Locally Weighted Predictive Modeling for Regression and Classification of Process Data

by

Alireza Kheradmand

A thesis submitted in partial fulfillment of the requirements for the degree of

Master of Science

in

Process Control

Department of Chemical and Materials Engineering

University of Alberta

Abstract

Predictive modeling has proven to be a valuable tool in process industry to estimate hard-to-measure variables that cannot be measured online. Those variables usually require LAB analysis to be quantified, which is time-consuming and costly. Predictive modeling can be used for both regression and classification. Predictive models in regression are often used as soft sensors or inferential sensors, and play an important role in control of processes.

Traditional methods in building predictive models usually suggest global approaches, where a training set of data is used to build the model, and the model performance is validated on the testing set. However, the performance of the model usually degrades over time as a result of nonlinearity, time varying issues or curse of dimensionality, all of which point out the necessity of model updating and good maintenance techniques. There are a number of existing methods in providing solutions for model updating and maintenance. Nonetheless if the operation mode keeps on changing rapidly, those methods are not able to adapt well with the process.

To address above mentioned issues, locally weighted modeling has been proposed, which strives to build as many local predictive models as necessary to address highly nonlinear process modeling problem. Locally weighted modeling is also known as Just-In-Time learning, which builds a local model for each query sample that arrives. Just-In-Time relies on a similarity measurement step to highlight the most relevant samples to the query sample. The local model is built mostly based on those samples. Traditional similarity metrics only account for magnitude of samples and mostly disregard the effect of time sequence. A novel Just-In-Time

learning method is proposed in this thesis as "Trend-Based Just-In-Time", which accounts for magnitude, direction and trend of changes in database with respect to the query sample. The proposed method takes advantage of Principal Component Analysis (PCA) and a moving horizon approach to evaluate similarity. Additionally, traditional Just-In-Time methods have limitations in dealing with missing data. By using Probabilistic PCA (PPCA), one is able to evaluate similarity in presence of missing data.

Similarity measurement is usually carried out in input subspace of a data set. Commonly, by including output space information in similarity calculation, improvement in prediction can be anticipated. Therefore, the proposed Trend-Based Just-In-Time is extended to similarity calculation in output space as well.

As mentioned, predictive models are also used for building classification models (or predictive classifiers). Similar to regression, global models might not be able to provide good classifications when data set is not linearly separable. Another issue that might occur to a data set is curse of dimensionality, which requires a variable (or variable) selection technique. In this thesis, a novel approach for locally weighted classification of high dimensional data with variable selection is proposed. The proposed approach addresses curse of dimensionality problem by using Neighborhood Component Analysis (NCA) as a variable selection technique and Kernel PCA as dimension reduction method. To address high degree of nonlinearity, locally weighted approach is employed, which performs similarity measurement between query sample and historical database in latent space. Afterwards, by considering the weights obtained from similarity calculation, a locally weighted Support Vector Machine (SVM) model is built.

All proposed approaches have been implemented on various Near InfraRed (NIR) data sets as well as synthetic data sets.

Acknowledgements

I am so fortunate to work as a member of Computer Process Control group at University of Alberta, where I could achieve what I desired in my professional life in addition to making friends, the company of whom is always delightful to me.

First and foremost, I would like to thank my supervisor and life mentor Prof.Biao Huang who provided this fantastic opportunity for me to continue my graduate studies. His constant patience and guidance along with his insightful comments and constructive criticism enabled me to investigate my potentials in research and industrial projects. In addition to being my academic supervisor, I always think of him as a life mentor who thinks of no limit in what an individual can achieve. I would like to express my sincerest gratitude to him for trusting me.

Afterwards, I would like to thank Dr.Yousef Alipouri. He was always there for me as both a teacher and a friend when I needed help. I learned how to think innovatively, and how to complete tasks in academia from him. In fact, without his help and support, completing this thesis would have been very difficult. Additionally, I would like to thank Dr.Nabil Magbool Jan, from whom I learned how to become patient, and managing multiple tasks. He is certainly one of the best teachers I have ever seen.

A special thanks to three of my friends in Computer Process Control group: Rahul Raveendran, Yanjun Ma and Shabnam Sedghi (Currently at Spartan Controls) who always received me kindly when I had question. I would also like to thank my other friends and colleagues in CPC group for their precious friendship including but not limited to: Hossein Shanhande, Agustin Vicente, David Scott, Yashas Mohankumar, Seraphina Kwak, Hareem Shafi, Lei Fan, Mengqi Fang, Ranjith Chiplunkar, Rui Nian,

Anahita Sadeghian, Atefeh Daemi, Arun Senthil, Sanjula Kammammettu, Nirwair Singh Bajwa, Dr.Shunyi Zhao, Dr.Fadi Ibrahim.

I would also like to acknowledge my other friends at University of Alberta whose presence helped me during this program: Farshad MohammadTabar, Hamid Niazi, Ali Talaie, Ali Khani, Firouz Khodayari, Sara Imani, Vahid Vajihi, Amin Pourasghar, Amene Sheikhjafari, Anuar Caldera, Daniel Moran, Vadim Kistlitsin, Abdorahaman Albeladi, Fernanda Carrozi, Gissele Uzcategui, Linda Contreras, Nagesh Kasturi Pai, Ananthan Santhakrishnan, and many others.

I would like to express my true gratitude to Dr.Ramesh Kadali from Suncor Energy for his his support and trust in completing projects with Suncor Energy.

I would like to acknowledge the Department of Chemical and Materials Engineering, University of Alberta, National Science and Engineering Council Canada and Industrial partners of CPC group especially Suncor Energy and Spartan Controls.

Last but not least, I cannot imagine my life without my family, above everyone, my mother and my father for everything I have in my life, and my sisters Aida, Azin and my brother-in-law Samad for their love and support. I would also like to thank three of my life-time friends: Hossein Yousefpour, Vahid Bahrie and Mehran Yaghoubi. I am expressing my gratitude to all my loved ones in life. Without their understanding and encouragement, I could not be where I am today.

Table of Contents

List of Tables

List of Figures

List of Abbreviations and Acronyms

OLS	*Ordinary Least Squares*
PCA	*Principal Component Analysis*
PCR	*Principal Component Regression*
PLS	*Partial Least Squares*
NIR	*Near InfraRed*
JIT	*Just-In-Time*
LWPLS	*Locally Weighted Partial Least Square*
Co-JIT	*Correlation-based Just-In-Time*
TB-JIT	*Trend-Based Just-In-Time*
PPCA	*Probabilistic Principal Component Analysis*
EM	*Expectation-Maximization algorithm*
IO TB-JIT	*Input-Output Trend-Based Just-In-Time*
NCA	*Neighborhood Component Analysis*
SVM	*Support Vector Machine*
KNN	*K Nearest Neighbor*

Chapter 1

Introduction

1.1 Motivation

Industrial processes possess a large volume of data owing to increasing number of sensors installed in a plant. Despite abundant process data available, there are some critical variables which cannot be measured online. These variables, known as quality variables or target properties, play a significant role in process operations. Quality variables are often measured through LAB techniques which are expensive and time consuming. To overcome this problem, predictive data-driven modeling has been widely practiced through which estimates of quality variables can be obtained. Predictive models can be used for different purposes such as regression or classification. A predictive model that is capable of providing estimate of a target property is known as soft sensor. Classification is another application of predictive models through which the class, category, or the type of the material used in a sample can be identified. The term predictive classifier refers to predictive models built for the purpose of classification.

Development, implementation and maintenance of predictive models can sometimes be a laborious task. Control actions are performed based on real time process measurements or outputs of predictive models. Diversion of estimation in predictive models might mislead the controller to take inappropriate control actions, and thus lead the process to undesired operation zones. Therefore, adopting updating and maintenance techniques in predictive models is of great importance.

The main contribution of this thesis is to address problems such as high degree of nonlineairty, curse of dimensionality and varying operation modes in building predictive models to ultimately provide satisfactory performance for a predictive model.

1.2 Thesis Outline and Contributions

The overall view of the problems taken, existing methods in research and industry, proposed solution as well as case studies are given in Figure 1.1.

In chapter 2, a novel Just-In-Time modeling framework is proposed as "Trend-Based Just-In-Time". It considers a new similarity measurement which is an important part of the framework. The proposed method takes advantage of sequential correlation of samples in a time-series data set. Traditional methods in Just-In-Time only consider the magnitude of samples with respect to the query sample (newly arrived sample). Trend-Based Just-In-time is able to calculate similarities considering magnitude, direction and trends between training samples and query sample. In other words, the method computes similarity considering similar trends of changes in the historical samples with respect to query sample. To calculate similarities, instead of query sample as in traditional Just-In-Time (JIT), a query window is formed which consists of query sample and a few of the most recent samples. By using a moving horizon approach, it compares the query window with different windows in historical database by employing PCA. Furthermore, traditional Just-In-Time methods cannot deal with missing data, since similarity measurement is done in a deterministic manner. Owing to the use of PPCA, similarity calculation can be done in presence of missing input data. The model structure adopted is LW-PLS. The efficacy of the proposed approach is examined on a synthetic data set as well as an Near-InfraRed (NIR) data set from an oil refinery.

In chapter 3, based upon the method proposed in chapter 2, a novel input/output similarity measure is proposed as "Input-Output similarity function for Trend-Based Just-In-Time learning". To evaluate similarity in input space, Trend-Based Just-In-

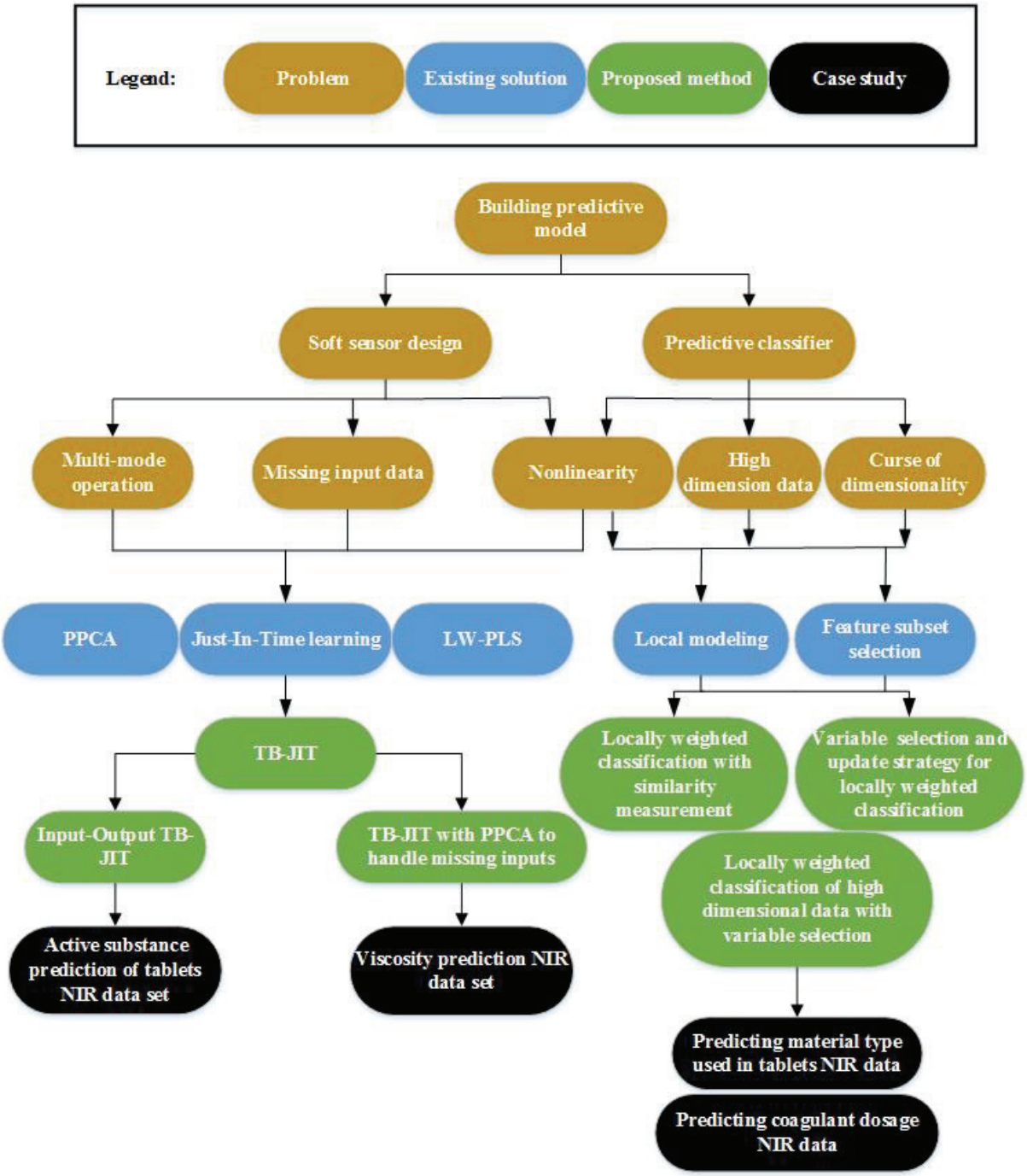

Figure 1.1: *Overall view of problems and solutions in this thesis.*

Time proposed in previous chapter is used. Then, an output similarity measurement technique is employed to evaluate similarity in output space. Prior to this step, an

initial model needs to be built to provide an initial estimation of outputs. Based on the initial estimation, similarity is calculated in output space, and thus the overall input-output space similarity is obtained. Model structure used in this chapter is LW-PLS. An NIR data set as well as a synthetic data set are used to test the performance of the proposed method .

In chapter 4, locally weighted modeling is extended to classification of high dimensional data sets when data set is not linearly separable. A new approach as " Locally weighted classification of high dimensional data with variable selection" is presented. The proposed approach is able to deal with curse of dimensionality, since a variable selection technique - Neighborhood Component Analysis (NCA)- is employed. After variable selection, a Kernel PCA model is built with respect to the query sample, which transforms training samples and query sample into latent space. Afterwards, a similarity measurement technique, such as Euclidean distance, calculates the distances between training samples and query sample to assign weights to training samples. Obtained weights are applied to a locally weighted Support Vector Machine (SVM) model to predict the class of query samples. The proposed approach is beneficial when data set is high dimensional, and not linearly separable. It is able to provide appropriate predictions owing to use of NCA and locally weighted modeling. Two NIR data sets, one from pharmaceutical application and the other from real-time operation of an oil sands extraction site, are used to test the performance of the proposed approach.

Finally, concluding remarks of this thesis and recommendations for future work are discussed.

Chapter 2

Trend-Based Just-In-Time Learning with Missing Input Data

2.1 Introduction

Predictive modeling plays a vital role in advanced process control and automation. Often, linear models are preferred in the industry since they require minimum computation cost, and they are relatively easy to build. Ordinary Least Squares (OLS) is the most prominent linear modeling technique. The prediction capability of the OLS model is often poor when there exists strong linear correlation among input variables. To deal with this issue, dimensionality reduction is typically carried out prior to building a regression model. This can be accomplished using latent variable extraction. Principal Component Analysis (PCA) [1] is the most popular tool for extracting linear latent variables. To take advantage of PCA in supervised learning, one can use the latent variables to build a regression model between them and outputs. This modeling technique is known as Principal Component Regression (PCR) [1]. Nonetheless, PCR only accounts for input data to extract the latent variables, which stands to the fact that the variation is captured with respect to only input data. On the other hand, Partial Least Squares (PLS) structure [2] extracts latent variables while considering both input and output information. Thus, PLS is quite useful in many applications in data analytic including Near InfraRed (NIR) modeling [3] and image processing [4].

Satisfactory performance of linear methods can be anticipated provided that the process is linear intrinsically. However, for nonlinear process data, predictive performance cannot be guaranteed. There are several techniques to enable a linear approach applicable to nonlinear processes. Such methods usually integrate out nonlinear features to extract linear ones. Methods such as quadratic PLS [5], neural network PLS [6] and fuzzy PLS [7] lie under this category. In these methods, linear feature extraction is carried out after a nonlinear mapping. As an alternative, Non Linear PLS models (NLPLS) [8] can extract nonlinear features. All of the above mentioned methods can be categorized as parametric global models which are trained offline, and the built models remain unchanged for all operating conditions. Although the computation cost is lesser when a global model is trained offline, its accuracy is not guaranteed especially in cases where process undergoes sudden and frequent changes. This is because a global model is usually built based on certain operating conditions.

To deal with nonlinearity and changing process conditions, the model should be adaptable. In this regard, Atkeson et al. [9] proposed a locally weighted learning approach. This is also known as "Just-In-Time-Learning (JIT)" or "Lazy learning". The main idea behind this approach is to delay modeling unless it is really needed. When a query sample arrives, a local model is built to make a prediction. Once it has been used, it is discarded. The whole procedure is repeated for each query sample that arrives. The advantage of locally weighted learning is that the model is built at the current operating condition, therefore, the accuracy of the local model prediction relies on the samples that are chosen. Thus, the vital part of locally weighted modeling relies on how relevant samples are chosen for building the local models. This can be accomplished by assigning weights to samples in the historical database. Locally weighted modeling or JIT, assigns weights to samples based on their importance in prediction. Those with higher weights draw more attention compared to others. Weights are obtained using distance which can be measured by geometric, statistical or any other metric. When a query sample arrives, the distances between the new sample (query sample) and samples in historical

database are computed. Intuitively speaking, lower values in distance stands to the fact that those points are closer to each other. This part in JIT is known as similarity measurement or relevance calculation. Once the weights are computed, the local model is then built. Another important aspect of the JIT model is the structure of the local model that is built. Locally Weighted PLS (LW-PLS) is a prominent structure being employed for many applications. Typically, Euclidean distance is used as a similarity metric to assign weights for LW-PLS. There are other types of similarity measurements such as Correlation-Based JIT (Co-JIT) [10], and angle-based [11]. Co-JIT considers PCA model to evaluate similarities under JIT framework. On the other hand, ensemble similarity measurements have been proposed in JIT framework. In this regard, Yang et al. [12] proposed a two-layer ensemble learning method in JIT, which uses a combination of various similarity metrics to finalize the prediction. The crucial matter in ensemble JIT is to determine the influence of each learning technique in overall prediction, as an inappropriate choice might lead to biased results. Therefore, single-similarity metric (e.g. Euclidean distance, Co-JIT) still prevails over ensemble approaches owing to its less computation cost.

In traditional JIT, a query sample as well as the data points in the historical database are considered to be independent samples. Therefore, they do not account for the time-series nature of the process data. This enabled one to use point-by-point distance based similarity measurement to compute weights for local modeling. Since almost all of the industrial process data are sequentially correlated time-series data, they contain information on magnitude as well as direction of change in the process variables that contain information on varying operating states. This information has not been used in the traditional similarity calculations to the best of our knowledge. Therefore, the objective of this work is to develop a novel similarity measure that accounts for the direction information in the sample selection step to achieve better prediction.

In this work, "Trend Based JIT" approach is therefore proposed. It uses a receding window of recent samples as query window (rather than a single query sample as in

traditional JIT) for weighing the importance of samples in historical database. In other words, the weights for the local learning is based on the query window rather than a point-to-point computation of weights. Given the fixed window size, with moving the window in a receding fashion in the historical data set, one can carry out the trend-based similarity measurement. Then, a LW-PLS is developed based on the proposed similarity calculation. Further, the traditional JIT has limitations to deal with missing data, as practical data are often subject to missing data problem. Therefore, the second objective of this chapter is to extend the proposed JIT to dealing with missing input data. To this end, PPCA is used to impute missing input data prior to local modeling. The efficacy of the proposed methodologies is demonstrated on a synthetic data set and a Near-Infrared (NIR) spectroscopy data set from oil sands industry.

The rest of this chapter is organized as follows. Trend-Based JIT is proposed in Section 2.3 which uses PCA to perform similarity calculations in deterministic framework. In section 2.4, in order to deal with missing input, Probabilistic PCA (PPCA) is used to compute similarity, which is solved using Expectation-Maximization (EM) algorithm. In case studies section, a numerical test is first presented to illustrate the attributes of the proposed method. Further, the effectiveness of both proposed approaches is demonstrated using the NIR data set obtained from the oil sands industry. Finally, conclusions are presented.

2.2 Traditional Just-In-Time Learning

Just-In-Time learning is an online framework to model processes with non linearity or frequently changing operating conditions. It tries to build as many local models as necessary considering the data points most relevant to a newly arriving sample, also called query sample. Just-In-Time (JIT) learning consists of four main steps as shown in Figure 2.1. They are as follows: (a) arrival of a query sample, (b) relevance calculation or similarity measurement, (c) building a local model, and (d) making a prediction.

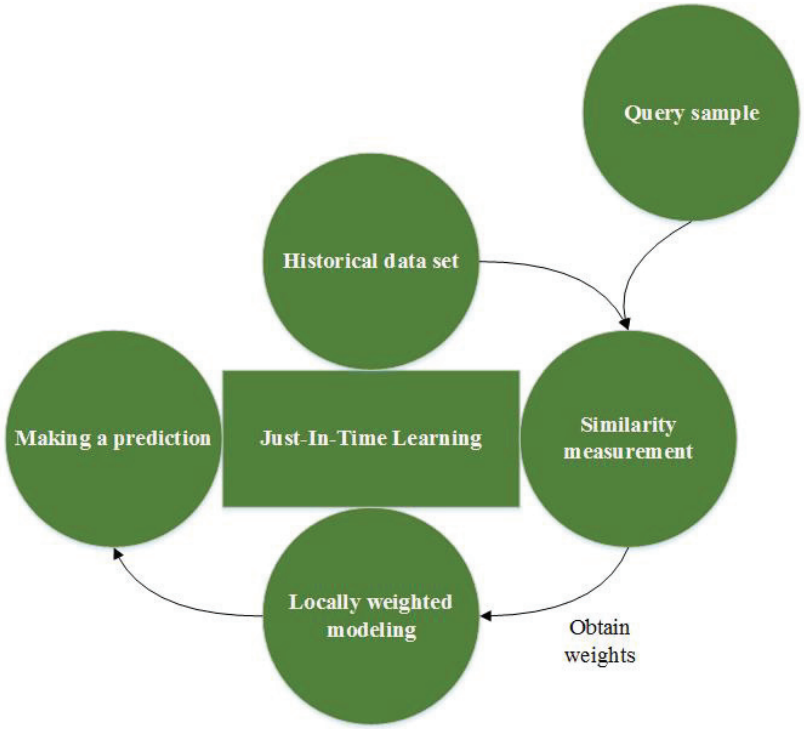

Figure 2.1: *Steps in JIT modeling*

JIT is a space-weight method meaning that weight is assigned to individual samples in the historical database during local modeling. In general, the weights are computed on the basis of the assumption that the samples are independent but spatially related, and are not treated as time-series data. This step is known as similarity measurement or relevance calculation step. There are a number of different similarity measurements such as Euclidean distance [13], angle-based similarity [11], and Correlation-based Just-In-Time (Co-JIT) [10].

Let us consider the M-variable input vector denoted by $\mathbf{x_i} \in R^{M\times 1}$ and single output variable $y_i \in R$ indexed by the sampling instant.

$$X = [\mathbf{x_1}\ \mathbf{x_2} \cdots\ \mathbf{x_N}]^T \tag{2.1}$$

$$\mathbf{y} = [y_1\ y_2 \cdots\ y_N]^T \tag{2.2}$$

Also, let the newly arriving (query) sample be denoted as $\mathbf{x_q} \in R^{M\times 1}$ where N denotes the number of samples and M denotes the number of input variables. The

most routine and convenient similarity measure is Euclidean distance:

$$d_i = \sqrt{(\mathbf{x_i} - \mathbf{x_q})(\mathbf{x_i} - \mathbf{x_q})^T} \qquad i = 1, 2, \cdots, N \tag{2.3}$$

where d_i denotes the distance between each sample in historical database and a query sample. To assign weight to each sample, different mapping functions can be employed among which exponential mapping is one of the most popular ones, and is given by

$$w_i = exp(-\frac{d_i^2}{\sigma_d}) \qquad i = 1, 2, \cdots, N \tag{2.4}$$

where w_i is the weight assigned to each sample and σ_d is a hyperparameter usually tuned through Leave-One-Out-Cross-Validation (LOOCV)[14]. Let us denote a diagonal weighting matrix:

$$W = diag(w_1, w_2, ..., w_i, ..., w_N) \tag{2.5}$$

Owing to the chemometric applciation under consideration, in this work, we use PLS as our local model structure due to its innate capability of addressing collinearity issue. Mathematically, the PLS model can be expressed as:

$$X = TP^T + E_X \tag{2.6}$$

$$\mathbf{y} = T\mathbf{q}^T + \mathbf{e_y} \tag{2.7}$$

where E_x and $\mathbf{e_y}$ are the error terms for input and output respectively. The widely used algorithm for LW-PLS is as follows [15], [16]:

1. Parameter definition:

$T = [\mathbf{t_1}, \mathbf{t_2}, ..., \mathbf{t_k}, ..., \mathbf{t_L}]$ are latent variables where each $\mathbf{t_k}$ is a vector with N samples. L denotes the number of latent variables for LW-PLS model.

$P \in R^{M \times L}$ is the loading matrix of X, and $\mathbf{p_k}$ is a vector with M variables

$\mathbf{q} \in R^{1 \times L}$ is the regression coefficients where each q_k is a scalar

$\mathbf{u_k}$ a weight vector with M elements

2. Evaluate similarity metric d_i for a given query sample, $\mathbf{x_q}$.

3. Assign weights using the mapping function $w_i = f(d_i)$ defined in equation (2.4), or any other suitable weighting function.

4. Obtain the corresponding mean-centered input matrix, output vector and query sample using

$$\bar{\mathbf{x}} = \frac{\sum_{i=1}^{N} w_i \mathbf{x_i}}{\sum_{i=1}^{N} w_i} \tag{2.8}$$

$$\bar{y} = \frac{\sum_{i=1}^{N} w_i y_i}{\sum_{i=1}^{N} w_i} \tag{2.9}$$

$$\tilde{X} = X - 1_N \bar{\mathbf{x}} \tag{2.10}$$

$$\tilde{\mathbf{y}} = \mathbf{y} - 1_N \bar{y} \tag{2.11}$$

$$\tilde{\mathbf{x_q}} = \mathbf{x_q} - \bar{\mathbf{x}} \tag{2.12}$$

where $\bar{\mathbf{x}}$ is a vector with M elements, $\bar{y}$ is scalar, and 1_N is a column vector with N elements of one.

5. Set the number of latent variables for LW-PLS as L. Set k=1.

6. Initialize the prediction of output at query point $\hat{y_q} = \bar{y}$.

7. Evaluate loading, score and weights for each latent variable:

$$\mathbf{u_k} = \tilde{X}^T W \tilde{y} \tag{2.13}$$

$$\mathbf{t_k} = \tilde{X} \mathbf{u_k} \tag{2.14}$$

$$\mathbf{p_k} = \frac{\mathbf{t_k}^T W \tilde{X}}{\mathbf{t_k}^T W \mathbf{t_k}} \tag{2.15}$$

$$q_k = \frac{\mathbf{t_k}^T W \tilde{\mathbf{y}}}{\mathbf{t_k}^T W \mathbf{t_k}} \tag{2.16}$$

8. Perform the deflation step as

$$\tilde{X} = \tilde{X} - \mathbf{t_k} \mathbf{p_k} \tag{2.17}$$

$$\tilde{y} = \tilde{y} - \mathbf{t_k} q_k \tag{2.18}$$

$$t_{q,k} = \tilde{\mathbf{x_q}} \mathbf{u_k} \tag{2.19}$$

where $t_{q,k}$ is the latent variable vector specifically for query sample.

$$\tilde{\mathbf{x_q}} = \tilde{\mathbf{x_q}} - t_{q,k} \mathbf{p_k} \tag{2.20}$$

9. Predict the query sample output as

$$\hat{y_q} = \hat{y_q} + t_{q,k} q_k \tag{2.21}$$

10. If $k = L$, terminate. Otherwise, go to step 4.

2.3 Trend-based Just-In-Time Learning

In traditional JIT, the distance between a query sample and each sample in database is usually calculated by Euclidean distance. This method is capable of finding the most relevant samples to the query sample in terms of spatial relations. However, if the query sample is part of a changing trend, single point will not be able to capture the magnitude and the direction of that change simultaneously. Rather, it only finds their closeness in space. In other words, a similarity measure that is able to distinguish changes with respect to their direction and magnitude can eventually improve the prediction in JIT framework. The illustration of this concept is given in Figure 2.2. Let us assume a one dimensional case where t_q is the query sample with a value of x_q. There are six points having the similar value as t_q. In traditional Euclidean distance-based JIT, all points from t_1 to t_6 are treated equally. However, t_1, t_3 and t_5 are actually part of a rising trend similar to t_q. On the other hand, t_2, t_4 and t_6 are part of a descending trend. If t_1, t_3 and t_5 receive higher weights than t_2, t_4 and t_6, prediction performance might be improved.

In order to identify the direction, a set of recent sequential samples should be taken into consideration. However, a query sample is a single point in space, which means the direction cannot be applied to it. To resolve this issue, a window of recent preceding samples to query sample can be chosen to form a query window. Having done so, the direction can be defined, since several consecutive points in time are under assessment. Thus, the direction of changes is accounted for in the similarity calculation and it is called trend based similarity measure in this chapter. Correspondingly, a window of the most recent historical data points are chosen and appended to query sample, which means matrices should be used to compare trends against each other. It is important to note that traditional similarity calculations are

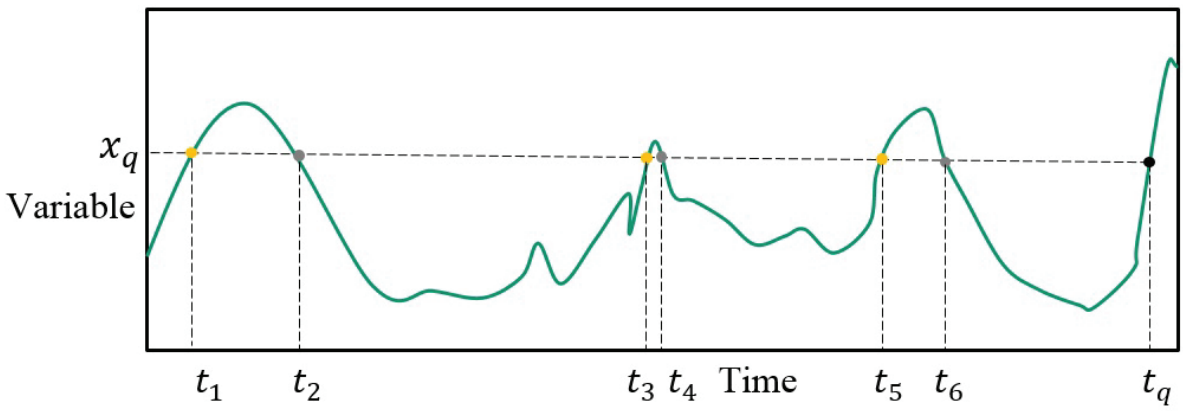

Figure 2.2: *Illustration of Trend-based JIT learning*

based on point calculations of query sample. In this work, we compare a window of samples forming a query matrix to the historical database to do similarity calculation. One of the ways to evaluate similarity of two matrices is through PCA which is a powerful tool in predicting similarities through its error term. When two subspaces are compared, for a given number of latent variables, the subspace in which the samples are more correlated, has a smaller error. In other words, when the query window and a window in historical data base are appended, PCA error is lower if two matrices have high correlation with each other. By using this fact, similarities can be evaluated using correlation. Another advantage of PCA is that it can be used in both deterministic and stochastic settings. Probabilistic PCA (PPCA) can be used as a similarity tool. In traditional JIT approaches, similarity metrics used often have limitation to handle uncertainty. Therefore, a new JIT approach is proposed in this work called " Trend-Based JIT", acronymed as TB-JIT, which uses PCA or PPCA for similarity computation.

Since trends are to be studied here, the method only accounts for time series data. Let us consider that $x_q \in R^{M\times 1}$ is the query sample. Instead of calculating distance point-by-point, a window of the most recent samples is chosen from the historical database X. The essence of applying TB-JIT is to choose a proper window size. The window size is treated as a fixed parameter, meaning that it does not change when the query sample index moves forward in time. Therefore, a query window $X_q \in R^{w\times M}$

is defined where w is the number of samples inside X_q which includes $w-1$ preceding samples from historical data along with the query sample x_q.

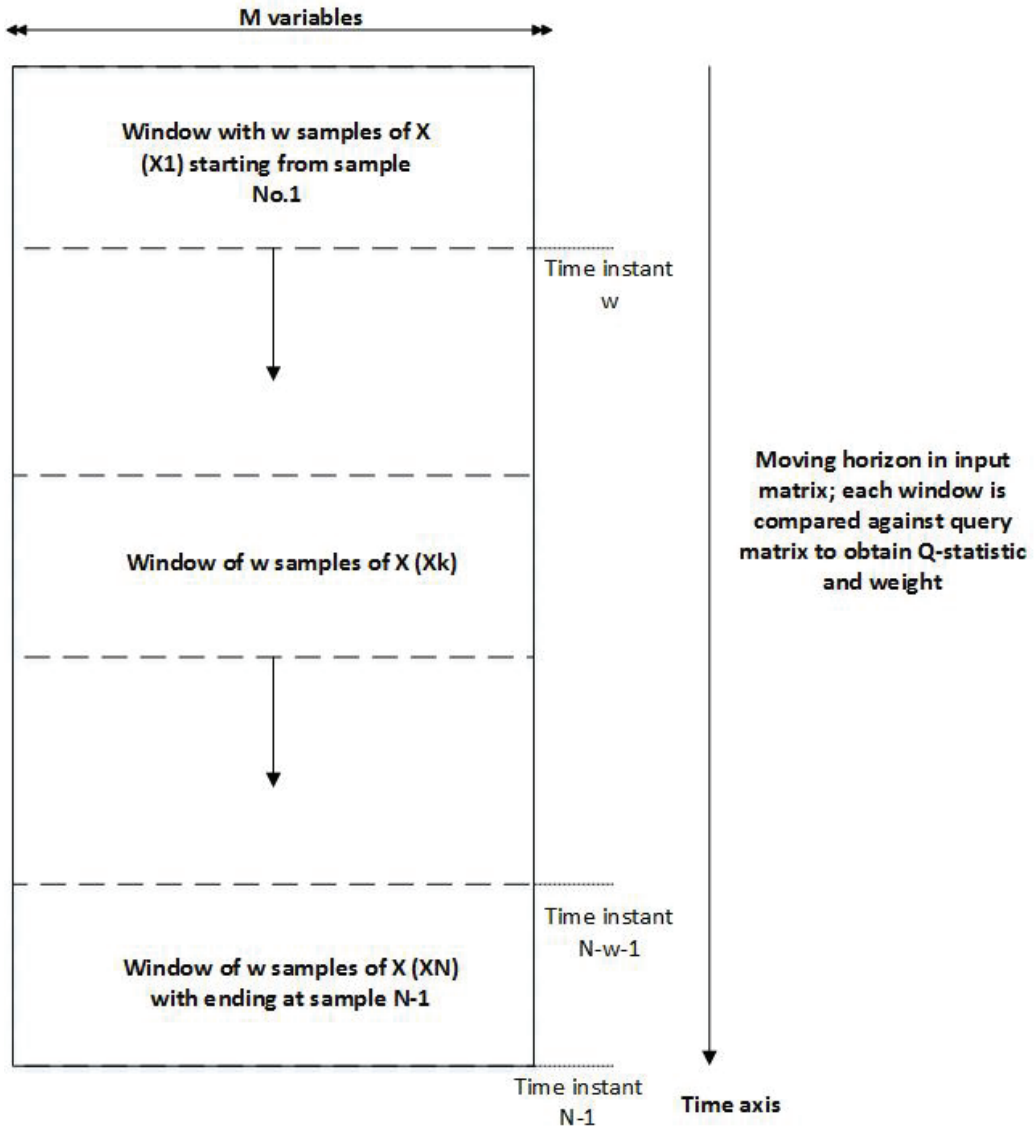

Figure 2.3: *Moving horizon approach in TB-JIT.*

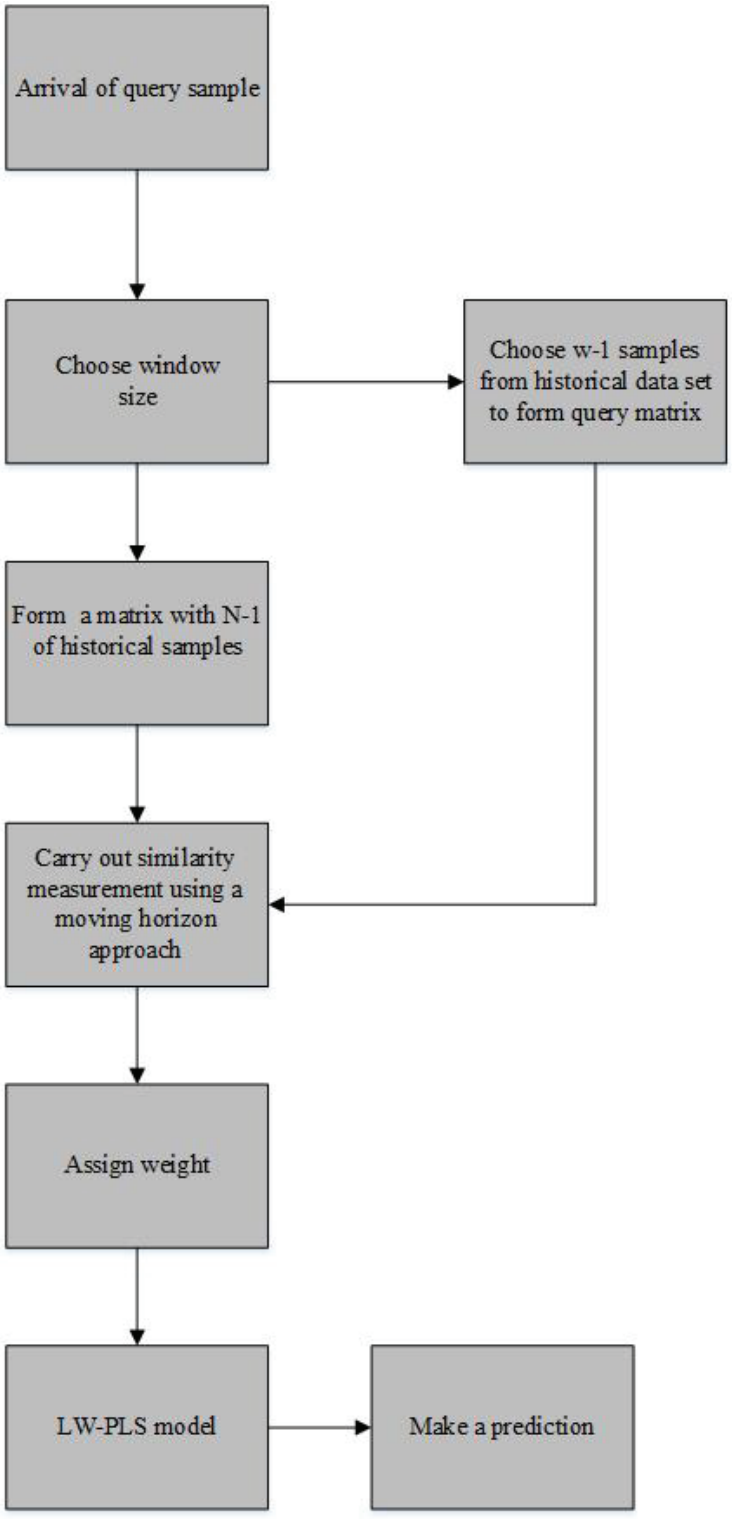

Figure 2.4: *Schematic of TB-JIT.*

Let X_k be a window of historical data where its last sample is the k-th sample from historical data. The number of samples in X_k (or size of X_k) can take any value. In other words, it is not necessary to take same window size between the query window and historical data window. However, we have assumed same window size for both of the windows for ease of presentation .

The comparison metric will be discussed later. Once the similarity measurement calculation is done for the pair $[X_k; X_q]$, the horizon is moved forward one step to become $([X_{k+1}; X_q])$, where $k = w+1, w+2, \cdots, N-1$. It is important to note that the underlying data for our model building is time series data. The moving horizon approach is illustrated in Figure 2.3. Figure 2.4 presents the schematic of the proposed TB-JIT approach.

Following the approach used in Correlation-based JIT [10], a new similarity measure is proposed in this work. In each step for $k = w+1, w+2, \cdots, N-1$, a new matrix $[X_k; X_q]$ is formed. Then, applying PCA on the new matrix yields Q-statistic. As mentioned before, Q-statistic signifies similarity between X_k and X_q. Figure 2.5 demonstrates that instead of a single query sample, a window of query samples are considered in determining the Q statistics which accounts for trends and magnitude in similarity. According to Figure 2.5, the trend of samples and their magnitudes in query space are more similar to samples in subspace II than subspace I. Thus, the overall PCA error when a query space is projected onto subspace II is lower than when it is projected onto subspace I.

Now, let us consider the computation of Q statistic given the window of query samples. To this end, we first define the PCA model as follows:

$$X = SV^T + E \tag{2.22}$$

$$\hat{X} = SV^T \tag{2.23}$$

with $X = [\mathbf{x_1}, \mathbf{x_2}, \cdots, \mathbf{x_N}]^T$ as the observation matrix, $S = [\mathbf{s_1}, \mathbf{s_2}, ..., \mathbf{s_N}]^T$ as latent variables and V as the loading matrix. Each $\mathbf{s_i}$ is a column vector with $\mathbf{s_i} \in R^{H\times 1}$. E is also the PCA error where $E \in R^{N\times M}$. From equations (2.22) to (2.23), the

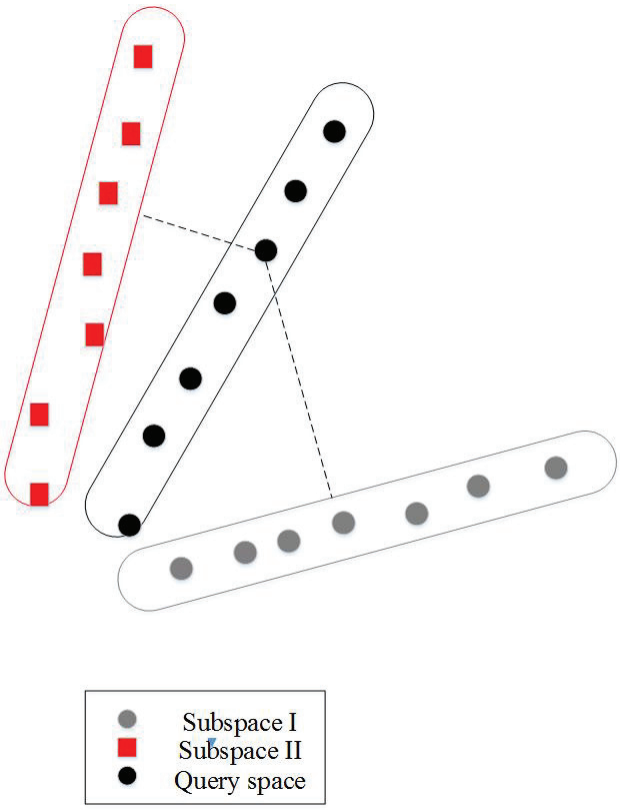

Figure 2.5: *Projection of query space on two subspaces.*

Q-statistic can be obtained for trend-based application:

$$Q = \sum_{i=1}^{2w} \sum_{j=1}^{M} (x_{i,j} - \hat{x_{i,j}})^2 \tag{2.24}$$

where $\hat{x_{i,j}}$ denotes the estimation of $x_{i,j}$.

Likewise, the Hotelling T^2 statistic can also be obtained as,

$$T^2 = \sum_{i=1}^{2w} \sum_{h=1}^{H} \frac{s_{i,h}^2}{\sigma_{s_{i,h}}^2} \tag{2.25}$$

The linear combination of both T^2 statistic and Q-statistic has been proposed by Raich and Sinar [17], which is used in similarity measurement:

$$J = \lambda T^2 + (1 - \lambda)Q \tag{2.26}$$

where λ is a tuning parameter between 0 and 1.

$$0 \leq \lambda \leq 1 \tag{2.27}$$

Using the defined parameter J, one can actually assign weights to samples for the local modeling using the following weighting function:

$$w_i = \frac{1}{1 + J_i} \tag{2.28}$$

where w_i is the weight of each sample, and J_i is obtained using equation (2.26).

2.4 Development of Trend-based Just-In-Time approach with missing input data

2.4.1 PPCA description of missing input data

Missing data is commonly encountered in industrial data sets. Trend-based similarity measure uses deterministic PCA in similarity measurement. To deal with missing input data, regular PCA requires one to impute the inputs prior to performing similarity measurement. On the other hand, Probabilistic PCA (PPCA) has been developed to handle missing input data in dimension reduction and latent variable extraction methods [18].

The main reason for which missing data happens is usually sensor failures. If the missing occurs instantaneously at some sample points, or it occurs for a very short period of time, then one can simply replace the missing values with their previous values. However, the overall prediction might suffer particularly when the number of missing data is large. Probabilistic methods can inherently handle this problem. In the previous section, a similarity measure is developed based on deterministic PCA for the proposed Trend Based JIT modeling. In this section, previously discussed similarity calculation is extended to enable the use of PPCA such that missing inputs can be handled effectively, owing to the probabilistic nature of the feature extraction method adopted.

Consider a PPCA model for each sample as follows [18]:

$$\mathbf{x_i} = V\mathbf{s_i} + \mu + \mathbf{e_i} \qquad i = 1, 2, \cdots, 2w \tag{2.29}$$

where $\mathbf{x_i}$ is a vector of observations with M variables, $\mathbf{s_i}$ is a vector of latent variables with H features, $V \in R^{M \times H}$ is the loading matrix, and μ is the bias term which is also

a vector with M parameters. Further, the noise term $\mathbf{e_i}$ is assumed to be isotropic Gaussian noise.

$$\mathbf{e_i} \sim N(0, \sigma^2 I) \tag{2.30}$$

The latent variables are assumed to follow a zero mean distribution with variance of I.

$$\mathbf{s_i} \sim N(0, I) \tag{2.31}$$

The resulting conditional distributions can be expressed as below [18]:

$$\mathbf{x_i} \sim N(\mu, VV^T + \sigma^2 I) \tag{2.32}$$

$$\mathbf{x_i}|\mathbf{s_i} \sim N((V\mathbf{s_i} + \mu), \sigma^2 I) \tag{2.33}$$

$$\mathbf{s_i}|\mathbf{x_i} \sim N(Z^{-1}W^T(\mathbf{x_i} - \mu), \sigma^2 Z^{-1}) \tag{2.34}$$

where $Z = W^T W + \sigma^2 I$ is a symmetric matrix of size H.

These are the general equations for solving a PCA problem from a Maximum-Likelihood point of view. In the presence of missing data, an element-by-element approach is often considered[19]:

$$x_{i,j} = \mathbf{v_j}\mathbf{s_i} + \mu_j + e_{j,i} \tag{2.35}$$

with $i = 1, 2, \cdots, 2w$ signifying sample index, and $j = 1, 2, \cdots, M$ signifying variable index. v_j is also the j-th input variables vector consisting of H latent variables.

$$P(\mathbf{s_i}) \sim N(0, I) \tag{2.36}$$

$$P(e_{i,j}) \sim N(0, \sigma^2) \tag{2.37}$$

The objective here is to find the Q-statistic for calculating the similarity. The Q-statistic is the variance of error of PPCA, in this case σ^2. The model parameters in PPCA are:

$$\theta = (V, \sigma^2, \mu) \tag{2.38}$$

To find the model parameters, one needs to solve a Maximum-Likelihood problem by Expectation- Maximization(EM) algorithm owing to the presence of missing data.

2.4.2 Obtaining model parameters

In this subsection, parameters of PPCA model are derived. To build local PPCA models for evaluating similarity, one needs to use EM algorithm due to presence of missing data. Let X_o denote the observed part of the data and X_m the missing part. The first step of EM (E-step) is evaluating expectation of the complete log-likelihood of data with respect to marginal distribution of missing part. The marginal distribution is:

$$P(X_m|\theta) = \prod_{j=1}^{M}\prod_{i=1}^{2w} N(\hat{x}_{i,j}, \sigma^2) \tag{2.39}$$

where $\hat{x}_{i,j} = \mathbf{v_j^T s_i} + \mu_j$ which is the noise-free observation.

The details on obtaining PPCA parameters and how the problem is solved can be found in [19]. The final expected log-likelihood for the observed part will be as

$$log(P(X_o|\theta)) = -\frac{2wM}{2} log(2\pi\sigma^2) - \frac{1}{2\sigma^2}\sum_{observed}(x_{i,j} - \hat{x}_{i,j})^2 - \frac{1}{2\sigma^2}\sum_{missed}((\bar{x}_{i,j} - \hat{x}_{i,j})^2 + \sigma^2) \tag{2.40}$$

In the M-step, the objective is to find model parameters (V, μ, σ^2) through maximizing the expected log-likelihood. In presence of missing values in PPCA, the equations over which we iterate are as follows:

$$\Sigma_{\mathbf{s_i}} = \sigma^2(\sigma^2 I + \sum_{j=1}^{M} \mathbf{v_j^T v_j})^{-1} \tag{2.41}$$

$$\bar{s}_i = \frac{\Sigma_{\mathbf{s_i}}}{\sigma^2}\sum_{j=1}^{M} \mathbf{v_j^T}(x_{i,j} - \mu_j) \tag{2.42}$$

$$\mu_j = \frac{1}{M}\sum_{i=1}^{2w}(x_{i,j} - \mathbf{v_j}\bar{\mathbf{s}}_\mathbf{i}) \tag{2.43}$$

$$\mathbf{v_i} = (\sum_{i=1}^{2w} \bar{\mathbf{s}}_\mathbf{i}(x_{i,j} - \mu_j))(\sum_{i=1}^{2w}(\Sigma_{\mathbf{s_i}} + \bar{\mathbf{s}}_\mathbf{i}\bar{\mathbf{s}}_\mathbf{i}^\mathbf{T})^{-1}) \tag{2.44}$$

$$\sigma^2 = \frac{1}{2w}\sum_{j=1}^{M}\sum_{i=1}^{2w}(x_{i,j} - \mathbf{v_j}\bar{\mathbf{s}}_\mathbf{i} - \mu_j)^2 + \mathbf{v_j}\Sigma_{\mathbf{s_n}}\mathbf{v_j^T} \tag{2.45}$$

where $\Sigma\mathbf{s_i}$ denotes the covariance for posterior of each $\mathbf{s_i}$, $\bar{\mathbf{s}}_\mathbf{i}$ the mean of latent variables in each sample, μ_j the biased term of each variable, and $\mathbf{v_j}$ defines each row in the loading matrix. An arbitrary set of values can be chosen to initialize model parameters. It can be inferred from equation (2.45) that the variance of observations depends not only on the reconstruction error but also on the variance of the latent variables. Also, it should be noted that this is indeed the Q statistic used in the relevance calculation. To find the optimal set of parameters, one should iterate over equations (2.41) to (2.45) until convergence. Once σ^2 is obtained, it can be used to evaluate J using equation (2.26).

2.5 Implementation procedure

The procedure to implement TB-JIT with or without missing inputs is described as follows:

1. Arrival of a query sample.
2. Choose a window size w.
3. Form a query matrix X_q that consists of the query sample and the most recent $w-1$ samples immediately before the current query sample
4. Start the moving horizon approach; set $k = w+1$, and form a matrix of X that starts from sample 1 and ends at sample $w+1$.
5. Append query matrix X_q and window k of X to form a new matrix on which similarity measurement is performed.
6. If there is no missing data, follow equations (2.22), (2.23), (2.24) and (2.25).

In presence of missing data, iterate over equations (2.41) to (2.45) until convergence. Impute the estimated missing values into historical data.

7. Obtain J value from equation (2.26).
8. Assign weight to sample k based on obtained value J and equation (2.28).
9. Set $k = k+1$ and go to step 4. If $k = N-1$ stop.
10. Use the obtained diagonal weight matrix to train the LW-PLS model.
11. Make the prediction for the query sample.

2.6 Case Studies

This section serves to demonstrate the effectiveness of the proposed TB-JIT in presence of missing data in the inputs. A numerical example and an Near-InfraRed (NIR) data set from a refinery are studied here. First, the overall performance of TB-JIT is compared against traditional JIT. Second, TB-JIT with missing input data is also studied to assess its performance against simple methods in treating missing values. In all the cases, model structure used is LW-PLS. When one intends to start prediction using TB-JIT, a window size is taken, for instance 10. In TB-JIT, it is not possible to make predictions for samples 1 to 10, which means prediction has to start from sample 11. That is because the window size is 10, sample indexes 1 to 10 cannot have a window of length 10. This is applicable to both the illustrations.

2.6.1 Numerical Example

First, a numerical example is designed to test the proposed JIT approach. Consider a model that has three inputs and one output. Figure 2.6 presents the generated inputs which consist of 200 data points. The output is generated based on inputs shown in Figure 2.6 through a nonlinear function, and can be found in Figure 2.7. The model is nonlinear with quadratic terms on inputs.

$$y_i = \frac{(a{x_{i,1}}^2 + bx_{i,2}^2 + cx_{i,3}^2)}{200} + \epsilon_i \tag{2.46}$$

where each ϵ_i is the noise term for sample i, and all ϵ_i follow a Gaussian distribution. The coefficients a,b,c are random numbers between -0.5 and 0.5. Here they are taken as [-0.1, 0.15, -0.02] respectively. True values are those generated based on equation (2.46.

i) Trend-Based JIT vs Traditional JIT

To apply TB-JIT on a regular data set, we follow the descriptions presented in section 2.5 and use equations (2.24), (2.25), (2.26) and (2.28) to assign weights to the historical data samples. In this example, the window size is fixed to be 7. Figure

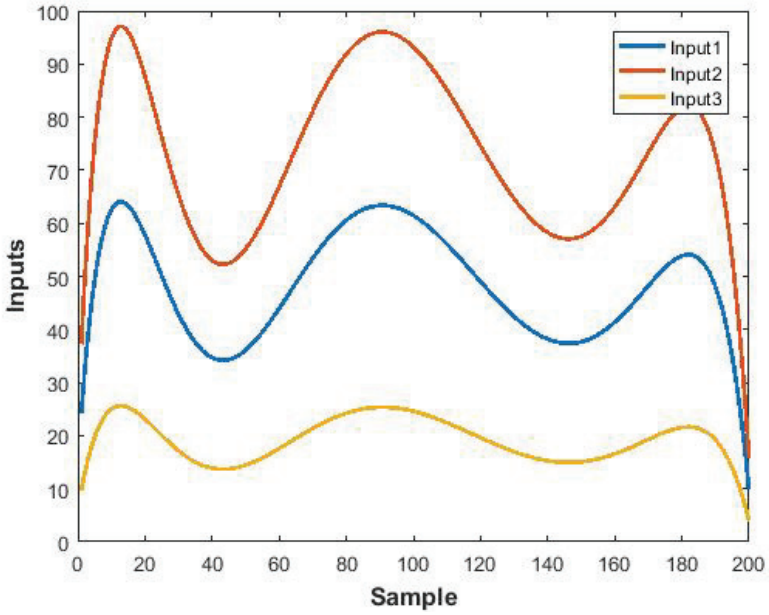

Figure 2.6: *Generated inputs for numerical example.*

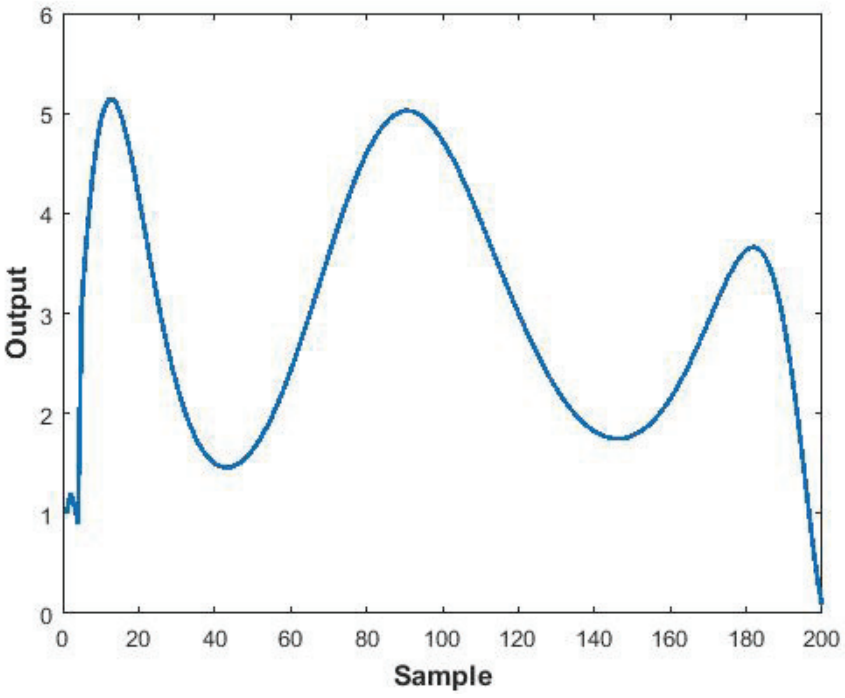

Figure 2.7: *Generated output for numerical example.*

2.8 shows the prediction for traditional Euclidean distance based JIT with proposed TB-JIT when there are no missing input data. It can be seen that TB-JIT provides a better prediction than traditional JIT. Table 2.1 presents the Root Mean Square Error (RMSE) and R^2 values obtained using both the methods. It can be noted that the proposed TB-JIT has lower error and higher R^2 compared to Traditional Euclidean distance based JIT.

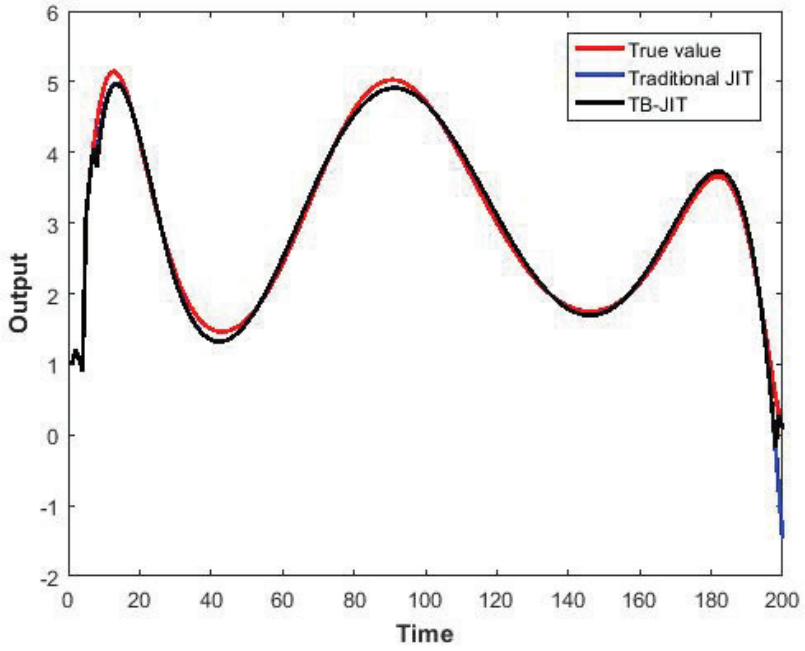

Figure 2.8: *Output prediction for traditional JIT vs.TB-JIT using numerical data set.*

In Figure 2.9, the predictions are validated using reference values.

Table 2.1: *RMSE values for Traditional JIT and TB-JIT for numerical example,* $\lambda = 0.9$.

Methods	**Number of latent variables for LW-PLS**	**RMSE**	R^2
Traditional JIT	2	0.1751	0.95
Trend-Based JIT	2	0.1192	0.98

From Table 2.1, it can be inferred that TB-JIT has a better performance for prediction. The most important step in any JIT modeling procedure is the weighting. Here, weights are assigned based on equation (2.28). Therefore, TB-JIT is able to

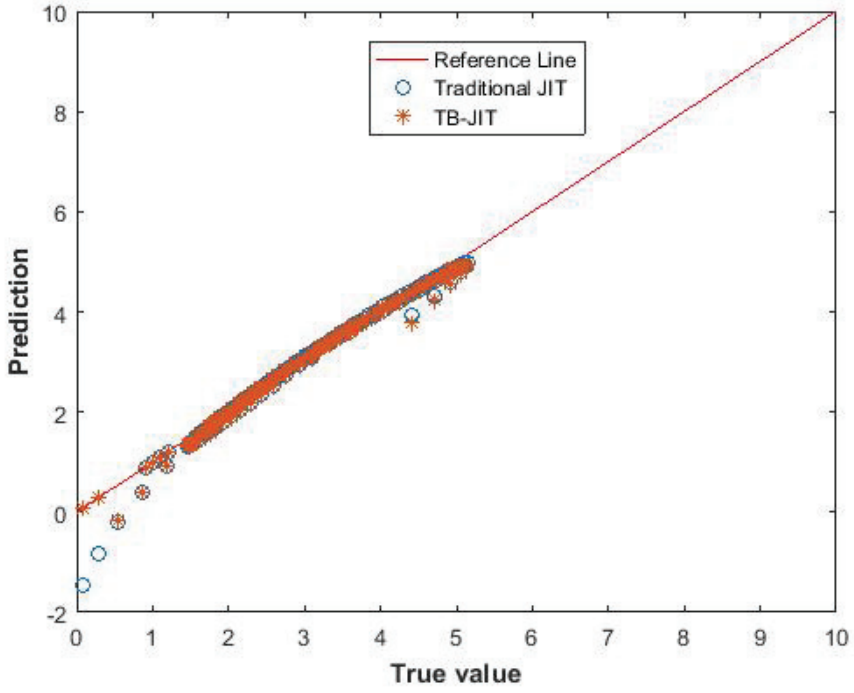

Figure 2.9: *Parity plot of performance of Traditional JIT and TB-JIT using numerical data set.*

improve the prediction due to a better weighting used in our approach in comparison with the traditional JIT approach. As an example, weights in query sample 120 are shown in Figure 2.10. Specifically, let us take a look at samples 75 and 30, as they have almost the same magnitude as that of sample 120. This would be an illustration of Figure 2.2. Euclidean distance assigns similar weight to samples 30 and 75 regardless of their trends in time. However, sample 30 is part of a descending trend just similar as sample 120. On the other hand, sample 75 is part of rising trend. In TB-JIT, we expect that sample 30 receives higher weight than sample 75, which is illustrated in Figure 2.10. This difference in weighting is the reason for better performance of TB-JIT compared to traditional JIT approach.

ii) Trend-Based JIT with respect to missing input

To demonstrate that TB-JIT has the capability to handle missing inputs, the values of certain variables in some samples are randomly missed in the presented numerical data set. TB-JIT with missing input data presented in section 2.4 is applied to the

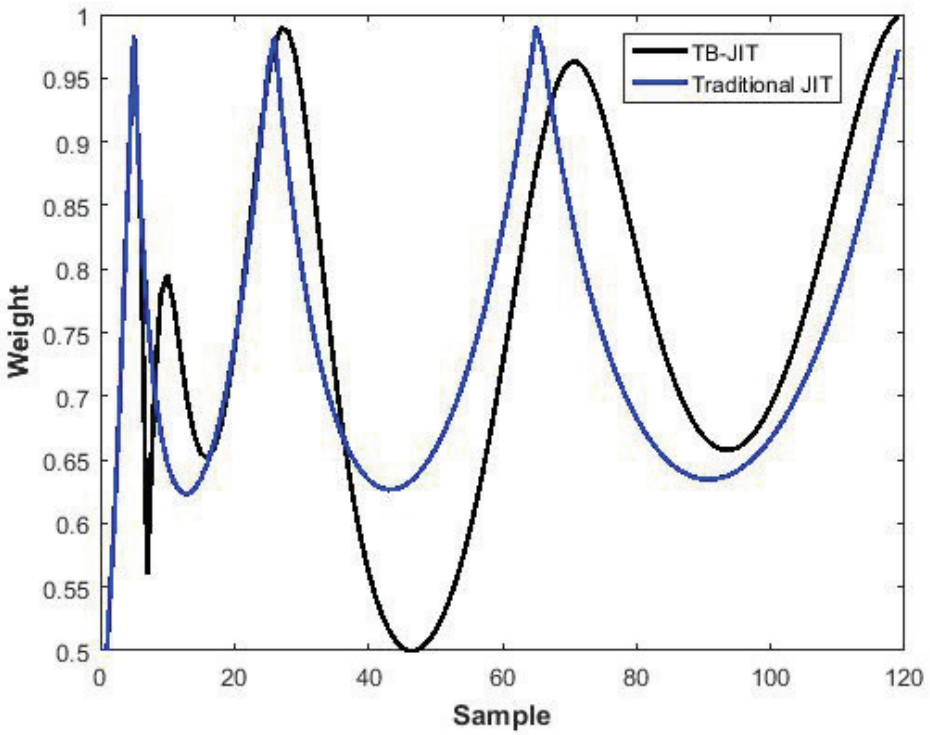

Figure 2.10: *Comparing weights using TB-JIT vs. Traditional JIT for sample 120.*

numerical example. It uses PPCA as a tool to calculate similarities. Furthermore, it can also estimate the missing inputs as part of the solution. In other words, the input imputation for missing inputs is obtained using PPCA.

In this section, the performance of the proposed TB-JIT with missing input data is compared against mean replacement TB-JIT. In this example, the missing inputs are imputed using mean values in the TB-JIT method. Since the predictive performance of TB-JIT in the no missing input case has been shown in the previous example, the difference in performance can be solely attributed to the ability to handle missing values in the PPCA based similarity measure as proposed for TB-JIT with missing input data. Window size is fixed to be 5. The predicted output obtained using TB-JIT with missing input data is shown in Figure 2.11. Also, the predictive performance of TB-JIT with mean imputation is presented.

Figure 2.12 shows the correlation of predictions using both methods. Prediction

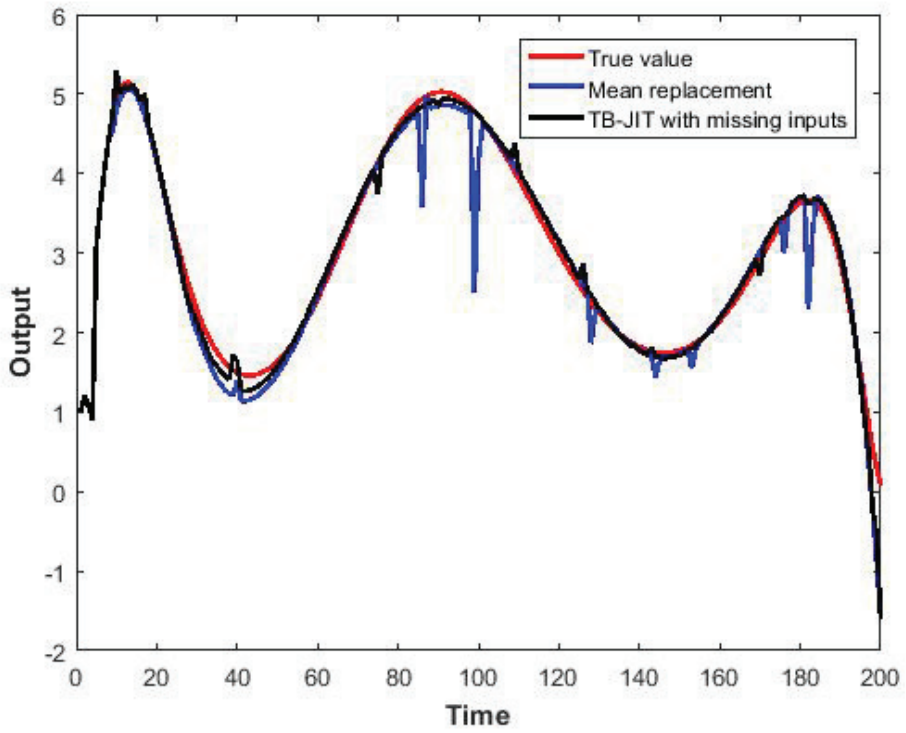

Figure 2.11: *Comparing output prediction using TB-JIT with missing inputs vs. mean replacement.*

through TB-JIT with missing inputs has higher correlation with the real values compared to mean replacement method.

This can also be noted from Table 2.2. Both methods utilize J values from TB-JIT for assigning weights and building locally weighted PLS models.

Table 2.2: *RMSE and R^2 values for Mean replacement and TB-JIT with missing inputs for numerical data set, $\lambda = 0.9$.*

Methods	**Window size**	**RMSE**	R^2
Mean replacement	8	0.300	0.928
TB-JIT with missing inputs	8	0.181	0.981

2.6.2 Viscosity prediction using NIR data

An NIR data set from a refinery is used to test the proposed algorithm. The data set contains spectrum as the input and the target property of interest is viscosity.

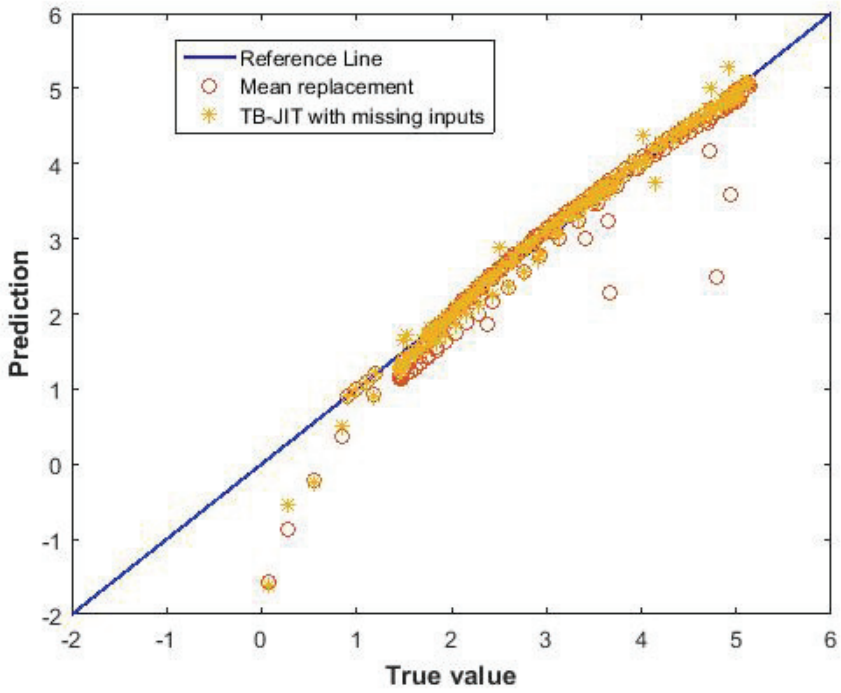

Figure 2.12: *Parity plot of performance of Mean replacement and TB-JIT with missing inputs on numerical data set.*

NIR model has wavelength as the input variables and thus is high dimensional. The wavelengths are highly correlated. For proprietary reason, the data is normalized. The data set consists of 395 samples with 901 recorded wavelengths. Thus, data compression method such as PLS is necessary. The objective of this study is to predict the viscosity of an oil blend. Figure 2.13 presents the NIR scans of the data set in all wavelengths with their spectrum values removed due to propriety reasons. The true values in this case study are viscosity values measured in the LAB.

i) Trend-Based JIT vs Traditional JIT

In this study, a fixed window size of 10 is taken which means the query matrix consists of 10 samples. The λ value used in evaluating J from equation (2.26) is taken as 0.9. Figure 2.14 shows the predictive performance of traditional JIT along with TB-JIT. The model structure under consideration for both approaches is LW-PLS. The prediction obtained using TB-JIT is slightly better than that of Traditional JIT when there is no missing input. Table 2.3 compares the RMSE and R^2 values obtained

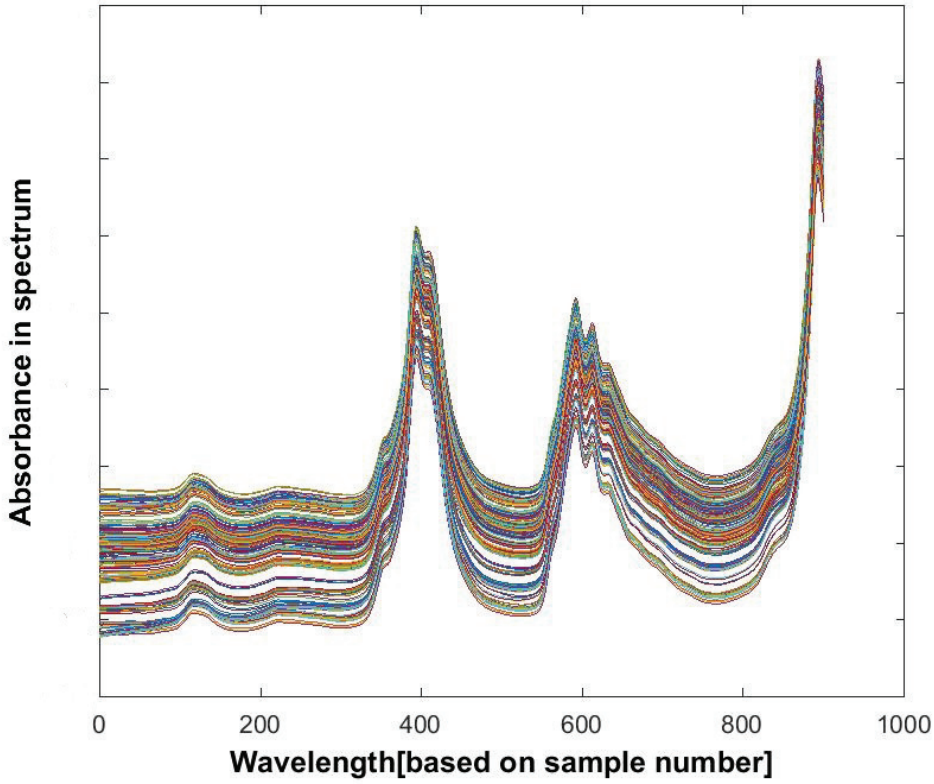

Figure 2.13: *NIR spectrum of the viscosity data set.*

using both methods. In this case study, 10 latent variables are chosen for the LW-PLS model.

Table 2.3: *RMSE values for Traditional JIT and TB-JIT,* $\lambda = 0.9$.

Methods	**Number of latent variables**	**RMSE**	R^2
Traditional JIT	10	0.1801	0.778
Trend-Based JIT	10	0.1641	0.842

Predictions with respect to true values for both methods can be seen in Figure 2.15 where the 45 degree line is the reference line. Closer to it means better agreement between the predicted and the real value. It can be seen that TB-JIT prediction is closer to the reference line. Thus, the proposed TB-JIT is better than the traditional JIT when there is no missing values.

ii) Trend-Based JIT with respect to missing input data

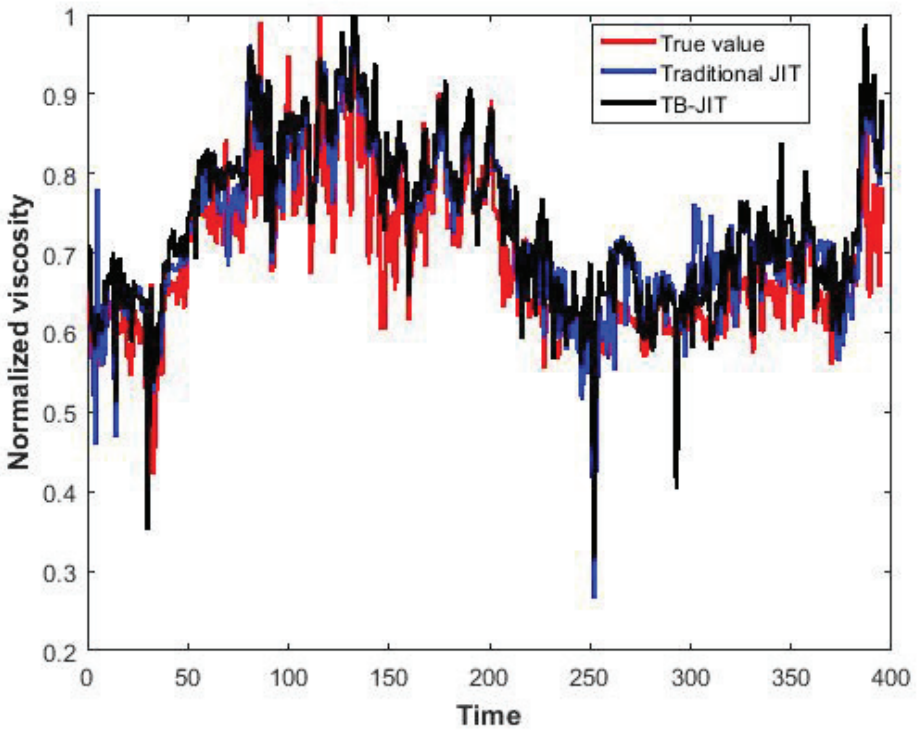

Figure 2.14: *Viscosity prediction for Traditional JIT and Trend-Based JIT.*

Window size is chosen as 5 for both the methods. Iterating over equations (2.41) to (2.45) results in optimal model parameters for PPCA. The loading matrix is used to obtain an estimation of the missing values. As mentioned previously, the variance directly leads to the Q-statistic equation, to ultimately obtain the weights needed for building the local model which is LW-PLS in this case. As for the missing data case, 5% of the whole input data is removed at random.

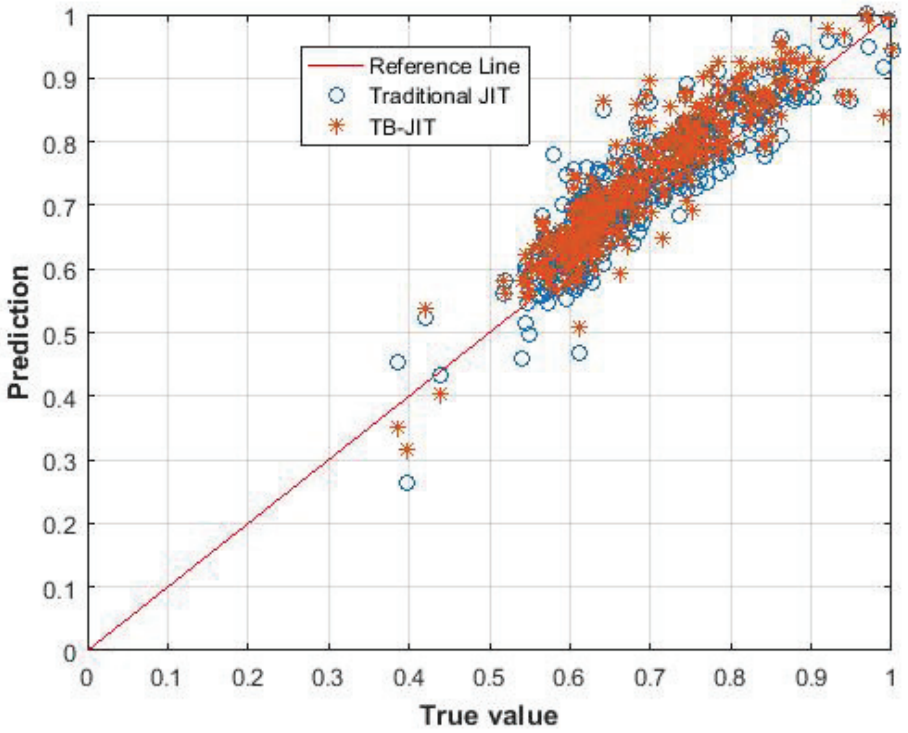

Figure 2.15: *Parity plot of performance of Traditional JIT and TB-JIT.*

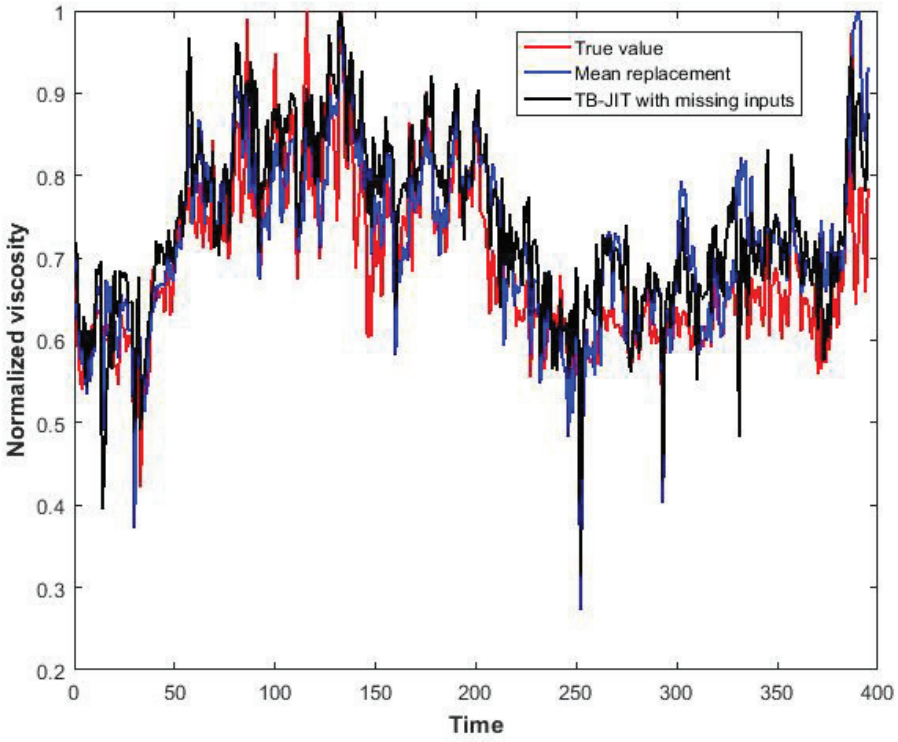

Figure 2.16: *Comparison of viscosity prediction between TB-JIT with mean replacement and TB-JIT with missing inputs.*

Figure 2.16 illustrates the prediction of normalized viscosity using two methods. In mean replacement method, the value of the missing element is estimated by averaging over preceding and succeeding samples of the missing input data point. Then, TB-JIT is used for prediction. On the other hand, TB-JIT with missing inputs utilizes PPCA to estimate the missing input data.

Table 2.4: *RMSE values for mean replacement and TB-JIT with missing inputs,* $\lambda = 0.9$.

Methods	**Window size**	**RMSE**	R^2
Mean replacement	5	0.2283	0.66
TB-JIT with missing inputs	5	0.172	0.81

Table 2.4 presents the RMSE values obtained using the TB-JIT with missing inputs and TB-JIT with mean replacement. It can be noted that PPCA based on the new proposed weight calculation and probabilistic estimation of missing input points result in higher predictive accuracy than the mean replacement approach used in the deterministic PCA. Further, the potential advantage of JIT is its ability in dealing with nonlinear problem using local modeling. Figure 2.17 shows that TB-JIT with missing inputs is closer to true values than mean replacement. In summary, case studies show that TB-JIT with and without missing inputs results in clearly better predictive performance than traditional JIT .

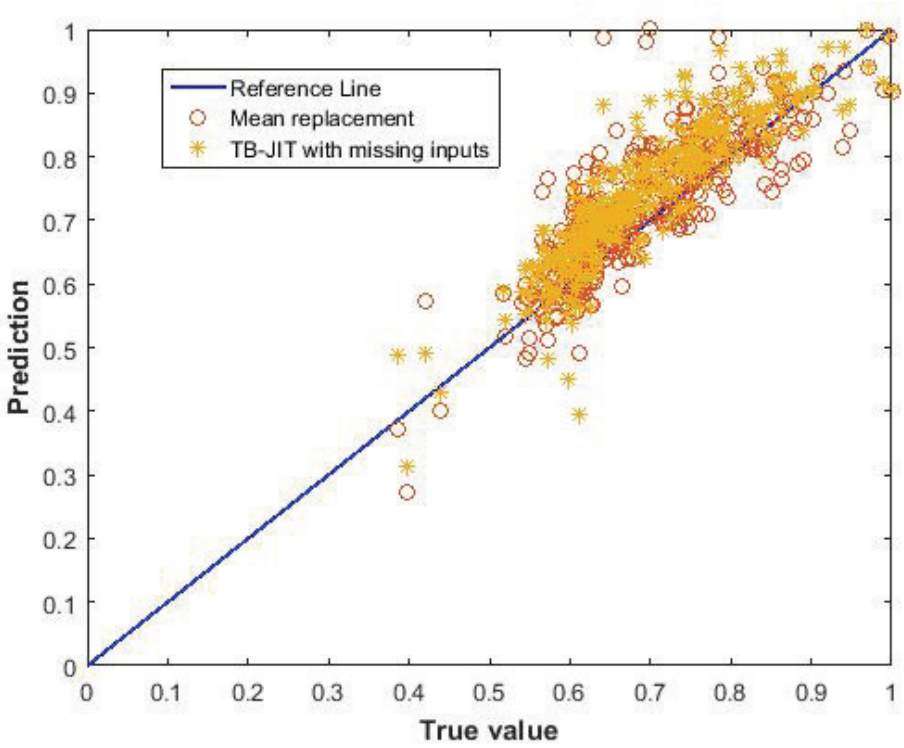

Figure 2.17: *Parity plot of performance of TB-JIT with missing inputs vs mean replacement.*

2.7 Conclusion

In order to account for correlation of samples in similarity calculation of JIT, a novel similarity measure for JIT modeling has been proposed. The proposed approach has considerable advantages over traditional methods in JIT: (a) It can select more appropriate samples for local modeling by considering the sequential correlation in input data. (b) It can account for trends and changes in data set upon arrival of a query sample by employing a moving horizon approach in evaluating similarity. (c) It can be adapted to handle missing input data problem by using PPCA. Owing to the probabilistic nature of PPCA, missing data problem can be handled effectively unlike mean imputation based approaches in deterministic PCA to achieve better predictive accuracy of the JIT model. These characteristics of the proposed method have been demonstrated through two case studies.

Chapter 3

Input-Output Similarity Function for Trend-Based Just-In-Time Learning with Missing Input Data

3.1 Introduction

Global predictive models in regression are widely practiced in process modeling, monitoring [22] and advanced process control. There are a number of different modeling approaches such as Ordinary Least square (OLS) [23], Principle Component Regression [24] (PCR), Partial Least Squares [25] (PLS) and Support Vector Regression [28], some of which build predictive models while keeping the variation of data after dimension reduction.

However, global linear models might fail to provide good prediction when process is nonlinear. To address this problem, recursive and adaptive methods such as recursive PLS [29], [30], [31] or recursive PCR [32] can be useful. Nonetheless, when process is highly nonlinear or operating point changes rapidly, they might not be useful. As a result, predictive locally weighted modeling in regression [33], which is also known as Just-In-Time (JIT) learning [13], has been proposed and can be applied to such problems.

JIT approach and its algorithms have been thoroughly discussed in the previous chapter. In this chapter, the previously proposed Trend-Based JIT (TB-JIT) in chapter 2 is further extended. TB-JIT tries to take advantage of any underlying

correlation within samples to calculate similarities. To do that, the query sample is appended to a number of the most recent historical samples to form query window. Then, the query window is compared against all windows in historical database. All these steps are implemented in input space of a given data set, which stands to the fact that output data is not used in similarity calculation step of JIT. Improvement in prediction of a locally weighted model is likely to be anticipated when output information is also considered in similarity measurement. Hence to enable TB-JIT to deal with output data in similarity calculation, "Input-Output similarity function for Trend-Baesd JIT" (accronymed as IO TB-JIT), is proposed. It considers trends of input data in similarity calculation as proposed in the previous chapter as well as the output data information. In addition to calculating similarity in input space, IO TB-JIT evaluates similarity in output space by providing an initial prediction for the query sample. Once similarity calculation is done for both input and output spaces, weights are assigned to historical samples considering the integrated input/output distance. Further, JIT approach has limitations to deal with missing data. Therefore, the second objective of this chapter is to extend the proposed IO TB-JIT to dealing with missing input data as also considered in the previous chapter. To this end, PPCA is used to impute missing inputs prior to local modeling. The efficacy of the proposed methodologies is demonstrated on a synthetic data set and a Near-Infrared (NIR) spectroscopy data set, respectively.

This chapter is an extension of the previous chapter. To minimize the overlap, fundamentals are not repeated and are referred to the previous chapter wherever is needed. The rest of this chapter is organized as follows. IO TB-JIT is proposed in section 3.2 which uses PCA to perform similarity calculations in deterministic framework, and calculates similarity in output space using an initial PLS model to predict the output of the query sample. In section 3.3, in order to deal with missing input, Probabilistic PCA (PPCA) is used to compute similarity. In the case studies section, a numerical test is first presented to illustrate the features of the proposed method. Further, the effectiveness of both proposed approaches is demonstrated using the NIR data set. Finally, conclusions are presented.

3.2 Trend-Based Just-In-Time Learning with Input-Output similarity measurement

3.2.1 Similarity calculation for input space

In traditional JIT, the distance between a query sample and each sample in database is usually calculated by Euclidean distance. This method is capable of finding the most relevant samples to the query sample in terms of spatial relations. However, if the query sample is part of a changing trend, single point will not be able to capture the magnitude and the direction that may change simultaneously. Rather, it just finds their closeness in space. In other words, a similarity measure that is able to distinguish changes with respect to their direction and magnitude can eventually improve the prediction in JIT framework. This similarity measure is TB-JIT as presented in the previous chapter.

3.2.2 Similarity calculations for output space

Output information can be incorporated in similarity calculation, and use of it will improve the prediction. There are a number of different methods in evaluating similarity in the output space. In this work, the following equation calculates the distances in output space and thus, evaluates similarity:

$$d_{i,y} = \frac{|y_i - \hat{y}_{q,in}|}{\sum_{i=1}^{N} |y_i - \hat{y}_{q,in}|} \tag{3.1}$$

where $\hat{y}_{q,in}$ is an initial prediction for query sample. Normally, a regular model frame is adopted to provide an initial prediction of the query sample, such as global PLS. By using the calculated $\hat{y}_{q,in}$, one would be able to perform similarity calculation in output space.

3.2.3 Similarity measurement for Input-Output Trend-Based Just-In-Time

Using the defined parameter J in equation (2.26) for input space and equation (3.1) for output space, overall distance/dissimilarity in input-output space can be evaluated:

$$D_i = \alpha J + (1 - \alpha) d_{i,y} \tag{3.2}$$

where D_i denotes the overall distance/dissimilarity for historical sample i, and α is a tuning parameter between 0 and 1. It is important to normalize input space PCA error J, and output space distance $d_{i,y}$ prior to combining them in equation (3.2).

One may assign weights to samples for the local modeling using the following weighting function:

$$w_i = \frac{1}{1 + D_i} \tag{3.3}$$

where w_i is the weight of each sample.

3.3 Development of input-output based similarity measure for Trend-based Just-In-Time with missing input data

In this section, calculating input-output similarities in presence of missing data in inputs is discussed. There is normally no missing data in output space since the outputs are usually through manual laboratory analysis. Hence, evaluating similarities in output is similar to that of section 3.3.

3.3.1 Evaluating similarities in input space in presence of missing data for Trend-Based Just-In-Time

Missing data is commonly encountered in industrial data sets. Trend-based similarity measure uses deterministic PCA in similarity measurement. To deal with missing inputs, regular PCA requires one to impute the inputs prior to performing similarity measurement. On the other hand, Probabilistic PCA (PPCA) has been developed to

handle missing input in dimension reduction and latent variable extraction methods [18]. The detailed procedure has been in chapter 2.

3.3.2 Input-Output similarity measurement for Trend-Based Just-In-Time with missing input data

Once PPCA model parameters (2.41) to (2.45) converge, σ^2 is used in equation (2.26) to evaluate similarity in input space. Equation (3.1) is used to evaluate similarity in output space. Similar to output similarity measurement when there is no missing data, one needs to make an initial prediction for query sample. Both similarities are added up in equation (3.2). Finally, equation(3.3) is used to assign weights.

3.3.3 Implementation procedure

The procedure to implement Input-Output TB-JIT with or without missing inputs is summarized as follows:

1. Arrival of a query sample.

- **Layer for input space similarity measurement**

2. Choose a window size w.

3. Form a query matrix X_q that consists of the query sample and the most recent $w-1$ samples from historical data.

4. Start the moving horizon approach; set $k = w+1$, and form a matrix of X that starts from sample number 1 and ends at sample $w+1$.

5. Append query matrix X_q and window k of X to form a new matrix on which similarity measurement is performed.

6. If there is no missing data, follow equations (2.22), (2.23), (2.24) and (2.25).

In presence of missing data, iterate from equations (2.41) to (2.45) until convergence. Impute the estimated missing values into historical data.

7. Obtain J value from equation (2.26).

- **Layer for output space similarity measurement**

8. Build an initial model to predict the output for query sample using regular PLS.

9. Calculate similarity in output space using equation (3.1).

- **Input-Output similarity measurement**

10. Assign weight to sample k based on obtained values using equations (3.3) and (3.2).

11. Set $k = k + 1$ and go to step 4. If $k = N - 1$ stop.

12. Use the obtained diagonal weight matrix in training the LW-PLS model.

13. Make the prediction for the query sample.

3.4 Case Studies

This section serves to demonstrate the effectiveness of the proposed Input-Output TB-JIT (IO TB-JIT) in presence of missing data in the inputs. A numerical example and an Near-InfraRed (NIR) data set are studied here. First, the overall performance of IO TB-JIT is compared against traditional JIT and TB-JIT presented in chapter 2. Second, IO TB-JIT with missing input data is also studied to assess its performance against mean replacement in traditional JIT and previously proposed TB-JIT with missing input data in chapter 2. In all the cases, model structure used is LW-PLS.

3.4.1 Numerical Example

A numerical example is developed to examine the effectiveness of IO TB-JIT. Consider a model that has three inputs and one output. Figure 3.1 presents the generated inputs which consist of 200 data points. The output is generated based on inputs shown in Figure 3.1 through a nonlinear function, and its trajectory can be found in Figure 3.2.

$$\mathbf{y} = \frac{\mathbf{x_1}\ \mathbf{x_2} + log(\mathbf{x_2}\ \mathbf{x_3}) + sin\ \mathbf{x_3}}{2} + \epsilon \tag{3.4}$$

where ϵ is the noise term, and it follows a Gaussian distribution. The true values are those generated based on equation (3.4).

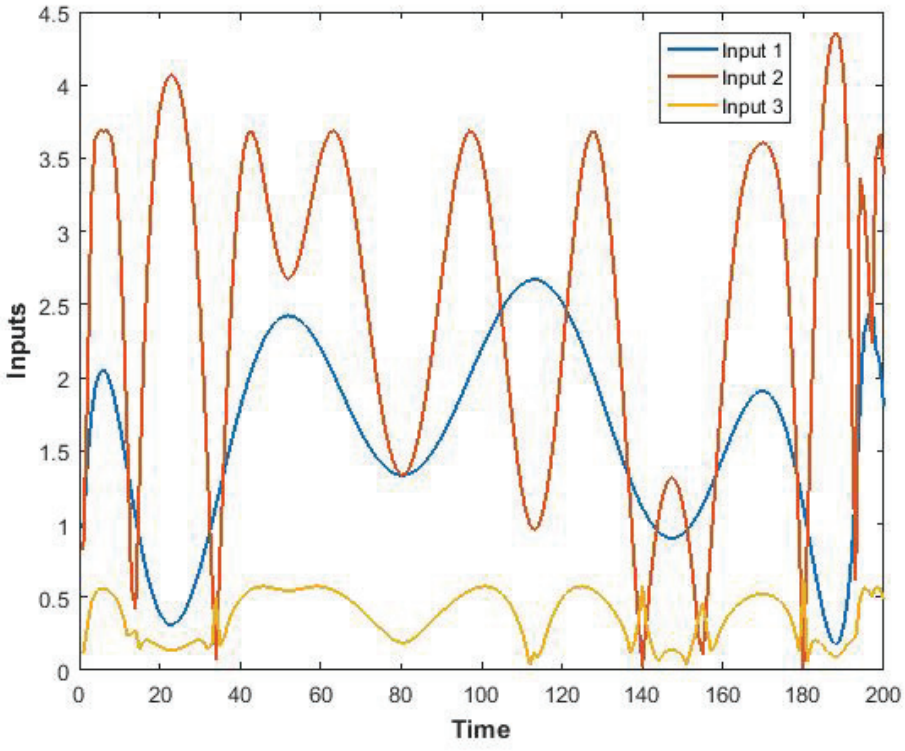

Figure 3.1: *Generated inputs for numerical example.*

i) IO TB-JIT vs Traditional JIT and TB-JIT

To apply IO TB-JIT on a regular data set, we follow the descriptions presented in section 3.3.3. In this example, the window size is fixed to be 10. Figure 3.3 shows the prediction for traditional Euclidean distance based JIT and TB-JIT (with window size of 10) against the proposed IO TB-JIT when there are no missing input data. It can be seen that IO TB-JIT provides a better prediction than other two methods. Table 3.1 presents the Root Mean Square Error (RMSE) and R^2 values obtained using each method. It can be noted that the proposed IO TB-JIT has lower error and higher R^2 compared to traditional Euclidean distance based JIT and TB-JIT.

In Figure 3.4, the predictions are validated using reference values.

From Table 3.1, it can be inferred that IO TB-JIT has a better performance for prediction than traditional JIT and TB-JIT. The most important step in any JIT modeling procedure is weighting. Here, weights are assigned based on equation (3.3)

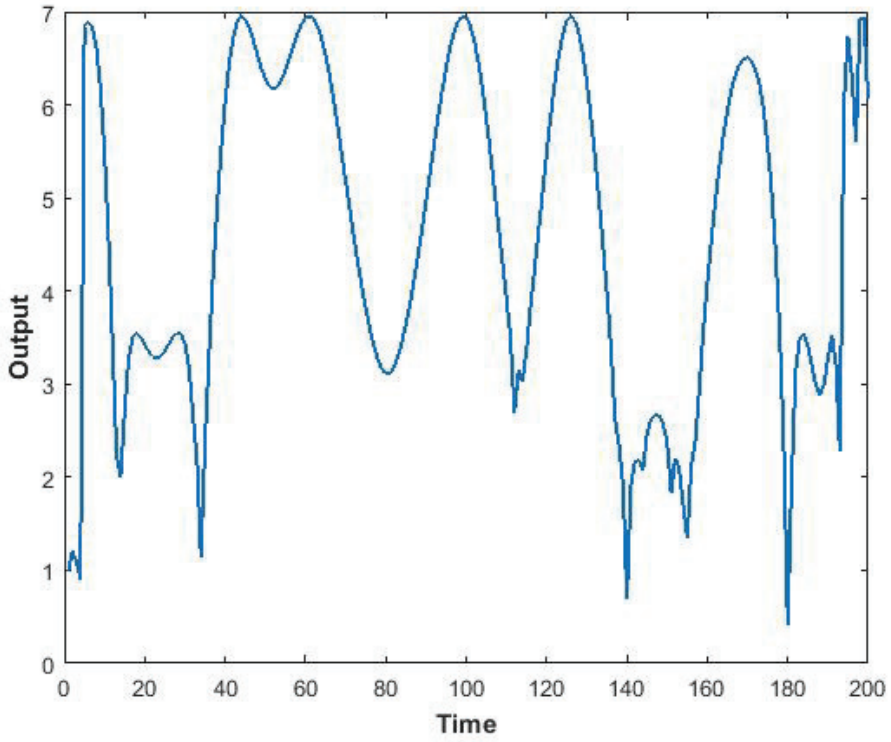

Figure 3.2: *Generated output for numerical example.*

Table 3.1: *RMSE values for Traditional JIT, TB-JIT and IO TB-JIT for numerical example,* $\lambda = 0.9$.

Methods	**Number of latent variables for LW-PLS**	**RMSE**	R^2
Traditional JIT	2	0.7833	0.8057
TB-JIT	2	0.5671	0.862
IO TB-JIT	2	0.5382	0.9083

which uses similarity in both input space and the output space. Therefore, IO TB-JIT is able to improve the prediction due to a better weighting used in this approach in comparison with the traditional JIT approach and TB-JIT. As an example, weights in query sample 90 are shown in Figure 3.5. Specifically, let us take a look at samples 70 and 40, as they have almost the same magnitude as sample 90. Euclidean distance assigns similar weights to samples 40 and 70 regardless of their trends in time. However, sample 40 is part of a rising trend just similar as sample 90. On the other hand, sample 70 is part of descending trend. In

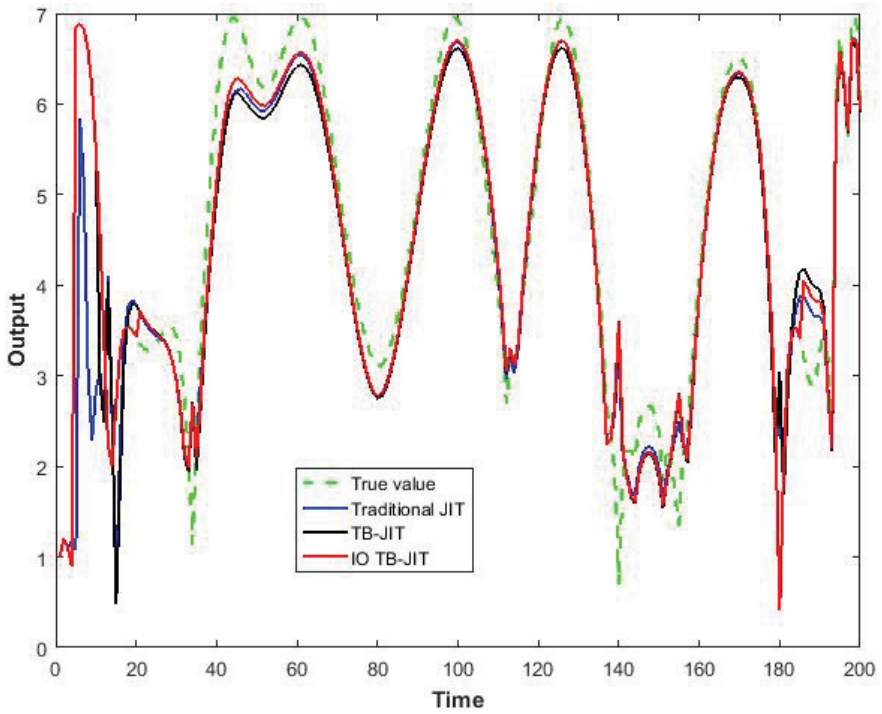

Figure 3.3: *Output prediction for traditional JIT and TB-JIT vs. IO TB-JIT using numerical data set.*

IO TB-JIT, we expect that sample 40 receives higher weight than sample 70, which is illustrated in Figure 3.5. In addition, the impact of including output information in weighting can also be seen, as IO TB-JIT yields better prediction compared to TB-JIT. These differences in weighting are the reason for better performance of IO TB-JIT compared to traditional JIT approach and TB-JIT.

ii) IO TB-JIT with missing input

To demonstrate that IO TB-JIT has the capability to handle missing inputs, the values of certain variables in some samples are randomly missed in the presented numerical data set. IO TB-JIT with missing input data presented in section 3.3 is applied to the numerical example. It uses PPCA as a tool to calculate similarities. Furthermore, it can also estimate the missing inputs as part of the solution. In other words, the input imputation for missing inputs is obtained using PPCA.

In this section, the performance of the proposed IO TB-JIT with missing input

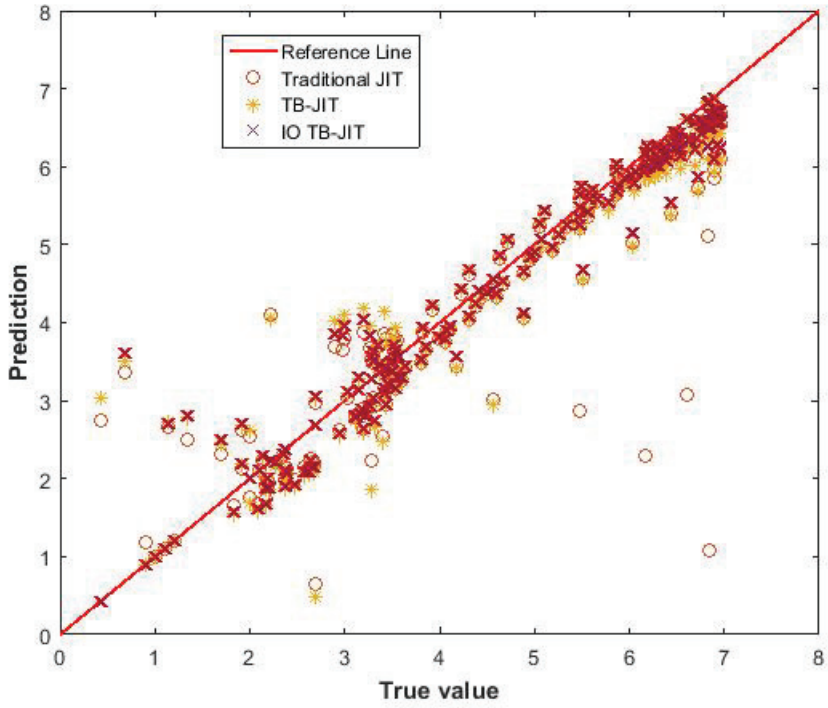

Figure 3.4: *Parity plot of performance of traditional JIT and TB-JIT vs. IO TB-JIT using numerical data set.*

data is compared against mean replacement in traditional JIT and TB-JIT with missing input data. In this example, the missing inputs are imputed using mean values for traditional JIT method. Window size is fixed as 5 for IO TB-JIT and TB-JIT. The predicted output obtained using IO TB-JIT with missing input data is shown in Figure 3.6. Also, the predictive performance of traditional JIT and TB-JIT with missing input data is presented.

Figure 3.7 shows the correlation of predictions for all three methods. Prediction through IO TB-JIT with missing input data has higher correlation with the real values compared to mean replacement method. This can be noted from Table 3.2.

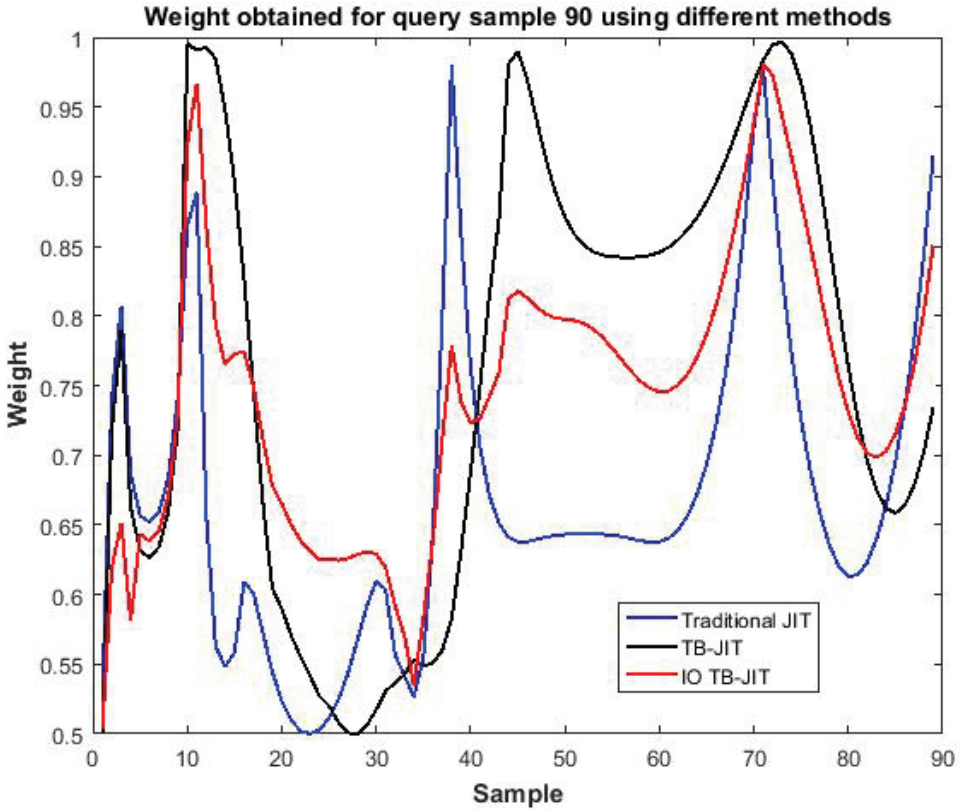

Figure 3.5: *Comparing weights using traditional JIT and TB-JIT vs. IO TB-JIT for sample 90.*

Table 3.2: *RMSE and R^2 values for Mean replacement in traditional JIT and TB-JIT with missing input vs. IO TB-JIT with missing input for numerical data set, $\lambda = 0.9$.*

Methods	No. of latent variables	RMSE	R^2
Mean replacement in traditional JIT	2	0.8167	0.762
TB-JIT with missing input	2	0.7841	0.801
IO TB-JIT with missing inputs	2	0.7421	0.835

3.4.2 Predicting active substance Escitalopram using NIR data

An NIR data set [36] for pharmaceutical application is used to test the proposed algorithm. The data set contains spectrum as the input and the target property of interest is the amount of a chemical known as Escitalopram (which is the active substance in tablets studied in the given data set). NIR model has wavelength as the

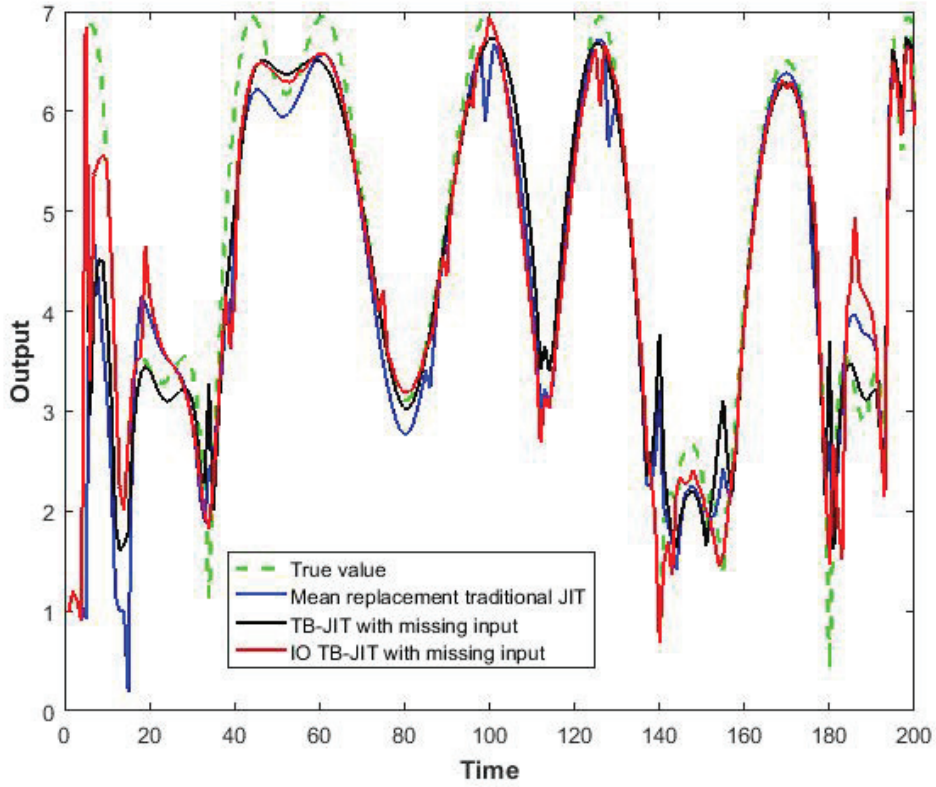

Figure 3.6: *Comparing output prediction using IO TB-JIT with missing inputs vs. mean replacement in traditional JIT and TB-JIT with missing input.*

input variables and thus is high dimensional. The data set consists of 310 samples with 404 recorded wavelengths. The objective of this study is to predict the amount of Escitalopram in certain tablets. Figure 3.8 presents the absorbance spectrum of the NIR scan of the data set. The true values of active substance are obtained based on LAB measurements.

i) IO TB-JIT vs Traditional JIT

In this study, a fixed window size of 10 is taken which means the query window consists of 10 samples. Figure 3.9 shows the predictive performance of traditional JIT and TB-JIT discussed in chapter 2 against IO TB-JIT. The model structure under consideration for both of the approaches is LW-PLS. The prediction obtained using IO TB-JIT is slightly better than that of Traditional JIT or TB-JIT when there is

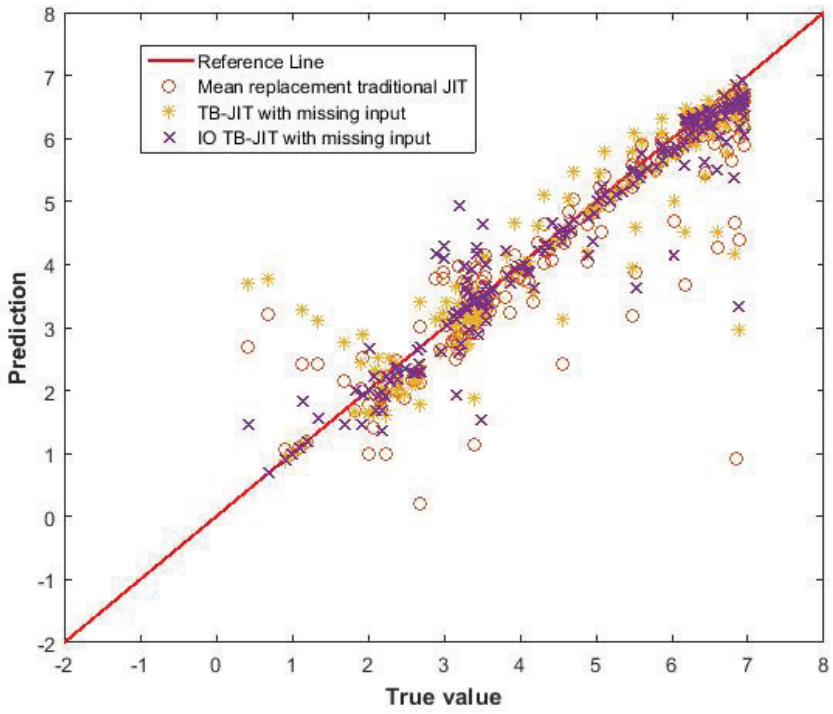

Figure 3.7: *Parity plot of performance of Mean replacement in tradiotnal JIT and TB-JIT with missing input vs. IO TB-JIT with missing inputs on numerical data set.*

no missing input. Table 3.3 compares the RMSE and R^2 values obtained using each method. In this case study, 5 latent variables are chosen for the LW-PLS model.

Table 3.3: *RMSE and R^2 values for traditional JIT and TB-JIT vs. IO TB-JIT, $\lambda = 0.9$* .

Methods	**Number of latent variables**	**RMSE**	R^2
Traditional JIT	5	0.4644	0.854
TB-JIT	5	0.436	0.881
IO TB-JIT	5	0.4215	0.892

Predictions with respect to true values for both methods can be seen in Figure 3.10 where the 45 degree line is the reference line. Closer to it means better agreement between the predicted and the real value. It can be seen that IO TB-JIT prediction is closer to the reference line. Thus, the proposed IO TB-JIT is better than the traditional JIT when there is no missing values.

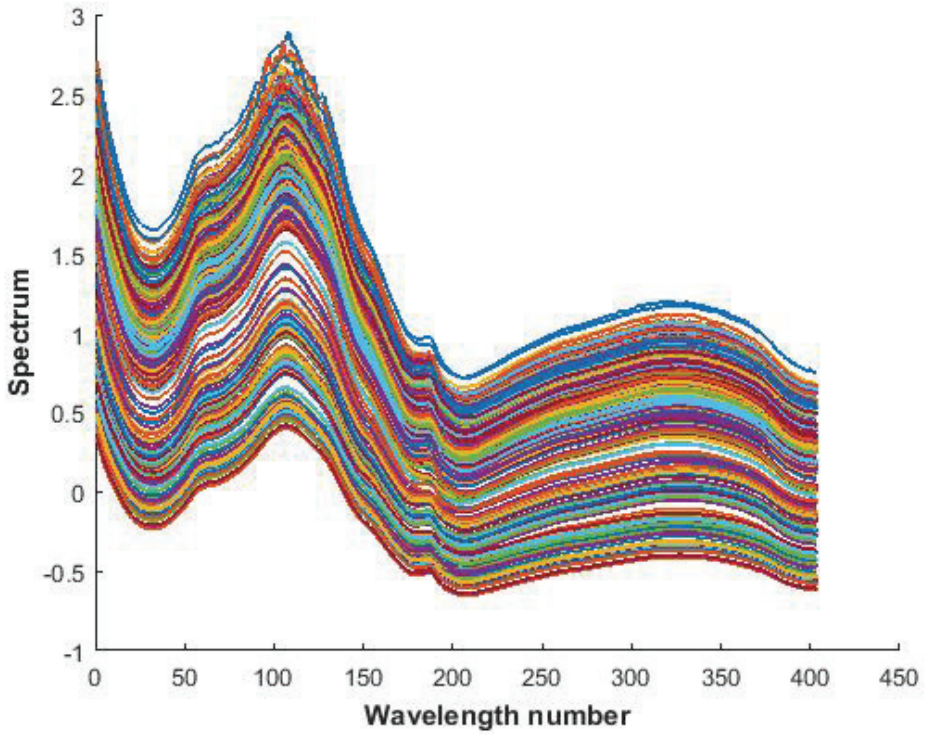

Figure 3.8: *NIR spectrum of the dat set.*

ii) IO TB-JIT with missing input data

Window size is chosen as 5 for IO TB-JIT with missing inputs. Iterating over equations (2.41) to (2.45) results in optimal model parameters for PPCA for input space similarity calculation. Similarity calculation in output space follows the procedure mentioned in section 3.2.2. As for the missing part, 5% of the whole input data is missing at random.

Figure 3.11 compares predictive performance of IO TB-JIT with missing input data against mean replacement for traditional JIT and TB-JIT with missing input data, which has been discussed in the previous chapter.

Table 3.4 presents the RMSE values achieved using the IO TB-JIT with missing input data, the traditional JIT with mean replacement and TB-JIT with missing input data. It can be noted that PPCA based on the new proposed weight calculation and probabilistic estimation of missing input points result in higher predictive accuracy

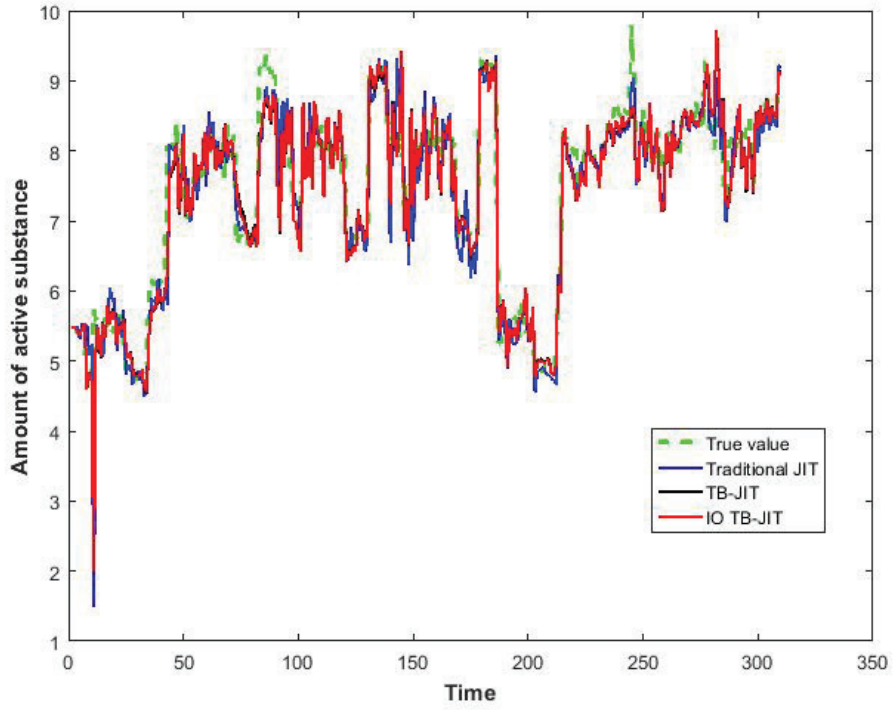

Figure 3.9: *Active substance prediction for Traditional JIT and TB-JIT vs. IO TB-JIT.*

Table 3.4: *RMSE values for mean replacement traditional JIT and TB-JIT with missing input data vs. IO TB-JIT with missing input data,* $\lambda = 0.9$.

Methods	**No.of latent variables**	**RMSE**	R^2
Mean replacement	5	0.5263	0.824
TB-JIT with missing inputs	5	0.481	0.851
IO TB-JIT with missing inputs	5	0.4778	0.865

than the mean replacement approach. In addition, considering output in similarity measurement also improves the performance. Figure 3.12 shows that IO TB-JIT with missing input data is closer to true values than mean replacement. In summary, case studies show that IO TB-JIT with and without missing inputs results in clearly better predictive performance than traditional JIT as well as than previously discussed TB-JIT.

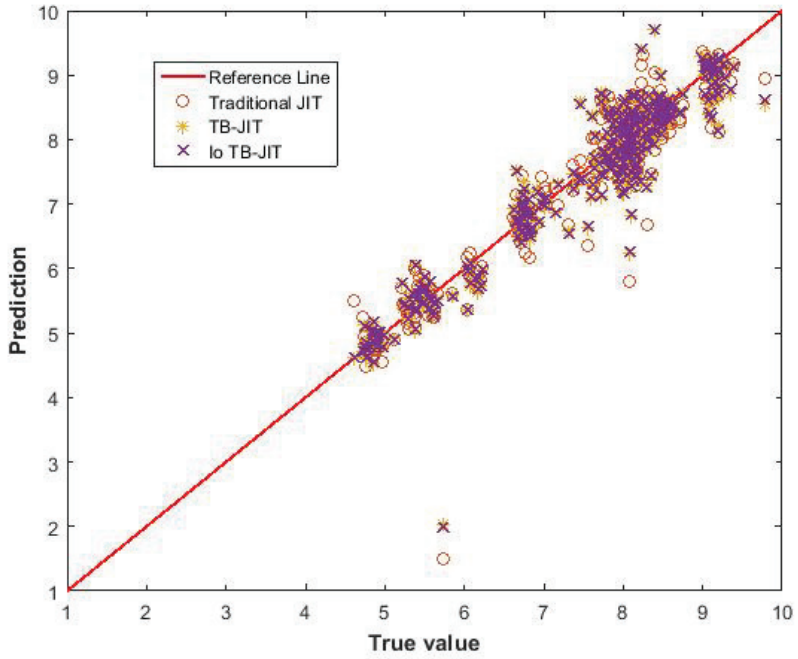

Figure 3.10: *Parity plot of performance of Traditional JIT and TB-JIT vs. IO TB-JIT.*

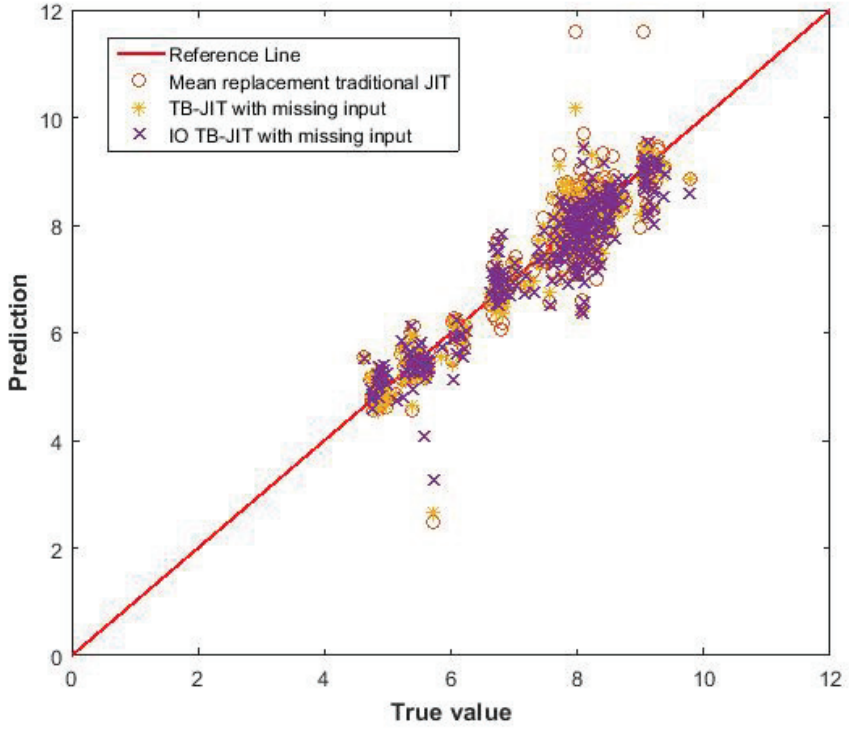

Figure 3.12: *Parity plot of performance of mean replacemen in traditional JIT and TB-JIT with missing input data vs. IO TB-JIT with missing input data.*

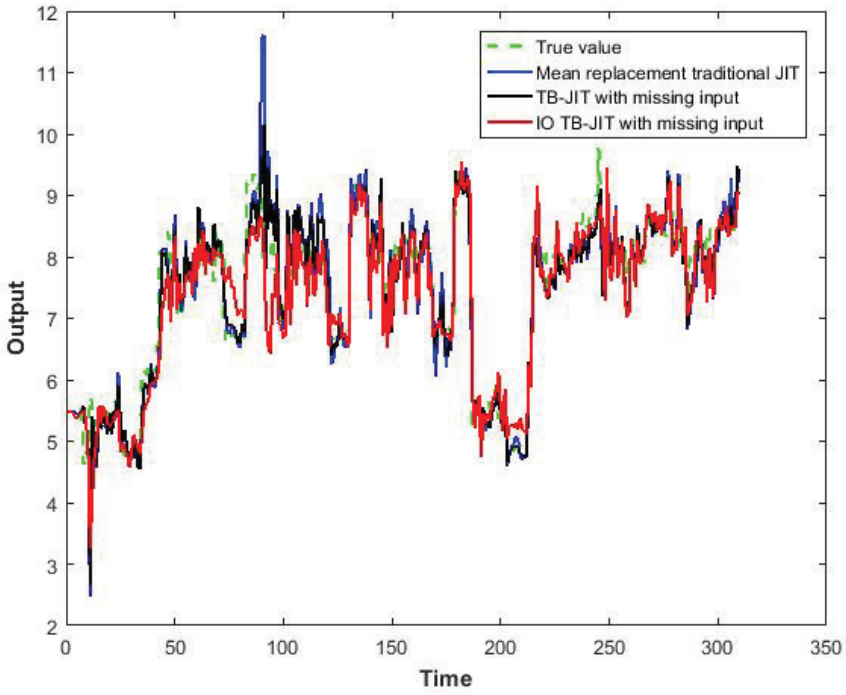

Figure 3.11: *Comparison of active substance prediction between mean replacement in traditional JIT and TB-JIT with missing input data vs. IO TB-JIT with missing input data.*

3.5 Conclusion

A similarity measure for JIT modeling has been proposed. The proposed approach has considerable advantages over traditional similarity measures in JIT: (a) It can select more appropriate samples for local modeling by considering the sequential correlation in input data. (b) It can account for trends and changes in data set upon arrival of a query sample by employing a moving horizon approach in evaluating similarity. (c) It considers output data in evaluating similarity. Thus, it can improve prediction performance. (d) It can be adapted to handle missing input data problem through the use of PPCA. Owing to the probabilistic nature of PPCA, missing input data problem can be handled effectively unlike mean imputation based approaches , thus achieving better predictive accuracy. These characteristics of the proposed method have been demonstrated through two case studies.

Chapter 4

Locally Weighted Classification of High Dimensional Data with Variable Selection

4.1 Introduction

Classification is a powerful machine learning tool for categorizing data sets, and it is widely used in various fields of data analysis such as image processing [37], tumor diagnosis in biomedical engineering [38], and process data analysis [39], [40]. Classification is a supervised learning procedure through which data points are categorized into different classes. These classes are actually the outputs of a classification model (also called "predictive classifier", or classifier). If the true outputs of classification (classes) are not known prior to modeling, classification becomes a clustering problem. In other words, clustering is an unsupervised machine learning technique which groups data points into clusters based on their statistical characteristics and spatial relations. In process data analysis, classification is used for various applications. In process monitoring and fault detection, classification is used to separate normal and abnormal data to enable one to predict a failure prior to its occurrence [41]. In data-driven modeling approaches for Near-InfraRed (NIR) spectroscopy, classification is used for separating samples based on their constituent materials [42]. For instance, for pharmaceutical applications, classification models are built based on NIR scans of samples to predict

the type of material used in each sample[43]. There are many classification methods which have been employed in process data analysis such as K-Nearest Neighbor (KNN) [44], Linear Discriminant Analysis [45], Naive Bayes Classifiers [46], Support Vector Machine(SVM) [47] and Artificial Neural Network(ANN) [48], each of which is applicable to certain situations. For instance, if data is linearly separable, Linear Discriminant Analysis (LDA) may provide high accuracy in prediction of classes. On the other hand, linear methods might fail to provide a good prediction [49] when data is not linearly separable. Alternatively, nonlinear methods such as ANN or Kernel SVM were developed to classify data sets that are not linearly separable.

Another factor that might affect classification is presence of irrelevant features or variables in a data set. In high dimensional data sets, some of the variables might not be helpful in classification, and they might even deteriorate the performance of the predictive classifier. Curse of dimensionality is another problem seen in high dimensional data sets. When the number of variables is significantly larger than the number of samples, curse of dimensionality will occur. To overcome this challenge, a variable selection or feature selection technique can be adopted. Various variable selection techniques have been proposed, some of which rely only on input data to select subset of variables, which are known as Filter methods [50]. Filter methods use a statistical test or correlation to do variable selection. On the other hand, there are methods which require output information (class) in addition to input data. These methods are known as Wrapper methods [51]. Wrapper methods take advantage of a classifier to select subset of variable. The third categroy of methods which is known as Embedded methods [52], is a combination of Filter and Wrapper methods. In comparison, Filter methods are generally easier to apply and have less computation cost, but they are less accurate than Wrapper methods.

Most of the structures used for building predictive classifiers are global models. Global models are built based on a training set, and they are expected to function well over validation and testing sets as well. They are relatively easy to build and implement. However, if a process undergoes unforeseen changes or experiences high degree of nonlinearity, performance of global models might deteriorate. Various model

updating techniques have been proposed through which model parameters or training set of data are updated. However, if the operating condition changes rapidly, or the degree of nonlinearity is relatively high, global models may fail. In order to resolve this problem, one can use a set of local models instead of a global model.

There are a number of local modeling approaches, one of which is mixture modeling [53]. If a nonlinear process can be assumed linear within certain regions, then one can build an individual model for each region. By putting all the models together, a mixture model can be built. Nonetheless, mixture models have limitations when operating conditions change rapidly. A framework to model highly nonlinear processes is locally weighted modeling [54] which builds as many local models as needed. Locally weighted modeling delays the learning until a new sample (also known as query sample) arrives. Then, the points in historical database are normalized with respect to the query sample. Afterwards, the distance between the query sample and each sample in historical database is calculated, which is known as the similarity measurement. Once distances are calculated, each sample in historical database receives a weight in compliance with the calculated distance. A weight matrix is formed and incorporated into the main modeling process. Once the local model is built and the prediction is made, it will be discarded. In this way, a nonlinear process is linearly approximated according to each new sample. In comparison, local modeling requires more computation, though it is more suitable for highly nonlinear systems.

There exist a number of locally weighted modeling approaches for building predictive classifiers. Frank et.al [55] proposed a locally weighted approach for building Naive Bayes classifiers. However, it does not address variable selection in dealing with high dimensional data sets. Domeniconi et.al [56] also proposed a method which addresses variable selection in classification. Nonetheless, it utilizes sample selection to build local models rather than building the local model based on weighting all samples, which might cause issues when the number of selected samples is small. In addition, to the best of authors' knowledge, such weighting approaches for locally weighted classification has not been well investigated in

process data analysis application such as NIR spectroscopy. In order to incorporate all training samples in building predictive classifier as well as choosing important variables, a novel approach "Locally weighted classification of high dimensional data with variable selection" is proposed in this chapter. In this approach, a training set of nonlinear input/output data points is required. A variable selection method called "Neighborhood Component Analysis" [57] (NCA) is employed on the training set. The employed variable selection technique is a supervised method, and lies under the category of Wrapper methods in variable selection. Then, only the selected variables are used in modeling. Afterwards, a dimensionally reduced model, which is Kernel (linear, Gaussian, etc) PCA, is trained to transform the data to latent space. In the next step, the distance in the latent space between query sample and each historical sample in database is calculated using a similarity metric. Historical samples in database receive weight according to the similarity measurement. At the end, the weights are applied to a local SVM model to predict the class of the query sample. Once a prediction is made, the local models (Kernel PCA and SVM) are both discarded. The proposed approach is tested on two NIR case studies. NIR data set is a high dimensional data set, and its variables are wavelengths. The first case study is regarding pharmaceutical application, and the second one is a real time operation data set from oil sands industry.

The rest of this chapter is organized as follows: First, variable selection techniques are discussed. Afterwards, basic principles of the proposed approach including NCA, Kernel PCA, SVM, and implementation procedure are discussed in section 4.3. At the end, results of applying the proposed approach in two case studies are discussed, and the effectiveness of the proposed approach is demonstrated using NIR spectrum in section 4.4. Finally, conclusions are presented in section 4.5.

4.2 Variable selection for classification

Variable selection (also known as variable selection, attribute selection or variable subset selection) has emerged as a powerful method in improving the goodness of

prediction in high dimensional data sets. In many modeling approaches such as locally weighted modeling, some samples are more important than others. To distinguish between samples for modeling, weights are assigned to them. Recently, attention has been drawn to weight assignment on variables in high dimensional data sets such as tumor diagnosis data sets or NIR spectroscopy data sets. High dimensional data sets have possibly many variables, some of which might be noisy which can jeopardize the performance of a model to be built. Presence of irrelavant variables might not affect training performane but deteriorate prediction performane on testing set, and thus cause overfitting problem [52]. Additionally, another common occurrence in dealing with high dimensional data is curse of dimensionality. When the number of variables is significantly larger than the number of samples, curse of dimensionality will occur [58]. Obviously, dimension reduction techniques are helpful in dealing with a high dimensional data set. Prior to that however, implementing a variable selection technique contributes in selecting only relevant variables. Hence, a subset of variables are usually selected for modeling purposes for high dimensional data sets.

Variable selection algorithms vary based on the purpose of modeling. For instance, the variables one chooses for regression might not be identical to that of classification. For classification, it is usually more appropriate to choose the variables that can highlight the differences of classes. There are many techniques developed for variable selection, three of which are discussed briefly here. Figure 4.1 reviews the three methods in variable selection in terms of their application, and example(s) of each method is (are) given. It can be seen that they can be categorized into three classes: Filter methods, Wrapper method, and Embedded methods. Wrapper methods are briefly discussed in section 4.2.1 as they are adopted in this work.

4.2.1 Wrapper method

Filter methods in variable selection try to find the most significant variables in a model-free way. Unlike Filter methods, Wrapper methods [59] rely on a model to highlight variable importance in a given data set [52]. The built model is not used for prediction, and it is only useful for selecting a subset of variables, which helps in

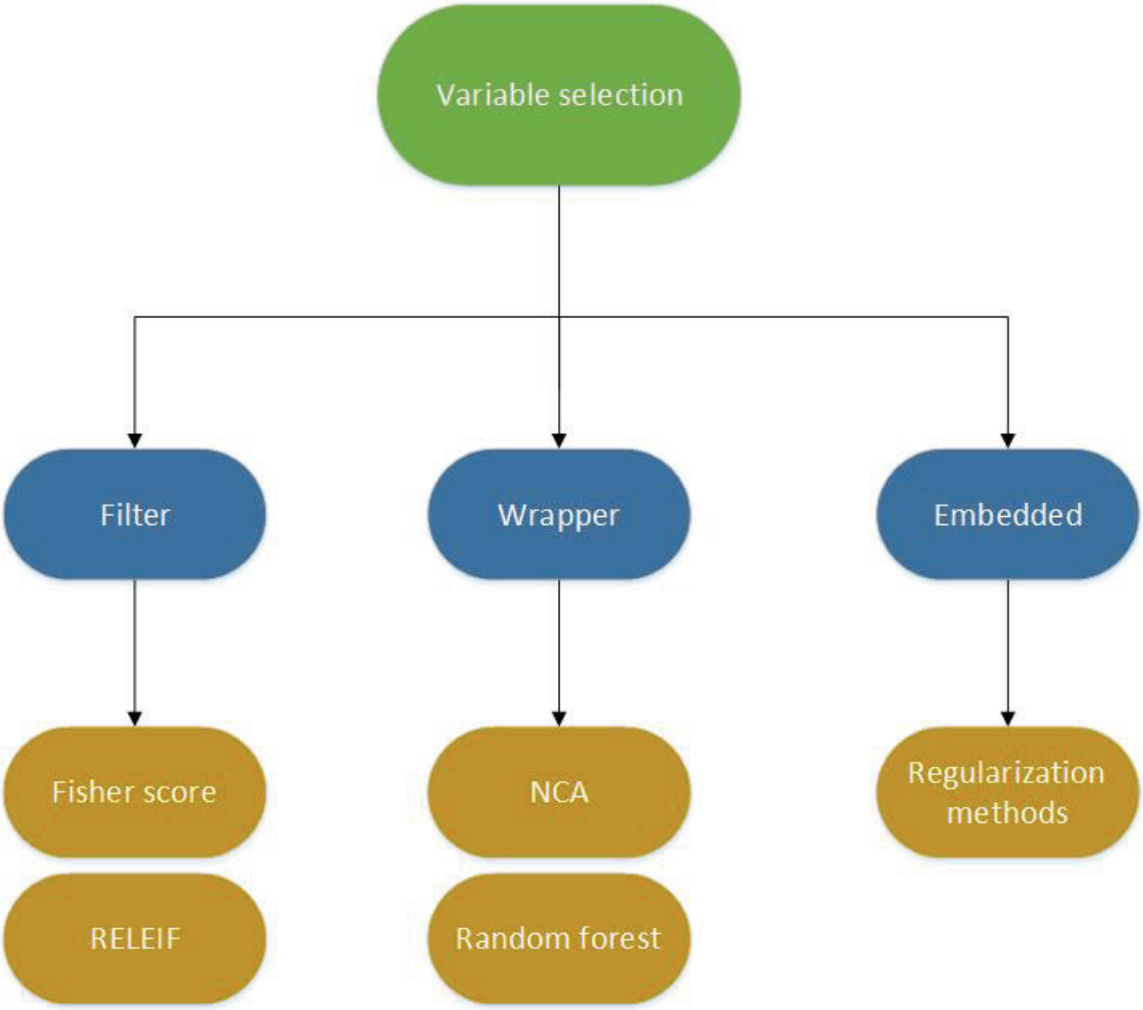

Figure 4.1: *Overview of variable selection methods in classification.*

overcoming curse of dimensionality.

There are many different approaches in Wrapper methods. Basically, one should run a classifier first, such as SVM, random forest or other types to obtain the variable weights. Once the weights are determined, a subset of them is chosen and further analysis is carried out based on them. If the classifier used for variable selection, and the one used for prediction are not the same, "variable(feature) exportability" [60] might occur. Variable exportability may happen when variables selected during variable selection step is not appropriate to be used for the predictive classifier. In other words, algorithms might not converge to a certain set of variables. However, in

many cases, the variables selected are almost identical.

4.3 Proposed algorithm for classification of high dimensional data with variable selection strategy

In this chapter, we are addressing two common issues in building predictive classifiers: Curse of dimensionality in high dimensional data sets, and nonlinearity of data in classification. NCA and Kernel PCA in sections 4.3.1 and 4.3.2 are adopted to handle curse of dimensionality. Locally weighted approach is used to tackle the nonlinearity of data for classification, which is discussed in section 4.3.4.

Regarding nonlinearity of data for classification, nonlinear classification models have been developed to make prediction. A number of different methods exist for nonlinear classification such as ANN, fuzzy SVM [61], and Kernel SVM. All of these methods are global models, and they are built once. Due to their nature, nonlinear classifiers are more complicated than linear ones. Furthermore, when the data set exhibits high degree of nonlinearity, global nonlinear models might not be able to provide good prediction. To resolve this issue, local models are employed to provide prediction when the data set is locally separable.

In this work, locally weighted approach for classification of high dimensional data with variable selection is proposed. In the following, the main components of the proposed approach are discussed. Afterwards, they are put together to present the proposed method.

4.3.1 Neighborhood component analysis

Neighborhood Component Analysis (NCA) [57] is a classification method which can also be extended as a variable selection method [62]. Thus, it can be categorized among Wrapper methods in variable selection. In terms of its nature, it is similar to K-Nearest Neighbor (KNN) classifier but with extensions. Therefore, KNN is explained first. Let a training data set with N samples be defined as:

$$L = \{\mathbf{x_i}, y_i\} \tag{4.1}$$

where $\mathbf{x_i}$ is the i-th sample with P variables, i.e $\mathbf{x_i} \in R^{P\times 1}$. The vector $\mathbf{y}$ is considered as the output in this work $\mathbf{y} = [y_1\ y_2\ \cdots\ y_i \cdots\ y_N]$ where $y_i \in \{1, 2, \cdots h, \cdots c\}$, and c represents the number of classes. The input space is rearranged as: $X = [\mathbf{x_1}\ \mathbf{x_2}\ \cdots\ \mathbf{x_N}]^T$.

KNN is a non-parametric instance-based classifier that predicts the class of a new sample based on K nearest neighbors around it. There are two main steps in KNN:

i) The distances between the new sample to all historical samples in database are computed using Euclidean distance. K samples that are closer to the new sample are selected regardless of their classes. Let the selected set be denoted as S.

ii) The conditional probability for each class given the new sample is calculated-that is the fraction of samples in S given class label:

$$P(y_{new} = h|\ x_{new}) = \frac{1}{K}\sum_{i=1}^{K} I(y_{(i)} = h) \tag{4.2}$$

where I becomes one if $y_{(i)}$ belongs to the same class as h, and zero otherwise.

Figure 4.2 provides an example on KNN. A new sample (blue point) should be classified using KNN. There are two classes: Class A shown with black points, and Class B shown with green points. The red circle determines the K neighbors selected, in this case $K = 5$. The probability that the new observation (blue point) belongs to class A and B are 0.6 and 0.4, respectively. Hence, the new point is classified in class A.

NCA is an extension of KNN with the following changes. First, K is assumed to be one here. Second, a similarity metric similar to Mahalanobis distance is defined. The distance is defined as follows:

$$d_{i,j} = \sqrt{(\mathbf{x_i} - \mathbf{x_j})^T \Sigma_x (\mathbf{x_i} - \mathbf{x_j})} \tag{4.3}$$

where $d_{i,j}$ is the distance between $\mathbf{x_i}$ and $\mathbf{x_j}$, and $\Sigma_x \in R^{P\times P}$ is a diagonal matrix as:

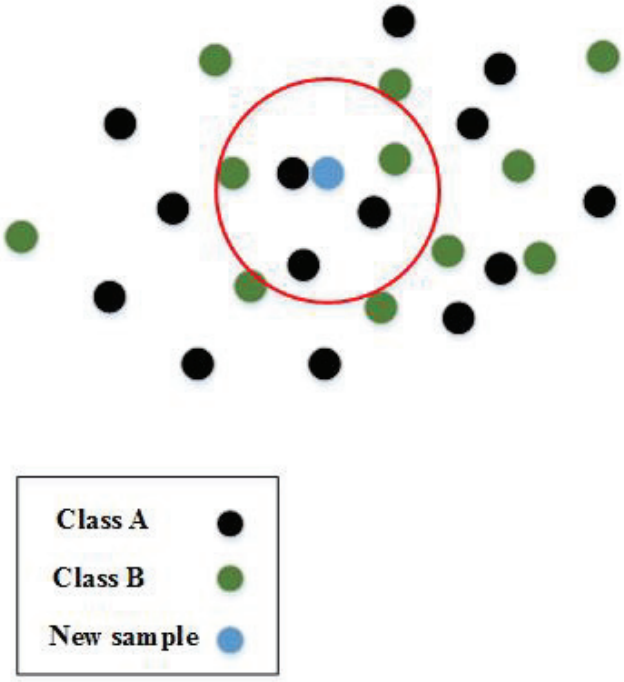

Figure 4.2: *An example on KNN classifier based on two classes*

$$\Sigma_x = \begin{pmatrix} {a_1}^2 & 0 & \dots & 0 \\ 0 & {a_2}^2 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & {a_P}^2 \end{pmatrix} \tag{4.4}$$

where each a_j determines the weight of corresponding variable j of the data set.

The third extension of KNN to NCA is replacing equation (4.2) with equation (4.5). For a given sample $\mathbf{x_i}$, the probability of it being in the same class as an another sample $\mathbf{x_j}$ can be defined as:

$$p_{i,j} = \frac{exp(-||d_{i,j}||^2)}{\sum_{k=1}^{N} exp(-||d_{i,j}||^2)} \tag{4.5}$$

The probability of point $\mathbf{x_i}$ being classified correctly is defined as:

$$p_i = \sum_{j \in C_i} p_{i,j} \tag{4.6}$$

where C_i consists of the points which belong to the same class as $\mathbf{x_i}$.

The main purpose of employing NCA as a variable selection step in high dimensional data analysis is to obtain the weight matrix Σ_x, as it reveals the weight of each variable in classification. The objective function is defined as:

$$\max_{\Sigma_x} \quad \sum_i \sum_{j \in C_i} p_{i,j} \tag{4.7}$$

The objective is to maximize equation (4.7) which represents the performance of the NCA. Since we want to solve an optimization problem, the cost function should be defined such that it becomes differentiable. Considering the weight matrix, the optimization problem mentioned in equation (4.7) should be solved such that the optimal values for Σ_x are obtained.

A gradient optimization method such as deltabar-delta [63] can be used to determine Σ_x. A number of the variables should be chosen once weights are obtained. After selecting appropriate variables, a subspace of inputs is denoted as $U \in R^{N \times D}$ with $U = [\mathbf{u_1}\ \mathbf{u_2}\ \cdots \mathbf{u_i}\ \cdots \mathbf{u_N}]^T$ where each $\mathbf{u_i} \in R^{D \times 1}$, and D ($D \ll P$) represents the number of variables selected after NCA. It is worth to note that $U \subseteq X$.

4.3.2 Kernel PCA

In a high dimensional data set, the number of variables might still be high even after a variable selection step that is conducted in the original variable space. Therefore, it requires one to employ a dimension reduction technique through projection to latent space, thus obtaining latent variables. One of the useful tools in dimension reduction is Principal Component Analysis (PCA). In our proposed approach, NCA is first applied to the training set to select the most important variables. Then, in order to carry out dimension reduction , PCA or Kernel PCA is employed. If the data set is highly nonlinear, Kernel PCA can be used. The choice of Kernel function depends on the nature of data. Recall U as the input data here, and let Φ denote the data in kernel space.

$$U \xrightarrow{g} \Phi \tag{4.8}$$

where g is called the kernel function or kernel trick. In kernel PCA, original data is first mapped into a higher dimension using kernel trick. There are different types of kernel tricks. In this paper, Gaussian Kernel defined in equation (4.9) is used.

$$g(\mathbf{u_i}, \mathbf{u_j}) = exp(-\frac{(\mathbf{u_i} - \mathbf{u_i})^2}{2\sigma^2}) \tag{4.9}$$

where $\mathbf{u_i}$ and $\mathbf{u_j}$ are two different samples, and σ^2 is the hyperparameter which determines how wide or narrow the points are distributed in latent space. To extract latent variables, the eigenvalue decomposition is applied to covariance matrix of Φ as follows:

$$B = \frac{1}{N}\sum_{i=1}^{N} \Phi_i \Phi_i^T \tag{4.10}$$

After solving an eigenvalue decomposition problem, principal components can be obtained [64]. Principal components for training set are denoted as $T = [\mathbf{t_1}\ \mathbf{t_2}\ \cdots \mathbf{t_i}\ \cdots \mathbf{t_N}]^T$ with each $\mathbf{t_i} \in R^{H\times 1}$ where H is the number of latent variables taken.

4.3.3 Support Vector Machine

In this section, SVM is explained assuming two classes I and II. The extension to multi-class SVM can be found in [65]. A classifier should be built based on historical points in database to predict the class for any future data. SVM tries to find a hyperplane (i.e. decision boundary) that can separate two classes of data from each other. Any point above the hyperplane will be in Class I, and any point below the hyperplane will belong to Class II. Figure 4.3 gives an overall view of SVM classifier. If the hyperplane is defined as a line, SVM model is linear. A linear SVM model is formulated as:

$$f(\mathbf{t_i}) = < \mathbf{w} \cdot \mathbf{t_i} > + b \tag{4.11}$$

where $\mathbf{w} \in R^{1\times H}$, b is the intercept term, and $< \cdot >$ is the dot product of two vectors. The output vector is $\mathbf{y} = [y_1, y_2, \cdots y_i, \cdots y_N]$ where $y_i \in \{-1, +1\}$.

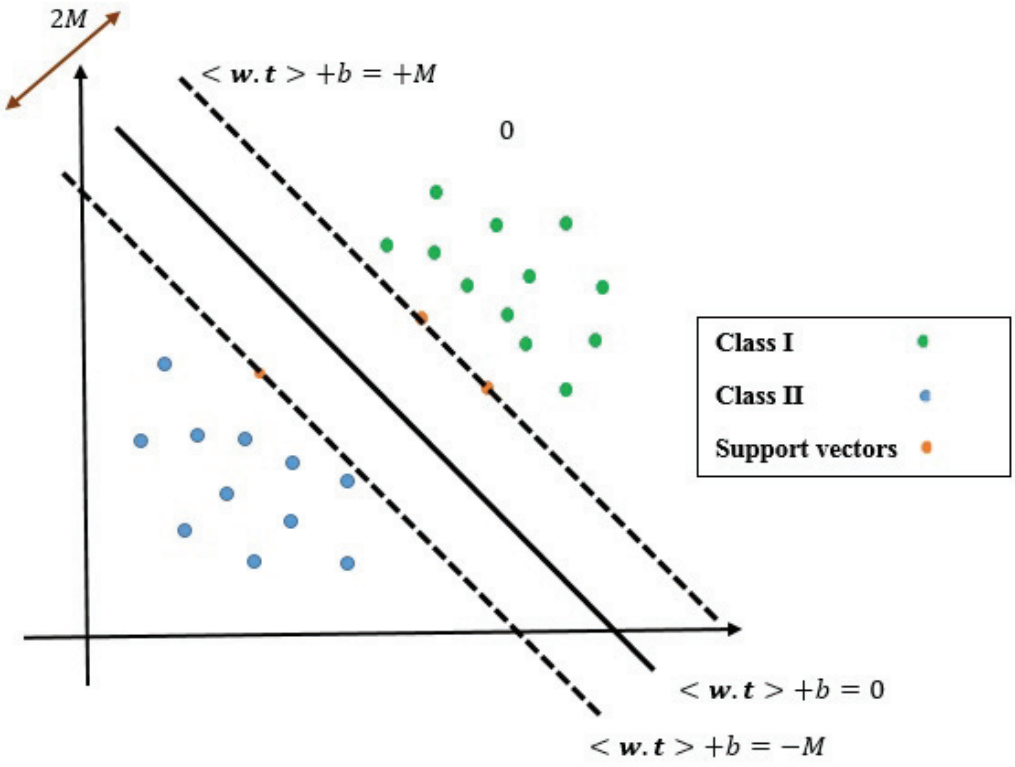

Figure 4.3: *Schematic of SVM in classification.*

To formulate nonlinear SVM using Kernel functions, one can use equation (4.9) to transform the data to Kernel space. Then, equation (4.11) becomes:

$$f(\phi(\mathbf{t_i})) =< \mathbf{w} \cdot \phi(\mathbf{t_i}) > +b \tag{4.12}$$

We proceed with linear SVM. According to equation (4.11), the decision boundary is defined by model parameters **w** and b which are determined by SVM.

Let $\mathbf{t_I}$ be a point from Class I , and $\mathbf{t_{II}}$ be a point from Class II. Note that these two points are the closest pair of points from two classes, and they are support vectors in SVM. The lines that pass from point $\mathbf{t_I}$ and $\mathbf{t_{II}}$ are parallel with each other, and thus we can write:

$$< \mathbf{w}.\mathbf{t_I} > +b = +M \tag{4.13}$$

$$< \mathbf{w}.\mathbf{t_{II}} > +b = -M \tag{4.14}$$

where $2M$ denotes the margin of the two classes. By subtracting equation (4.13) with equation (4.14), one can calculate the distance between the points:

$$\mathbf{w}.(\mathbf{t_I} - \mathbf{t_{II}}) = 2M \tag{4.15}$$

$$(\mathbf{t_I} - \mathbf{t_{II}}) = \frac{2M}{\mathbf{w}} \tag{4.16}$$

Therefore, the distance between support vectors becomes $\frac{2M}{\mathbf{w}}$. To normalize, equation (4.16) is divided by M. Hence, one can write an objective function for maximizing the distance between support vectors:

$$\max_{\mathbf{w},b} \quad \frac{2}{\sqrt{\mathbf{w^T}w}} \tag{4.17}$$

$$subject\ to \quad y_i(\mathbf{wt_i} + b) \leq 1$$

The cost function is strictly positive. Thus, equation (4.17) is equivalent to:

$$\min_{\mathbf{w},b} \quad \frac{1}{2}w^T w \tag{4.18}$$

$$subject\ to \quad y_i(\mathbf{wt_i} + b) \leq 1$$

By adding soft margin and slack variables, the cost function becomes as equation (4.19) shown below. The second term on the right hand side of equation (4.19) is a relaxation term which sets penalty for the points violating the constraint. μ_i is slack variable, and C^* is a constant.

$$\min_{w,b,\mu_i} \quad \frac{1}{2}\mathbf{w}\mathbf{w}^T + C^* \sum_{i=1}^{N} \mu_i \tag{4.19}$$

$$subject\ to \quad y_i(\mathbf{wt_i} + b) \leq 1 - \mu_i$$

The primal form of equation (4.19) is converted to its dual form and then solved [66].

4.3.4 Locally weighted classification of high dimensional data with variable selection strategy

In this section, the proposed approach to classify high dimensional data using locally weighted modeling and variable selection is presented.

Figure 4.4 sketches the proposed method. Let us take the training set $X \in R^{N \times P}$ where N represents the number of training samples. NCA is implemented on training set to screen the most significant variables in the original variable space. Then, the new training data based on the selected variables are obtained. The new training set after NCA is named $U \in R^{N \times D}$ where $D \ll P$. D is the number of variables selected by NCA.

When a query sample (${\mathbf{x_q}}^T \in R^{1 \times P}$) arrives, the D variables from NCA step are determined from the query sample, and hence, the query sample becomes ${\mathbf{u_q}}^T \in R^{1 \times D}$. The variables of the historical data and query sample are appended to build a new matrix $Z = [U; {\mathbf{u_q}}^T]$, then it is normalized.

Although the dimension of the data is diminished, a dimension reduction technique is still required to compress the data further. This is done by projecting data to latent space to obtain latent variables. Let the extracted set of training latent variables be $T \in R^{N \times H}$ where H represents the number of latent variables in latent space. The PCA model is built only based on the training set. Using the built PCA model (or Kernel PCA), the query sample is also transformed to latent space, and it is denoted as ${\mathbf{t_q}}^T \in R^{1 \times H}$.

In the next step, similarity measurement part, which is the essential part in locally weighted learning, is assessed. Using a similarity metric, the distance between ${\mathbf{t_q}}^T$ and each sample in $T \in R^{N \times H}$ is calculated. In this work, City Block is used as similarity metric. Nonetheless, there is no limitation in adopting other similarity metrics. City Block similarity metric is defined as [67]:

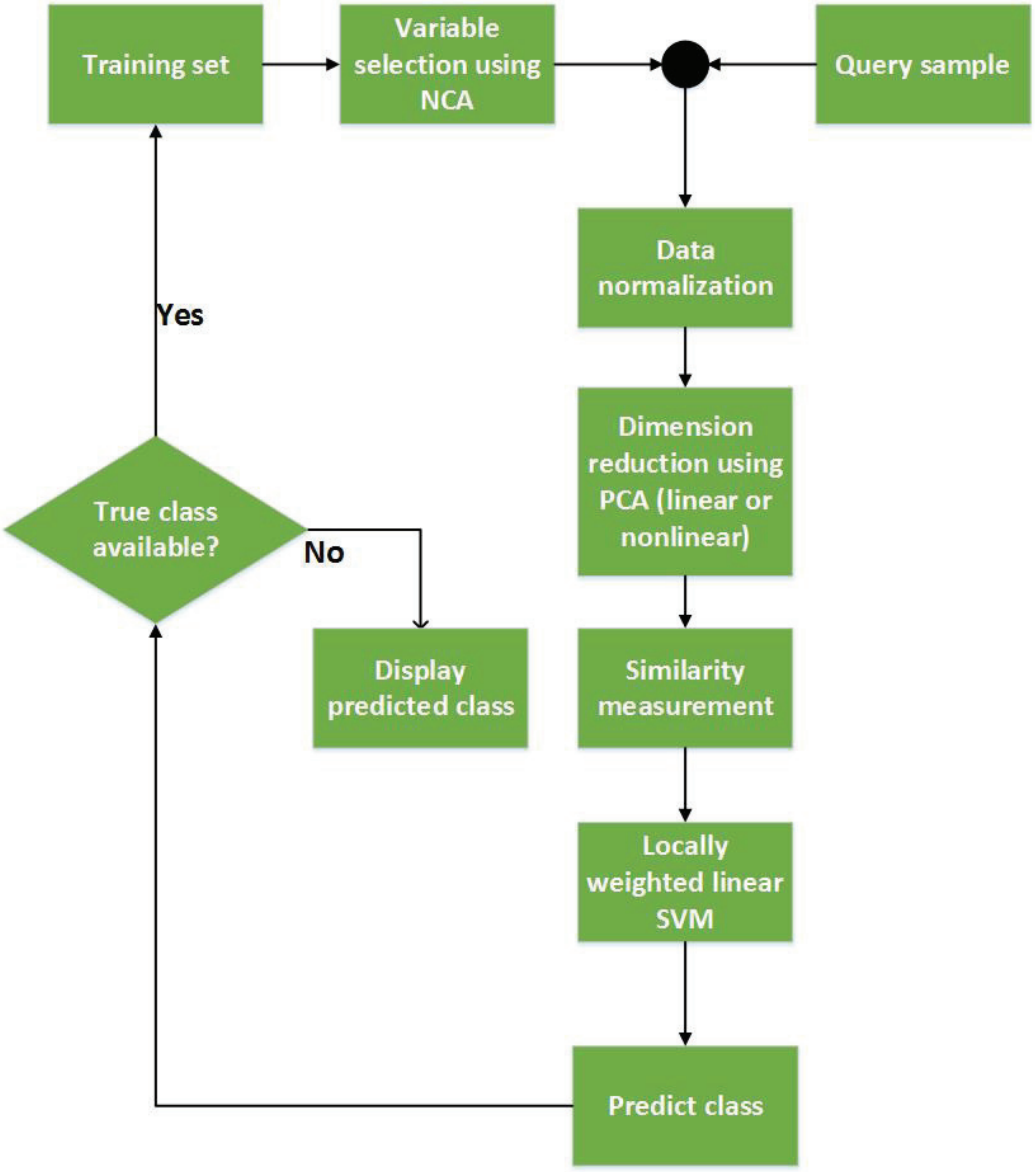

Figure 4.4: *Schematic of the proposed method.*

$$s_{i,q} = \sum_{j=1}^{H} |u_{i,j} - u_{q,j}| \tag{4.20}$$

where $s_{i,q}$ is the distance between query sample and a given sample in training set in latent space. Using the calculated distance, weights can be assigned using equation (4.21):

$$v_{i,q} = exp(-{s_{i,q}}^2) \tag{4.21}$$

where v_i is the weight assigned to each sample in training set in latent space. Then the weight matrix for each query sample is formed as:

$$V = \begin{pmatrix} v_{1,q} & 0 & \dots & 0 \\ 0 & v_{2,q} & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & v_{N,q} \end{pmatrix} \tag{4.22}$$

The weight matrix is $V \in R^{N \times N}$, and all the elements are zero except for those lie on the diagonal.

Afterwards, a locally weighted SVM model should be built for each query sample as followed:

$$\hat{y_i} = \langle \mathbf{w}.v_{i,q}\mathbf{t_i} \rangle + b \tag{4.23}$$

where $\hat{y_i}$ is the predicted class for sample i in the training set. When the model parameters are identified, they are used to make a prediction for the query sample:

$$\hat{y_q} = \langle \mathbf{w}.\mathbf{t_q} \rangle + b \tag{4.24}$$

4.3.5 Implementation procedure

1. Take the training set X and $\mathbf{y}$.

2. Perform variable selection using NCA on training set to choose the most important variables.

3. Choose D variables from NCA in training set and name them U.

4. Arrival of query sample $\mathbf{x_q}$. Choose the D variables from query sample and name it $\mathbf{u_q}$.

5. Append training set and query sample to from a new matrix $[U_t; \mathbf{u_q}]$. Normalize the appended matrix. After normalization, separate training set and query sample.

6. Select the desired number of latent variables. Build a local Kernel (linear, Gaussian, etc) PCA on training set to obtain training set of latent variables, and name it T.

7. Transform the query sample to latent space using the built Kernel PCA model, and obtain $\mathbf{t_q}$.

8. Calculate the similarity between $\mathbf{t_q}$ and T using a similarity metric such as that defined in equation (4.20).

9. Assign weight to samples in training set using a weighting function such as (4.21). Form a diagonal weight matrix.

10. Build a locally weighted SVM model based on training set in latent space.

11. Make a prediction for query sample in latent space.

12. If the true class of query sample is available, add the query sample to training set to update the NCA model.

4.4 Case Studies

This section consists of two Near InfraRed (NIR) spectroscopy data sets to demonstrate the effectiveness of the proposed method for classification of high dimensional data in process data analysis. NIR data sets are high dimensional, and thus they are suitable for exammining the proposed method. The first data set is for pharmaceutical purposes to analyze the contents of tablets [36]. The second data set was taken from an oil sands extraction operation. For proprietary reason, actual spectrum values are removed in the presentation of second case study. It is important to note that the variables in NIR data sets are wavelengths, and instead of variable selection, the term wavelength selection might also be used here after. Both cases are compared against a global model structure in classification which consists of the following steps:

i. A wavelength selection method, NCA.

ii. Dimension reduction, Kernel PCA.

iii. Global modeling, Kernel SVM.

For Kernel PCA, The choice of Kernel functions vary from the proposed local method to global modeling method. In the case of global modeling method, a Gaussian Kernel is used for PCA and SVM. In the proposed local approach, the SVM model is linear while the PCA model might use Kernel function depending on the data.

4.4.1 NIR data set to classify tablets based on their constituent materials

In this case study, the data set consists of 310 samples with 404 wavelengths (variables). The samples are taken from NIR probes examined on four different types of tablets. Figure 4.5 shows the spectrum of the samples. The purpose is to build a classifier that is able to predict the type (class) of the tablets.

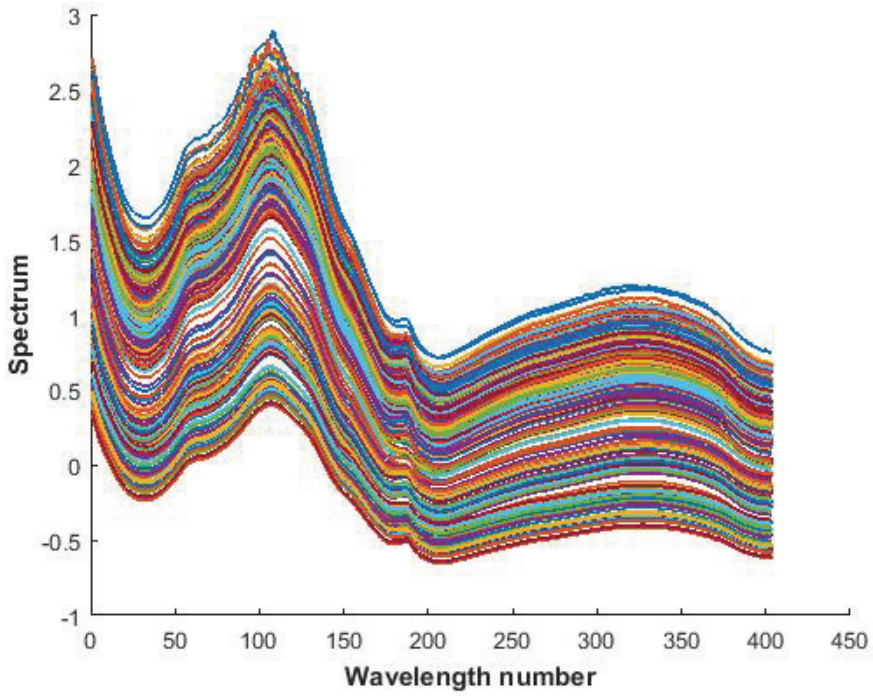

Figure 4.5: *NIR spectrum for tablets.*

In addition, Figure 4.6 shows the spectrum in their true classes. It can be seen that there are four different classes. According to Figure 4.6, class 2 (Blue class) can

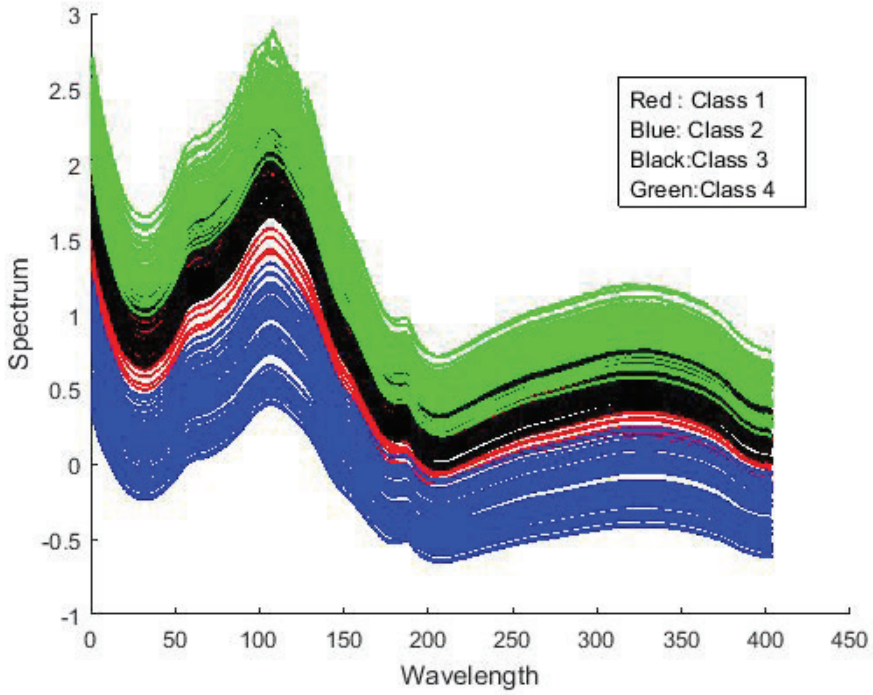

Figure 4.6: *NIR scans with their true classes.*

be linearly separated. However, other three classes are not linearly separable. As a result, the proposed locally weighted modeling technique can be tested on this data set.

In order to build the classifier using global modeling or local modeling, the first step is splitting the data into training and testing sets. 65% of the samples are allocated for training, and the remaining 35% for testing. Data segmentation occurs completely at random. Therefore, it is recommended to run Monte Carlo simulation to have unbiased results. In the following, the proposed approach is explained using only one typical set of data segmentation. The results however, are from a large number of random data segmentation coming from Monte Carlo simulations. First, the results concerning global nonlinear classification are presented.

a) Global modeling

In the global modeling approach after data segmentation, NCA is used to obtain the most important wavelengths. The criterion used is to select wavelengths that have

the highest and lowest weights, as simulations prove both are useful in improving the performance of a classifier.

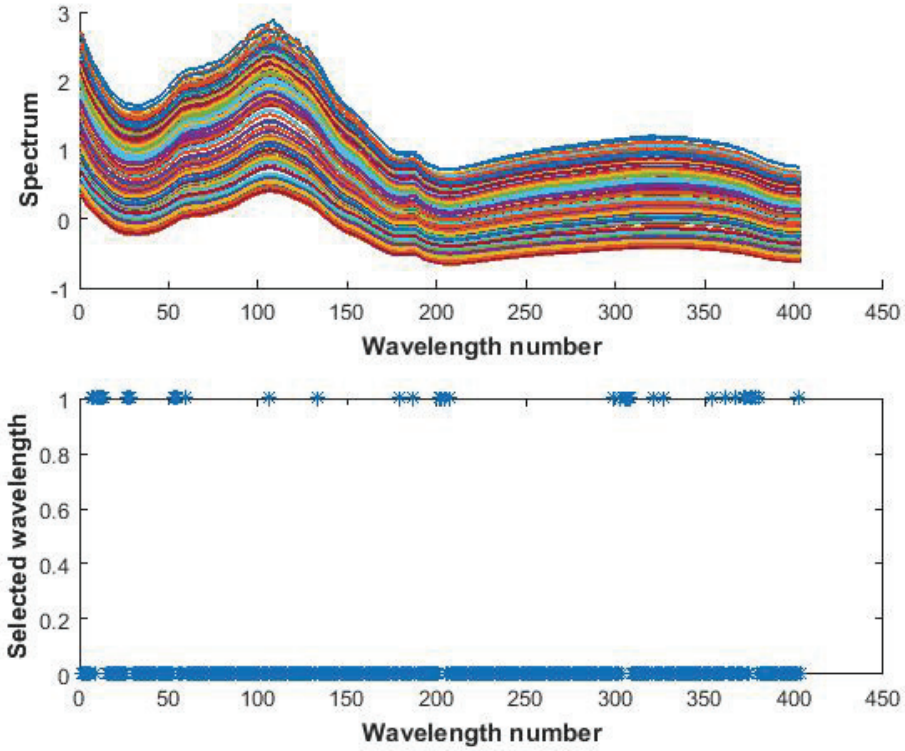

Figure 4.7: *Wavelengths selected after NCA.*

Figure 4.7 shows the wavelengths selected for this case study in global modeling. The number of wavelengths has been diminished from 404 to 40 by using NCA. To reduce the dimension even further while keeping the variance, one needs to do Kernel PCA. Figure 4.8 shows the training samples in latent space, and it can be seen that they are not linearly separable.

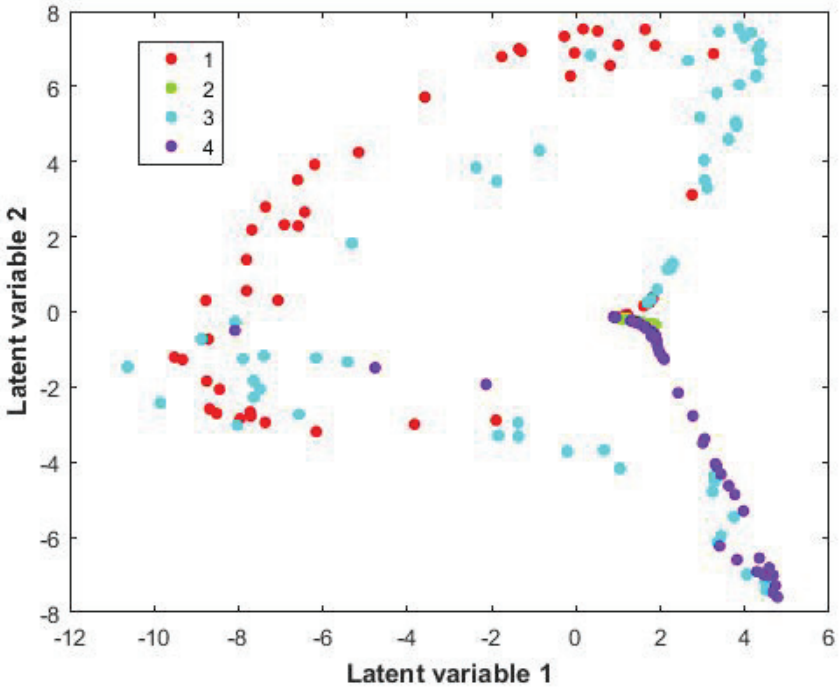

Figure 4.8: *Kernel PCA on data after NCA.*

b) The Proposed approach for locally weighted classification

According to Figure 4.4, the step after data splitting is wavelength selection through NCA. The dimension of the data has decreased from 404 to 14 here. Although wavelength selection technique for both global modeling and proposed local modeling approach is NCA, the wavelengths selected are different according to Figures 4.7 and 4.9. The reason is the fact that the training set for global modeling is different from that of the proposed approach due to individual random data segmentation in Monte Carlo simulation. The next step is performing PCA. When a query sample arrives, a linear PCA model is built based on the training set to divide local latent variables in a two-dimensional space, as two latent variables are able to explain most of the variation. Once the local PCA model is built, the query sample will also be transformed to latent space using the built local linear model. Figure 4.10 shows the training samples in latent space after PCA.

The locally weighted modeling is applied at this stage. For a given query sample,

the distance between the query sample in latent space and all training samples in latent space is computed using the similarity metric shown in equation (4.20). Afterwards, weights are assigned to samples based on the computed distances by using the weighting function given in equation (4.21). Then, the weights are incorporated to the linear SVM model in equation (4.24) to predict the class of the query sample. Finally, both local models (the dimension reduction and the classifier) are discarded after each prediction.

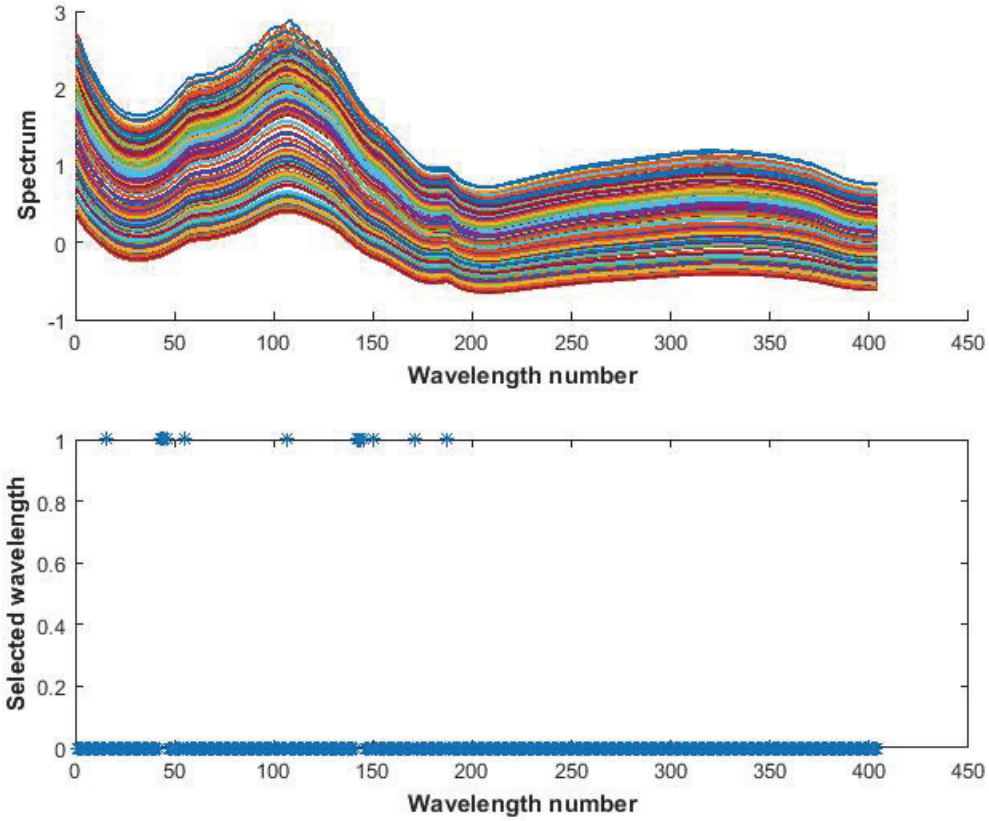

Figure 4.9: *Wavelength selection using NCA in locally weighted modeling.*

By running the Monte Carlo simulations for all the samples in testing set, the average and the lowest misclassification error are calculated. Tables 4.1 compares the results for global nonlinear classification and locally weighted classification.

Two different Kernel functions -linear and Gaussian- are used for the global

Table 4.1: *Misclassification error on testing set for both global modeling and proposed approach*

Global modeling			
Kernel for dimension reduction	**Kernel for SVM**	**Min error**	**Average error**
Linear	Linear	0.667	0.741
Linear	Gaussian	0.75	0.85
Gaussian	Linear	0.269	0.375
Gaussian	Gaussian	0.222	0.4
Proposed approach			
Kernel for dimension reduction	**Kernel for SVM**	**Min error**	**Average error**
Linear	Linear	0.06	0.179
Gaussian	Linear	0.185	0.317

classification. The best result is obtained when Gaussian Kernel is used for both PCA and SVM.

It can be seen that the testing error is lower for the proposed approach. Additionally, Table 4.1 reveals that the appropriate Kernel function for the the proposed approach is linear in this case study. It is worth to note that the Kernel for SVM in the proposed approach is always assumed to be linear, since the purpose is to solve a nonlinear problem by considering as many local linear models as necessary.

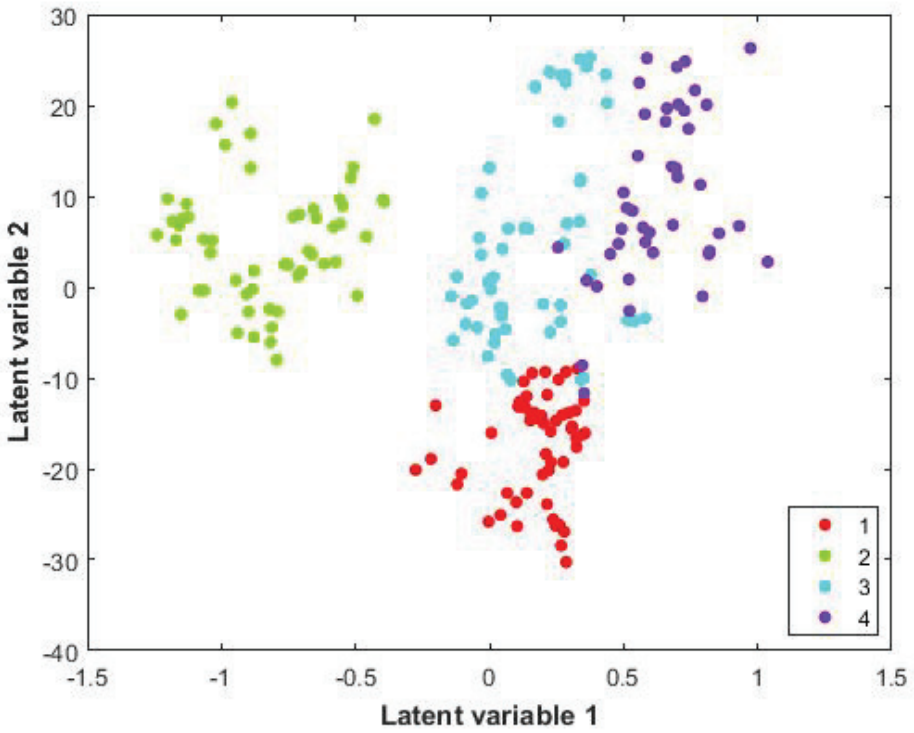

Figure 4.10: *Training points in latent space.*

4.4.2 Classification of NIR data from oil sands extraction

In this case study, NIR is used to analyze the samples taken from different locations of an oil sands tailing pond to obtain their contents and chemical properties. The samples are Mature Fine Tailing (MFT) [68], and they are accumulated in a tailing pond over time. Major components of MFTs are solid, water and small portion of bitumen (less than 5%). To facilitate solid settling, it is necessary to remove the water from MFT samples. There are a few methods applied in industry to facilitate water removal. It is necessary to use those methods due to the fact that solid portion of MFT require quite long time to separate from water if they are left alone. Adding coagulant is an effective method that is widely practiced in oil sands operation. Coagulant tries to neutralize surface charges of particles in MFT, and makes them form larger particles. Eventually, particles drop to the bottom of the pond, and water can be removed from the top.

In this study, there are 280 NIR scans of MFT sample in 1107 wavelengths. Additionally, there are two different coagulant dosages which are shown in Table 4.2.

Table 4.2: *Coagulant dosages considered as output for classification.*

Coagulant dosage	Class Number
Low	2
High	1

The samples used in this case study come from laboratory, and thus the coagulant dosage injected to samples (classes) are actually known. Nonetheless, for future MFT samples to be received, the coagulant dosage of a given sample is not necessarily labeled or accurately known, which is simply because injection cannot be quantified or quantified accurately in real time operation. By training a classifier, one would be able to predict the coagulant dosage online. Figure 4.11 shows the NIR spectrum of the data set.

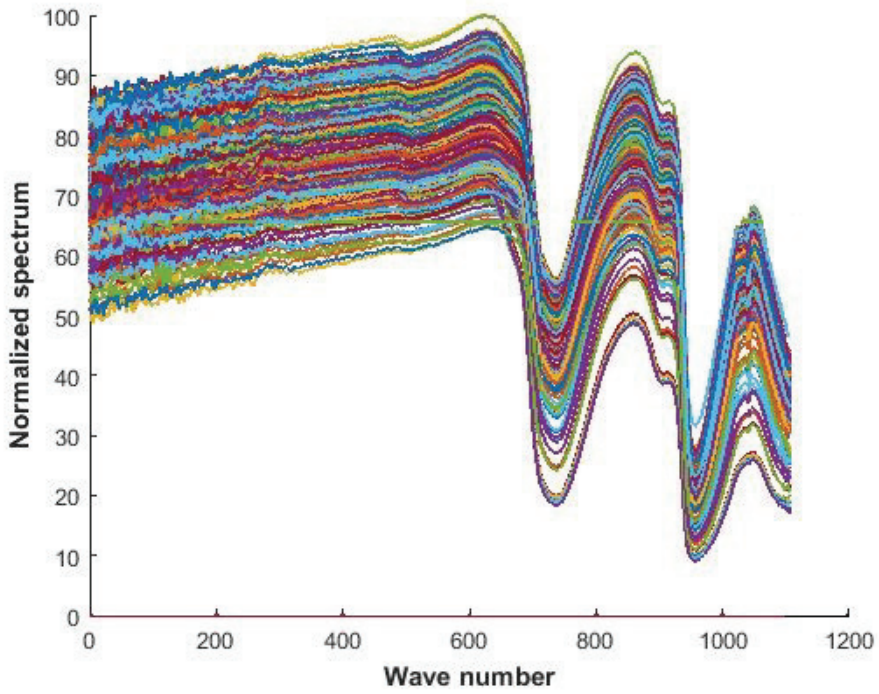

Figure 4.11: *NIR spectrum from MFT samples.*

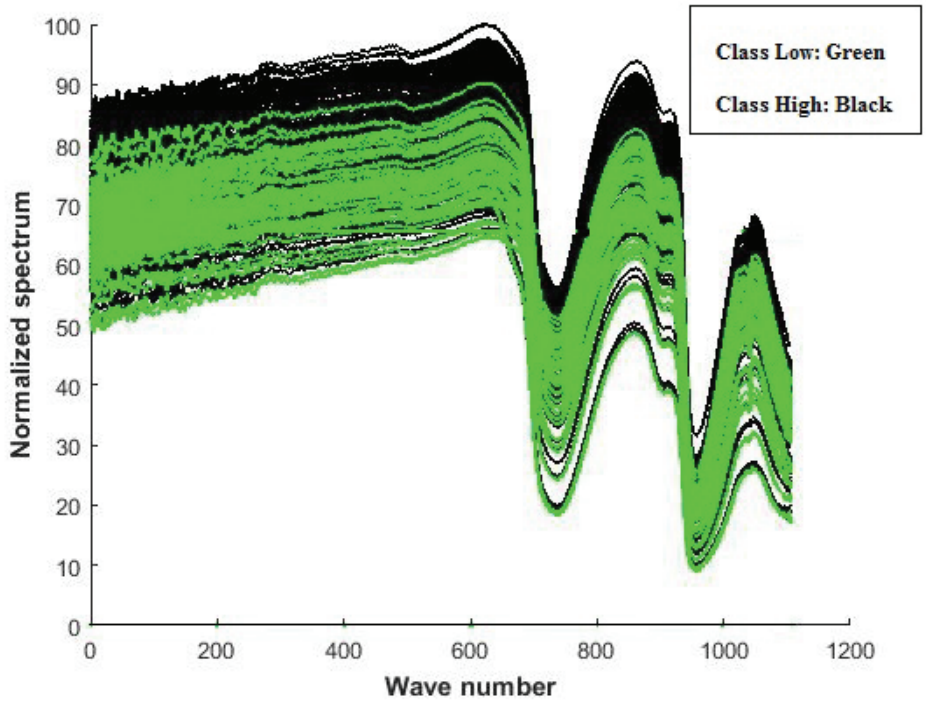

Figure 4.12: *NIR spectrum from MFT samples considering their respective coagulant dosage.*

According to Figure 4.12, there are two different colors, each representing a coagulant dosage. It can be seen that this problem faces high degree of nonlinearity. Hence, it is worth to apply Kernel PCA in the dimension reduction step of the proposed method. First, we assess the performance of global modeling in this case study.

In this part, we start with one data segmentation. At the end, the average performance along with the best performance of all different data segmentation will be discussed. From the available data, 65% of the scans are used for training, and the rest are used for testing .

a) Global modeling

After data segmentation, NCA is applied to highlight the most important wavelengths. It can be seen in Figure 4.13 that the chosen wavelengths are mostly from a specific range of wavelength for this combination of data segmentation. The number of selected wavelengths is diminished to 22 after this step from 1107.

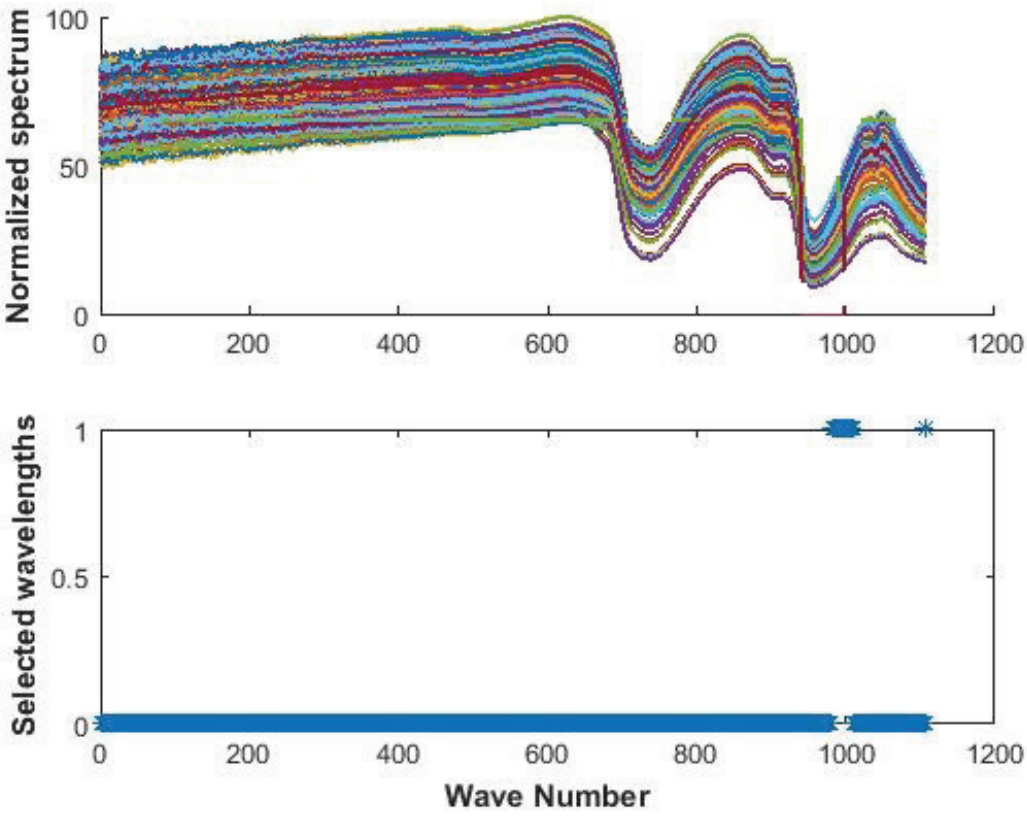

Figure 4.13: *Selected wavelengths after NCA for global modeling.*

In the next step, Kernel PCA is applied to do dimension reduction. The number of latent variables is selected to be two, as two latent variables are able to explain most of the variation.

Figure 4.14 shows the training samples in latent space after Kernel PCA for global modeling. It can be seen that the samples exhibit high degree of nonlinearity even after a nonlinear dimension reduction technique, which points out the need of a local modeling approach. The final results after running Monte Carlo simulations will be discussed along with the results of the proposed local approach.

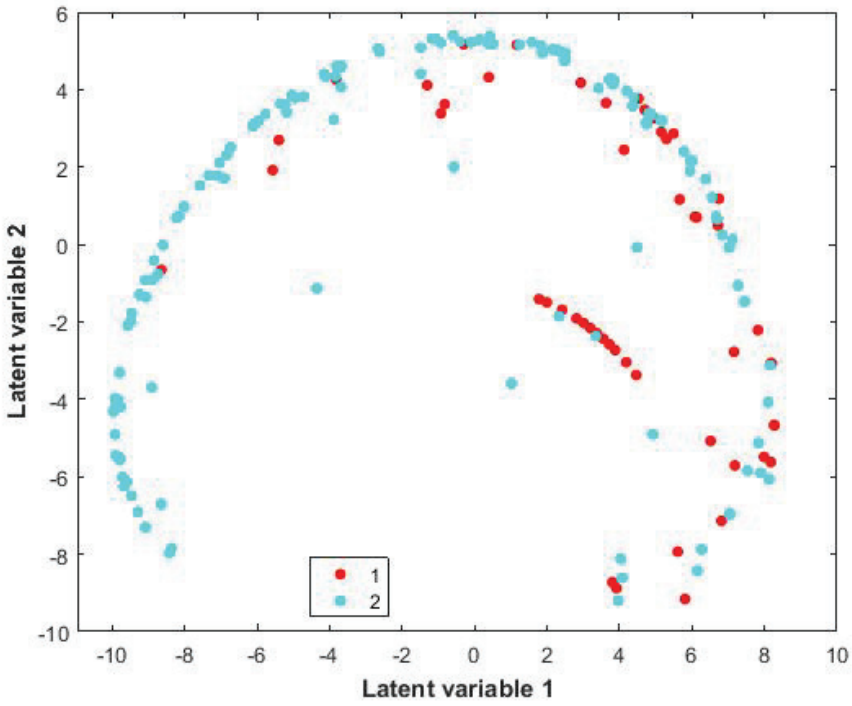

Figure 4.14: *Training samples in latent space after Kernel PCA for global modeling.*

b) The proposed approach for locally weighted classification

In the proposed approach after data segmentation, NCA is used to select the most significant wavelengths. In the next step, Kernel PCA is applied on the training set. The latent space is selected to be two-dimensional, as two latent variables are able to explain most of the variation.

Figure 4.16 shows the training samples in latent space when a given query sample arrives. Each query sample will have its exclusive latent space and classification model, which means Figure 4.16 is specifically for one query sample. Computing the weights based on arrived query sample is done after latent space modeling through Kernel PCA. Computed weights for each query sample will be applied in building the local SVM model for class prediction.

Once all the steps are completed for one query sample, the algorithm goes to the next query sample. When prediction is complete for all query samples, the algorithm starts another combination of data segmentation to ultimately finish Monte Carlo

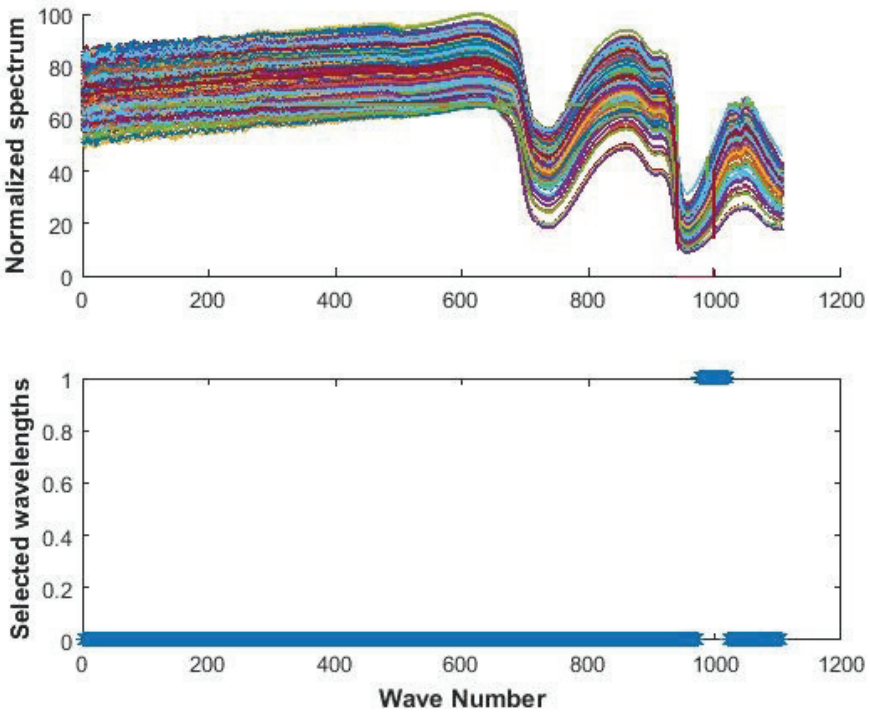

Figure 4.15: *Wavelength selection for locally weighted classification.*

simulation.

Table 4.3 shows the results obtained from global modeling approach as well as the proposed approach. The performance in Table 4.3 is based on the misclassification error only on testing set of data.

Table 4.3: *Comparison of misclassification error for global modeling and the proposed approach on testing set.*

Global modeling			
Kernel for PCA	**Kernel for SVM**	**Min error**	**Average error**
Gaussian	Gaussian	0.1837	0.2652
Proposed approach			
Gaussian	Linear	0.1649	0.2422

The Kernel function used for SVM in the proposed approach is linear, the reason

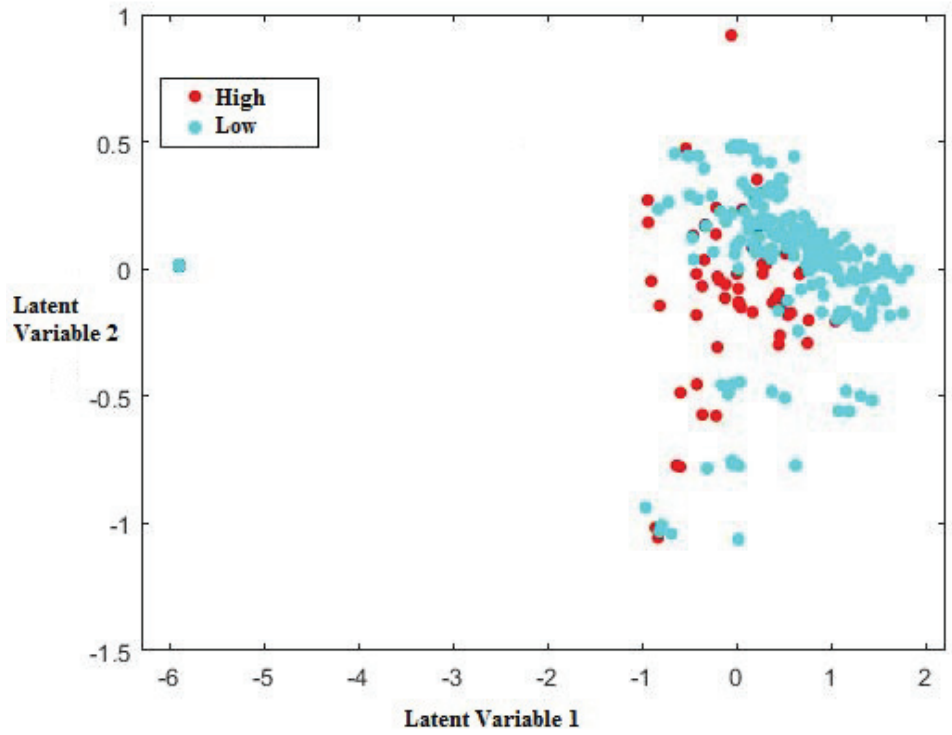

Figure 4.16: *Training samples in latent space after Kernel PCA for the proposed approach.*

of which has been discussed before.

By using Table 4.3, one can conclude that the proposed approach can indeed improve the prediction for classification. The locally weighted approach presented here considers all training samples but with different weights, creating a local model for each given query sample. In summary, it can be said that the proposed approach improves the prediction on testing set when data is highly nonlinear, and it performs better than the global approach.

4.5 Conclusion

In conclusion, a classification approach for high dimensional nonlinear data is proposed in this chapter which utilizes NCA for variable selection, Kernel PCA for dimension reduction, and locally weighted SVM for building classifier. The proposed

approach has the following advantages: (a) It can select the most important variables or variables for classification. (b) It is a local modeling method, meaning that it can deal with nonlinearity of data in classification. (c) It uses a similarity metric to allocate weights to training sample with respect to the query sample. (d) The approach is designed to deal with curse of dimensionality. These characteristic have been successfully tested on two NIR data sets, one to analyze contents of tablets, and the other comes from real operation from an oil sands extraction site.

Chapter 5

Conclusions

5.1 Summary of thesis

This thesis has focused on solving some key challenges in development and implementation of predictive models using JIT and locally weighted modeling. There are various model structures designed to deal with nonlinear processes. However, their complex maintenance and updating requirements raise certain concerns. Traditional model structures such as OLS, PCR, PLS and SVM present a global modeling framework to build predictive models. However, their performance might deteriorate over time due to high degree of nonlinearity of the process or its parameter-varying characteristics. Locally weighted modeling has been proposed to address above mentioned issues in predictive modeling by building as many local models as needed. Hence, the local process is assumed to be linear for each query sample, and a local model is built based on this assumption. The built model will be discarded after making the prediction, and another local model will be built and used for other query samples to come.

The performance of local models highly depends on sample selection or weight assignments, which is the result of similarity measurement step. Existing similarity metrics in JIT only account for magnitude of samples. A similarity metric that can incorporate the correlation of samples with respect to the query sample will improve the prediction. In order to find the correlation of samples over time, one needs to compare two subspaces. A subspace is first formed through a query window which

consists of query sample and the most recent samples. By forming the query window, the trend of samples can be observed over time given a time-series data. Then, a window is taken in historical data, and this window can be compared against query sample window. By calculating PCA error term, one can evaluate the similarity of two windows (subspaces). To compare all the samples in historical database, a moving horizon approach is used. Once all the PCA error terms are obtained, the weight matrix is formed and a prediction is made. LW-PLS is the model structure used. Furthermore, traditional JIT methods have limitations in dealing with missing data. Since the proposed similarity measurement approach utilizes PCA, it can be easily extended to PPCA for evaluating similarity. The proposed approach is accronymed as TB-JIT (**T**rend- **B**ased **J**ust- **I**n- **T**ime).

Additionally, JIT methods mostly evaluate similarity using input space data. Including output information in similarity calculation may also improve the prediction. Thus, the proposed TB-JIT is extended to account for output data in similarity measurement. Therefore, similarity measurement is carried out simultaneously in both input and output space.

Application of locally weighted modeling is not limited to regression purposes. Rather, it can be extended to classification. In a high dimensional data set when the data is not linearly separable, local modeling framework can be used instead of global modeling. Additionally, when we are dealing with curse of dimensionality, a variable selection method can assist in improving the prediction by choosing the most important variables with respect to a classification model. In this work, when a query sample arrives, NCA is used to select a subset of variables to overcome curse of dimensionality. To further compress the data, Kernel PCA is built based on training set of data to transform the set to latent space. By using the built Kernel PCA model, query sample is also transformed to latent space, where similarity measurement step between query sample and training samples is implemented to ultimately obtain weight of each sample in training set with respect to a given query sample. Once the weights are obtained, they can be applied to a SVM model to build a locally weighted model which predicts the output of the query sample.

The case studies in this thesis are through NIR data sets, which are high dimensional. In order to test the efficacy of methods proposed in this thesis, NIR data sets from an oil refinery, an oil sands extraction site as well as from pharmaceutical application are employed, and improved performance is observed on all of them.

5.2 Recommendations for future work

To further improve the proposed approaches in this thesis, a few recommendations for future work are listed here as follows:

1. The proposed TB-JIT approach in chapter 2 and chapter 3 uses a fixed window size to calculate similarities. By using Bayesian inference, the posterior of the window size as a hyperparameter can be computed for each query sample. This way, the method can be extended to varying window size. In other words, window sizes can be defined locally, meaning that there is an optimal window size for each query sample.

2. Missing input data problem in TB-JIT is addressed by using EM algorithm. Variational Bayesian approach can also be taken to model the missing part of the input data.

- **Remark**: The way PCA is able to capture exact similarity still needs to be investigated further in a future work. Possibly, PLS can be an alternative for similarity calculation.

3. Normal Gaussian distribution is used to parameterize error term and latent variables in PPCA. In order to make TB-JIT robust, the approach can take advantage of Student's t distribution which enables the proposed approach to deal with missing inputs as well as being robust with respect to outliers.

4. The similarity metric adopted for evaluating similarities in output space in chapter 3 can also be replaced with other similarity metrics.

5. The variable selection step in the proposed approach in chapter 4 can also be modified. The local predictive classifier structure adopted is SVM. There are methods

which can take advantage of SVM in performing variable selection step. A common issue that might befall to high dimensional data classification problems is variable (feature) exportability. By using SVM for variable selection and local predictive classifier, the possibility for this issue to occur can be decreased.

Bibliography

[1] I. T. Jolliffe and J. Cadima, "Principal Component Analysis: a review and recent developments," *Phil. Trans. R. Soc. A*, vol. 374, no. 2065, p. 20150202, 2016.

[2] H. Abdi, "Partial Least Squares regression and projection on latent structure regression (PLS regression)," *Wiley interdisciplinary reviews: computational statistics*, vol. 2, no. 1, pp. 97–106, 2010.

[3] M. H. Killner, J. J. Rohwedder, and C. Pasquini, "A PLS regression model using NIR spectroscopy for on-line monitoring of the biodiesel production reaction," *Fuel*, vol. 90, no. 11, pp. 3268–3273, 2011.

[4] A. Sharma and D. W. Jacobs, "Bypassing synthesis: PLS for face recognition with pose, low-resolution and sketch," 2011.

[5] S. Wold, N. Kettaneh-Wold, and B. Skagerberg, "Nonlinear PLS modeling," *Chemometrics and intelligent laboratory systems*, vol. 7, no. 1-2, pp. 53–65, 1989.

[6] S. J. Qin and T. J. McAvoy, "Nonlinear PLS modeling using neural networks," *Computers & Chemical Engineering*, vol. 16, no. 4, pp. 379–391, 1992.

[7] Y. H. Bang, C. K. Yoo, and I.-B. Lee, "Nonlinear PLS modeling with fuzzy inference system," *Chemometrics and intelligent laboratory systems*, vol. 64, no. 2, pp. 137–155, 2002.

[8] S. Kim, R. Okajima, M. Kano, and S. Hasebe, "Development of soft-sensor using locally weighted PLS with adaptive similarity measure," *Chemometrics and Intelligent Laboratory Systems*, vol. 124, pp. 43–49, 2013.

[9] S. SCHAAL, "Christopher g. atkeson1'3, andrew w. moore2 and," *Lazy Learning*, vol. 11, pp. 75–113, 1997.

[10] K. Fujiwara, M. Kano, and S. Hasebe, "Correlation-based just-in-time modeling for soft sensor design," *Transactions of the Society of Instrument and Control Engineers*, vol. 44, no. 4, pp. 317–324, 2008.

[11] C. Cheng and M.-S. Chiu, "A new data-based methodology for nonlinear process modeling," *Chemical Engineering Science*, vol. 59, no. 13, pp. 2801–2810, 2004.

[12] X. Yang, D. Lo, X. Xia, and J. Sun, "Tlel: A two-layer ensemble learning approach for Just-In-Time defect prediction," *Information and Software Technology*, vol. 87, pp. 206–220, 2017.

[13] Z. Ge and Z. Song, "A comparative study of Just-In-Time-learning based methods for online soft sensor modeling," *Chemometrics and Intelligent Laboratory Systems*, vol. 104, no. 2, pp. 306–317, 2010.

[14] R. Kohavi *et al.*, "A study of cross-validation and bootstrap for accuracy estimation and model selection," in *Ijcai*, vol. 14, pp. 1137–1145, Montreal, Canada, 1995.

[15] S. Schaal, C. G. Atkeson, and S. Vijayakumar, "Scalable techniques from nonparametric statistics for real time robot learning," *Applied Intelligence*, vol. 17, no. 1, pp. 49–60, 2002.

[16] M. Chen, S. Khare, and B. Huang, "A unified recursive Just-In-Time approach with industrial near infrared spectroscopy application," *Chemometrics and Intelligent Laboratory Systems*, vol. 135, pp. 133–140, 2014.

[17] A. Raich and A. Cinar, "Statistical process monitoring and disturbance diagnosis in multivariable continuous processes," *AIChE Journal*, vol. 42, no. 4, pp. 995–1009, 1996.

[18] M. E. Tipping and C. M. Bishop, "Probabilistic Principal Component Analysis," *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, vol. 61, no. 3, pp. 611–622, 1999.

[19] A. Ilin and T. Raiko, "Practical approaches to Principal Component Analysis in the presence of missing values," *Journal of Machine Learning Research*, vol. 11, no. Jul, pp. 1957–2000, 2010.

[20] F. A. Souza, R. Araújo, and J. Mendes, "Review of soft sensor methods for regression applications," *Chemometrics and Intelligent Laboratory Systems*, vol. 152, pp. 69–79, 2016.

[21] B. Lin, B. Recke, J. K. Knudsen, and S. B. Jørgensen, "A systematic approach for soft sensor development," *Computers & chemical engineering*, vol. 31, no. 5-6, pp. 419–425, 2007.

[22] P. Kadlec, B. Gabrys, and S. Strandt, "Data-driven soft sensors in the process industry," *Computers & chemical engineering*, vol. 33, no. 4, pp. 795–814, 2009.

[23] D. Wang, J. Liu, and R. Srinivasan, "Data-driven soft sensor approach for quality prediction in a refining process," *IEEE Transactions on Industrial Informatics*, vol. 6, no. 1, pp. 11–17, 2010.

[24] Z. Ge, F. Gao, and Z. Song, "Mixture probabilistic PCR model for soft sensing of multimode processes," *Chemometrics and Intelligent Laboratory Systems*, vol. 105, no. 1, pp. 91–105, 2011.

[25] R. Sharmin, U. Sundararaj, S. Shah, L. V. Griend, and Y.-J. Sun, "Inferential sensors for estimation of polymer quality parameters: Industrial application of a PLS-based soft sensor for a LDPE plant," *Chemical Engineering Science*, vol. 61, no. 19, pp. 6372–6384, 2006.

[26] H. Wang, "Gaussian process and its application to soft-sensor modeling," *JOURNAL OF CHEMICAL INDUSTRY AND ENGINEERING-CHINA-*, vol. 58, no. 11, p. 2840, 2007.

[27] J. Gonzaga, L. A. C. Meleiro, C. Kiang, and R. Maciel Filho, "ANN-based soft-sensor for real-time process monitoring and control of an industrial polymerization process," *Computers & chemical engineering*, vol. 33, no. 1, pp. 43–49, 2009.

[28] P. Jain, I. Rahman, and B. Kulkarni, "Development of a soft sensor for a batch distillation column using Support Vector Regression techniques," *Chemical Engineering Research and Design*, vol. 85, no. 2, pp. 283–287, 2007.

[29] S. J. Qin, "Recursive PLS algorithms for adaptive data modeling," *Computers & Chemical Engineering*, vol. 22, no. 4-5, pp. 503–514, 1998.

[30] B. S. Dayal and J. F. MacGregor, "Improved PLS algorithms," *Journal of Chemometrics: A Journal of the Chemometrics Society*, vol. 11, no. 1, pp. 73–85, 1997.

[31] B. S. Dayal and J. F. MacGregor, "Recursive exponentially weighted PLS and its applications to adaptive control and prediction," *Journal of Process Control*, vol. 7, no. 3, pp. 169–179, 1997.

[32] L. M. Elshenawy, S. Yin, A. S. Naik, and S. X. Ding, "Efficient recursive Principal Component Analysis algorithms for process monitoring," *Industrial & Engineering Chemistry Research*, vol. 49, no. 1, pp. 252–259, 2009.

[33] S. Schaal, C. G. Atkeson, and S. Vijayakumar, "Real-time robot learning with locally weighted statistical learning," in *Robotics and Automation, 2000. Proceedings. ICRA'00. IEEE International Conference on*, vol. 1, pp. 288–293, IEEE, 2000.

[34] S. Vijayakumar and S. Schaal, "Locally weighted projection regression: An O (n) algorithm for incremental real time learning in high dimensional space," in *Proceedings of the Seventeenth International Conference on Machine Learning (ICML 2000)*, vol. 1, pp. 288–293, 2000.

[35] A. Saptoro, "State of the art in the development of adaptive soft sensors based on Just-In-Time models," *Procedia Chemistry*, vol. 9, pp. 226–234, 2014.

[36] M. Dyrby, S. B. 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*, vol. 56, no. 5, pp. 579–585, 2002.

[37] R.-S. Kwan, A. C. Evans, and G. B. Pike, "MRI simulation-based evaluation of image-processing and classification methods," *IEEE transactions on medical imaging*, vol. 18, no. 11, pp. 1085–1097, 1999.

[38] S. Ramaswamy, P. Tamayo, R. Rifkin, S. Mukherjee, C.-H. Yeang, M. Angelo, C. Ladd, M. Reich, E. Latulippe, J. P. Mesirov, *et al.*, "Multiclass cancer diagnosis using tumor gene expression signatures," *Proceedings of the National Academy of Sciences*, vol. 98, no. 26, pp. 15149–15154, 2001.

[39] B. Lavine and J. Workman, "Chemometrics," *Analytical chemistry*, vol. 80, no. 12, pp. 4519–4531, 2008.

[40] R. Isermann, "Supervision, fault-detection and fault-diagnosis methodsan introduction," *Control engineering practice*, vol. 5, no. 5, pp. 639–652, 1997.

[41] S. Yin, S. X. Ding, A. Haghani, H. Hao, and P. Zhang, "A comparison study of basic data-driven fault diagnosis and process monitoring methods on the benchmark Tennessee Eastman process," *Journal of Process Control*, vol. 22, no. 9, pp. 1567–1581, 2012.

[42] M. Kim, Y.-H. Lee, and C. Han, "Real-time classification of petroleum products using Near-InfraRed spectra," *Computers & Chemical Engineering*, vol. 24, no. 2-7, pp. 513–517, 2000.

[43] Y. Roggo, P. Chalus, L. Maurer, C. Lema-Martinez, A. Edmond, and N. Jent, "A review of Near InfraRed spectroscopy and chemometrics in pharmaceutical

technologies," *Journal of pharmaceutical and biomedical analysis*, vol. 44, no. 3, pp. 683–700, 2007.

[44] Z. Deng, X. Zhu, D. Cheng, M. Zong, and S. Zhang, "Efficient knn classification algorithm for big data," *Neurocomputing*, vol. 195, pp. 143–148, 2016.

[45] J. Ye, R. Janardan, and Q. Li, "Two-dimensional linear discriminant analysis," in *Advances in neural information processing systems*, pp. 1569–1576, 2005.

[46] A. McCallum, K. Nigam, *et al.*, "A comparison of event models for Naive Bayes text classification," in *AAAI-98 workshop on learning for text categorization*, vol. 752, pp. 41–48, Citeseer, 1998.

[47] T. S. Furey, N. Cristianini, N. Duffy, D. W. Bednarski, M. Schummer, and D. Haussler, "Support Vector Machine classification and validation of cancer tissue samples using microarray expression data," *Bioinformatics*, vol. 16, no. 10, pp. 906–914, 2000.

[48] W. Wu, B. Walczak, D. Massart, S. Heuerding, F. Erni, I. Last, and K. Prebble, "Artificial Neural Networks in classification of NIR spectral data: design of the training set," *Chemometrics and intelligent laboratory systems*, vol. 33, no. 1, pp. 35–46, 1996.

[49] C. Zhu, X. Ji, C. Chen, R. Zhou, L. Wei, and X. Zhang, "Improved linear classifier model with NYSTRöm," *PloS one*, vol. 13, no. 11, p. e0206798, 2018.

[50] S. Das, "Filters, wrappers and a boosting-based hybrid for feature selection," in *Icml*, vol. 1, pp. 74–81, 2001.

[51] L. Talavera, "An evaluation of filter and wrapper methods for feature selection in categorical clustering," in *International Symposium on Intelligent Data Analysis*, pp. 440–451, Springer, 2005.

[52] J. Tang, S. Alelyani, and H. Liu, "Feature selection for classification: A review," *Data classification: Algorithms and applications*, p. 37, 2014.

[53] S. W. Choi, J. H. Park, and I.-B. Lee, "Process monitoring using a gaussian mixture model via principal component analysis and discriminant analysis," *Computers & chemical engineering*, vol. 28, no. 8, pp. 1377–1387, 2004.

[54] C. G. Atkeson, A. W. Moore, and S. Schaal, "Locally weighted learning for control," in *Lazy learning*, pp. 75–113, Springer, 1997.

[55] E. Frank, M. Hall, and B. Pfahringer, "Locally weighted naive bayes," in *Proceedings of the Nineteenth conference on Uncertainty in Artificial Intelligence*, pp. 249–256, Morgan Kaufmann Publishers Inc., 2002.

[56] C. Domeniconi and D. Gunopulos, "Adaptive Nearest Neighbor classification using Support Vector Machines," in *Advances in Neural Information Processing Systems*, pp. 665–672, 2002.

[57] J. Goldberger, G. E. Hinton, S. T. Roweis, and R. R. Salakhutdinov, "Neighbourhood Components Analysis," in *Advances in Neural Information Processing Systems 17* (L. K. Saul, Y. Weiss, and L. Bottou, eds.), pp. 513–520, MIT Press, 2005.

[58] N. M. Nasrabadi, "Pattern recognition and machine learning," *Journal of electronic imaging*, vol. 16, no. 4, p. 049901, 2007.

[59] R. Kohavi and G. H. John, "Wrappers for feature subset selection," *Artificial intelligence*, vol. 97, no. 1-2, pp. 273–324, 1997.

[60] Y. Sun, S. Todorovic, and S. Goodison, "Local-learning-based feature selection for high-dimensional data analysis," *IEEE transactions on pattern analysis and machine intelligence*, vol. 32, no. 9, pp. 1610–1626, 2010.

[61] C.-F. Lin and S.-D. Wang, "Fuzzy Support Vector Machines," *IEEE transactions on neural networks*, vol. 13, no. 2, pp. 464–471, 2002.

[62] W. Yang, K. Wang, and W. Zuo, "Neighborhood component feature selection for high-dimensional data.," *JCP*, vol. 7, no. 1, pp. 161–168, 2012.

[63] H. D. Harris, "Evidence that incremental delta-bar-delta is an attribute-efficient linear learner," in *European Conference on Machine Learning*, pp. 135–147, Springer, 2002.

[64] B. Schölkopf, A. Smola, and K.-R. Müller, "Kernel Principal Component Analysis," in *International Conference on Artificial Neural Networks*, pp. 583–588, Springer, 1997.

[65] J. Weston and C. Watkins, "Multi-class Support Vector Machines," tech. rep., Citeseer, 1998.

[66] C.-J. Hsieh, K.-W. Chang, C.-J. Lin, S. S. Keerthi, and S. Sundararajan, "A dual coordinate descent method for large-scale linear SVM," in *Proceedings of the 25th international conference on Machine learning*, pp. 408–415, ACM, 2008.

[67] R. A. Melter, "Some characterizations of city block distance," *Pattern recognition letters*, vol. 6, no. 4, pp. 235–240, 1987.

[68] T. J. Penner and J. M. Foght, "Mature Fine Tailings from oil sands processing harbour diverse methanogenic communities," *Canadian journal of microbiology*, vol. 56, no. 6, pp. 459–470, 2010.