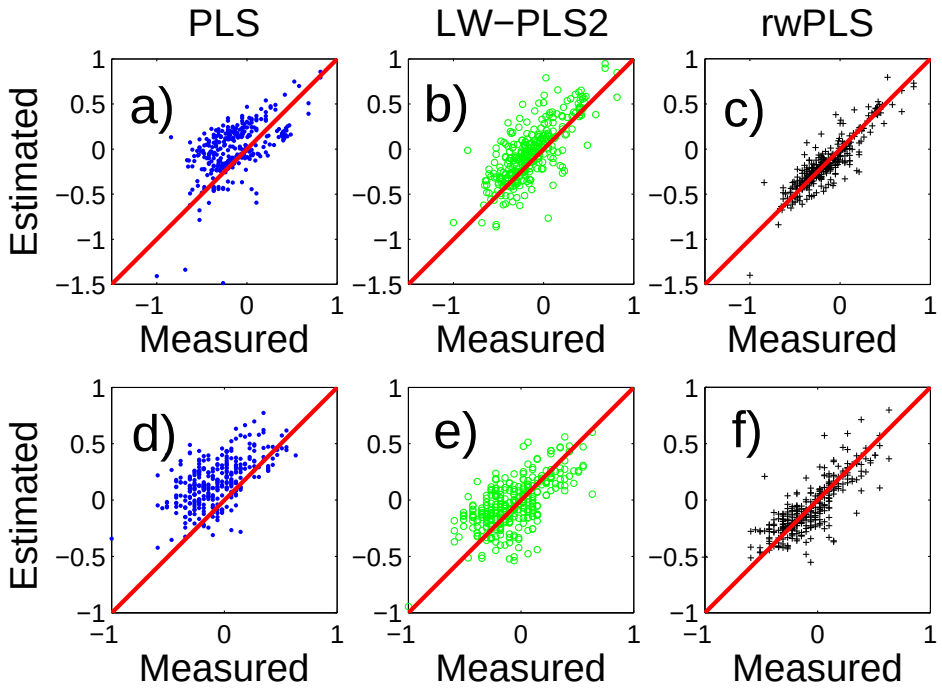

Figure 2.6: Scatter plots of model predictions: Flash point (a,b,c) and Clound point (d,e,f)

The large bias terms indicate that a fixed PLS model is not adequate to explain the data. Moreover, the behavior of bias term indicates that some structures have not been captured by the bias updating. Especially some peaks are seen from the bias updating trajectory. Therefore the simple bias correction is not enough to compensate the changes in the process properties, which confirms the results presented in Table 2.1. The peaks of the bias update located in the shaded region A and D explain the sudden wavelength changes in the corresponding periods. In fact, the relationship

between the bias term and wavelength updating can also be seen from other shaded areas. The results show that the model accuracy is not only affected by the correctable bias error (which may be due to some systematic errors) but also by certain structural variations in the process. The latter should be compensated by updating both the model structure (wavelength selected) and the model coefficients.

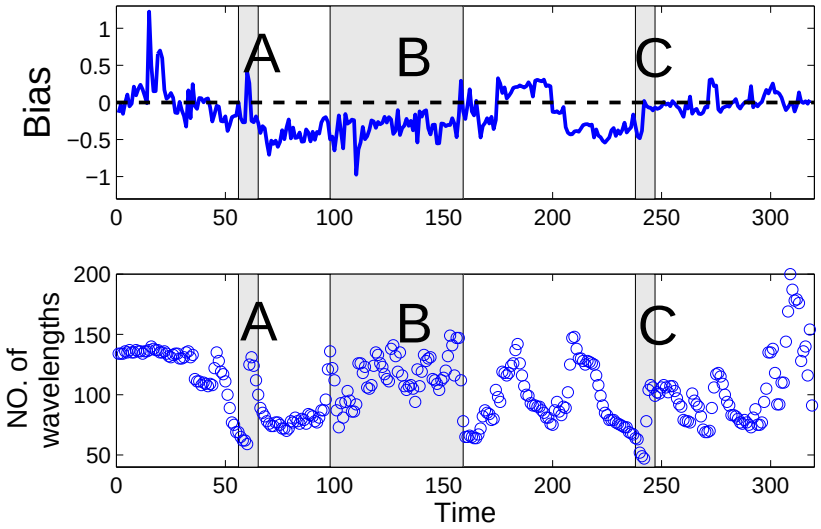

Figure 2.7: Wavelengths updating for flash point estimation

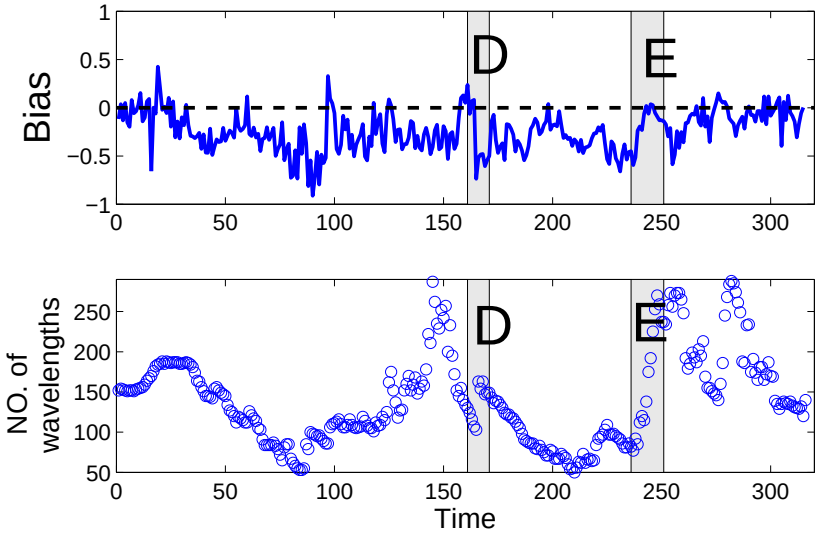

Figure 2.8: Wavelengths updating for cloud point estimation

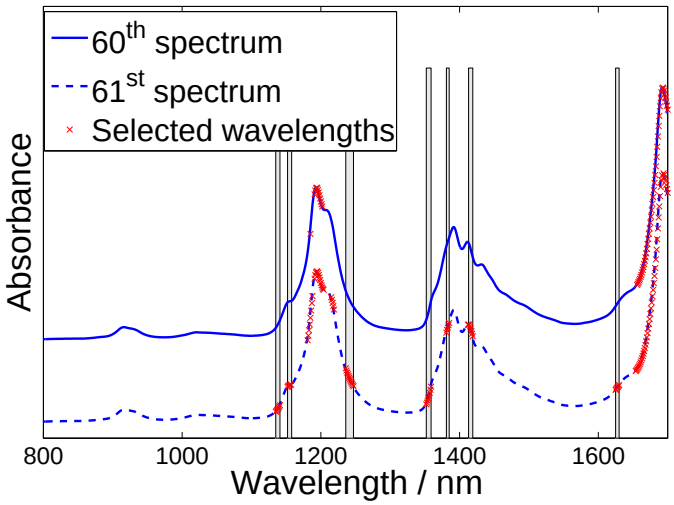

Figure 2.9: Wavelength selection from 60th to 61th sample (flash point)

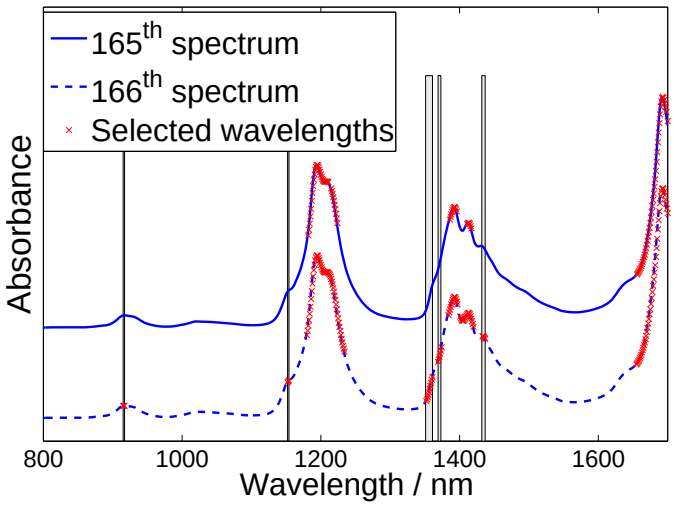

Figure 2.10: Wavelength selection from 165th to 166th sample (cloud point)

Figure 2.9 contains the details of wavelength selection for region A in Figure 2.7. The shaded wavelength intervals (around 1150nm, 1250nm 1350-1430nm and 1620nm) in Figure 2.9 were not selected at the 60th sample but later included into the model by the 61st sample. These new wavelengths can be approximately attributed to first overtone, second overtone and combination bands of C-H groups [52]. This indicates that new chem-

ical components were emerging in that period. Since new chemicals had contribution to the Flash point, the correlation among wavelengths were affected and an updated model structure was needed. Similarly, Figure 2.10 shows the additionally selected wavelengths (920nm, 1150nm and 1400nm) in region D of Figure 2.8. The 920nm interval can be attributed to the third overtone of methylene. Since the new emerged chemical groups had influence on the Cloud point, the corresponding model structure should be updated by rwPLS.

To further justify the usefulness of rwPLS, three other wavelength updating strategies were compared with rPLS and rwPLS. In these three strategies, whenever a new sample was available, the database was updated by a moving-window approach (delete the oldest samples from the database and add the latest sample). Then a new model was constructed using corresponding wavelength selection method followed by PLS regression. It is evident that both model structure and model coefficients are updated in each iteration. The details are as below:

1. VIP-PLS: update wavelength by VIP, then reconstruct the PLS model and the threshold α was set the same as in rwPLS.

2. UVE-PLS: update wavelength by UVE [15] and then reconstruct the PLS model.

3. iPLS-PLS: update wavelength by iPLS and then reconstruct the PLS model. The iToolbox for MATLAB contributed by L. Norgaard [21] was used.

The results are shown in Table 2.2

Table 2.2: Comparison of different wavelength updating strategies

	Flash point		Cloud point	
Model	RMSEP	R	RMSEP	R
VIP-PLS	0.1631	0.8548	0.1584	0.7685
UVE-PLS	0.1542	0.8722	0.1712	0.7266
iPLS-PLS	0.1472	0.8893	0.1607	0.7631
rPLS	0.1530	0.8814	0.1488	0.7815
rwPLS	0.1287	0.9009	0.1504	0.8067

Compared with Table 2.1, these three wavelength updating strategies achieve much better results than PLS, LW-PLS and LW-PLS2, which confirms that the process was changing frequently, and the latest sample should be learned in time. Also, since the model coefficients are updated in these three strategies, they outperform the PLS+bias approach.

For Flash point, the iPLS-PLS approach is better than rPLS, while UVE-PLS has comparable results with rPLS. For Cloud point, both VIP-PLS and iPLS-PLS have similar results with rPLS. Although rPLS approach do not update the wavelength, it has two advantages: 1) the bias change can be corrected by the mean vector updating shown in Equation 2.16-2.17; 2) more weight can be assigned to the latest samples by adjusting the forgetting factor λ. On the other hand, the information of the latest sample may be diluted by the large number of samples in the database if we simply add samples into the database and delete corresponding number of oldest samples, just as VIP-PLS, UVE-PLS and iPLS-PLS do.

Overall, rwPLS has advantages from both rPLS and wavelength selection. By wavelength updating, rwPLS can detect changes in correlation among wavelengths, which rPLS cannot achieve. So the lowest $RMSEP$ and largest R can be seen in Table 2.2.

2.4.3 Results for wheat kernels data

In the benchmark wheat kernels dataset, the test samples were stored for additional two months so that the prediction results will be influenced by the drift in both samples and instrumentation. To check how wavelength selection and recursive algorithm enhance the model performance, the following methods are tried:

1. PLS: a constant PLS model without wavelength selection or any updating.
2. UVE: a constant PLS model using variables selected by UVE.
3. iPLS: a constant PLS model using variables selected by iPLS.
4. rPLS and rwPLS: as described before, the forgetting factor λ is set as 0.97 selected according to cross validation and the VIP threshold α is set as 0.75.

The scatter plot of 3 models can be found in Figure 2.11. In the range of 7%-11% (Measured protein content), there is obvious drift which cannot be removed by PLS.

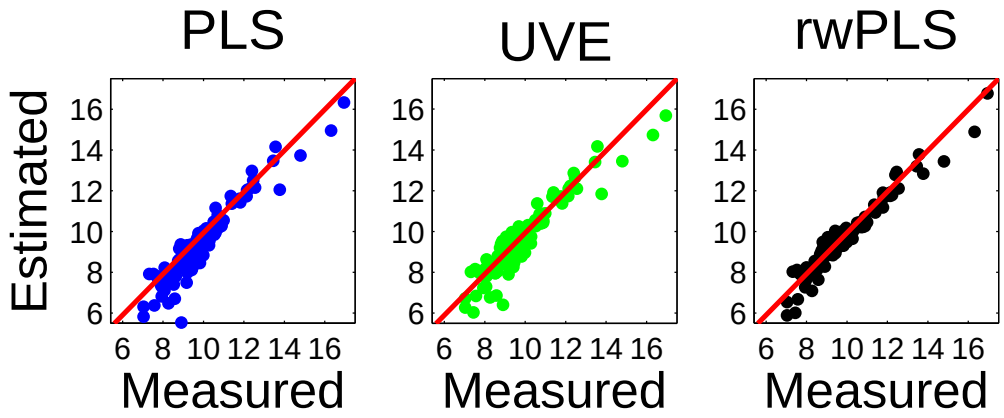

Figure 2.11: Scatter plots of protein content (%) predictions

From Table 2.3, it can be concluded that both wavelength selection and recursive PLS can improve the model accuracy. rPLS is even better than

wavelength selection because the drift caused by the two months' storage can be corrected by learning from the samples in the test dataset. The rwPLS algorithm has the best performance for the reason that it take advantages of both rPLS and wavelength selection.

Table 2.3: Comparison for protein content (%) predictions

Model	RMSEP	R
PLS	0.7807	0.9472
UVE	0.6544	0.9409
iPLS	0.6566	0.9343
rPLS	0.5459	0.9540
rwPLS	0.4583	0.9706

2.5 Conclusions

This study presents a recursive wavelength selection method for updating the model structure (wavelength) as well as model coefficients in NIR application. Based on the exponentially weighted recursive PLS and variance importance in the projection (VIP) wavelength selection, the proposed method is shown to be able to select the model structure in real time according to the process variations. In each updating iteration, the original database is gradually discounted enabling the wavelength selection in accordance with the latest process information. With applications to a real refinery dataset and a benchmark wheat dataset, it is demonstrated that the proposed method gives an improved predictive performance over traditional PLS, locally weighted PLS, bias updated PLS and recursive PLS. Our results also indicate that wavelength updating is necessary for real-time NIR applications.

Chapter 3

Just-in-time modeling using input-output similarity measurement

3.1 Introduction

Soft sensors are widely used in refinery, pharmaceutical, steel and many other processes. Recently, soft sensors have become increasingly popular among process modeling, control and fault detection applications. Hardware analyzers are commonly used in industry but they are usually expensive and difficult to maintain. In the meantime, traditional lab measurements have time delay issues which prevent them from being used in real time control. On the other hand, soft sensors based on mathematical modeling can provide fast rate prediction and require no additional capital cost. So whenever possible, these advantages promote the use of soft sensors for estimating product quality in real time along with hardware analyzers and lab analysis.

Based on the extent of first-principle information that is involved in the modeling, the modeling techniques are divided into three categories: white-

box, black-box and grey-box modeling. White-box models are also called knowledge-driven models since they make full use of the process knowledge. On the other hand, black-box models are also referred to as data-driven models which are developed when the process knowledge is not available or it is too difficult to develop a white-box model. An alternative approach is the grey-box modeling which combines the above two approaches [53]. Compared to white-box models, one of the main advantages of data-driven models is that they can generally be developed quickly without requiring substantial understanding of the phenomenology involved [54]. Moreover, process historical data have become widely available with digital instrumentation in most plants nowadays, which provides a great incentive to use the data-driven methods.

In most of the empirical data-driven modeling approaches, researchers usually utilize the global modeling method, where only one model is built from the historical data and applied to all the future inputs. However, if the process operates in various operating modes, the global modeling approach is not suitable. This problem becomes prominent when the historical data cannot cover the whole process variation ranges or operation patterns. One of the solutions that is widely adopted to solve this problem is the multi-model approach, where several sub models are built beforehand according to different operating regions or data patterns. Once the new query data is available, it can be associated with one of the sub models to generate the estimation. The switching between the sub models is governed by predefined criterion based on the operation modes of scheduling variables. However, the multi-model approach suffers from the drawback of requiring prior knowledge to determine the partition of the operating space [55]. When this information is unavailable, a complex training strategy needs to be resorted in order to determine both optimal model partition and parameters for each local model.

In the case of lacking process or expert knowledge, the Just-in-time (JIT) learning method [2, 56, 54], which is also called 'lazy learning' or 'locally weighted modeling' method, has been proposed in the machine learning literature. It has been considered as an attractive alternative for chemical process modeling, monitoring and soft sensor development. Under the JIT modeling architecture, a local model is built after the query sample is available, using the most relevant historical samples to the query sample. The local model is used to estimate the output value; after which the local model is discarded. Since JIT approach 'creates' a unique model for every query sample, it is different from traditional offline global modeling or multi-model approach. Generally, the JIT method exhibits a local model structure and it is built online using the most relevant data. Thus it is believed that JIT can be used to cope with the process non-linearity and track the abrupt changes of a process [57].

As can be seen in Figure 3.1, JIT approach is more applicable when the process operation modes overlap with each other. Without process knowledge, it is often difficult to identify the range of each pattern, which is a major problem in multi-model approach. However, JIT does not need the prior knowledge, and it simply searches for the relevant 'process mode' based on data-driven methods. This could be an advantage of JIT over regular multi-model approach.

It should be noted that JIT itself is not a regression technique, but it can be considered as a framework under which data-driven modeling techniques can be applied in a more efficient way. For example, partial least squares (PLS) regression can be applied within JIT framework, which results in the well-known method called locally weighted PLS. Also, nonlinear regression methods like artificial neural network (ANN) [24], least square support vector machines (LS-SVM) [25] and other kernel learning methods have also been used within the JIT framework [54].

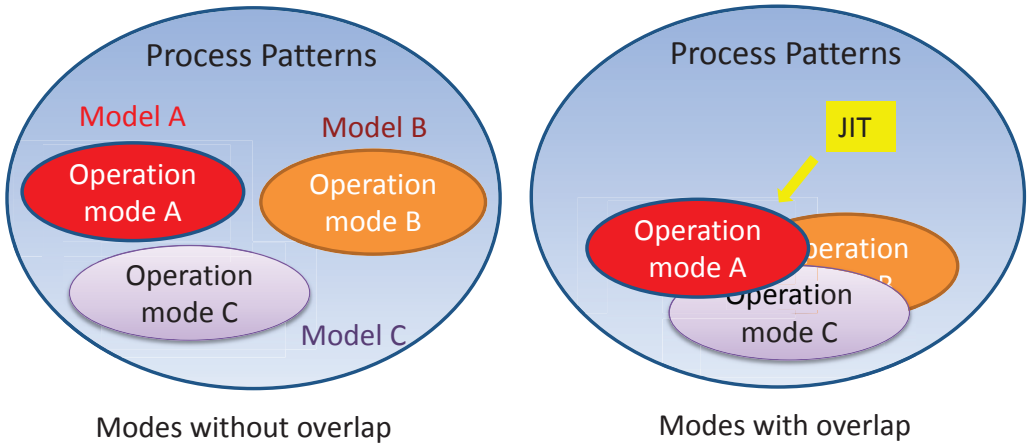

Figure 3.1: Situations for applying multi-model and JIT approach

3.2 Framework of Just-in-time modeling

There are four key components in Just-in-time approach, namely, historical database, similarity measurement, weight function and modeling technique, which are illustrated in Figure 3.2.

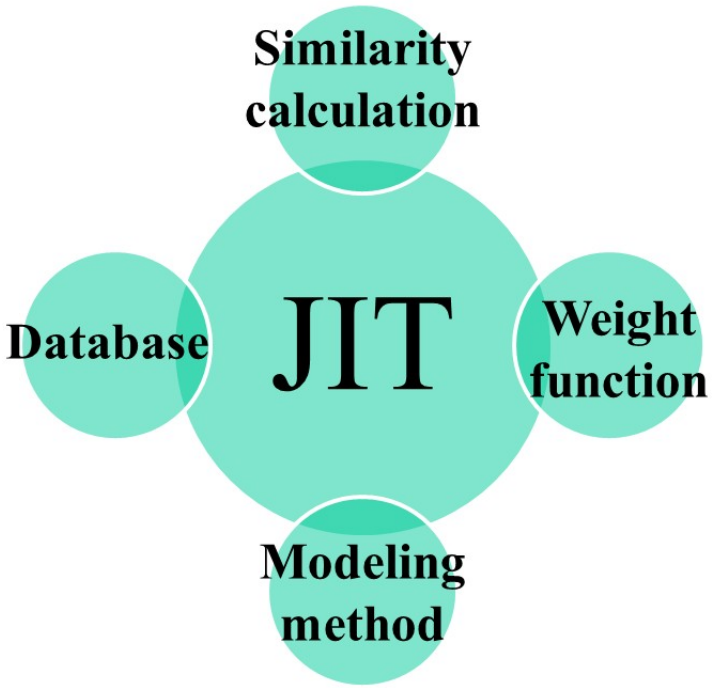

Figure 3.2: The four key componets in JIT modeling

The database consists of the data samples which are used as training data to build local models for the future query samples. The similarity measurement calculates the distances between query sample and the historical data. The similarity quantifies the relevance of the historical data with the current query sample. Thus similarity measurement is important to the success of JIT modeling since it tells which data points should be given priority to build a local model for the current query sample. The weight function is a formula which calculates the weights for each historical data based on their similarity values. A data point in the database which exhibits a high degree of similarity with the query sample is assigned a higher weight. The weight function is a bridge between similarity and modeling technique. After assigning weights to the historical data points, a modeling technique (either linear or nonlinear) is used to generate a local model.

In summary, for a query sample $\mathbf{x}_q$ there are four main steps involved in a JIT model for online prediction:

1. Search for samples relevant to $\mathbf{x}_q$ from the historical database based on the similarity measurement.

2. Calculate the weight for each sample using the weight function.

3. Build a local JIT model $f(\mathbf{x}_q)$ using weights and regression method.

4. Estimate the output y_q online for the current input $\mathbf{x}_q$ and then discard the local model $f(\mathbf{x}_q)$.

The difference between global modeling and JIT modeling can be seen from Figure 3.3 [2].

It is clear that, in JIT approach, the current input (query sample) is involved in the modeling steps, which means that the local model utilizes the information from the query sample. It is quite different from the existing global model approach where the query sample does not affect the model at all.

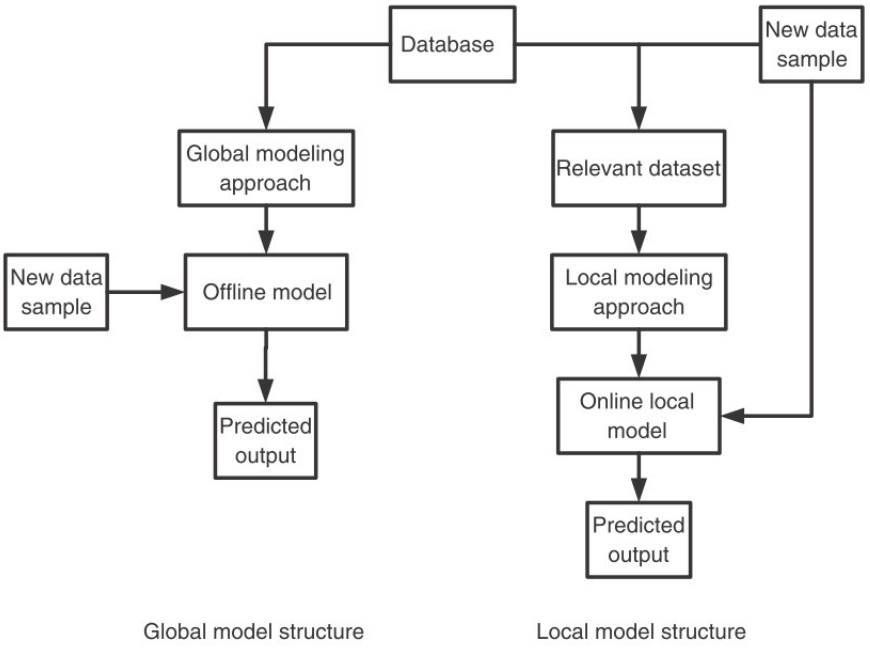

Figure 3.3: A comparison between global and JIT modeling [2]

3.2.1 Database construction

A reliable historical database is not only important in JIT approach, but also important in any other data-driven modeling approaches. Without a good database, it is not possible to build a good JIT model. A thumb of rule in database construction is that the database should cover as wide range of process variations as possible; otherwise extrapolation is needed for the query samples that are located out of the known operation range, which will deteriorate the model performance.

Even if there is a representative database, it would be still questionable when process varies. The following methods describe how the database can be updated so that the new process variations can be learned.

1. Continuous addition of most recent samples: In this method, other than the originally built database, the recently obtained samples which contain the latest information of the process are added into the database continuously. Though it is safe to say that no information is

lost, it requires large memory to store a large amount of data. Also, as the number of data points increases, the computation load becomes heavier and it becomes more difficult to search for the most relevant data.

2. Moving window update of database: In order to avoid heavy computational load, it is desired to keep the database size constant. In moving window approach, the oldest sample is removed from while the most recent sample is added into the database. However, this method is still risky because it may result in losing informative data and adding less informative data. For example, if the process undergoes an abnormal operation for a certain period, the new window would replace the good data with abnormal data, which is detrimental to modeling.

3. Core database approach: This method addresses the shortcomings mentioned in the previous methods. In core database approach, the database consists of two parts: the 'core' part which consists of the most reliable points (typically reflecting time-invariant part of the process) selected from historical database; and the 'updating' part which is updated using the moving window approach. The samples that belong to 'core' part are fixed and always used for modeling. The 'updating' part learns the new process information by moving window approach. It is evident that this approach can effectively update the database while keeping the 'golden' data and database size constant.

3.2.2 Similarity measurements

Similarity measurement plays a key role in JIT modeling. There are many different approaches to define the distance between two samples. In this section, the similarity measurements proposed so far are summarized below and the future directions are discussed.

The most frequently used similarity measurement is the Euclidean distance [58, 59]. The distance between query sample $\mathbf{x}_q$ and the i-th sample $\mathbf{x}_i$ in the database can be calculated according to Equation 3.1:

$$d_{i,Eu} = \sqrt{(\mathbf{x}_i - \mathbf{x}_q)(\mathbf{x}_i - \mathbf{x}_q)^{\mathrm{T}}} \tag{3.1}$$

where $i = 1, 2...N$ represents the index of sample in the database. Usually, a larger distance indicates a less relevance of the data point.

One drawback of Euclidean distance is that it may not assign appropriate weights on the input variables. If the input variables have different units or magnitudes, the distance calculation will be misleading because the variables with larger magnitude may appear to have more contributions to $d_{i,Eu}$. One solution to this problem is to normalize the data before distance calculation, which is also known as the Mahalanobis distance method [53, 60]. The definition of Mahalanobis distance can be found in Equations 3.2 and 3.3.

$$d_{i,Ma} = \sqrt{(\mathbf{x}_i - \mathbf{x}_q)\mathbf{D}(\mathbf{x}_i - \mathbf{x}_q)^{\mathrm{T}}} \tag{3.2}$$

$$\mathbf{D} = h \cdot \begin{bmatrix} \sigma_1^2 & 0 & \cdots & 0 \\ 0 & \sigma_2^2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \sigma_M^2 \end{bmatrix}^{-1} \tag{3.3}$$

where σ_j^2 $(j = 1, 2...M)$ is the variance of each input variable and h is a scaling parameter.

Furthermore, other than using the variance as weight, the similarity based on the weighted Euclidean distance (WED) has been proposed to improve the performance of JIT models [56, 57]. As shown in Equations 3.4 and 3.5, a diagonal matrix $\mathbf{\Theta}$ is used to assign weights for each variable.

$$d_{i,WED} = \sqrt{(\mathbf{x}_i - \mathbf{x}_q)\mathbf{\Theta}(\mathbf{x}_i - \mathbf{x}_q)^{\mathrm{T}}} \tag{3.4}$$

$$\mathbf{\Theta} = diag(\theta_1, \theta_2, \theta_3 ... \theta_M) \tag{3.5}$$

where θ_j $(j = 1, 2...M)$ is the weight for the j-th variable. Shigemori et al. [61] proposed to use the absolute values of regression coefficients as the weight θ_j. The regression coefficients can be obtained from a global model that is built offline. Then the JIT models can be constructed by using the proposed similarity shown in Equations 3.4 and 3.5. The developed method was successfully applied to estimate the product quality of a real steel manufacturing process. Other than using the global model coefficients as weights for each variable, Kim et al. [56] defined θ_j as the absolute value of the j-th input variable's regression coefficient of a locally weighted PLS model for the query input, which was built without weights, i.e. the traditional Euclidean distance was used to generate the regression coefficients. The final JIT models were built using the defined $\mathbf{\Theta}$. Compared with Shigemori's method, this approach is more complicated since it requires the construction of two JIT models in one prediction iteration (one for determining the $\mathbf{\Theta}$ and the other for estimation). Both of these two approaches have been used to improve the performance of near infrared spectra (NIR) model for estimating active pharmaceutical ingredient (API) content in tablets.

In addition to distance calculation, the angular relationship between samples was also considered as an important factor to determine similarity [3]. The details can be found in Equation 3.6 to 3.9.

$$d_i = \|\mathbf{x}_i - \mathbf{x}_q\|_2 \tag{3.6}$$

$$\Delta\mathbf{x}_q = \mathbf{x}_q - \mathbf{x}_{q-1}, \quad \Delta\mathbf{x}_i = \mathbf{x}_i - \mathbf{x}_{i-1} \tag{3.7}$$

$$\cos\varphi_i = \frac{\Delta\mathbf{x}_q \Delta\mathbf{x}_i^{\mathrm{T}}}{\|\Delta\mathbf{x}_q\|_2 \cdot \|\Delta\mathbf{x}_i\|_2} \tag{3.8}$$

$$s_i = \begin{cases} \lambda\sqrt{e^{-d_i^2}} + (1-\lambda)\cos\varphi_i & \text{if } \cos\varphi_i \geq 0 \\ 0 & \text{if } \cos\varphi_i < 0 \end{cases} \tag{3.9}$$

where $\lambda \in [0, 1]$ is a balance parameter between distance and angle. The cosine function is used to discriminate the directionality between $\Delta\mathbf{x}_q$ and $\Delta\mathbf{x}_i$. When $\cos\varphi_i < 0$, the corresponding data point $\mathbf{x}_i$ in the database will not be involved in the subsequent JIT modeling procedure. For example, in the three-dimensional space shown in Figure 3.4 [3], although the distances of $\mathbf{x}_i$ and $\mathbf{x}_j$ from $\mathbf{x}_q$ might be the same, $\mathbf{x}_i$ is still more similar to $\mathbf{x}_q$ because they are in a similar direction. Consequently, the corresponding sample pair $(\mathbf{x}_j, y_j)$ in the database will not be involved in the subsequent JIT modeling procedure.

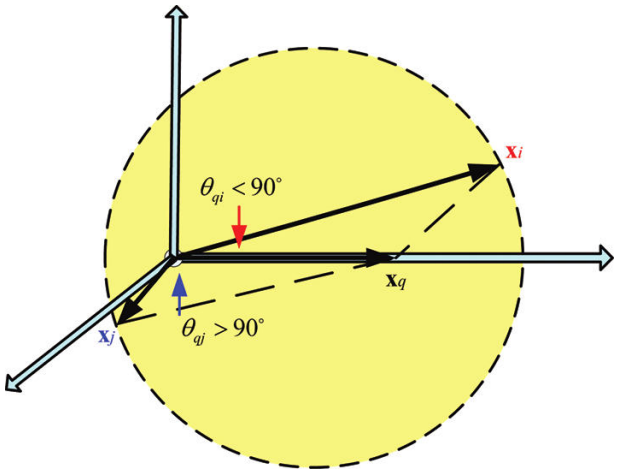

Figure 3.4: Similarity considering both distance and angle in a 3-dimensional space [3]

Fujiwara et al. [62, 63] recently proposed another correlation-based similarity criteria (CoJIT). Several sub data sets were partitioned from the original database beforehand, and the similarity index J for each sub data set was calculated using Hotelling's T^2 and Q statistics as in Equation 3.10.

$$J = \lambda T^2 + (1 - \lambda)Q \tag{3.10}$$

Then a local model was built using the sub data set whose J is the minimum. The CoJIT method was applied to a cracked gasoline fractionator and had better performance than recursive PLS. However, this approach

requires prior process knowledge to divide the database into several partitions. Based on the methods discussed above, there are also many modified versions of similarity measurements. For example, the principle component in PCA analysis was proposed to calculate the similarity instead of the original variables [64, 65, 66]. Kim et al. [67] modified the weighted Euclidean distance [56] by adaptively determining the similarity according to the strength of the non-linearity between each input variable and the output variable around a query sample.

All the above-mentioned methods only employ the input information to perform similarity calculation. However, the accuracy of JIT models might be enhanced by incorporating the information from output space. Wang et al. [55] proposed a similarity measurement which is based not only on the distance $d_{i,x}$ in the input space but also the distance $d_{i,y}$ in the output space, which can be seen in Equation 3.11 to 3.12:

$$s_i = \lambda d_{i,x} + (1 - \lambda) d_{i,y} \tag{3.11}$$

$$d_{i,y} = \frac{|y_i - \hat{y}_q|}{\sum_{i=1}^{N} |y_i - \hat{y}_q|} \tag{3.12}$$

where $d_{i,x}$ is the Mahalanobis distance in the input space and y_i is the output of i-th sample in the database. $\hat{y}_q$ is the estimated output of the query sample obtained by an initial global model. λ is a tuning parameter for balancing purpose. It was shown in [55] that this method achieved better performance with fewer latent variables and was less sensitive to noise. Chang et al. [65] also proposed a different method to calculate $d_{i,y}$, which can be seen in Equation 3.13.

$$d_{i,y} = \frac{|y_i - \hat{y}_q|}{max(|y_i - \hat{y}_q|)} \tag{3.13}$$

Note that, for all the methods discussed above to determine similarity

of query sample with the database samples, one cannot single out the 'best' method. The choice of one of these methods is based on the application.

3.2.3 Weight functions

In order to assign priority to the most relevant samples in the database, the weight of each sample is calculated from its similarity measurement. The weight w_i is a function of similarity s_i with the general rule that higher weight is assigned to more similar sample. The maximum value of the weight should be given to the data point with zero distance. Another desired property of the weight function is the smoothness; which means that the weight function should decay smoothly as the the similarity decreases. A smooth weight function would give a smooth estimation result. Depending on how quickly the weight drops with the increase of distance, there are several types of weight functions, which will be reviewed in the following section according to [58].

The simplest way to assign weights is to ignore some parts of the data, namely, the least relevant data, while the remaining parts of data are treated equally in the modeling step. Usually, in other weight assigning methods the weight becomes zero as the distance goes infinity. However, in this method, the weight can reach zero at a finite distance. One special question here is that how many data points or how much percentage of the data should be discarded. It may be determined through cross validation methods, where different numbers of samples are used and the one which gives the lowest prediction error is chosen as the optimum number. This type of weight function allows faster implementations or calculations, since data points that are too far away from the query can be ignored.

Another weight function can be defined as the negative power of the distance (Shepard 1968), where the power magnitude p determines drop-off

rate of the weight with distance, as can be seen in Equation 3.14.

$$w_i = \frac{1}{d_i^p} \tag{3.14}$$

where p is a positive integer and d_i is the distance between query sample and the i-th sample in the database. As the query point approaches a stored data point, the weight goes infinity.

Also, the inverse distance (Wolberg 1990) with limited magnitude was proposed to avoid over-interpolation due to data noise. The function is shown in Equation 3.15.

$$w_i = \frac{1}{1 + d_i^p} \tag{3.15}$$

Another smoothing weight function is Gaussian kernel (Deheuvels 1977) as shown in Equation 3.16, which is widely used in soft sensor applications.

$$w_i = e^{-d_i^2} \tag{3.16}$$

A related weight function is the exponential kernel, which has been used in psychological models (Aha and Goldstone 1992), which can be seen in Equation 3.17.

$$w_i = e^{-|d_i|} \tag{3.17}$$

The quadratic function (Epanechnikov 1969) was also proposed in the literature. Unlike Gaussian kernel and exponential kernel which has infinite extent, this type of weight function has finite extent and ignores the data points with distance larger than the radius of 1. However, due to the discontinuity in the derivative at $d = 1$, the function has been less frequently used in applications that need analytical treatments. The function can be

written in Equation 3.18.

$$w_i = \begin{cases} 1 - d_i^2 & \text{if } \mid d_i \mid < 1 \\ 0 & \text{otherwise} \end{cases} \tag{3.18}$$

When using this type of function, the distance d_i should be normalized first. A good way of normalization is as following:

$$d_{i,Nor} = \frac{d_i - d_{min}}{d_{max} - d_{threshold}} \tag{3.19}$$

where d_{min} and d_{max} are the minimal and maximal distance of the stored data points from query sample, respectively, and $d_{threshold}$ is a truncation value to ignore data points that are further from a particular threshold distance. If $d_{threshold} = d_{min}$, all the distances in Equation 3.18 will be smaller than 1; thus all the points will be used in modeling. However, if it is set as a value between d_{min} and d_{max}, part of the data points that have large distances will be truncated and ignored.

The tricube function was used by Cleveland (1979), and later widely applied to locally weighted regression for near infrared spectroscopy. Similar to quadratic function, it has finite extent and the distance has to be normalized . The function is written as below:

$$w_i = \begin{cases} (1 - \mid d_i \mid^3)^3 & \text{if } \mid d_i \mid < 1 \\ 0 & \text{otherwise} \end{cases} \tag{3.20}$$

The triangular function (Stone 1977) and its variant (Franke and Nielson 1980) were also proposed due to simplicity, which are shown in Equations 3.21 and 3.22.

$$w_i = \begin{cases} 1 - \mid d_i \mid & \text{if } \mid d_i \mid < 1 \\ 0 & \text{otherwise} \end{cases} \tag{3.21}$$

$$w_i = \begin{cases} \frac{1-|d_i|}{|d_i|} & \text{if } \mid d_i \mid < 1 \\ 0 & \text{otherwise} \end{cases} \tag{3.22}$$

The trends of these weight functions with the increase of distance are plotted in Figure 3.5. It is clear that some weight functions such as truncation and negative power functions have large drop-off as distance increases . Some have finite extent (e.g., tricube function) while others have infinite extent (e.g., negative power function). Some functions put zero weights to the far away data points, which implies that these data do not participate in the modeling. Different functions may be appropriate to different data sets.

One could create new weight functions by simply changing the power of the functions. The power can be either integral or non-integral. However, as pointed out by [58, 59], there is no clear evidence that the selection of weight function plays an important role in the performance of JIT models. Instead, it seems that it is the similarity rather than weight function that is crucial to the success of JIT modeling. So in the following case study, different similarity measurements are compared and the same weight function is adopted.

3.2.4 Modeling techniques

As discussed in the previous section, JIT approach is not a regression technique but a framework to improve the regular regression performance. In this subsection, we discuss various ways of combining the weights and regression techniques in order to achieve better prediction performance.

The Comparison Analysis Using Restructured Near Infrared and Constituent Data (CARNAC) method proposed by Davies et al. [68] is a rapid and simple approximation of local regression. The method predicts sample properties (outputs) directly through the reference data stored in the

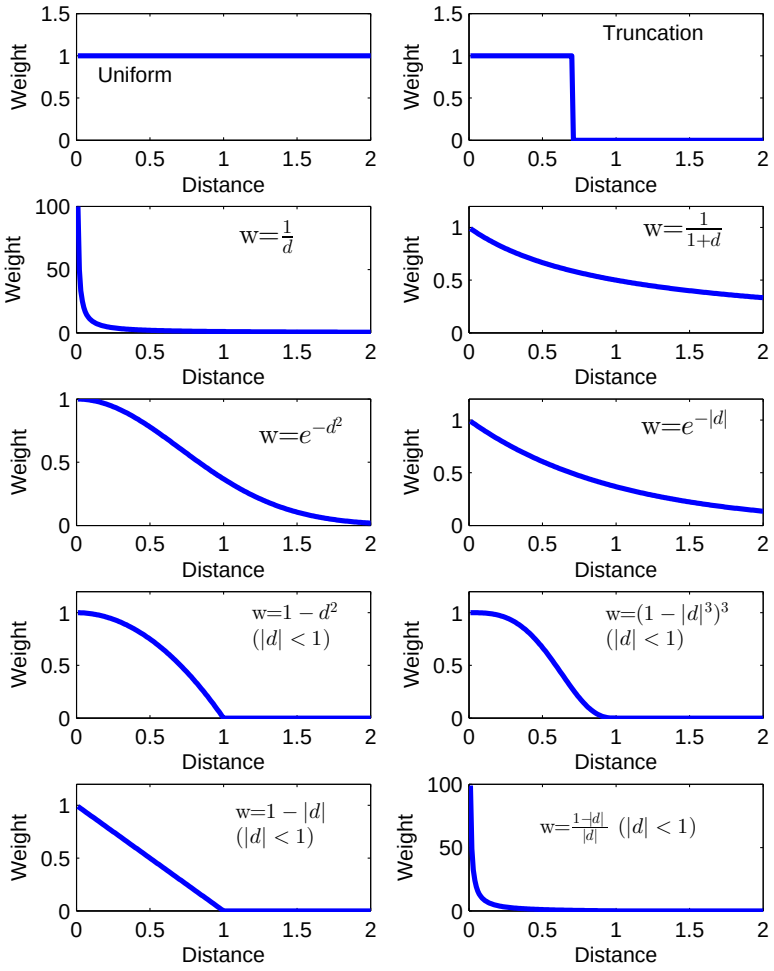

Figure 3.5: A comparison of different weight functions

database, rather than through regression model derived from historical data. The predicted value for the unknown sample is obtained by weighted average of the reference values of similar subset, as shown in Equation 3.23.

$$\hat{y}_q = \frac{\sum_{i=1}^{N} s_i y_i}{\sum_{i=1}^{N} s_i} \tag{3.23}$$

Similar samples are selected through similarity calculations as shown in

Equation 3.24.

$$s_i = \frac{1}{1 - r_i^2} \tag{3.24}$$

where r_i is the correlation coefficient between the query sample and a member in the database. Another rapid local regression algorithm is the Locally-Biased Regression (LBR) [69]. For each query sample, the preliminary output is generated from a global model, then the bias and skew terms are calculated from neighboring samples in order to correct the global model.

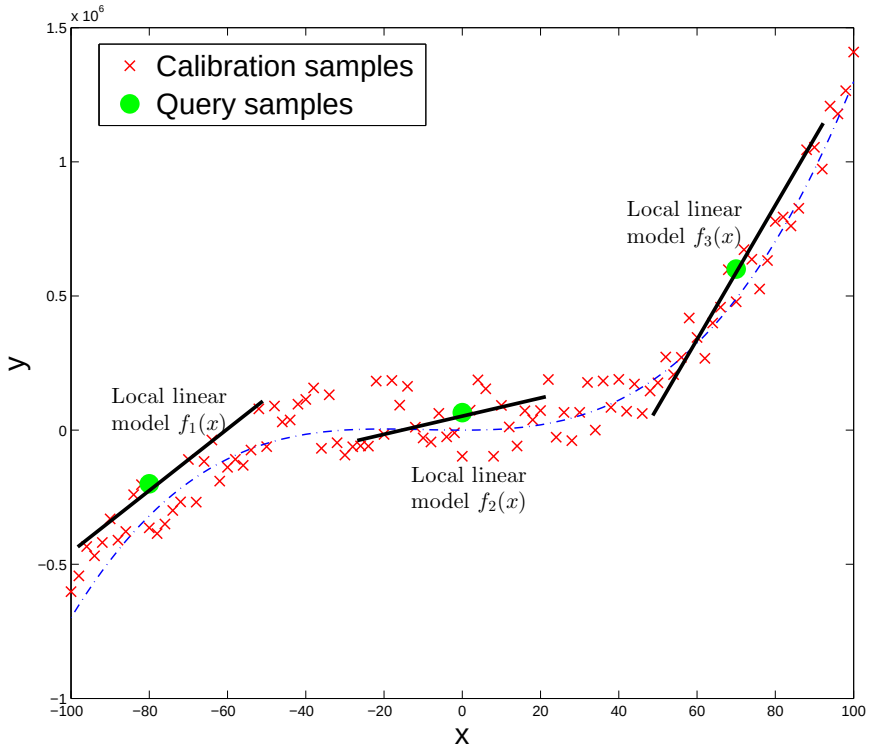

Figure 3.6: Piecewise linear approximation of a nonlinear function

There are no restrictions on what kind of model structure should be used when building local models. However, models that are linear in the unknown parameters, such as local polynomials, can be trained faster than more general models. Nonlinear modeling techniques such as Least Squares Support Vector Regression (LSSVR) [25] and Support Vector Regression (SVR) [70]

can also be fused into JIT framework. However, nonlinear methods often have much more computational load, which is especially not suitable for online model training. For linear models, the major component of training cost is the lookup procedure. In the meantime, the past research indicates that most regression surfaces can be fitted locally using linear models [64], which makes more sense to the popularity and practical applicability of linear model structure.

The most straightforward linear local algorithm is Locally Weighted Regression (LWR) [71]. The key concept of this method is to approximate nonlinear functions by piecewise linear models, which can be seen from Figure 3.6. The following weighted regression analysis is performed for every query sample:

$$\mathbf{W} = \begin{bmatrix} w_1 & 0 & \cdots & 0 \\ 0 & w_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & w_N \end{bmatrix} \tag{3.25}$$

$$\boldsymbol{\beta}_{LWR} = (\mathbf{X}\mathbf{W}\mathbf{X}^T)^{-1}\mathbf{X}\mathbf{W}\mathbf{y} \tag{3.26}$$

Another widely used JIT algorithm is the Locally Weighted Partial Least Squares (LW-PLS) [71]. There are two reasons for switching to LW-PLS from LWR: 1) the input dimension is too large; 2) the input variables are correlated. Under both situations, the matrix inversion in Equation 3.26 will become numerically unstable. So the PLS algorithm is synthesized with Just-in-time framework in order to compress the input variables into a projection space with much lower dimension. The following steps describe the details of LW-PLS.

1. When query sample arrives, calculate w_i and weight matrix $\mathbf{W}$.

2. Set $k = 1$ and determine the number of latent variables as A.

3. Mean center the data such that

$$\overline{\mathbf{x}} = \frac{\sum_{i=1}^{N} w_i \mathbf{x}_i}{\sum_{i=1}^{N} w_i} \tag{3.27}$$

$$\overline{y} = \frac{\sum_{i=1}^{N} w_i y_i}{\sum_{i=1}^{N} w_i} \tag{3.28}$$

$$\tilde{\mathbf{X}} = \mathbf{X} - \mathbf{1}_N \overline{\mathbf{x}} \tag{3.29}$$

$$\tilde{\mathbf{y}} = \mathbf{y} - \mathbf{1}_N \overline{y} \tag{3.30}$$

$$\tilde{\mathbf{x}}_q = \mathbf{x}_q - \overline{\mathbf{x}} \tag{3.31}$$

where $\mathbf{1}_N$ is a vector of ones.

4. Initialize the prediction $\hat{y}_q = \overline{y}$.

5. Calculate the weight vector, loading vector and score vector as

$$\mathbf{u}_k = \tilde{\mathbf{X}}^T \mathbf{W} \tilde{\mathbf{y}} \tag{3.32}$$

$$\mathbf{t}_k = \tilde{\mathbf{X}} \mathbf{u}_k \tag{3.33}$$

$$\mathbf{p}_k = \frac{\mathbf{t}_k^T \mathbf{W} \tilde{\mathbf{X}}}{\mathbf{t}_k^T \mathbf{W} \mathbf{t}_k} \tag{3.34}$$

$$q_k = \frac{\mathbf{t}_k^T \mathbf{W} \tilde{\mathbf{y}}}{\mathbf{t}_k^T \mathbf{W} \mathbf{t}_k} \tag{3.35}$$

$$t_k = \tilde{\mathbf{x}}_q \mathbf{u}_k \tag{3.36}$$

6. Deflate the matrix as

$$\tilde{\mathbf{X}} = \tilde{\mathbf{X}} - \mathbf{t}_k \mathbf{p}_k \tag{3.37}$$

$$\tilde{\mathbf{y}} = \tilde{\mathbf{y}} - \mathbf{t}_k q_k \tag{3.38}$$

$$\tilde{\mathbf{x}}_q = \tilde{\mathbf{x}}_q - t_k \mathbf{p}_k \tag{3.39}$$

7. Make the prediction: $\hat{y}_q = \hat{y}_q + t_k q_k$.

8. If $k = A$, terminate the algorithm. Otherwise, set $k = k + 1$ and return to Step 5.

Once the weights are calculated, the only tuning parameter in LW-PLS is the number of latent variables, which can be optimized by offline cross validation. Actually, the most important factor in LW-PLS is how to define the similarity, which distinguishes the modeling performance of LW-PLS from regular PLS. One advantage of LW-PLS and LWR is that the weights can be integrated into the regression steps; however, for nonlinear methods such as support vector machine (SVM) and least square support vector machine (LSSVM), JIT approach only provides a way to select the most relevant subset, and the regression step itself is not changed by nature [2]. Therefore, the details of nonlinear regression based JIT modeling are not discussed here.

3.3 Similarity calculation using both input and output information

Most of the existing similarity measurements only rely on input information. In these methods, although the selected relevant samples are close to the query sample in the input space, they may not necessarily be close in the space of the dependent variable (output y) [55]. Under this circumstance, Equation 3.11-3.13 have shown one of the possible solutions to involve output space information when calculating similarity. The similarity is calculated as a weighted average of the distances in both input and output spaces. However, this approach highly depends on the accuracy of the initial global model. In this section, two alternative approaches are attempted for the use of the output information when calculating the distance.

The previous research [64, 65, 66] utilized the first few principle components (PCs) of Principal Component Analysis (PCA) to calculate distance since it is less sensitive to noise. Based on this idea, we can extend the information contained in PCs by applying partial least squares (PLS). An alternative term for PLS is Projection to Latent Structures. PLS finds a linear regression model by projecting the input and output variables into a new space. The projections are chosen according to the correlation of input and output data. A PLS model tries to find the multi-dimensional direction in the input space that explains the maximum multi-dimensional variation direction in the output space. In the projection space, the original input variables can be represented by a few latent variables (LVs). Instead of original variables, the first few LVs can be used for calculating distance. The distance calculation is shown as in Equation 3.40.

$$d_i = \sum_{k=1}^{A} (t_{k,i} - t_{k,q})^2 \tag{3.40}$$

where $t_{k,i}$ is one entry of $\mathbf{t}_k$ which represents the k-th score value of the i-th sample. The corresponding score values of the query sample can be calculated from Equation 3.41.

$$t_{k,q} = \mathbf{x}_q \mathbf{u}_k \tag{3.41}$$

The latent variable $\mathbf{t}_k$ and weight vector $\mathbf{u}_k$ in the k-th projection loop can be obtained through the standard PLS algorithm. There are two advantages of this distance calculation: 1) it reduces noise effect through data compression; 2) the output information is integrated into the latent variables. Since the latent variables preserve the direction information of the output, the proposed distance measurement would make more sense in terms of searching for data points that are close to $\mathbf{x}_q$ in the y-space.

The second method which involves y-space information for similarity measurement is based on Orthogonal Signal Correction (OSC). The OSC algorithm is a filter that is used to remove the variations in $\mathbf{X}$-space which are linearly unrelated with the target variable $\mathbf{y}$ [72]. By subtracting the variations that are orthogonal to $\mathbf{y}$, the nuisance information in $\mathbf{X}$ is eliminated. Usually, a typical OSC filter consists of three parts: weight, score and loading vectors ($\mathbf{u}_k$, $\mathbf{t}_k$, $\mathbf{p}_k$, $k = 1...A$) which are selected based on the following optimization problems [72].

$$\mathbf{u}_k = \underset{\substack{\|\mathbf{u}\|=1 \\ (\mathbf{X}_k\mathbf{u})^T\mathbf{y}=0}}{\arg\max}\ \mathbf{u}^T\mathbf{X}_k^T\mathbf{X}_k\mathbf{u} \tag{3.42}$$

$$\mathbf{t}_k = \underset{\mathbf{t}}{\arg\min}\, \|\mathbf{X}_k - \mathbf{t}\mathbf{u}_k^T\|^2 = \mathbf{X}_k\mathbf{u}_k/(\mathbf{u}_k^T\mathbf{u}_k) \tag{3.43}$$

$$\mathbf{p}_k = \underset{\mathbf{p}}{\arg\min}\, \|\mathbf{X}_k - \mathbf{t}_k\mathbf{p}^T\|^2 = \mathbf{X}_k\mathbf{t}_k/(\mathbf{t}_k^T\mathbf{t}_k) \tag{3.44}$$

where $\mathbf{X}_k$ is the input matrix after k-th deflation.

Originally, OSC filter was considered as a preprocessing method to remove the systematic orthogonal variation by simply subtracting the first one or two OSC components (score vectors) from the raw data set. However, the OSC provides a useful way to summarize the output space information that cannot be captured by the traditional similarity measurement. It would be beneficial to filter the data before calculating the distance so that the similarity measurement is forced to be conducted in a space which has the maximum relation to the target variable $\mathbf{y}$. The general idea is shown in Figure 3.7.

Wold et al. [73] used nonlinear iterative partial least squares NIPALS methodology to compute the score vectors and loading vectors listed above. Fearn [74] proposed a modified OSC algorithm that removed the unrelated information based on PCA. In the presented study, the method of the orthogonal projections to latent structures (O-PLS) proposed by Trygg and

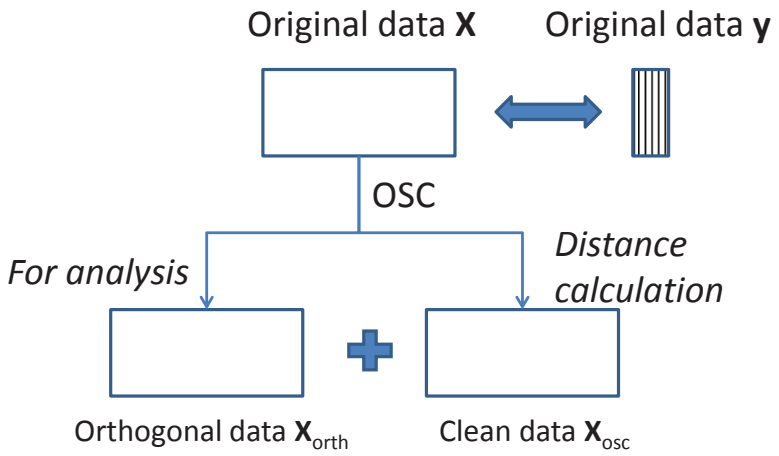

Figure 3.7: Utilizing y-space infromation for similarity measurement

Wold [75] is adopted for similarity calculation. The O-PLS is the latest algorithm which belongs to the OSC filter group and it has several advantages: 1) it can analyze the disturbing variation in each regular PLS component; 2) the orthogonal factors can be analyzed separately; 3) time-consuming internal iteration is avoided. The following steps explain the details of O-PLS algorithm. For simplicity, the data set is assumed to be centered and scaled beforehand, and there is only one variable in $\mathbf{y}$.

1. Perform NIPALS PLS calculation to obtain weight vector $\mathbf{u}$

$$\mathbf{u} = \mathbf{X}^T\mathbf{y}/(\mathbf{y}^T\mathbf{y}) \tag{3.45}$$

$$\mathbf{u} = \mathbf{u}/\|\mathbf{u}\| \tag{3.46}$$

2. Obtain the loading vector $\mathbf{p}$

$$\mathbf{t} = \mathbf{X}\mathbf{u}/(\mathbf{u}^T\mathbf{u}) \tag{3.47}$$

$$\mathbf{p} = \mathbf{X}^T\mathbf{t}/(\mathbf{t}^T\mathbf{t}) \tag{3.48}$$

3. Calculate the orthogonal factors $\mathbf{u}_\perp$, $\mathbf{t}_\perp$ and $\mathbf{p}_\perp$

$$\mathbf{u}_\perp = \mathbf{p} - [\mathbf{u}^T\mathbf{p}/(\mathbf{u}^T\mathbf{u})]\mathbf{u} \tag{3.49}$$

$$\mathbf{u}_\perp = \mathbf{u}_\perp/\|\mathbf{u}_\perp\| \tag{3.50}$$

$$\mathbf{t}_\perp = \mathbf{X}\mathbf{u}_\perp/(\mathbf{u}_\perp^T\mathbf{u}_\perp) \tag{3.51}$$

$$\mathbf{p}_\perp = \mathbf{X}^T\mathbf{t}_\perp/(\mathbf{t}_\perp^T\mathbf{t}_\perp) \tag{3.52}$$

4. Remove the orthogonal information from the training set $\mathbf{X}$

$$\mathbf{X}_o = \mathbf{X} - \mathbf{t}_\perp\mathbf{p}_\perp^T \tag{3.53}$$

5. Remove the orthogonal information from the query sample $\mathbf{x}_q$

$$t_{q,\perp} = \mathbf{x}_q\mathbf{u}_\perp/(\mathbf{u}_\perp^T\mathbf{u}_\perp) \tag{3.54}$$

$$\mathbf{x}_{q,o} = \mathbf{x}_q - t_{q,\perp}\mathbf{p}_\perp^T \tag{3.55}$$

6. Save the orthogonal factors as $\mathbf{U}_\perp = [\mathbf{U}_\perp \quad \mathbf{u}_\perp]$, $\mathbf{P}_\perp = [\mathbf{P}_\perp \quad \mathbf{p}_\perp]$, $\mathbf{T}_\perp = [\mathbf{T}_\perp \quad \mathbf{t}_\perp]$ and $\mathbf{t}_{q,\perp} = [\mathbf{t}_{q,\perp} \quad t_{q,\perp}]$

7. To remove additional orthogonal factors, set $\mathbf{x}_q = \mathbf{x}_{q,o}$ and $\mathbf{X} = \mathbf{X}_o$, then repeat Step 2-6

8. The orthogonal information can be summarized as follows

$$\mathbf{X}_\perp = \mathbf{T}_\perp\mathbf{P}_\perp^T \tag{3.56}$$

$$\mathbf{x}_{q,\perp} = \mathbf{t}_{q,\perp}\mathbf{P}_\perp^T \tag{3.57}$$

In this study, we propose to calculate the distance using the data filtered by OSC algorithm. So $\mathbf{X}_o$ and $\mathbf{x}_{q,o}$ are used to calculate the distance. Since

the output space information is integrated in Equation 3.45, the output will certainly affect the distance calculation. The orthogonal variations in the training data set can be removed offline, which reduces the online computation burden. Thus, the calculation in Equations 3.54 and 3.55 is performed online for the new query in every prediction instance. In order to analyze the orthogonal information extracted from the raw data set, one can perform PCA on $\mathbf{X}_{\perp}$ in Equation 3.56. In locally biased regression, Fearn [69] proposed to use the extracted orthogonal vectors to search for the similar data points. This idea is appropriate in terms of identifying the samples that have similar bias term; however, it may not be useful if one wants to identify the nearby samples in the output space as proposed in this thesis.

3.4 Application results

In order to evaluate the algorithms proposed above, an industrial near-infrared spectroscopy data set is used in this section. Several different similarity measurements are compared under the same JIT framework, which gives an insight into the advantage of utilizing output information for similarity measurement.

3.4.1 Benchmark NIR data for pharmacy tablets

The spectra data of Escitalopram tablets are measured by the pharmaceutical industry, which is a benchmark public data set for multivariate data analysis. The spectra as well as the reference values can be downloaded from *Department of Food Science, University of Copenhagen*. There are in total 310 tablet samples distinguished by four different dosage types. Each type contains tablet samples from 3 different scales of batch processes (full scale, pilot scale and laboratory scale). All raw materials were prepared into

pure powder form for spectral analysis. Since the samples come from different batch processes and the concentration of active substance has a wide range, the spectral data can be clustered into various regions in both input and output spaces. These factors imply that Just-in-time is an appropriate method for this set of data. The near-infrared transmittance spectra were measured using ABB Bomem FT-NIR model MB-160 with a spectral range of 4000-14000cm^{-1}. Each spectrum was the average of 128 scans in order to provide sufficient stability and reproducibility. There are 404 wavelengths in each spectrum with a resolution of 16cm^{-1}. The target variable (the content of active substance in each tablet % w/w) was measured by high performance liquid chromatography (HPLC). For details of the experiments and reference method, one can refer to [76].

The raw spectra of 310 samples are displayed in Figure 3.8. The training data set consists of 124 tablets with reference value ranges from 4.84% to 9.79%, while the validation set consists of 186 tablets with reference value ranges from 4.61% to 9.38%. The PCA score plot for all the 310 samples is

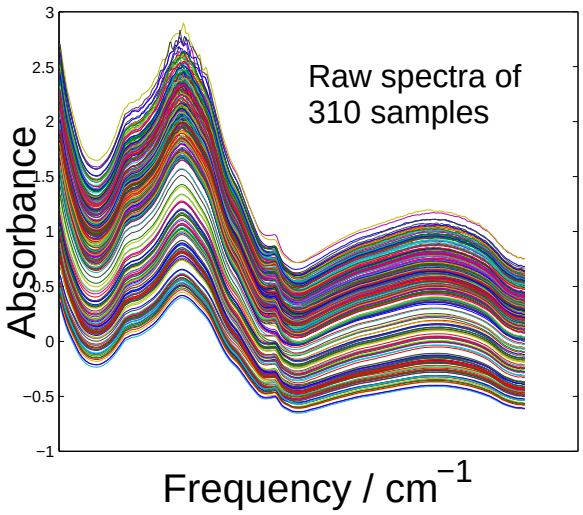

Figure 3.8: Raw spectra of pharmacy tablets

shown in Figure 3.9. Type A, B, C and D represent different concentration levels, i.e. different ranges in the output space. In the 2-dimensional pro-

jection space, the samples that belong to different types have been marked by different colors. Although Type A, C and D should have been separated into different clusters due to their different concentration levels (y information), it is observed that the samples from these three types are overlapping with each other in the principle component space (i.e. a subspace of X-space). This is because PCA only utilizes the input information for data compression. So similarity calculated from only input information would probably treat a sample close in the $\mathbf{X}$-space but far away in the y-space as relevant data, which can have negative effect on the model accuracy.

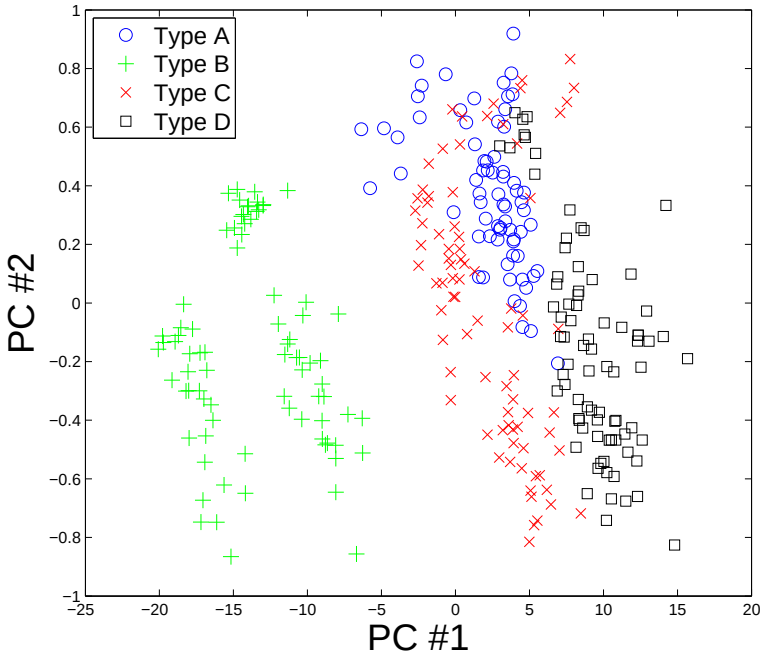

Figure 3.9: PCA score plot for tablet samples

3.4.2 Results and discussion

In order to make fair comparison between different similarity measurements, all the following methods except for regular PLS are put under the same

JIT framework, using the same weight function as shown in Equation 3.58.

$$w_i = \exp(-\frac{d_i\phi}{\sigma_{d_i}}) \tag{3.58}$$

where ϕ is a tuning parameter for optimization and σ_{d_i} is the standard deviation of $d_i(i = 1, 2...N)$. So the only difference between the following JIT methods is the definition of similarity. Five methods are summarized as follows:

1. PLS: Global PLS approach.

2. JIT: JIT approach using Euclidean distance (Equation 3.1).

3. JIT-Y1: JIT approach using estimated y-value to calculate distance (Equations 3.11 and 3.12).

4. JIT-Y2: JIT approach using PLS latent variables to calculate distance (Equations 3.40 and 3.41). The number of latent variables (LVs) for distance calculation is set the same as that used for local modeling.

5. JIT-Y3: JIT approach using OSC filtered data to calculate distance.

JIT-Y2 and JIT-Y3 are the methods proposed in Section 3.3. Other than the tuning parameter ϕ in weight function and the number of LVs for local models, some methods may have additional tuning parameters. Such as the balancing parameter in Equation 3.11 and the number of OSC factors in JIT-Y3. All the optimal values for these parameters are determined by offline cross validation. In this study, the optimal tuning parameters for these five methods are listed in Table 3.1. In the meantime, the Root Mean Square Errors of Prediction ($RMSEP$) and correlation coefficient R for each method are shown in Table 3.1.

Table 3.1: Tuning parameters and prediction results

Methods	LVs	ϕ	λ	OSC factors	$RMSEP$	R
PLS	7	-	-	-	0.5188	0.9334
JIT	5	0.85	-	-	0.4452	0.9464
JIT-Y1	4	0.51	0.6	-	0.4106	0.9495
JIT-Y2	4	0.4	-	-	0.4076	0.9503
JIT-Y3	4	1.1	-	3	0.3499	0.9635

From Table 3.1, it is interesting to see that all the JIT methods have used fewer LVs because of their local modeling strategy. The additional LVs used in global PLS may be helpful to account for the non-linearity in the data. Also, all the JIT methods achieved lower $RMSEP$ and higher R than global PLS, which shows the advantage of locally weighted PLS. The JIT methods which utilize both input and output information for similarity calculation have better performances than traditional JIT; among which JIT-Y3 has the best predictive ability while JIT-Y2 is just comparable with JIT-Y1. To visualize the results, a comparison between PLS, JIT and JIT-Y3 is plotted in Figure 3.10. From the scatter plots, one can easily see that the reference value (y-value) can be classified into several groups, and each group has its certain range in the y-space. This is the reason why the output space information is important in similarity measurement for this set of data. By integrating both the input and output information, more weights can be assigned to the data points that belong to the same group in the y-space without losing the important input information. Comparing the performance of PLS and JIT approach, it can be seen that some estimated points that are far from the reference values can be corrected by JIT method. Moreover, JIT-Y3 has shown even better results in terms of correcting those 'bad' points, especially the ones marked by the green circles in Figure 3.10. This evidence clearly shows the advantage of JIT-Y3 over

regular JIT method.

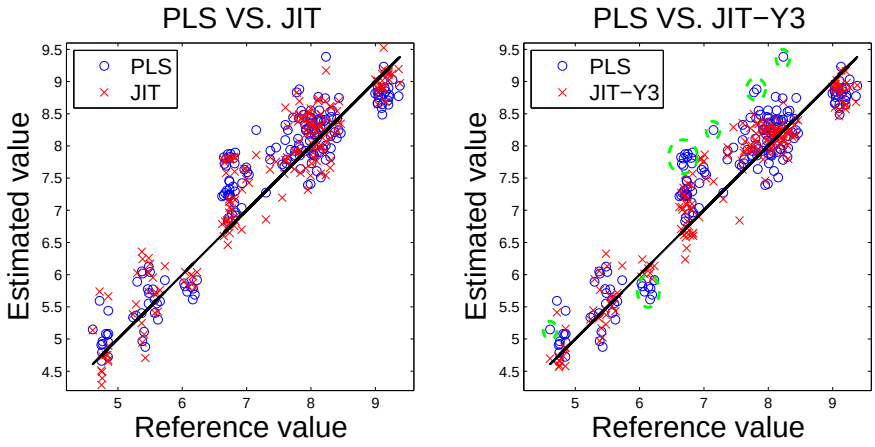

Figure 3.10: Scatter plot of JIT modeling results

Table 3.2: Tuning parameters and prediction results (Extrapolation case)

Methods	LVs	ϕ	λ	OSC factors	$RMSEP$	R
PLS	8	-	-	-	0.8267	0.7568
JIT	7	0.4	-	-	0.5444	0.7514
JIT-Y1	5	0.95	0.9	-	0.4804	0.8642
JIT-Y2	5	0.67	-	-	0.4978	0.8004
JIT-Y3	4	0.15	-	3	0.4590	0.8422

To further evaluate the usefulness of JIT approach as well as the modified JIT approaches, another simulation study has been conducted on the same NIR data set. In order to evaluate the algorithm under the extrapolation case, the training data set is chosen such that their output values are limited in the range of 4.61% to 8.71%, while the output values of the validation data set are chosen to be within the range of 6.62% to 9.79%. The division of data set clearly shows that, the training set is unable to cover the whole variation range of the validation set, which means that part of the estimation results can only be made by extrapolation. It is valuable to do so since the situation is commonly seen in real applications. Again, all the tuning parameters

have been determined by offline cross validation and the prediction results are listed in Table 3.2. From Table 3.2, it is clear that the global PLS

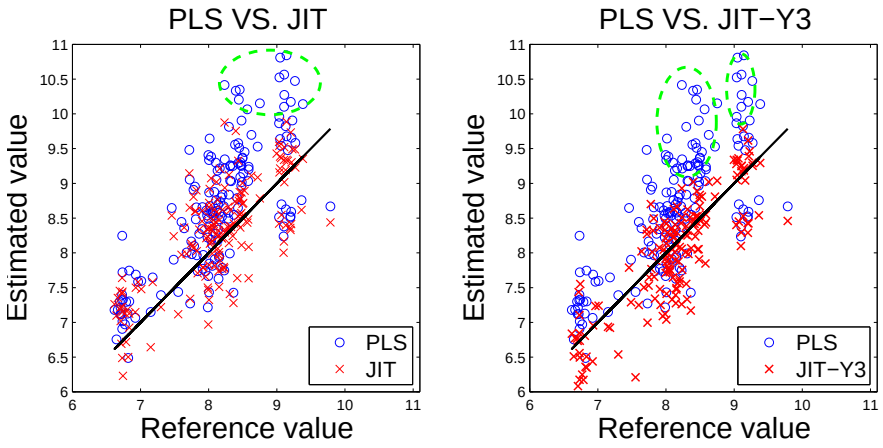

Figure 3.11: Scatter plot of JIT modeling results (Extrapolation case)

model has much poorer performance under the extrapolation case even by adding additional LVs. However, the prediction results of four JIT methods are still acceptable, which again proves the advantage of JIT approach, especially under extrapolation case. Even though the training data set cannot represent all the variation ranges of the validation data set, the JIT approach can still find the most relevant data points for modeling. Among all the JIT methods, JIT-Y3 again has the lowest $RMSEP$. JIT-Y1 obtains the highest R and has better performance than JIT-Y2. JIT has better performance than global PLS but is still worse than the methods that use the output space information to calculate similarity as proposed in this thesis. Both simulation studies show that involving y-space information for similarity calculation can improve the model performance.

The scatter plots for PLS, JIT and JIT-Y3 are plotted in Figure 3.11. For the global PLS approach, it can be seen that for most of the data points whose reference values are higher than 8.71% (the maximum of training set), their estimation results obviously deviate from the reference values (as marked by the green circles). Therefore the global PLS model cannot be

used effectively for extrapolation. However, JIT and JIT-Y3 can obtain good estimation results for these 'bad' points that need extrapolation. The lowest prediction error is achieved by JIT-Y3.

In order to check the performance of OSC algorithm and the degree of the nuisance information removed from the raw data, the following statistics are calculated. First, the angle θ between the orthogonal factor $\mathbf{t}_\perp$ and $\mathbf{y}$ is calculated as in Equation 3.59:

$$\cos(\theta) = \frac{\mathbf{t}_\perp^T \mathbf{y}}{\|\mathbf{t}_\perp\|\|\mathbf{y}\|} \frac{180}{\pi} \tag{3.59}$$

Second, the fraction of variations that remains in $\mathbf{X}$ after OSC filtering (φ) is calculated according to Equation 3.60:

$$\varphi = \frac{(\sum_{i=1}^{N} \sum_{j=1}^{M} x_{i,j}^2)_{OSC}}{(\sum_{i=1}^{N} \sum_{j=1}^{M} x_{i,j}^2)_{Raw}} \tag{3.60}$$

where $x_{i,j}$ is the element of matrix $\mathbf{X}$ at row i and column j. In Table 3.3, the angle and remaining variations are shown for both training data set and testing data set. From the table it can be observed that as the number of OSC factors increases, more variations will be removed from the raw data. It can also be noted that, after the removal of first OSC factor, about 80% of variation information was removed. For the next OSC factors, the algorithm slowly removed minor potions of the remaining variations. This actually indicates that most of the variations in $\mathbf{X}$ are uncorrelated with $\mathbf{y}$. So when two data points are close to each other in the original input space, they may not be close to each other in the $\mathbf{y}$-space. Even worse, it may happen that two data points are close to each other in $\mathbf{X}$-space just because they are corrupted by noise, and renders them meaningless to be used as relevant samples in model building stage. Further, note that the removed orthogonal vector $\mathbf{t}_\perp$ is almost perpendicular to the $\mathbf{y}$ vector, which

confirms that only the information orthogonal to output has been removed.

Table 3.3: Angle and RemainX statistics for the OSC method

	training set		Testing set	
NO. of OSC factors	Angle	RemainX	Angle	RemainX
1	90.00°	22.9%	89.99°	19.1%
2	90.00°	19.6%	90.02°	15.6%
3	90.00°	17.2%	89.98°	17.4%
4	90.00°	16.9%	90.13°	15.9%
5	90.00°	16.9%	90.41°	15.6%
6	90.00°	16.6%	90.65°	15.1%

To understand why the proposed method achieves better performance in selecting similar data points, a special sample, the 145-th query sample is selected for investigation. As can be seen in Table 3.4, both the PLS and regular JIT methods yield a large prediction error for this sample, while JIT-Y3 gives a much smaller prediction error. For the 145-th query sample, the weight of each training sample assigned by both JIT and JIT-Y3 methods is plotted in Figure 3.12, where the x-axis represents the reference y-value of corresponding training samples. Because the 145-th query sample has a y-value of $y_q = 8.2329$, both JIT and JIT-Y3 have assigned large weights to the samples with y-value around 8.2329. However, the difference is how JIT and JIT-Y3 treat the remaining samples that are obviously far away from 8.2329. For JIT method (only input space information is used to calculate similarity), it is clear that a branch of data points with y-values around 5 have been given weights as large as the ones around 8.2329. But for JIT-Y3, this branch of data points almost have zero weights. This is an evidence that JIT-Y3 method has considered not only the input space but also the output space information when calculating similarity. By assigning

less weights to the samples that are too far away in the output space, the algorithm avoids using wrong information for model construction.

Table 3.4: Prediction error for the 145-th query sample

Reference value	Error by PLS	Error by JIT	Error by JIT-Y3
8.2329	1.1508	0.9380	0.4453

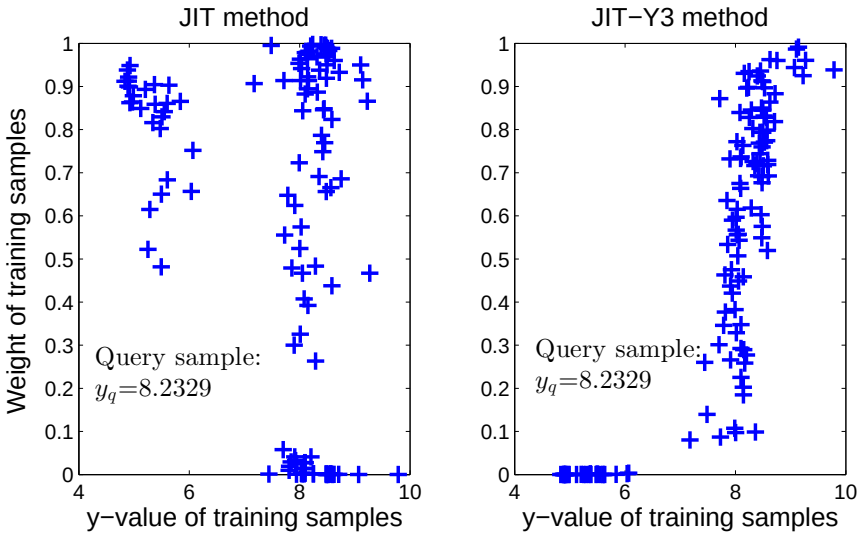

Figure 3.12: Weights of traning data at the 145-th query sample

For the extrapolation case study, the 85-*th* query sample is selected for investigation. Table 3.5 shows the prediction error and reference value for this point. From Figure 3.13, one can see that the regular JIT method assigned higher weights to the data points that are far away from the query sample in the output space, which is misleading in terms of finding similar data. On the other hand, JIT-Y3 avoids this problem by assigning almost zero weights to these data points and higher weights to the ones around $y_q = 9.1619$. Therefore, a lower prediction error is acheved by JIT-Y3.

Table 3.5: Prediction error for the 85-th query sample (Extrapolation case)

Reference value	Error by PLS	Error by JIT	Error by JIT-Y3
9.1619	0.3162	0.1634	0.1195

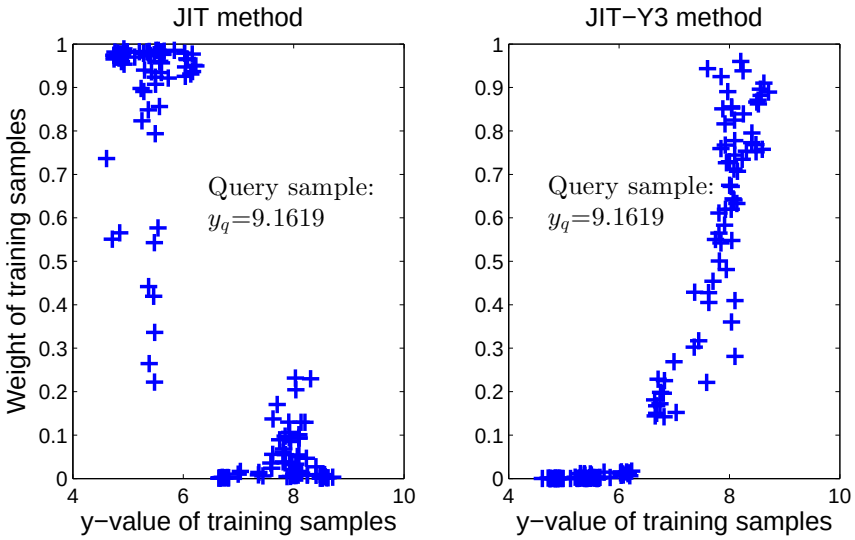

Figure 3.13: Weights of traning data at the 85-th query sample (Extrapolation case)

3.5 Conclusions

In this chapter, the framework of Just-in-time (JIT) approach was summarized into four components and the detailed description for each component has been provided. To enhance the model performance, two novel approaches for similarity measurement were proposed by incorporating both input and output information. The first approach simply utilizes the PLS latent variables for Euclidean distance calculation instead of using the original variables. The second approach applies orthogonal signal correction (OSC) algorithm to filter the raw data in order to remove the information uncorrelated with output. Then the distance can be calculated using the filtered data. The proposed methods have been applied on an NIR data set from pharmaceutical industry. Compared with traditional PLS and regular JIT methods, the model performance is improved by assigning more weights to the samples that are close to the query sample in both the input and output space. It is also found that, the proposed method could also be effective under the extrapolation situation, where the output range of training samples are not able to cover the whole range of process variation.

Chapter 4

Time varying issues in Just-in-time modeling

4.1 Introduction

Soft sensors are often built in order to monitor the hard-to-measure target variables in process industry. An important step for building a successful soft sensor is the offline model identification using the recorded historical data. However, it is observed that the prediction performance of the offline model deteriorates with respect to time. To ensure the acceptable performance of the offline model for a longer time, the historical data should contain all possible future states and conditions of the process, which includes not only the process operational conditions but also the external factors such as environmental changes and changes of the material quality. Practically, there are always numerous factors that cannot be foreseen during the offline model design phase. For example, process raw materials changes, catalyst activity changes, instrument drift and external environmental changes (e.g. weather, temperature, season). Even if all these factors can be estimated and collected from the historical database, covering the whole process variation would require a high model complexity and huge amount of process

data, which increases the burden of model development [77]. As a result of these facts, it is commonly admitted that the performance of invariant models will deteriorate gradually once they are put for online operation [78]. Due to the time-varying behavior of the process, it often requires to adapt the offline soft sensor to the varying operation conditions during the online operation phase.

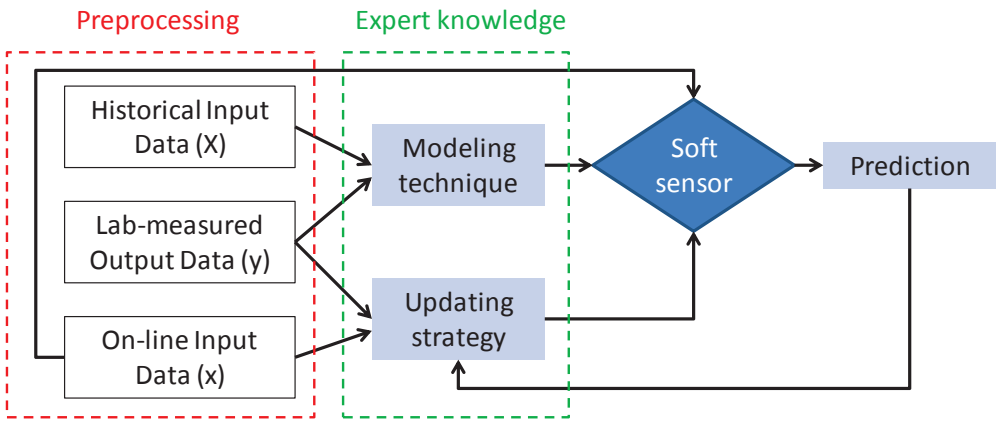

Figure 4.1: Online adaptation of soft sensors

The procedure for the adaptation of soft sensors is summarized in Figure 4.1. Initially, offline models are built using the historical input data $\mathbf{X}$ and lab reference data $\mathbf{y}$. Usually, data preprocessing is applied before modeling in order to deal with issues like outliers, missing values, noise, etc., and expert knowledge about the process is often introduced to identify the important variables and time delays. During online operation, the fast-rate input data (usually based on hardware sensors or instruments) is associated with the soft sensor to generate the prediction or estimation results for the hard-to-measure variable. As time goes by, the online input data $\mathbf{x}$ and corresponding lab reference data are accumulated and stored in historical database, which provides the source for model updating. In the online updating phase, two things need to be considered carefully, e.g. data preprocessing and performance feedback. The preprocessing step is also included

in the model updating phase in order to ensure that the model is adapted to the useful data preventing the model from absorbing disturbances or noises, especially when the hardware sensors deliver faulty measurements [79]. Furthermore, by monitoring the performance of soft sensors, a criterion is set to determine when to apply adaptation on the soft sensor, which can be achieved by comparing the model prediction results and the lab reference values. However, in practical scenarios, this step is often replaced by adapting the soft sensor at a certain frequency. Another point that should be mentioned in online adaptation is the involvement of expert knowledge. Usually, available knowledge of process dynamics should be applied when choosing the window size or forgetting factor for adaptation [77].

Tsymbal & Alexey [80] have categorized the existing adaptation methods into three approaches:

- Instance selection [81]
- Instance weighting [82]
- Ensemble methods [83]

The first two approaches have been named as Moving Window technique and Recursive Adaptation technique in [77], respectively. Both linear and non-linear models can be updated using the above mentioned three adaptation approaches. Least Squares (LS), Principal Component Regression (PCR) and Partial Least Squares (PLS) methods are often associated with moving window approach and recursive approach to adapt the models to process changes. The adaptive versions of non-linear modeling techniques such as Support Vector Machine (SVM), Least Square Support Vector Machine (LSSVM) and Adaptive Kernel Learning (AKL) have also been reported. However, to the best of the author's knowledge, the time varying problems in Just-in-time (JIT) framework are rarely discussed and the adaptation strategy for the local models in JIT approach remains an open issue.

Since JIT approach itself is able to handle non-linearity in the process, it would be beneficial if it can cope with time varying issue in the same time. Based on this motivation, an adaptive algorithm is proposed under the JIT framework with its usefulness demonstrated by industrial case studies.

4.2 Data-driven soft sensors using adaptive methods

In this section, a review of the existing adaptive soft sensing techniques is provided. Usually, two frameworks are widely used for prioritizing the most recent samples, the first one of which is the moving window approach, as shown in Figure 4.2. As new samples accumulate in the database, the window slides along the time axis enabling the involvement of the latest data. In the meantime, instead of storing all the available data, the same amount of the oldest samples are removed from the window due to the allowable data library constraint. Depending on the updating frequency, the model can be recalculated on a sample-wise (the most frequent case) or block-wise (less frequent case) basis [84]. The number of samples used in each calculation is called the 'Window size'. Usually the modeling techniques (SVM, PCR) are performed in each window when update is needed. The number of samples used to update the window is called 'Step size'. Both window size and step size are crucial to the performance of adaptive soft sensors. An inappropriate setting of these two parameters may result in the deterioration of model performance (in some cases even worse than a constant model). The second framework for adaptive soft sensing is the recursive approach, in which the past data are discounted gradually by a forgetting factor λ. In this way, the most recent samples are assigned higher weights than the distant past samples. The role of forgetting factor is similar to the window size as mentioned in the moving window approach, which quantifies

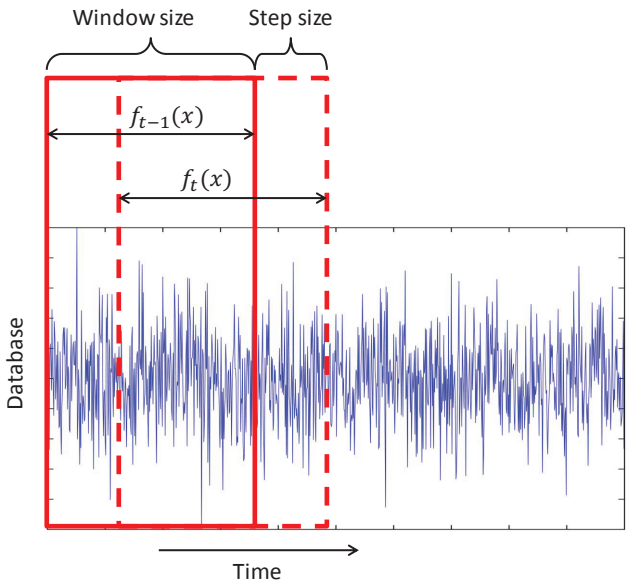

Figure 4.2: An illustration of Moving Window adaptation approach

the importance of the recent samples over the distant past samples. Benchmark algorithms such as recursive least square and recursive partial least square will be discussed later in this section.

4.2.1 Adaptation of linear models

Two of the most representative adaptive linear modeling methods are introduced here, i.e. moving window PCR and recursive least square, which are based on moving window technique and recursive technique, respectively.

Typically, moving window PCR is performed by retraining the PCR model periodically after accumulating a certain number of new samples. The latest N data points are used to retrain the model, where N is the window size. Usually the modeling technique (PCR) is combined with a parameter optimization method (such as cross validation) in order to estimate the optimal parameters (number of principle components). It should be noted that the parameter in an updating iteration may be different from the previous iterations.

Wang et al. have proposed an optimized version of the moving window PCR called Fast Moving Window Principal Component Analysis (FMWPCA) [85]. The algorithm incorporates the moving window technique with the recursive adaptation of the PCA model. It was shown that the new algorithm is computationally more efficient than conventional block-wise moving window techniques, under the assumption that the window size is at last 3 times the number of variables. The principle behind this algorithm is to update the latest correlation matrix from the previous one, which avoids the direct calculation using the process data. Kadlec et al. [77] have summarized the FMWPCA into a two-step algorithm, the first step of which is the downdating phase where the oldest sample $\mathbf{x}_{t-N}$ in the window is discarded as shown in Equation 4.1-4.2.

$$\mathbf{b}^* = \frac{N}{N-1}\mathbf{b}_{t-1} - \frac{1}{N-1}\mathbf{x}_{t-N} \tag{4.1}$$

$$\mathbf{R}^* = \mathbf{R}_{t-1} - \Sigma_{t-1}^{-1}(\mathbf{b}_{t-1} - \mathbf{b}_t)(\mathbf{b}_{t-1} - \mathbf{b}_t)^T\Sigma_{t-1}^{-1} - \frac{1}{N-1}\tilde{\mathbf{x}}_{t-N}\tilde{\mathbf{x}}_{t-N}^T \tag{4.2}$$

where $\mathbf{b}^*$ is the mean vector after removing the oldest data point, N is the window size, Σ is the standard deviation matrix for scaling purpose, $\tilde{\mathbf{x}}_{t-N}$ is the oldest data after mean subtraction and $\mathbf{R}^*$ is the intermediate correlation matrix. Then the second step (updating phase) incorporates the latest sample $\mathbf{x}_t$ by performing Equation 4.3-4.4.

$$\mathbf{b}_t = \frac{N-1}{N}\mathbf{b}^* + \frac{1}{N}\mathbf{x}_t \tag{4.3}$$

$$\mathbf{R}_t = \Sigma_t^{-1}\Sigma_{t-1}\mathbf{R}^*\Sigma_{t-1}\Sigma_t^{-1} + \Sigma_t^{-1}(\mathbf{b}_t - \mathbf{b}^*)(\mathbf{b}_t - \mathbf{b}^*)^T\Sigma_t^{-1} + \frac{1}{N-1}\tilde{\mathbf{x}}_t\tilde{\mathbf{x}}_t^T \tag{4.4}$$

where $\mathbf{b}_t$ is the updated mean vector, $\tilde{\mathbf{x}}_t$ is the newest data after mean subtraction and $\mathbf{R}_t$ is the updated correlation matrix. It can be seen from the above equations that FMWPCA algorithm obtains the computational

efficiency by updating the correlation matrix using procedures derived from recursive PCA algorithm.

Recursive adaptation technique is an example of instance weighting method; the dominating algorithms belonging to this class are recursive versions of least squares regression, principle component regression and partial least squares regression. The goal of recursive least squares (RLS) is to accommodate the latest data point $(\mathbf{x}_t, y_t)$ into the model coefficients β_t. The details are summarized in Equation 4.5-4.7.

$$\beta_t = \left(\begin{bmatrix} X_{t-1} \\ \mathbf{x}_t \end{bmatrix}^T \begin{bmatrix} X_{t-1} \\ \mathbf{x}_t \end{bmatrix} \right)^{-1} \left(\begin{bmatrix} X_{t-1} \\ \mathbf{x}_t \end{bmatrix}^T \begin{bmatrix} \mathbf{y}_{t-1} \\ y_t \end{bmatrix} \right) \tag{4.5}$$

$$P_t = (X_t^T X_t)^{-1} = P_{t-1} - \frac{P_{t-1}\mathbf{x}_t\mathbf{x}_t^T P_{t-1}}{1 + \mathbf{x}_t^T P_{t-1}\mathbf{x}_t^T} \tag{4.6}$$

$$\beta_t = \beta_{t-1} + P_t\mathbf{x}_t(y_t - \mathbf{x}_t^T\beta_{t-1}) \tag{4.7}$$

where P_t is the current covariance matrix. The above equations show that all the samples (instances) are treated equally; however, in real applications the more recent samples should be given higher weights. This can be achieved by introducing a weighting scheme using a forgetting factor which gradually discounts the old data. Equation 4.5 then becomes:

$$\beta_t = (X_t^T W X_t)^{-1}(X_t^T W \mathbf{y}_t), \quad \text{with} \quad W = diag(\lambda^{t-1}, \ldots, \lambda, 1) \tag{4.8}$$

where $0 < \lambda < 1$ and W is a diagonal matrix. It should be noted that the past data is gradually discounted by λ in each iteration. Thus the RLS regression becomes a weighted least square regression with less weights on the past data. The weight for the most recent data point is always 1.

Consequently, the solution to Equation 4.8 is:

$$P_t = \frac{1}{\lambda}\left(P_{t-1} - \frac{P_{t-1}\mathbf{x}_t\mathbf{x}_t^T P_{t-1}}{\lambda + \mathbf{x}_t^T P_{t-1}\mathbf{x}_t^T}\right) \tag{4.9}$$

From Equations 4.7 and 4.9, it is not difficult to find that there is no need to store all the data in memory for RLS algorithm. Instead, only the covariance matrix P_t needs to be stored. The forgetting factor plays a key role in RLS algorithm.

The instance weighting adaptation technique is also useful in PLS algorithm. Qin (1998) [42] used the following equation for sample-wise adaptation:

$$\mathbf{X}_t = \begin{bmatrix} \lambda P_{t-1}^T \\ \mathbf{x}_t \end{bmatrix} \quad \mathbf{y}_t = \begin{bmatrix} \lambda\beta_{t-1}Q_{t-1}^T \\ \mathrm{y}_t \end{bmatrix} \tag{4.10}$$

where P_{t-1} and Q_{t-1} are loading matrix obtained in the last updating iteration, and β_{t-1} is the model coefficients. The algorithm reconstructs the data matrix by merging the old model and new sample $(\mathbf{x}_t, y_t)$. The forgetting factor λ determines the strength of the adaptation. A lower value of this factor results in a faster adaptation to the current process condition. The reconstructed data matrix will be used to recalculate the model coefficients β_t.

Dayal and MacGregor [41] proposed a computationally efficient recursive PLS based on kernel version PLS algorithm. The key point in this recursive PLS is the update of covariance matrix:

$$\begin{aligned} R_t^{xx} &= \lambda R_{t-1}^{xx} + \mathbf{x}_t^T\mathbf{x}_t \\ R_t^{xy} &= \lambda R_{t-1}^{xy} + \mathbf{x}_t^T\mathrm{y}_t \end{aligned} \tag{4.11}$$

Then the updated covariance matrix can be used in the improved PLS kernel algorithm [49]. Since the improved kernel PLS is many times faster than regular PLS, it speeds up the computation during the adaptation procedure.

Both of the above two recursive PLS algorithms do not need to store all the data in memory. The first version only needs P_{t-1}, Q_{t-1} and B_{t-1}, while the second version only needs R_{t-1}^{xx} and R_{t-1}^{xy}.

4.2.2 Adaptation of non-linear models

Non-linear modeling methods have gained increasing popularity in building data-driven soft sensors since they can interpret the complex relationship between inputs and outputs. However, as time-varying issue always exists in processes, the success of these non-linear methods highly depends on how much variation information the historical data can cover. In other words, the appropriate adaptation on non-linear models is not only helpful but also necessary. So periodical updating is often performed on non-linear soft sensors in order to keep desired accuracy.

One of the most straightforward ways to model non-linearity in data is to modify the existing linear regression methods, e.g. PCR and PLS. The basic idea is to map the input space into a feature space through some non-linear mappings (also called kernel functions), and then the linear regression between the feature space and output space is performed [86]. It should be noted that the non-linear mapping can be applied on either the original input variables or the score vectors, as shown in Figure 4.3. Since this type of non-linear modeling does not involve any non-linear optimization procedures during the regression, it is quite easy to apply adaptation techniques to these methods. Therefore, the contents introduced in Section 4.2.1 are also applicable to this kind of non-linear modeling. Here a moving window kernel PCA algorithm and a recursive non-linear PLS algorithm are discussed as below.

Liu et al. have proposed a Moving Window Kernel PCA (MWKPCA) [87], which is similar to the FMWPCA discussed in Section 4.2.1. First, the non-linear mapping is performed on the input variables to formalize the

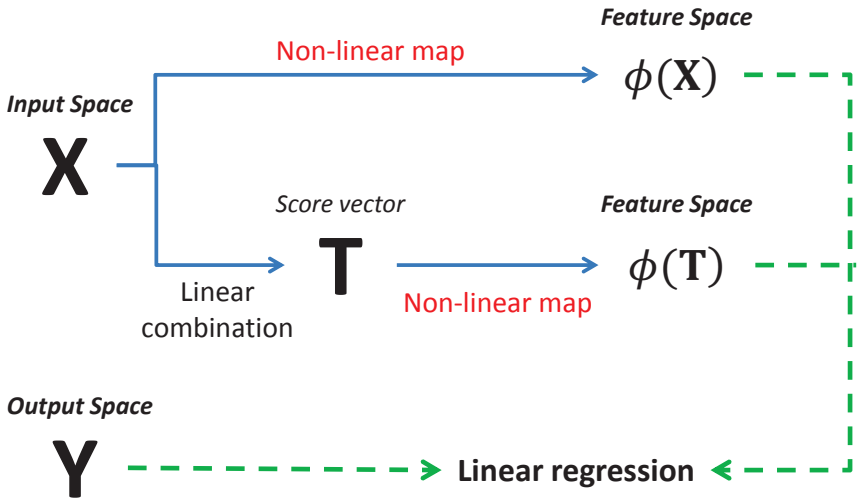

Figure 4.3: Two non-linear mapping approaches

covariance matrix as shown in Equations 4.12 and 4.13.

$$\mathbf{b}_\Phi = \frac{1}{n}\Phi(\mathbf{X})\mathbf{1}_n \tag{4.12}$$

$$C_\Phi = \frac{1}{n-1}\sum_{n}^{i=1}(\Phi(\mathbf{x}_i) - \mathbf{b}_\Phi)(\Phi(\mathbf{x}_i) - \mathbf{b}_\Phi)^T \tag{4.13}$$

where $\mathbf{b}_\Phi$ is the sample mean in the feature space, n is the number of samples, C_Φ is the sample covariance matrix in the feature space and Φ is the kernel function (non-linear mapping such as logarithm, polynomial etc.). After the construction of non-linear mapping, a two step adaptation scheme to remove the oldest sample and add the newest sample is performed using the moving window approach. The details can be found in [87].

Li & Ye have proposed a recursive non-linear PLS algorithm to solve both non-linearity and time varying issues encountered in process modeling [88]. The work is based on the recursive PLS algorithm proposed by Qin [42]. In order to handle the non-linearity issue, the Radial Basis Function (RBF) network is integrated with PLS regression, where the regression is performed between the output space and the extended input space. The

extended input is constructed by the original input variables, the hidden nodes outputs of the RBF network and a constant column with all elements being 1, as shown in Equation 4.14.

$$\mathbf{Y} = [\mathbf{1} \quad \mathbf{X} \quad \mathbf{G}] \begin{bmatrix} \mathbf{b}^T \\ \mathbf{A} \\ \mathbf{H} \end{bmatrix} = [\mathbf{1} \quad \mathbf{X} \quad \mathbf{G}]\beta_{PLS} \tag{4.14}$$

where $\mathbf{G}$ is the outputs of hidden nodes from the RBF network. Apparently, the matrix $\mathbf{G}$ is used to deal with non-linearity of the process and the column $\mathbf{1}$ is used for adapting the variations in process variable means. The outputs of the RBF network are linear combinations of the hidden nodes and a basis function has to be chosen for calculating the hidden layer. Usually the Gaussian kernel is chosen as the basis function, e.g. the l-th hidden node is calculated as in Equation 4.15:

$$G_l(\mathbf{x}) = exp(-\|\mathbf{x} - \mathbf{c}_l\|^2/\sigma_l^2) \tag{4.15}$$

where $\mathbf{c}_l$ is the centre of l-th node and σ_l is the corresponding width. In general, the RBF structure is illustrated in Figure 4.4.

In the recursive updating step, the latest sample $\mathbf{x}_t$ is also augmented into $[\mathbf{1} \quad \mathbf{x}_t \quad \mathbf{g}_t]$, and then a new PLS model is calculated the same way as shown in Equation 4.10. However, if $\mathbf{x}_t$ is too far away from the existing RBF nodes, a new hidden node will be added into the structure for adaptation purpose.

The above mentioned non-linear modeling techniques are modifications of linear regression techniques in the sense that they do not involve any non-linear optimization problem in the regression procedure. However, benchmark regression technique such as support vector machine (SVM) is a true non-linear modeling technique, where the quadratic programming

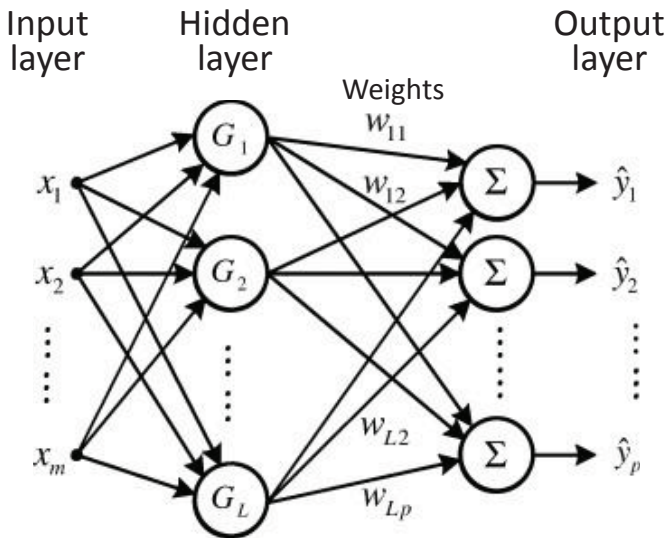

Figure 4.4: Non-linear mapping: The RBF structure

techniques are applied to solve the following optimization problem [89]:

$$f(\mathbf{x}) = \sum_i (\alpha_i - \alpha_i^*) k(\mathbf{x}_i, \mathbf{x}) + b \tag{4.16}$$

$$|y - f(\mathbf{x})| < \epsilon \tag{4.17}$$

where $k(\mathbf{x}_i,\mathbf{x}_j) = \varphi(\mathbf{x}_i)^T \varphi(\mathbf{x}_j)$, φ is the kernel function mapping the the original data into a high-dimensional space and b is a constant that can be calculated by applying the KarushKuhn-Tucker conditions [89]. Recently, Liu et al. [90] have proposed the adaptive version of SVM named online kernel learning algorithm (OKL), which is based on the Adaptive Kernel Learning framework proposed by Wang et al. [91]. The proposed OKL algorithm can adaptively learn the process dynamics using relatively small samples. The first step in OKL is referred to as Growing (Forward Recursive Learning), in which the number of nodes is increased with new incoming process information. The second step is referred to as Pruning (Backward Recursive Learning), in which the old information is removed recursively.

4.2.3 Adaptive algorithms in Just-in-time framework

Although Just-in-time approach is a local learning method, the prediction performance of the model deteriorates under certain circumstances, e.g., the change of the process characteristics [92]. Chen et al. proposed a new local modeling algorithm, adaptive local kernel-based learning scheme (ALKL), where an adaptive weighted least squares support vector regression (AW-LSSVR) was employed to establish the local model [92]. In ALKL, for each query data, the trade-off parameters of LSSVR are adjusted iteratively along the gradient descend direction so that the local model is updated recursively. The self-tuning of trade-off parameters is achieved by performing the fast leave-one-out cross-validation (FLOO-CV) criterion [93] on the most relevant l samples in the database. Similarly, the FLOO-CV strategy has been adopted by Liu et al. in [54] for adaptive selection of kernel parameters in a JIT-RLSSVR modeling framework. Since the kernel parameters in SVM algorithm are crucial to the success of a predictive model, the adaptive selection of parameters is beneficial in building local models. The implementation of FLOO-CV reduces the computational load during online phase. In the above two adaptive JIT algorithms, only the local kernel parameters are updated recursively and the database is always fixed. In this regard, these two methods are quite different from traditional recursive methods where the most recent sample pair $(\mathbf{x}_t, y_t)$ is merged into the database for model update. Under this circumstance, our motivation is to update the JIT models by adopting and weighting recent samples instead of refreshing the local tuning parameters. In the previous research, although JIT method builds a new model for each query data (which can be seen as a unique local model), the database is never changed. Actually, unless the original database contains sufficient information about the future process conditions, the models will become invalid after a certain period of time (even though they are local models). In this section, an algorithm which

prioritizes the most recent samples as well as nearby samples is proposed to solve both time varying and non-linearity problems.

4.3 Proposed algorithm: prioritize recent samples in JIT

4.3.1 Space weight and Time weight

Just-in-time approach can also be referred to as Relevance-in-space approach while the recursive algorithm can also be referred to as Relevance-in-time approach. It is not difficult to understand that JIT and recursive algorithm outperform traditional modeling methods by prioritizing samples that are relevant in terms of both distance and time. The nature of recursive adaptation algorithm is to assign higher weights on more recent samples and gradually forget the old data. In this case, the forgetting factor λ can be considered as the weight w as mentioned in Just-in-time framework. Then, as opposed to weight being determined by space relevance in JIT approach, the weight in this case is determined by time relevance. Thus it is possible to merge the recursive algorithm into the JIT framework so that both non-linearity and time varying issues can be solved in the same time. In the following discussion, we propose a formulation wherein the forgetting factor is transferred into the weight which can be used in locally weighted PLS.

The success of traditional locally weighted PLS depends on two main loading vectors as shown in Equations 4.18 and 4.19, where the loading vectors are derived using the diagonal weight matrix $\mathbf{W}$.

$$\mathbf{p}_k = \frac{(\mathbf{X}^T\mathbf{W}\mathbf{y})^T(\mathbf{X}^T\mathbf{W}\mathbf{X})}{(\mathbf{X}^T\mathbf{W}\mathbf{y})^T(\mathbf{X}^T\mathbf{W}\mathbf{X})(\mathbf{X}^T\mathbf{W}\mathbf{y})} \tag{4.18}$$

$$q_k = \frac{(\mathbf{X}^T\mathbf{W}\mathbf{y})^T(\mathbf{X}^T\mathbf{W}\mathbf{y})}{(\mathbf{X}^T\mathbf{W}\mathbf{y})^T(\mathbf{X}^T\mathbf{W}\mathbf{X})(\mathbf{X}^T\mathbf{W}\mathbf{y})} \tag{4.19}$$

It is clear that both $\mathbf{p}_k$ and q_k are derived from two weighted covariance matrix, as shown in Equations 4.20 and 4.21.

$$\mathbf{R}_x = \mathbf{X}^T\mathbf{W}\mathbf{X} \tag{4.20}$$

$$\mathbf{R}_y = \mathbf{X}^T\mathbf{W}\mathbf{y} \tag{4.21}$$

To investigate the influence of weights on the covariance matrix, let:

$$\mathbf{X} = \begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \\ \vdots \\ \mathbf{x}_N \end{bmatrix} \quad \mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{bmatrix} \quad \mathbf{W} = \begin{bmatrix} w_1 & 0 & \cdots & 0 \\ 0 & w_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & w_N \end{bmatrix} \tag{4.22}$$

Then the weighted covariance matrix in Equations 4.20 and 4.21 can be written as

$$\begin{aligned} \mathbf{R}_x &= w_1\mathbf{x}_1^T\mathbf{x}_1 + w_2\mathbf{x}_2^T\mathbf{x}_2 + \cdots w_N\mathbf{x}_N^T\mathbf{x}_N \\ \mathbf{R}_y &= w_1\mathbf{x}_1^T y_1 + w_2\mathbf{x}_2^T y_2 + \cdots w_N\mathbf{x}_N^T y_N \end{aligned} \tag{4.23}$$

It is clear that for the i-th ($i = 1, 2...N$) sample pair $(\mathbf{x}_i, y_i)$ the weight can be expressed as:

$$W(\mathbf{x}_i) = w_i = f(d_i) \tag{4.24}$$

where $f(d_i)$ is a function of the distance d_i between $\mathbf{x}_i$ and $\mathbf{x}_q$.

Similarly for recursive PLS, the most important step is to discount the covariance matrix by putting a forgetting factor λ on the past data, as shown in the following equations:

$$\begin{aligned} \mathbf{R}_t^x &= \lambda\mathbf{R}_{t-1}^x + \mathbf{x}_t^T\mathbf{x}_t \\ \mathbf{R}_t^y &= \lambda\mathbf{R}_{t-1}^y + \mathbf{x}_t^T y_t \end{aligned} \tag{4.25}$$

where $\mathbf{X}_0$ is the initial data matrix containing a certain number of training samples (used for initial model building) and t represents the current time. The current covariance matrix can be further written into:

$$\begin{aligned}\mathbf{R}_t^x &= \lambda^t \mathbf{X}_0^T \mathbf{X}_0 + \lambda^{t-1}\mathbf{x}_1^T\mathbf{x}_1 + \lambda^{t-2}\mathbf{x}_2^T\mathbf{x}_2 + \cdots \lambda \mathbf{x}_{t-1}^T\mathbf{x}_{t-1} + \mathbf{x}_t^T\mathbf{x}_t \\ \mathbf{R}_t^y &= \lambda^t \mathbf{X}_0^T \mathbf{y}_0 + \lambda^{t-1}\mathbf{x}_1^T\mathbf{y}_1 + \lambda^{t-2}\mathbf{x}_2^T\mathbf{y}_2 + \cdots \lambda \mathbf{x}_{t-1}^T\mathbf{y}_{t-1} + \mathbf{x}_t^T\mathbf{y}_t\end{aligned} \tag{4.26}$$

Comparing Equation 4.23 and Equation 4.26, it is not difficult to find that both locally weighted algorithm and recursive algorithm belong to the instance weighting technique; thus recursive algorithm can also be expressed as a weighting algorithm as shown in Equations 4.27 and 4.28.

$$\begin{aligned}\mathbf{R}_t^x &= \mathbf{X}_t \mathbf{W}_t \mathbf{X}_t \\ \mathbf{R}_t^y &= \mathbf{X}_t \mathbf{W}_t \mathbf{y}_t\end{aligned} \tag{4.27}$$

$$\mathbf{X}_t = \begin{bmatrix}\mathbf{X}_0 \\ \mathbf{x}_1 \\ \vdots \\ \mathbf{x}_t\end{bmatrix} \quad \mathbf{y}_t = \begin{bmatrix}\mathbf{y}_0 \\ y_1 \\ \vdots \\ y_t\end{bmatrix} \quad \mathbf{W}_t = \begin{bmatrix}\lambda^t & 0 & \cdots & 0 \\ 0 & \lambda^{t-1} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \lambda^0\end{bmatrix} \tag{4.28}$$

Then the 'Time' weight for the sample at time τ can be expressed as in Equation 4.29. The weight for the current sample ($\tau = t$) is 1.

$$W(\mathbf{x}_\tau) = \lambda^{t-\tau} = f(t) \tag{4.29}$$

From Equations 4.24 and 4.29, one can find that both space weight and time weight for the historical samples can be merged into the JIT framework. A novel JIT algorithm which can handle both time varying and non-linearity issues is thus proposed and the detailed description is given in the following part.

4.3.2 Recursive locally weighted PLS

Based on the above discussions, the motivation of the proposed algorithm is to prioritize both the recent samples (time relevance) and the nearby samples (space relevance) in the JIT framework when building a local PLS model. In the proposed algorithm, a moving window approach is adopted to update the database. Then two weighting matrix $\mathbf{W}_1$ and $\mathbf{W}_2$ are formulated to account for time varying and non-linearity issues, respectively. A balancing parameter ρ is introduced in order to make a balance between $\mathbf{W}_1$ and $\mathbf{W}_2$. The details are described as below.

Step 1. Update the database using moving window approach. The most recent sample is added into the database while the oldest one is removed.

Step 2. Mean center the database and query sample using Equation 4.30-4.34:

$$\overline{\mathbf{x}} = \frac{\sum_{i=1}^{N} \mathbf{x}_i}{N} \tag{4.30}$$

$$\overline{y} = \frac{\sum_{i=1}^{N} y_i}{N} \tag{4.31}$$

$$\tilde{\mathbf{X}} = \mathbf{X} - \mathbf{1}_N \overline{\mathbf{x}} = \begin{bmatrix} \tilde{\mathbf{x}}_1 \\ \tilde{\mathbf{x}}_2 \\ \vdots \\ \tilde{\mathbf{x}}_N \end{bmatrix} \tag{4.32}$$

$$\tilde{\mathbf{y}} = \mathbf{y} - \mathbf{1}_N \overline{y} \tag{4.33}$$

$$\tilde{\mathbf{x}}_q = \mathbf{x}_q - \overline{\mathbf{x}} \tag{4.34}$$

Step 3. Set up the *Time Weight Window* size K as the number of most recent samples to be prioritized.

Step 4. Assign weights to the most recent K samples using the forgetting

factor λ so that the weighting matrix $\mathbf{W}_1$ is obtained.

$$\mathbf{W}_1 = \begin{bmatrix} \lambda^{K-1} & 0 & \cdots & 0 \\ 0 & \lambda^{K-2} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 \end{bmatrix} \tag{4.35}$$

Step 5. Assign weights to the remaining $N-K$ samples using Euclidean distance.

$$d_i = \|\tilde{\mathbf{x}}_i - \tilde{\mathbf{x}}_q\|_2, \qquad i = 1, 2...N-K \tag{4.36}$$

$$w_i = \frac{d_i - d_{min}}{d_{max} - d_{min}} \tag{4.37}$$

where d_{min} is the minimum distance from query sample and d_{max} is the maximum distance.

$$\mathbf{W}_2 = \begin{bmatrix} w_1 & 0 & \cdots & 0 \\ 0 & w_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & w_{N-K} \end{bmatrix} \tag{4.38}$$

Step 6. Combining $\mathbf{W}_1$ and $\mathbf{W}_2$ using the balancing parameter ρ.

$$\mathbf{W} = \begin{bmatrix} \mathbf{W}_2 & 0 \\ 0 & \rho\mathbf{W}_1 \end{bmatrix} = \begin{bmatrix} w_1 & & & & & \\ & \ddots & & & 0 & \\ & & w_{N-K} & & & \\ & & & \rho\lambda^{K-1} & & \\ & 0 & & & \ddots & \\ & & & & & \rho \end{bmatrix} \tag{4.39}$$

Step 7. Perform locally weighted PLS algorithm using $\mathbf{W}$ (follow the same procedure as shown in Section 3.2.4 of Chapter 3).

The first step adopts moving window approach in order to keep the number of historical samples constant; otherwise the time required to perform search for the relevant samples will increase as database keeps growing. Step 2 is used for detecting the bias change, which is a commonly observed phenomenon during online phase. Step 3 and 4 construct weighting matrix for the Time Weight Window by putting forgetting factor λ on the most recent K samples ($0 < \lambda < 1$). The number K is the size of this window which can be determined according to both prior knowledge and offline validation results. Step 5 assigns space weights to the remaining samples in the database using Euclidean distance calculation. The space weights are normalized between 0 and 1 by Equation 4.37 so that their values are comparable with the time weights calculated in Equation 4.35. In Step 6, a balancing parameter ρ ($0 < \rho \leq 1$) is put on $\mathbf{W}_1$ in order to determine the strength of adaptation to the most recent samples.

To make a better interpretation of the algorithm, the weight for each data point is divided into two parts, i.e. the space weight and time weight, as descried in the following equation:

$$w = w_S \cdot w_T \tag{4.40}$$

As a result, the final weighting matrix can be explained in the following form:

$$\mathbf{W} = \mathbf{W}_S \cdot \mathbf{W}_T = \begin{bmatrix} f(d_1) & & & \\ & \ddots & 0 & \\ & & \rho & \\ 0 & & & \ddots \\ & & & & \rho \end{bmatrix} \cdot \begin{bmatrix} 1 & & & \\ & \ddots & & 0 \\ & & \lambda^{K-1} & \\ 0 & & & \ddots \\ & & & & 1 \end{bmatrix} \tag{4.41}$$

For samples in the time weight window, the space weight are all forced to be the balancing factor $w_S = \rho$ while the time weight is a function of time $w_T = \lambda^{t-\tau}$. On the other hand, for the remaining samples, the space weight is a function of distance $w_S = f(d_i)$ while the time weight is forced to be $w_T = 1$.

4.4 Application results

4.4.1 NIR spectra from refinery

The NIR data used for case study were obtained from a refinery plant, located in Edmonton, Canada. A total of 565 diesel samples are collected from online operation between January 2010 and April 2012, the sampling rate of which is one sample per day. The spectra of diesel samples were measured using an NIR spectrometer having the wavelength range of 800-1700 nm and nominal spectral resolution of 1 nm. The target property to be estimated is the Final Boiling Point (FBP), which is measured in lab using standard ASTM testing methodologies. For proprietary reasons, the property values are normalized between -1 and 1. The earliest 300 samples were used to form the initial database, and the remaining 265 samples were used for testing the algorithm. All the spectra were subject to first order derivative preprocessing according to the Savitzky-Golay method. The original spectra and the preprocessed spectra can be seen in Figure 2.2

4.4.2 Results and discussion

The proposed recursive locally weighted PLS (RLWPLS) in Section 4.3.2 is applied to the NIR data mentioned above. In this study (NIR modeling for the refinery product), the number of samples in the database, which is also the moving window size, is $N = 300$. The forgetting factor in Equation

4.35 is set as 0.95 in order to keep a certain number of samples to be effective. For example, the value of the 40-th latest sample will become 0.13 time that of the actual value ($0.95^{40} \approx 0.13$). The effects of the other two parameters, i.e. the Time Weight Window size K and the balancing parameter ρ are investigated. For each pair of the two parameters, the corresponding prediction performance is demonstrated by $RMSEP$ and correlation coefficient R, which are shown in Figure 4.5 and 4.6.

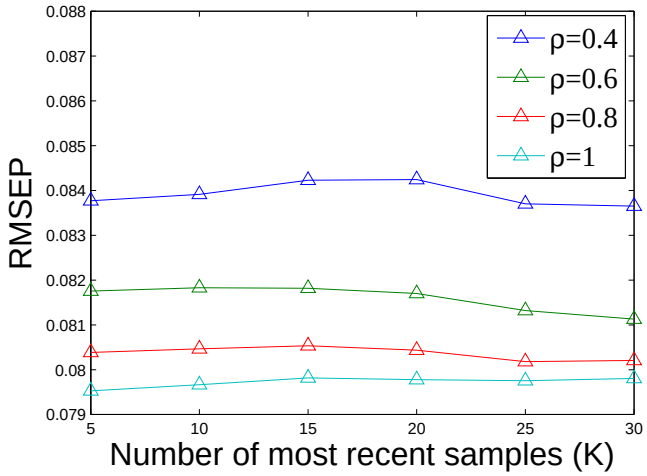

Figure 4.5: Prediction error under different tuning parameters

From Figure 4.5, it is clear that the balancing parameter plays an important role in the model performance. When $\rho = 1$, the model gets lowest $RMSEP$, which means that large time weights should be put onto the latest samples. However, the Time Weight Window size (K) does not have too much influence on the model performance. As long as the balancing parameter ρ is fixed, the $RMSEP$ does not change much with K. Similarly in Figure 4.6, it is the balancing parameter that plays a crucial role in the prediction results. The highest correlation coefficient (above 0.85) is obtained when $\rho = 1$ and the correlation coefficient does not change too much with K. Thus, the inference drawn from Figure 4.6 matches with that from Figure 4.5. In conclusion, the optimal values for these two parameters

are $K_{opt} = 5$ and $\rho_{opt} = 1$. It should be noted that the optimal parameters depend on the data set. For processes with more non-linearity and less time varying issues, a larger K and a smaller ρ might be more helpful. In Figure

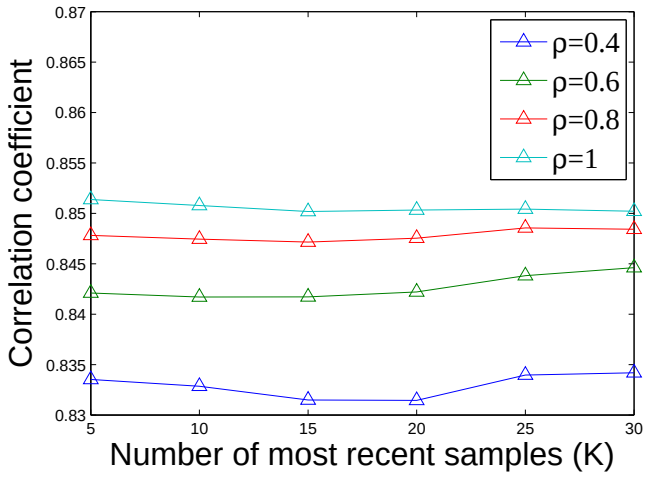

Figure 4.6: Correlation coefficient under different tuning parameters

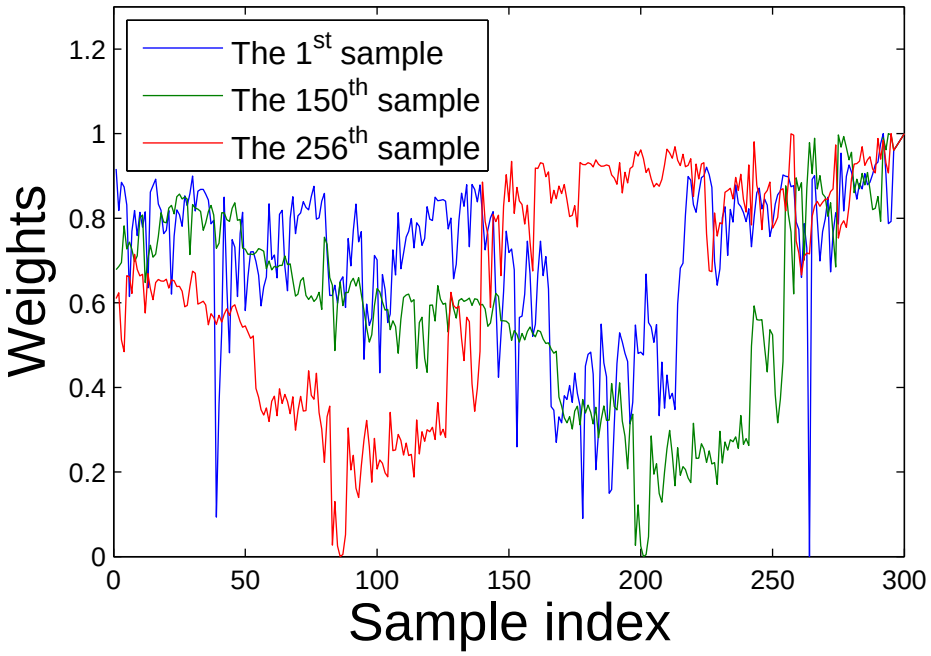

Figure 4.7: Training data weights at 3 different query samples

4.7, the weights of training samples assigned by RLWPLS are plotted. The

window size is $N = 300$, and the most recent 5 samples ($296^{th} - 300^{th}$) are assigned relatively higher weights $(0.95^4, 0.95^3, 0.95^2, 0.95, 1)$. It should be noted that samples that have large space weights are not necessarily the most recent ones, which means that they have lower time weights. On the other hand, some of the recent samples are assigned with low space weights. In real applications, both non-linearity and time varying issues exist in the same time. The two problems cannot be solved by applying a simple locally weighted algorithm or simple adaptive algorithm separately. The proposed algorithm solves both problems by forcing a few most recent samples to have large time weights while assigning space weights to the remaining samples.

The prediction results of the proposed algorithm are compared with another four methods listed below.

1. PLS: A fixed PLS model.

2. LWPLS: Locally weighted PLS with a fixed database, using the similarity measurement from Equation 4.36 and 4.37.

3. MWPLS: Moving window PLS with window size=300 and step size=1.

4. MW-LWPLS: Locally weighted PLS with a moving window database. The similarity measurement is the same as LWPLS method and window update is the same as MWPLS method.

5. RLWPLS: The proposed algorithm with forgetting factor $\lambda = 0.95$, balancing parameter $\rho = 1$ and $K = 5$.

The $RMSEP$ and R for the above mentioned methods are summarized in Table 4.1. It is clear that the proposed RLWPLS can significantly reduce the $RMSEP$ by 45.26% compared with PLS method. The R can be improved by 69.27% using RLWPLS.

Table 4.1: Estimation results for JIT and adaptive methods

Methods	$RMSEP$	Reduced by%	R	Increased by%
PLS	0.1453	0%	0.5030	0%
LWPLS	0.1299	10.59%	0.6399	27.23%
MWPLS	0.1082	25.51%	0.7157	42.30%
MW-LWPLS	0.0887	38.93%	0.8131	61.67%
RLWPLS	0.0795	45.26%	0.8514	69.27%

Traditional methods such as LWPLS and MWPLS are also able to improve the model performance, but not as much as the proposed RLWPLS. Specially, MWPLS achieves more improvements than LWPLS, which indicates that the time varying issue is more evident than non-linearity issue in this data set. The MW-LWPLS gets comparable results as RLWPLS, which indicates that the non-linearity and time varying issues both exist in the same time.

The detailed estimation results are shown in Figure 4.8 and 4.9. From Figure 4.8, it is clear that LWPLS is not able to track the process dynamics during certain periods, which is due to the limited process information contained in the database. This is an evidence that if the JIT approach adopts a fixed database, the future process information will be missed and thus the model performance will deteriorate in real applications as time goes by. From Figure 4.9, one can notice that MWPLS achieves better performance than LWPLS in terms of tracking the process dynamics, but there are still some instances where the model performance is not satisfactory. This may be due to the remaining non-linearity issues which cannot be removed by an adaptive algorithm. In both Figure 4.8 and 4.9, RLWPLS gets a better prediction performance since RLWPLS takes advantage of both adaptive and locally weighted algorithms. Furthermore, the scatter plots for the four modeling methods (PLS, LWPLS, MWPLS, RLWPLS) are shown in

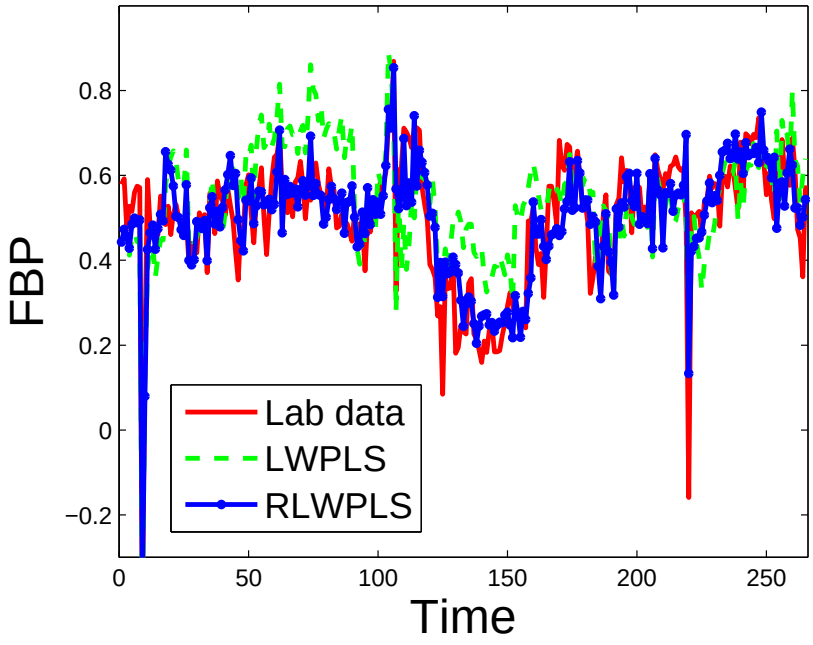

Figure 4.8: Prediction results for LWPLS and RLWPLS

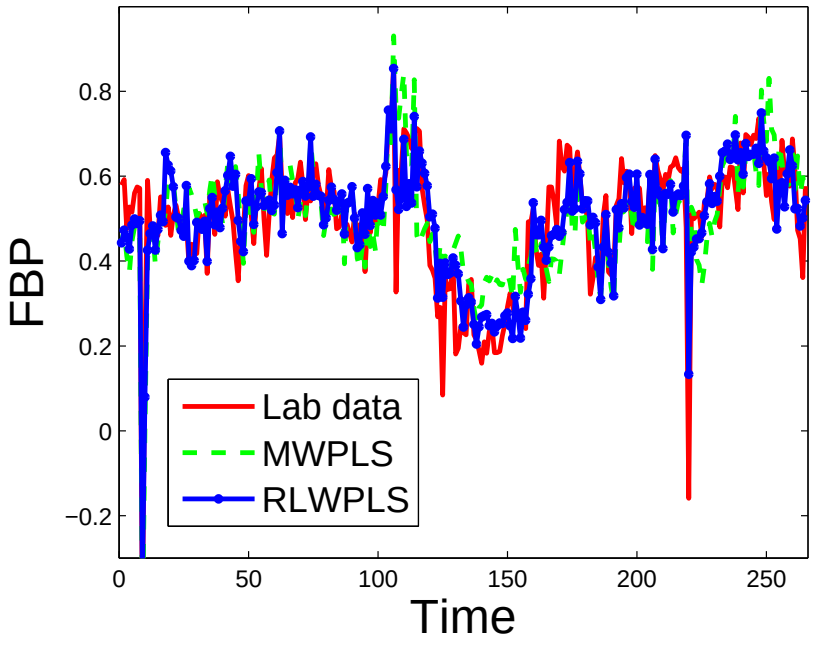

Figure 4.9: Prediction results for MWPLS and RLWPLS

Figure 4.10.

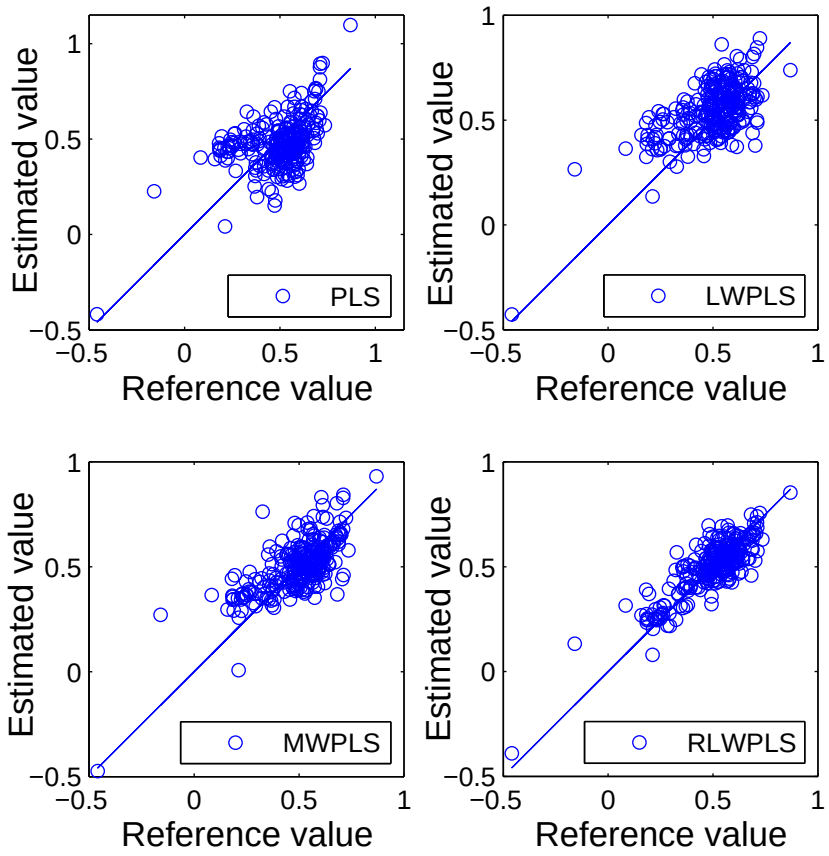

Figure 4.10: Scatter plots for RVP predictions

4.5 Conclusions

In this chapter, the traditional methods for adaptation of soft sensors have been reviewed. Both linear regression techniques and non-linear regression techniques can be updated using corresponding adaptive methods. Traditionally, recursive algorithm and locally weighted algorithm prioritize certain samples by assigning higher weights to more relevant samples in terms of time and distance, respectively. In order to deal with both time varying and non-linearity issues in the same time, the Recursive Locally Weighted

Partial Least Square (RLWPLS) algorithm is proposed. RLWPLS is formulated by merging recursive and locally weighted algorithms into the same Just-in-time framework. The moving window database update is adopted to renew the information stored in the database. Then both recent samples and nearby samples are given higher weights to form the weighting matrix. Finally the regression is conducted using locally weighted PLS algorithm and the weighting matrix. The usefulness of this algorithm is illustrated by an industrial NIR case study. Compared with traditional PLS, regular JIT methods, and moving window algorithms, the proposed method is shown to significantly enhance the prediction performance. Furthermore, a balancing parameter is proposed to adjust the weights assigned on the most recent samples.

Chapter 5

Conclusions and Recommendations

5.1 Summary of this thesis

This thesis is concerned with the model development and model update problems for near infrared (NIR) spectroscopy applications. Traditional invariant models have shown limitations when there are non-linearity and/or time varying issues in the process. The limitation of conventional hardware sensors and the necessity of building robust soft sensors motivate us to explore the model development for NIR applications.

The background material about near infrared spectroscopy was presented in Chapter 1. Chapter 2 dealt with the online model update problem for NIR applications. By incorporating the wavelength selection method with recursive partial least squares regression, an update strategy which can adjust both model structure and model coefficients was proposed. The variable importance in the projection (VIP) method was adopted to perform wavelength selection in each update iteration. The recursive exponentially weighted PLS (rPLS) was adopted to calculate the score vectors, loading vectors and model coefficients. Industrial case studies were used to evaluate

the performance of proposed algorithm.

In practice, nonlinear relationship between spectra (input) and quality variable (output) always exists, especially for industrial processes with various operation ranges. The objective of Chapter 3 is to develop local models under the framework of Just-in-time learning in order to solve the nonlinearity problem in model development. The orthogonal signal correction (OSC) algorithm was used to combine both input and output information for similarity calculation. To illustrate the proposed model construction scheme, an NIR data set from pharmaceutical industry was used. The comparison results with other traditional JIT methods proved the superiority of this proposed method.

Although non-linearity issue can be handled using JIT modeling, time varying issue is still a problem if the process condition changes over time. Chapter 4 solved both non-linearity and time varying issues under the JIT framework. The recursive algorithm was formulated under the JIT framework and a novel method was proposed to combine recursive algorithm with locally weighted algorithm. An industrial case study was used to prove the usefulness of the proposed method.

5.2 Recommendations for future work

Throughout the whole thesis, we have worked on the modeling and model update issues that are associated with near infrared spectroscopy. Several techniques have been adopted such as recursive algorithm, moving window approach, Just-in-time approach and orthogonal signal correction. With the effectiveness of these methods being demonstrated in this thesis, there are several open issues and directions which can further enhance the performance of these techniques. Firstly, the recursive wavelength selection adopts a fixed number of latent variables and a constant threshold for VIP

wavelength selection which obtained by offline cross validation. It would be more beneficial to check if the optimal setting of these parameters changes with time and then formulate a framework to adjust the optimal values of these parameters. Secondly, numerous wavelength selection methods are available in the literature so that they could also be incorporated with the recursive algorithm. Thirdly, the similarity criterion using input-output information is only evaluated using NIR data sets, and the efficiency of this method can be expanded into other processes where multi-variate modeling is needed. Finally, how to define and quantify the time varying and non-linearity issues in the data sets remains to be an open question.

Bibliography

[1] Pavel Matějka. *NIR Spectrometry.*

[2] Zhiqiang Ge and Zhihuan Song. A comparative study of just-in-time-learning based methods for online soft sensor modeling. *Chemometrics and Intelligent Laboratory Systems*, 104(2):306–317, 2010.

[3] Cheng Cheng and Min-Sen Chiu. A new data-based methodology for nonlinear process modeling. *Chemical engineering science*, 59(13): 2801–2810, 2004.

[4] Zou Xiaobo, Zhao Jiewen, Malcolm J. W. Povey, Mel Holmes, and Mao Hanpin. Variables selection methods in near-infrared spectroscopy. *Analytica Chimica Acta*, 667(1-2):14–32, 2010.

[5] L.E. Agelet and C.R. Hurburgh Jr. A tutorial on near infrared spectroscopy and its calibration. *Critical Reviews in Analytical Chemistry*, 40(4):246–260, 2010.

[6] D.A. Burns and E.W. Ciurczak. *Handbook of near-infrared analysis*, volume 35. CRC, 3rd edition, 2007.

[7] R.M. Balabin and S.V. Smirnov. Variable selection in near-infrared spectroscopy: Benchmarking of feature selection methods on biodiesel data. *Analytica Chimica Acta*, 692(1):63–72, 2011.

[8] Asmund Rinnan, Frans van den Berg, and Soren Balling Engelsen. Review of the most common pre-processing techniques for near-infrared spectra. *TrAC-Trends in Analytical Chemistry*, 28(10):1201–1222, 2009.

[9] M Zeaiter, JM Roger, and V Bellon-Maurel. Robustness of models developed by multivariate calibration. Part II: The influence of pre-processing methods. *TrAC-Trends in Analytical Chemistry*, 24(5):437–445, 2005.

[10] H Martens, JP Nielsen, and SB Engelsen. Light scattering and light absorbance separated by extended multiplicative signal correction. application to near-infrared transmission analysis of powder mixtures. *Analytical Chemistry*, 75(3):394–404, 2003.

[11] A Savitzky and Mje Golay. Smoothing and differentiation of data by simplified least squares procedures. *Analytical Chemistry*, 36(8):1627–&, 1964.

[12] RJ Barnes, MS Dhanoa, and S.J. Lister. Standard normal variate transformation and de-trending of near-infrared diffuse reflectance spectra. *Applied Spectroscopy*, 43(5):772–777, 1989.

[13] S. Wold, H. Antti, F. Lindgren, and J. Öhman. Orthogonal signal correction of near-infrared spectra. *Chemometrics and Intelligent Laboratory Systems*, 44(1):175–185, 1998.

[14] MCU Araujo, TCB Saldanha, RKH Galvao, T Yoneyama, HC Chame, and V Visani. The successive projections algorithm for variable selection in spectroscopic multicomponent analysis. *Chemometrics and Intelligent Laboratory Systems*, 57(2):65–73, 2001.

[15] V Centner, DL Massart, OE deNoord, S deJong, BM Vandeginste,

and C Sterna. Elimination of uninformative variables for multivariate calibration. *Analytical Chemistry*, 68(21):3851–3858, 1996.

[16] IG Chong and CH Jun. Performance of some variable selection methods when multicollinearity is present. *Chemometrics and Intelligent Laboratory Systems*, 78(1-2):103–112, 2005.

[17] H. Swierenga, PJ De Groot, AP De Weijer, MWJ Derksen, and LMC Buydens. Improvement of PLS model transferability by robust wavelength selection. *Chemometrics and Intelligent Laboratory Systems*, 41 (2):237–248, 1998.

[18] GuangHui Fu, QingSong Xu, HongDong Li, DongSheng Cao, and YiZeng Liang. Elastic net grouping variable selection combined with partial least squares regression (EN-PLSR) for the analysis of strongly multi-collinear spectroscopic data. *Applied Spectroscopy*, 65(4):402–408, 2011.

[19] Hongdong Li, Yizeng Liang, Qingsong Xu, and Dongsheng Cao. Key wavelengths screening using competitive adaptive reweighted sampling method for multivariate calibration. *Analytica Chimica Acta*, 648(1): 77–84, 2009.

[20] R Leardi and AL Gonzalez. Genetic algorithms applied to feature selection in PLS regression: how and when to use them. *Chemometrics and Intelligent Laboratory Systems*, 41(2):195–207, 1998.

[21] L. Norgaard, A. Saudland, J. Wagner, J.P. Nielsen, L. Munck, and SB Engelsen. Interval partial least-squares regression (iPLS): a comparative chemometric study with an example from near-infrared spectroscopy. *Applied Spectroscopy*, 54(3):413–419, 2000.

[22] S. Kasemsumran, Y.P. Du, K. Maruo, and Y. Ozaki. Improvement of

partial least squares models for in vitro and in vivo glucose quantifications by using near-infrared spectroscopy and searching combination moving window partial least squares. *Chemometrics and Intelligent Laboratory Systems*, 82(1):97–103, 2006.

[23] R.M. Balabin, R.Z. Safieva, and E.I. Lomakina. Comparison of linear and nonlinear calibration models based on near infrared (NIR) spectroscopy data for gasoline properties prediction. *Chemometrics and Intelligent Laboratory Systems*, 88(2):183–188, 2007.

[24] M. Blanco, J. Coello, H. Iturriaga, S. Maspoch, and J. Pages. NIR calibration in non-linear systems: different PLS approaches and artificial neural networks. *Chemometrics and Intelligent Laboratory Systems*, 50 (1):75–82, 2000.

[25] F. Chauchard, R. Cogdill, S. Roussel, JM Roger, and V. Bellon-Maurel. Application of LS-SVM to non-linear phenomena in NIR spectroscopy: development of a robust and portable sensor for acidity prediction in grapes. *Chemometrics and Intelligent Laboratory Systems*, 71(2):141–150, 2004.

[26] T. Chen, J. Morris, and E. Martin. Gaussian process regression for multivariate spectroscopic calibration. *Chemometrics and Intelligent Laboratory Systems*, 87(1):59–71, 2007.

[27] D. Perez-Marin, A. Garrido-Varo, and JE Guerrero. Non-linear regression methods in NIRS quantitative analysis. *Talanta*, 72(1):28–42, 2007.

[28] E. Bouveresse, C. Hartmann, DL Massart, IR Last, and KA Prebble. Standardization of near-infrared spectrometric instruments. *Analytical Chemistry*, 68(6):982–990, 1996.

[29] Onno E De Noord. Multivariate calibration standardization. *Chemometrics and Intelligent Laboratory Systems*, 25(2):85–97, 1994.

[30] Z. Chen, D. Lovett, and J. Morris. Process analytical technologies and real time process control a review of some spectroscopic issues and challenges. *Journal of Process Control*, 21(10):1467–1482, 2011.

[31] W. Du, Z.P. Chen, L.J. Zhong, S.X. Wang, R.Q. Yu, A. Nordon, D. Littlejohn, and M. Holden. Maintaining the predictive abilities of multivariate calibration models by spectral space transformation. *Analytica Chimica Acta*, 690(1):64–70, 2011.

[32] M. Zeaiter, JM Roger, and V. Bellon-Maurel. Dynamic orthogonal projection. A new method to maintain the on-line robustness of multivariate calibrations. Application to NIR-based monitoring of wine fermentations. *Chemometrics and Intelligent Laboratory Systems*, 80 (2):227–235, 2006.

[33] Z.P. Chen, L.M. Li, R.Q. Yu, D. Littlejohn, A. Nordon, J. Morris, A.S. Dann, P.A. Jeffkins, M.D. Richardson, and S.L. Stimpson. Systematic prediction error correction: A novel strategy for maintaining the predictive abilities of multivariate calibration models. *Analyst*, 136(1): 98–106, 2010.

[34] Y. Wu, Y. Jin, Y. Li, D. Sun, X. Liu, and Y. Chen. NIR spectroscopy as a process analytical technology (PAT) tool for on-line and real-time monitoring of an extraction process. *Vibrational Spectroscopy*, 2011.

[35] W. Li, L. Xing, L. Fang, J. Wang, and H. Qu. Application of near infrared spectroscopy for rapid analysis of intermediates of *Tanreqing* injection. *Journal of Pharmaceutical and Biomedical Analysis*, 53(3): 350–358, 2010.

[36] C.L. Stork and B.R. Kowalski. Weighting schemes for updating regression modelsa theoretical approach. *Chemometrics and Intelligent Laboratory Systems*, 48(2):151–166, 1999.

[37] X. Capron, B. Walczak, OE De Noord, and DL Massart. Selection and weighting of samples in multivariate regression model updating. *Chemometrics and Intelligent Laboratory Systems*, 76(2):205–214, 2005.

[38] J.A. Farrell, K. Higgins, and J.H. Kalivas. Updating a near-infrared multivariate calibration model formed with lab-prepared pharmaceutical tablet types to new tablet types in full production. *Journal of Pharmaceutical and Biomedical Analysis*, 2011.

[39] B. Xu, Z. Wu, Z. Lin, C. Sui, X. Shi, and Y. Qiao. NIR analysis for batch process of ethanol precipitation coupled with a new calibration model updating strategy. *Analytica Chimica Acta*, 2012.

[40] K. Helland, H.E. Berntsen, O.S. Borgen, and H. Martens. Recursive algorithm for partial least squares regression. *Chemometrics and Intelligent Laboratory Systems*, 14(1):129–137, 1992.

[41] B.S. Dayal and J.F. MacGregor. Recursive exponentially weighted PLS and its applications to adaptive control and prediction. *Journal of Process Control*, 7(3):169–179, 1997.

[42] S. Joe Qin. Recursive PLS algorithms for adaptive data modeling. *Computers & Chemical Engineering*, 22(4):503–514, 1998.

[43] O. Haavisto, J. Kaartinen, and H. Hyötyniemi. Optical spectrum based measurement of flotation slurry contents. *International Journal of Mineral Processing*, 88(3):80–88, 2008.

[44] O. Haavisto and H. Hyötyniemi. Recursive multimodel partial least squares estimation of mineral flotation slurry contents using optical reflectance spectra. *Analytica Chimica Acta*, 642(1):102–109, 2009.

[45] O. Haavisto and H. Hyötyniemi. Reflectance spectroscopy in the analysis of mineral flotation slurries. *Journal of Process Control*, 21(2): 246–253, 2011.

[46] J.W. Jin, Z.P. Chen, L.M. Li, R. Steponavicius, S.N. Thennadil, J. Yang, and R.Q. Yu. Quantitative spectroscopic analysis of heterogeneous mixtures: The correction of multiplicative effects caused by variations in physical properties of samples. *Analytical Chemistry*, 84 (1):320–326, 2011.

[47] Z.P. Chen and J. Morris. Improving the linearity of spectroscopic data subjected to fluctuations in external variables by the extended loading space standardization. *Analyst*, 133(7):914–922, 2008.

[48] S. Roussel, B. Igne, D. Funk, and C. Hurburgh. Noise robustness comparison for near infrared prediction models. *Journal of Near Infrared Spectroscopy*, 19(1):23, 2011.

[49] B. Dayal and J.F. MacGregor. Improved PLS algorithms. *Journal of Chemometrics*, 11(1):73–85, 1998.

[50] S. Mu, Y. Zeng, R. Liu, P. Wu, H. Su, and J. Chu. Online dual updating with recursive PLS model and its application in predicting crystal size of purified terephthalic acid (PTA) process. *Journal of Process Control*, 16(6):557–566, 2006.

[51] S. Kim, M. Kano, H. Nakagawa, and S. Hasebe. Estimation of active pharmaceutical ingredients content using locally weighted partial least

squares and statistical wavelength selection. *International Journal of Pharmaceutics*, 2011.

[52] L. Yan-kun. Determination of diesel cetane number by consensus modeling based on uninformative variable elimination. *Analytical Methods*, 4(1):254–258, 2012.

[53] Petr Kadlec, Bogdan Gabrys, and Sibylle Strandt. Data-driven soft sensors in the process industry. *Computers & chemical engineering*, 33 (4):795–814, 2009.

[54] Yi Liu, Zengliang Gao, Ping Li, and Haiqing Wang. Just-in-time kernel learning with adaptive parameter selection for soft sensor modeling of batch processes. *Industrial & Engineering Chemistry Research*, 51(11): 4313–4327, 2012.

[55] Ziyi Wang, Tomas Isaksson, and Bruce R Kowalski. New approach for distance measurement in locally weighted regression. *Analytical Chemistry*, 66(2):249–260, 1994.

[56] Sanghong Kim, Manabu Kano, Hiroshi Nakagawa, and Shinji Hasebe. Estimation of active pharmaceutical ingredients content using locally weighted partial least squares and statistical wavelength selection. *International journal of pharmaceutics*, 421(2):269–274, 2011.

[57] Hiroshi Nakagawa, Takahiro Tajima, Manabu Kano, Sanghong Kim, Shinji Hasebe, Tatsuya Suzuki, and Hiroaki Nakagami. Evaluation of infrared-reflection absorption spectroscopy measurement and locally weighted partial least-squares for rapid analysis of residual drug substances in cleaning processes. *Analytical chemistry*, 84(8):3820–3826, 2012.

[58] Christopher G Atkeson, Andrew W Moore, and Stefan Schaal. Locally weighted learning. *Artificial intelligence review*, 11(1-5):11–73, 1997.

[59] Sanghong Kim, Manabu Kano, Shinji Hasebe, Akitoshi Takinami, and Takeshi Seki. Long-term industrial applications of inferential control based on just-in-time soft-sensors: Economical impact and challenges. *Industrial & Engineering Chemistry Research*.

[60] D Perez-Marin, A Garrido-Varo, and JE Guerrero. Non-linear regression methods in nirs quantitative analysis. *Talanta*, 72(1):28–42, 2007.

[61] Hiroyasu Shigemori, Manabu Kano, and Shinji Hasebe. Optimum quality design system for steel products through locally weighted regression model. *Journal of Process Control*, 21(2):293–301, 2011.

[62] Koichi Fujiwara, Manabu Kano, Shinji Hasebe, and Akitoshi Takinami. Soft-sensor development using correlation-based just-in-time modeling. *AIChE Journal*, 55(7):1754–1765, 2009.

[63] Koichi Fujiwara, Manabu Kano, and Shinji Hasebe. Development of correlation-based pattern recognition algorithm and adaptive soft-sensor design. *Control Engineering Practice*, 20(4):371–378, 2012.

[64] D Perez-Marin, A Garrido-Varo, and JE Guerrero. Non-linear regression methods in nirs quantitative analysis. *Talanta*, 72(1):28–42, 2007.

[65] Shih-Ying Chang, Ernie H Baughman, and Bruce C McIntosh. Implementation of locally weighted regression to maintain calibrations on ft-nir analyzers for industrial processes. *Applied Spectroscopy*, 55(9): 1199–1206, 2001.

[66] Are Halvor Aastveit and Petter Marum. Near-infrared reflectance spectroscopy: Different strategies for local calibrations in analyses of forage quality. *Applied spectroscopy*, 47(4):463–469, 1993.

[67] Sanghong Kim, Ryota Okajima, Manabu Kano, and Shinji Hasebe. Development of soft-sensor using locally weighted pls with adaptive similarity measure. *Chemometrics and Intelligent Laboratory Systems*, 2013.

[68] Anthony MC Davies, Heather V Britcher, Jeremy G Franklin, Susan M Ring, Alex Grant, and William F McClure. The application of fourier-transformed near-infrared spectra to quantitative analysis by comparison of similarity indices (carnac). *Microchimica Acta*, 94(1-6):61–64, 1988.

[69] Tom Fearn and Anthony Davies. Locally-biased regression. *Journal of near infrared spectroscopy*, 11(6):467–478, 2003.

[70] Vladimir Vapnik. *The nature of statistical learning theory*. springer, 1999.

[71] Stefan Schaal, Christopher G Atkeson, and Sethu Vijayakumar. Scalable techniques from nonparametric statistics for real time robot learning. *Applied Intelligence*, 17(1):49–60, 2002.

[72] Baibing Li, A Julian Morris, and Elaine B Martin. Orthogonal signal correction: algorithmic aspects and properties. *Journal of chemometrics*, 16(11):556–561, 2002.

[73] O Svensson, T Kourti, and JF MacGregor. An investigation of orthogonal signal correction algorithms and their characteristics. *Journal of chemometrics*, 16(4):176–188, 2002.

[74] Tom Fearn. On orthogonal signal correction. *Chemometrics and Intelligent Laboratory Systems*, 50(1):47–52, 2000.

[75] Johan Trygg and Svante Wold. Orthogonal projections to latent structures (o-pls). *Journal of Chemometrics*, 16(3):119–128, 2002.

[76] M Dyrby, SB Engelsen, L Nørgaard, M Bruhn, and L Lundsberg-Nielsen. Chemometric quantitation of the active substance (containing c≡ n) in a pharmaceutical tablet using near-infrared (nir) transmittance and nir ft-raman spectra. *Applied spectroscopy*, 56(5):579–585, 2002.

[77] Petr Kadlec, Ratko Grbić, and Bogdan Gabrys. Review of adaptation mechanisms for data-driven soft sensors. *Computers & chemical engineering*, 35(1):1–24, 2011.

[78] Petr Kadlec, Bogdan Gabrys, and Sibylle Strandt. Data-driven soft sensors in the process industry. *Computers & Chemical Engineering*, 33(4):795–814, 2009.

[79] GD Gonzalez. Soft sensors for processing plants. In *Intelligent Processing and Manufacturing of Materials, 1999. IPMM'99. Proceedings of the Second International Conference on*, volume 1, pages 59–69. IEEE, 1999.

[80] Alexey Tsymbal. The problem of concept drift: definitions and related work. *Computer Science Department, Trinity College Dublin*, 2004.

[81] Marcus A Maloof and Ryszard S Michalski. Selecting examples for partial memory learning. *Machine Learning*, 41(1):27–52, 2000.

[82] Ralf Klinkenberg. Learning drifting concepts: Example selection vs. example weighting. *Intelligent Data Analysis*, 8(3):281–300, 2004.

[83] Martin Scholz and Ralf Klinkenberg. Boosting classifiers for drifting concepts. *Intelligent Data Analysis*, 11(1):3–28, 2007.

[84] Min W Lee, Jea Y Joung, Dae S Lee, Jong M Park, and Seung H Woo. Application of a moving-window-adaptive neural network to the mod-

eling of a full-scale anaerobic filter process. *Industrial & engineering chemistry research*, 44(11):3973–3982, 2005.

[85] Xun Wang, Uwe Kruger, and George W Irwin. Process monitoring approach using fast moving window pca. *Industrial & Engineering Chemistry Research*, 44(15):5691–5702, 2005.

[86] Bernhard Schölkopf, Alexander Smola, and Klaus-Robert Müller. Nonlinear component analysis as a kernel eigenvalue problem. *Neural computation*, 10(5):1299–1319, 1998.

[87] Xueqin Liu, Uwe Kruger, Tim Littler, Lei Xie, and Shuqing Wang. Moving window kernel pca for adaptive monitoring of nonlinear processes. *Chemometrics and Intelligent Laboratory Systems*, 96(2):132–143, 2009.

[88] Chunfu Li, Hao Ye, GUIZENG Wang, and Jie Zhang. A recursive nonlinear pls algorithm for adaptive nonlinear process modeling. *Chemical engineering & technology*, 28(2):141–152, 2005.

[89] Alex J Smola and Bernhard Schölkopf. A tutorial on support vector regression. *Statistics and computing*, 14(3):199–222, 2004.

[90] Yi Liu, Haiqing Wang, and Ping Li. Modeling of fermentation processes using online kernel learning algorithm. In *Proceedings of IFAC World Congress*, pages 9679–9684, 2008.

[91] Haiqing Wang, Ping Li, Furong Gao, Zhihuan Song, and Steven X Ding. Kernel classifier with adaptive structure and fixed memory for process diagnosis. *AIChE journal*, 52(10):3515–3531, 2006.

[92] Kun Chen, Jun Ji, Haiqing Wang, Yi Liu, and Zhihuan Song. Adaptive local kernel-based learning for soft sensor modeling of nonlinear pro-

cesses. *Chemical Engineering Research and Design*, 89(10):2117–2124, 2011.

[93] Gavin C Cawley and Nicola LC Talbot. Fast exact leave-one-out cross-validation of sparse least-squares support vector machines. *Neural networks*, 17(10):1467–1475, 2004.