SOFT SENSOR DESIGN IN CHEMICAL PROCESSES USING STATISTICAL LEARNING METHODS

by

Aysun Urhan

B.S., Chemical Engineering, Boğaziçi University, 2016

Submitted to the Institute for Graduate Studies in
Science and Engineering in partial fulfillment of
the requirements for the degree of
Master of Science

Graduate Program in Chemical Engineering

Boğaziçi University

2018

ACKNOWLEDGEMENTS

The amount of support I've received during my work on this thesis is beyond fathomable, I am thoroughly grateful to everyone. Some of whom I must mention...

First, I'd like to thank my advisor, Assoc. Prof. Burak Alakent, for sparking my interest in statistics, and his continual supervision and help, but most importantly, for his never-ending tolerance for me from the day one. I feel truly privileged to be working with him, for he has not only guided me in the infancy of my academic career, but also in every aspect of my life. The freedom he has given me to speak my mind at all times is what made me build confidence as an independent researcher. While he often points out how much I remind him of himself, I can only hope to be half as good as he is, some day in the future \m/.

I would also like to thank my thesis committee, Assoc. Prof. Erdem Günay and Prof. Ramazan Yıldırım for their valuable questions, and comments which I have highly appreciated.

I want to thank my parents for bringing me up and instilling me with values, they have planted the seeds of my love of lifelong learning. Their infinite support, care and love are the main reasons for my academic achievements.

My friends have been largely responsible for cheering me up, and keeping me sane during my studies. It'd be too involving to name all my friends who have been there for me, and shown me great love and support at all times. I'm forever grateful to everyone, but I must particularly thank Ece Çiğdem Mutlu, who has greatly encouraged, and motivated me during our time as a "lab couple".

I owe many thanks to the Department of Chemical Engineering: my professors and the staff for all their help in the last 6 years, throughout my studies.

I would also like to thank Nevin Gerek İnce and Ronald Bondy from AVEVA Group, for providing the SimSci PROII^{TM} license, the simulation model of crude distillation unit and much appreciated technical support.

ABSTRACT

SOFT SENSOR DESIGN IN CHEMICAL PROCESSES USING STATISTICAL LEARNING METHODS

Today, production facilities employ rigorous approach to process control, monitoring and fault detection strategies, due to increased competition in the industry, and more severe manufacturing restrictions and safety concerns. A large number of process variables are measured online during operation, while quality variables, which may not be available online at the same rate as process variables need to be predicted using soft sensors. The traditional approach to data driven soft sensor design is based on statistical learning methods, such as, global ARX modeling, PLS and PCR. In this thesis, it is aimed to address the issues of multicollinearity and redundancy in process data, and concept drift in data driven soft sensor design. Experiments are performed on two synthetic datasets obtained from steady state and dynamic simulations, and one dataset comprised of dynamic industrial data. Incorporating feature selection and ensemble modeling into PLS and ANN models is shown to handle multicollinearity and redundancy in process data, and stabilize learners. RVM, an embedded feature selection method, yields superior prediction performance than PLS, under both virtual and real concept drift. RVM is also observed to handle redundancy in predictor space more effectively. Several RVM-based adaptive learning algorithms are developed to cope with concept drift. Adaptive window sizing in moving window models is shown to improve predictions, and the best overall performance is achieved in window size adjustment via explicit concept drift detection. Combining MW and JITL models in ensemble learning is suggested to further increase prediction accuracy, and it is observed to yield the best predictive performance among all algorithms. The suggested adaptive learners are shown to outperform conventional methods from the literature on both synthetic and real data, while complying with time limits of online prediction.

ÖZET

İSTATİSTİKSEL ÖĞRENME YÖNTEMLERİNİ KULLANARAK KİMYASAL SÜREÇLERDE HESAPSAL SENSÖR TASARIMI

Günümüzde artan rekabet, imalat kısıtlamaları ve güvenlik tedbirleri nedeniyle, süreç kontrol, izleme ve arıza tespit stratejileri kimyasal tesislerde daha titizlikle uygulanmaktadır. Bu sebeple, kimyasal süreçlerde çok sayıda süreç değişkeni çevrimiçi olarak ölçülmekte, diğer taraftan ürünün kalitesinde daha önemli rol oynayan kalite değişkenleri aynı sıklıkta ölçülememektedir. Son yıllarda popülerlik kazanan veri-bazlı hesapsal (soft) sensörler geleneksel olarak ARX modelleme, PLS ve PCR gibi evrensel istatistiksel öğrenme yöntemlerini kullanarak, kalite değişkenlerinin tahmin edilmesini sağlamaktadır. Bu tezde, doğrusal ilişkili ve gereğinden fazla sayıda süreç değişkenin izlendiği, ve kavram değişimi gibi sık rastlanılan problemlerin etkisi altında olan kimyasal süreçler için hassas tahmin becerisine sahip veri-bazlı hesapsal sensörler tasarlanmıştır. Tasarlanan sensörler, kararlı durum ve dinamik benzetimlerden elde edilen iki veri seti ve endüstriyel bir dinamik veri seti üzerinde test edilmiştir. Değişken seçimi ve topluluk öğrenimi, doğrusal ilişkinin üstesinden gelmek, ve tahmin modellerini kararlı hale getirmek için önerilmiştir. RVM bazlı hareketli pencere (MW) modellerinde, modellenen pencere boyutunun o anki kavrama adapte edilmesi, ve topluluk öğrenimi yaklaşımı çerçevesinde, MW ve JITL modellerinin birleştirilmesi, tahminleri iyileştirmek için önerilmiştir. Geliştirilen algoritmalar arasında en doğru tahminler, topluluk öğreniminde elde edilmiş, ayrıca bu algoritmaların, literatürden ilgili geleneksel yöntemlerden daha iyi performans gösterdikleri, ve hesap süresinin çevrimiçi zaman kısıtlamalarına uyduğu görülmüştür.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS

a_u	Random disturbance term for input variable u
$bias$	Bias operator
cov	Covariance operator
$\mathcal{D}$	Data
D	Dataset comprising observation as input and output pairs
$diag$	Diagonal operator in linear algebra
$\mathrm{DS}_{i,\mathrm{int}}$	Interpolation dataset of i^{th} simulation setting
$\mathrm{DS}_{i,\mathrm{ext}}$	Extrapolation dataset of i^{th} simulation setting
$\hat{d}$	Filtered distance in AD2
d_{max}	Distance threshold ın AD2
d_{M}	Mahalanobis distance in AD1A
E	Expectation operator
Eff_i	Efficiency of i^{th} learner
Eff'_i	Worst case scenario efficiency of i^{th} learner
$\mathcal{F}_{\nu_1,\nu_2}$	F-distributed random variable with dof ν_1 and ν_2
F	Ratio of two model variance estimates in AD1C
f	Model
$\hat{f}$	Estimated model
$\mathrm{FS}^{\mathrm{cal}}$	Forward selection of predictors using calibration error
$\mathrm{FS}^{\mathrm{val}}$	Forward selection of predictors using validation error
I	Mutual information *or* indicator function
I_{M}	Result of hypothesis test on Mahalanobis distance
K	Number of hidden units
k	Number of folds in CV
$\mathcal{L}$	Learner
L	Number of PLS components
LB	Lower bound of EWMA chart in AD1C
$logF$	Logarithm of ratio of two model variance estimates in AD1C

MW	Set of time stamps of training observations in MW model in AD2
$\widetilde{\text{MW}}$	Set of time stamps of training observations in JITL model in AD2
mp	Maximum number of past validation points in AD2
$\mathcal{N}$	Normal distribution
N	Size of a set of observations
NS	Neighborhood size
n	Number of lags
PLS_{Cr}	Ensemble PLS model constructed via crogging
PLS_{MW}	PLS based MW model
$\text{PLS}_{\text{QuadIn}}$	PLS model using quadratic inner relation
PLS_{Sc}	PLS model scaled using variance obtained from $\mathbf{X}$ residuals
p	Probability *or* number of predictors
p'	Effective number of parameters
R	Number of repetitions in repeated CV
R^2_{adj}	Adjusted R-squared
R^2_p	Predicted R-squared
$rank$	Rank operator in linear algebra
RC	Redundancy/correlation metric
Red_{Corr}	Feature selection using RC metric
RM	Redundancy/mutual information metric
Red_{MI}	Feature selection using RM metric
RR_i	Set of time stamps of observations in i^{th} local region in AD2
RVM_{MW}	RVM based MW model
S_v	Time stamp of validation points in AD2
S_{W}	A set of candidate window sizes in AD1B and AD1C
SS_{R}	Sum of squared errors
SS_{T}	Total sum of squares
t	Time
u_t	Value of input variable u at time t

UB	Upper bound of EWMA chart in AD1C
UB_α	Upper confident limit at significance level α in AD1C
V	Variance operator
VIF	Variance inflation factor
$\widetilde{\mathrm{W}}$	JITL model window size in AD2
W^{U}	Upper limit of window size
W_{L}	Lower limit of window size
w_{MW}	Ensemble weight of MW model in AD2
$\mathbf{x}_i$	Input of i^{th} observation
y_i	Output of i^{th} observation
$\mathbf{X}$	Input variable matrix
$\mathbf{y}$	Output variable vector
x	Model input
y	Model output
$\hat{y}$	Output estimate
Z	EWMA variable in AD1C
β	Regression coefficient
δ	Window size increments *or* number of observations slided
ϵ	Random measurement error
λ	Regularization parameter in RR and Lasso *or* EWMA weight
λ_f	Forgetting factor
μ	Mean
ν	Degrees of freedom
ϕ	AR coefficient
σ^2	Variance
σ_ϵ^2	Variance of random measurement error
$\hat{\sigma}^2$	Model error variance estimate
$\mathbf{\Sigma}$	Covariance matrix
ρ	Correlation coefficient
Θ	Set of all model parameters accessible

θ	Model parameters
τ	Set of time stamps

LIST OF ACRONYMS/ABBREVIATIONS

AD	Adaptive Learner
AD1	Moving window with adaptive window size
AD2	Ensemble learning with moving window and JITL
ADWIN	Adaptive Windowing
AET	Average Elapsed Time
AI	Artificial Intelligence
AIC	Akaike's Information Criterion
ANN	Artificial Nerual Network
AOSVR	Accurate Online Support Vector Regression
API	Active Pharmaceutical Ingredient
AR	Auto-Regressive
ARL	Average Run Length
ARX	Auto-Regressive with Exogenous Inputs
ASTM	American Society for Testing and Materials
BIC	Bayesian Information Criterion
BMA	Bayesian Model Averaging
CD	Concept Drift
CDM	Concept Drift Model
CDU	Crude Oil Distillation Unit
CoJITL	Correlation-Based Just-In-Time Learning
CSTR	Continuous Stirred Tank Reactor
CUSUM	Cumulative Sum
CV	Cross Validation
CVE	Cross Validation Error
DLOER	Dual Learning-based Online Ensemble Regression
DS	Data Set
DW	Dataset Windowing
ELM	Extreme Learning Machine

ELWPLS	Ensemble Locally Weighted Partial Least Squares
EWMA	Exponential Weighted Moving Average
FIR	Finite Impulse Response
FS	Feature Selection
FTS	Fuzzy Takagi-Sugeno
GMM	Gaussian Mixture Model
GPR	Gaussian Process Regression
i.i.d.	Independent and Identically Distributed
ILLSA	Incremental Local Learning Soft Sensing Algorithm
IV	Input Variable
JIT	Just-In-Time
JITL	Just-In-Time Learning
LARPLS	Localized and Adaptive Recursive Partial Least Squares
LARS	Least Angle Regression
LASS	Localized Adaptive Soft Sensor
LB	Lover Bound
LC	Level Controller
LOOCV	Leave-One-Out Cross Validation
LR	Linear Regression
LS	Least Squares
LS-SVM	Least Squares Support Vector Machine
LW-KPLS	Locally Weighted Kernel Partial Least Squares
LWR	Locally Weighted Regression
MAE	Mean Absolute Error
maxAE	Maximum Absolute Error
MI	Mutual Information
MLP	Multilayer Perceptron
MRMR	Minimum Redundancy Maximum Relevance
MSE	Mean Squared Error
MW	Moving Window
NGRJIT	Non Gaussian Regression Just-In-Time

NIPALS	Nonlinear Iterative Partial Least Squares
NN	Neural Network
NNPLS	Neural Net PLS
OEOA	On-line Ensemble of regressor models using Ordered Aggregation
OSVM	Online Support Vector Machine
PCA	Principle Component Regression
PCR	Principle Component Regression
PE	Prediction Error
PLS	Partial Least Squares
ReLu	Rectified Linear Units
RMSE	Root Mean Squared Error
RNN	Recurrent Neural Network
RPLS	Recursive Partial Least Squares
RR	Ridge Regression
RVM	Relevance Vector Machine
S	Simulation
sdRMSE	Standard Deviation of RMSE
SELPLS	Selective Ensemble of Local Partial Least Squares
SPC	Statistical Process Control
SS	Sum of Squares
SVDD	Support Vector Data Description
SVM	Support Vector Machine
SVR	Support Vector Regression
TC	Temperature Controller
UB	Upper Bound
VAE	Variational Autoenchoders
WLS	Weighted Least Squares

1. INTRODUCTION

Throughout the years, manufacturing industries have been known to be competitive for delivering goods to customers, and making profit. In the advent of the fourth industrial revolution, i.e. "industry 4.0", industrial companies are driven to increase efficiency, and integrate technological advances with the rest of the physical and human components of a plant in order to compete in today's world [1]. In addition, more recent issues regarding safety regulations in manufacturing, and environmental concerns have led to tougher restrictions on production facilities, thereby increasing the need for rigorous control and monitoring operations to ensure the production of final goods, which meet the desired quality. For that reason, process plants have been set up with measurement devices, which has resulted in an abundance of data collected from manufacturing processes. Stream flow rate, temperature, pressure and etc. are among examples of these process variables which can be efficiently measured online due to advances in hardware technology and data collection techniques. However, certain attributes of process components, such as product concentration, often cannot be measured online at the same rate as process variables. These attributes, referred to as "quality variables", are of great importance since they may define the product quality, hence most operational plans of process monitoring and control in a plant are centered around these quality variables. While it is required to monitor these attributes in real time during manufacturing [2], there exist problems in rendering online acquisition of quality variables, e.g. most of quality variables are difficult to measure and require specific laboratory analyses which are often tedious, and in some cases manufacturing conditions may hinder the use of measurement devices when problematic chemicals are considered. "Soft sensors" (also called inferential sensors, or virtual on-line analyzers) have been introduced for online prediction of quality variables in the absence of online sensor measurements. In addition to real time predictions, soft sensors may also be used for validation of hardware measurements, and as backup lest hardware failure [2]. There are three main groups of methodologies used for soft sensor design: model based, data driven and fuzzy modeling based design, also referred to as white, black and gray

box approaches, respectively. Recently, there has been a growing interest in data driven approach to soft sensor design, conceivably as a reflection of increased popularity of statistical sciences, in numerous fields of industry and academia, e.g. finance, computer science, machine learning and artificial intelligence (AI) research [3]. It may be argued that, data driven soft sensor design has become more attractive, as a natural extension to advances in rigorous data collection and computation technology, both of which have enabled accumulation of large databases, i.e. "big data" [4].

Statistical learning methods are used to extract information, and learn patterns and dependencies from data [5]. Problem of pattern recognition, and statistical inference for generalization are encountered in many fields, hence statistical learning is composed of multidisciplinary research areas [6]. The field of statistical learning comprises of vast research on diverse techniques, however at its core, statistical learning is basically approximating functional relationship between inputs (predictor variables) and outputs (response variables). The conventional approach to data driven time-dependent soft sensor design is based on global modeling techniques, e.g. autoregressive models with exogenous input (ARX), and finite impulse response (FIR) models are among the most widely used linear modeling methods in which the output (response variable) is estimated from past output and inputs (predictor variables). In addition, principle component regression (PCR) and partial least squares (PLS) are possibly the most popular latent variable models with a linear functional form [7]. Though not as much as linear techniques, nonlinear modeling techniques, such as support vector regression (SVR), and artificial neural network (ANN) models have also found many applications in soft sensor design [8].

Data collected from chemical processes, and environment in which soft sensors are to be operated both have certain characteristics, which may complicate soft sensor design. For instance, with the increase in number of measured process variables, one may encounter problems such as multicollinearity and redundancy among observations. Furthermore, in the absence of any knowledge on the process, it may be difficult to determine the subset of process variables, and the number of lagged measurements to

include as inputs to the predictive model [2]. Another major concern in industrial plants is maintenance of soft sensors; even if an accurate soft sensor model has been estimated from data initially, performance of the predictive model may deteriorate due to time varying characteristics of process data. This problem, often called "concept drift" in machine learning literature, invalidates most assumptions in statistical learning, e.g. independent and identically distributed (i.i.d.) observations, in which the underlying input-output relationships remain constant, do not hold. Therefore, different approaches to learning and evaluation of learning algorithms (or learners) should be adopted when data is received in the form of streams.

In the current thesis, the aim is to design, and evaluate soft sensors which can address major issues encountered in industrial process data using statistical learning methods. This thesis is organized as follows: The subject of soft sensor design is described, and different methodologies developed for designing soft sensors are outlined along with corresponding examples from the literature in Section 2. In Section 3, a detailed background on theory and mathematics of statistical learning methods used in this thesis is given. The framework for statistical learning in a data stream scenario, and specific problems regarding learning are introduced in Section 4. Soft sensor data, both synthetic and from a real industrial process are presented and described in Section 5. Various combinations of statistical learning methods and novel adaptive methods for designing soft sensors are proposed and evaluated in three parts in Section 6. Finally, a summary of major conclusions from this study, and recommendations for future work are given in Conclusions and Recommendations.

2. SOFT SENSORS

Increased competition in chemical industry has led to tougher measures on operational tasks to regulate production quality, and maximize profits. In addition, industrial plants of today are responsible for operating within strict safety regulations, and heightened standards with environmental concerns. These issues can be handled by employing tight control and monitoring of the entire plant systems. In order to maintain stricter production specifications, plants have been equipped with numerous measurement devices; developments in hardware technology have enabled such gratuitous use of process measurements. In an industrial process, measurements of properties such as flow rate, temperature, pressure, are collected online during operation. These variables are referred to as process variables, in general. Quality variables, on the other hand, are direct indicators of product quality; for instance concentration or boiling temperature of a crude oil fraction in a distillation column effluent, melt index of the product of a polymerization process or the concentration of product in reactor outlet stream [2]. Most process control, monitoring and fault detection strategies in a plant are based on such quality variables, and should be performed in real time to proceed with operation. However, online measurements of quality variables may not always be available since they are often too difficult to measure due to lengthy laboratory analyses, and obtained with time delay, or harsh production conditions, which may prevent the use of hardware devices in the first place. In the absence of online sensor measurements, soft sensors may be used for online prediction of quality variables. Even in cases where measurements are readily available, soft sensors can still be utilized in parallel with the existing hardware system as a validation tool, and as a back up for the online measuring devices to reduce maintenance costs [2]. In summary, soft sensors are vital in the sense that they provide efficient, cost effective and solid help for solving technical issues.

Generally speaking, there are three different strategies for soft sensor design: (i) model based design using mechanistic laws, (ii) data driven design based on statisti-

cal learning techniques, and (iii) fuzzy modeling, which relies on both (i) and (ii). In practice, the first approach for soft sensor design is infeasible as industrial processes are highly complex systems, for which it is too difficult to write down mechanistic equations; there are too many unknowns and uncertainties in the systems to deduce accurate mechanistic models. Data based approach, on the other hand, has become attractive, particularly with recent advances in hardware technology, leading to a collection of abundant process data, i.e. big data. It should be noted that more data does not entail higher quality of data; data collection from industrial plants, and consequent preprocessing are both separate research topics, not discussed within this thesis. However, the importance of sampling should still be emphasized with the "garbage in garbage out" principle in information and data related fields: output from a statistical model is only as good as its input.

Data driven soft sensor design differs from the model driven alternative, in the sense that model building stage is unallied to physical characteristics of the process. It is a black box approach to process modeling, which relies solely on data, instead of white box modeling strategies developed on physical laws. In gray box modeling, expert knowledge on a process is combined with statistical learning tools, however this has proven to be quite challenging. Gray box approach for developing soft sensors is attractive in many ways, it can provide interpretable results which can be further used for production planning and optimization purposes, and engineers would be contributing to soft sensor development, by adding their valuable expert knowledge and hence, play a more proactive role in the process [4]. Even though there have been some practical applications of gray box techniques, e.g. an evolving fuzzy rule-based model was used for predicting crude oil quality in a distillation column [9], perhaps due to growing interest in purely data based approaches, published research on black box techniques seems to be higher in number than that on gray box techniques [10–12].

The current thesis aims to provide a systematic approach, which incorporates recent advances in fields, such as statistics, machine learning and artificial intelligence, for soft sensor design. In this section, data driven soft sensor design methodology is

discussed, and majors issues in soft sensor design and some prominent examples from the literature are examined.

2.1. Data Based Soft Sensor Design

Soft sensor design consists of a number of consecutive, yet connected steps (Figure 2.1). With the advances in hardware technology, it is possible to obtain a large amount of online measurements from a chemical plant. However, quantity does not necessarily imply quality, and additional issues arise in collecting data. Mismatches in sampling times of different devices, occasional hardware failure resulting in missing data, outliers, and inconsistent measurement accuracy are among the most common problems with data acquisition. Preprocessing, in this context, encapsulates the aforementioned steps with all techniques employed on sensor data, between the collection instant from the process to the point, at which data is to be directly used by various tools of statistical learning.

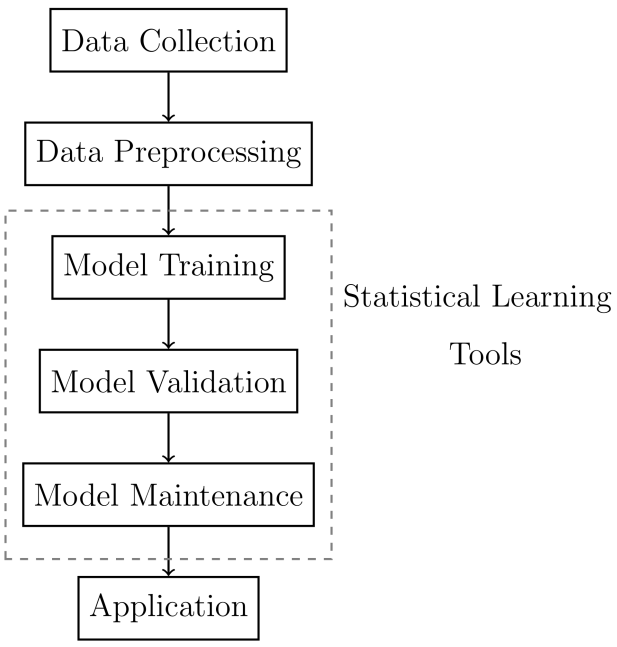

Figure 2.1. Steps of data based soft sensor design [4].

Missing values in sensor measurements result from physical causes related to the hardware itself; sensors need regular maintenance and repair as they are subject to continuous abrasion, or they might be replaced entirely in the case of failure. Techniques developed for handling missing data can be classified into two groups: ad-hoc methods and statistical methods [13]. Removing the missing values, along with the accompanying measurements obtained from other sensors is an example of an ad-hoc technique. Statistical approaches to deal with incomplete data fall under the general term, data imputation; a trivial solution to missing values is to replace the missing value with the mean or median of the process variable. More sophisticated statistical methods based on latent variable models such as PCA and PLS have also been developed [14]. Data imputation can be employed in a Bayesian framework as well [13].

Outliers in sensor data can arise not only from the same causes as missing values, but also due to changes in process operating conditions, or process disturbances. Adequate detection and, if necessary, removal of outliers are crucial steps in data based soft sensor design, since predictive performance of statistical learning algorithms can severely deteriorate in the presence of outliers. Simplest approach to outlier detection is using the mean and variance of observations in one dimension to compute 3σ limits [2]. While, robust counterparts of these statistics, median and median absolute deviation from the median, respectively, are used in Hampel identifier to determine limits for detection [15]. T^2 statistic and Jolliffe parameter obtained from PCA are used for multivariate outlier detection [16].

While the soft sensor model is only as good as the data fed to it, developing a decent statistical learning algorithm is the heart of data based soft sensor design. Model training, validation and maintenance can be identified as three steps of statistical modeling. Global modeling techniques had been the traditional approach to soft sensor design, for instance, ARX models were among the most popular choices for soft-sensor design [17]. In ARX modeling, all observations in the historical dataset are used to build a single model, output is a function of past input and output values and the model is trained offline. Parameters in ARX models are easy to estimate, and can

be extended to include higher order polynomial terms to explain more complex input-output relations [18]. FIR models are another example of popular linear modeling techniques in soft sensor design. In addition to the most recent measurements, lagged measurements of process variables are used as the model input in FIR models, and the model order can be increased by considering larger number of lagged inputs [19]. Other batch modeling techniques based on linearly parametrized models include latent variable models, such as PCR and PLS. Dimensionality reduction through feature extraction, particularly PLS, is quite popular in the soft sensor literature. Prominence of feature extraction is due to the highly collinear nature of process data, e.g. subsequent tray temperatures in a distillation column are bound to follow similar trajectories, and PLS serves as a simple model which can handle this problem while still being interpretable. Linear methods, however, can be inadequate to model the highly nonlinear dynamics; thus, more complex modeling techniques with greater number of parameters should be considered. More recently developed nonlinear models popularized in statistical learning and artificial intelligence fields, such as SVR, and ANN models with different structures, e.g. feedforward networks and recurrent neural networks, have been incorporated into soft sensor design methodology as well [4].

2.2. Issues in Soft Sensor Design

Several practical issues in soft sensor development and maintenance exist, and the most important ones can be grouped into two: (i) challenges in data collection, and process measurements, and (ii) issues in statistical modeling of a process (Figure 2.2). These two groups often overlap, as some of the points in the first group may lead to major problems in developing statistical models, and both groups should be considered simultaneously taking their interactions into account.

Collinearity and redundancy, while being issues that arise within the data collection step as a result of the nature of process data, they have major influence on the performance of statistical methods developed for soft sensor design.

Measurements of process variables are known to exhibit high collinearity due to the abundance of measurement devices as well as the choice of their placements. This largely results from the fact that the main purpose of measuring a process variable is for process control applications, not predictive modeling. Hence, process industries are often called "data rich, information poor" [20].

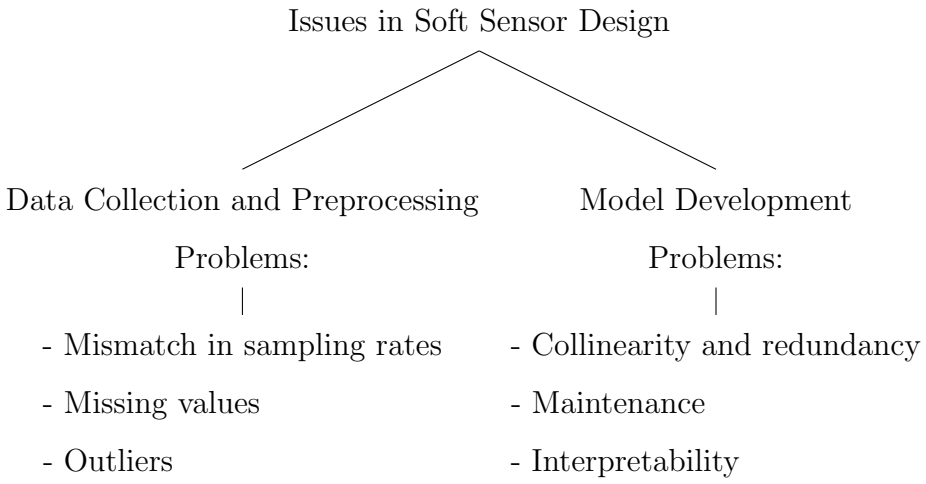

Figure 2.2. Outline of issues in soft sensor design.

Similarly, it is often desired to include a large number of lagged inputs in FIR models, however subsequent measurements of the same process variables lead to redundancy as they are highly correlated. To make accurate predictions in the presence of multicollinearity and redundancy, we need to adopt appropriate tools developed in the field of statistical learning. There are two main approaches to address this issue with dimensionality reduction techniques: feature extraction and selection. In feature extraction, inputs are projected (or transformed) onto a new space in which the collinearity is no longer a problem; the transformed variables then become the input to the model. This projection can be either linear, as in PCA and PLS, or nonlinear, as in variational autoencoders (VAE) [21]. Feature extraction is quite popular in soft sensor design, PLS being one of the most widely used models [22]. Results from feature extraction algorithms can often be interpreted to extract knowledge from the process dynamics [23].

In feature selection (or predictor subset selection), on the other hand, the aim is to remove inputs, which do not provide additional information about the process outputs, to reduce collinearity, and determine model parameters using the remaining inputs. Feature selection is particularly beneficial for predictive modeling; it can improve predictive performance while reducing computational burden and memory requirements [24]. The main idea in most feature selection algorithms is to rank inputs according to a criteria, and retain only a certain number of the inputs for prediction. Several PLS based feature selection methods have been utilized for soft sensor design [22]. Feature selection methods are often divided into three groups: filter, wrapper and embedded. Filter and wrapper approach differ in how they define the metric for evaluating the relevance of inputs. In filter algorithms, this metric is independent of the predictive method used following the feature selection step. However, in the wrapper approach, the prediction model needs to be considered when ranking variables. Embedded methods, on the other hand, perform feature selection during training as part of their parameter estimation algorithms. Lasso and relevance vector machine (RVM) are among some examples of embedded methods [6].

Soft sensor maintenance is currently the leading problem encountered by practitioners in the industry [22]. Soft sensors are expected to provide good estimation performance not only when they are first installed, but also throughout the process operation. Industrial processes, however, exhibit nonstationary behavior as the process dynamics change with time. External factors, such as changes in process feed and production targets, or internal factors such as decline in equipment performance, fouling and catalyst activation, or simply seasonal effects can all contribute to the time varying behavior of processes [25]. This phenomenon is called concept drift in statistical learning, and several algorithms have been developed to handle drifting data, since it is encountered in a diverse set of applications, such as signal processing, spam filtering, financial forecasting, anomaly detection [26]. There is a growing need for designing soft sensors, which can handle changing dynamics, and provide online predictions for process control, monitoring and fault detection purposes, while maintaining high accuracy in a dynamic environment by adapting itself.

Since process data is received in the form of streams, one-shot learning algorithms based on batch modeling techniques are not appropriate, as their assumptions no longer hold in such scenario. In addition, static approach to modeling is often unpractical when predictions are required in real time for online control and monitoring of the process. For that reason, PLS model has been extended with recursive PLS (RPLS), and dynamic PLS [27, 28]. RPLS is an incremental algorithm for updating PLS parameter estimates recursively with incoming data; it defines a forgetting factor for geometrically down weighting the effect of older observations on estimates, in favor of more recent ones. Similarly, online implementations of more recent statistical learning techniques, such as SVM, which have been traditionally used for offline training, have been studied [29]. Exact incremental learning algorithms for SVM have found limited applications, due to their extensive memory requirements. Hence, alternative approaches to memory organization and arithmetic operations had been proposed [30], in addition to different approximations to the exact solution [31]. An online least squares-support vector regression (LS-SVM) model, which adapts to the time changing characteristics of a process by selectively pruning samples, and implementing a varying window size for modeling, was developed for soft sensor design [32].

While the recursive modeling schemes allow for continuous updating and online predictions, they cannot handle abrupt changes in drifting environments as they do not adapt quickly enough. For that reason, moving window (MW) based modeling frameworks have been developed; model parameters are estimated using only a window of the most recent observations, enabling a quicker model adaptation [33]. The window size is not required to be constant, and various adaptive window size adjustment mechanisms have been proposed in the literature [34].

Just-in-time (JIT) learning, also known as lazy learning, or local modeling has been proposed to cope with changing dynamics in industrial processes. Influenced by memory based techniques and local modeling approaches, a local model is built upon query, and the model is discarded once the output is predicted. If the response variable of the query point is measured, then the query point is added to the reference dataset

[35]. Similar to MW modeling, just-in-time learning (JITL) aims to cope with time varying process behavior by building predictive models using only the samples relevant to the current dynamics. MW strategy is based on temporal relevance, whereas in JITL methodology, the relevance is defined in input space. Several relevance metrics have been developed for JITL, e.g. Euclidean distance, angle measurement, and Q and T^2 statistics from PCA [36–38]. JITL is an attractive choice for soft sensor applications as it can handle not only the changing characteristics of data streams with nonstationary behavior, but also cope with nonlinear dynamics of complex industrial processes.

Another way to deal with concept drift is to use an ensemble of models instead of a single one; final prediction output is usually obtained as the weighted average of outputs from multiple predictive models [39]. The ensemble approach is also known to increase generalization capability of single learners [40]. ILLSA method, for instance, is based on an ensemble of RPLS models which are combined with weights proportional to their predictive performances, and individual model parameters and the ensemble weights are updated recursively with each query point [41]. Several adaptive ANN ensembles have been proposed for soft sensor design, updating models via variable forgetting factors [42,43]. JITL may also be used in an ensemble setting. For instance, prediction outputs from multiple JITL models with different similarity metrics were combined to obtain the final output estimate for an industrial batch hydrocracking process [44]. Employing JITL ensemble based on a dual updating scheme for PLS estimates on an industrial batch process yielded predictions superior to those by conventional adaptive approaches [45]. The major drawback of ensemble models is their increased complexity; the model performance is directly influenced by adaptation schemes, such as model inclusion or pruning, updating parameter estimates of individual models and adapting model weights, employed within the ensemble framework [40].

Another open issue in soft sensor design in incorporating expert knowledge about the application into the statistical model, and extracting knowledge from model predictions. Model based design methodologies, developed from mechanistic laws do not achieve desired predictive performance in general; however, an alternative is to opt for

gray box modeling and combine practical information with statistical learning techniques. Some gray box modeling approaches have also been proposed to handle process changes, by incorporating prior knowledge on the general characteristics of a system. For instance, Fuzzy Takagi-Sugeno (FTS) modeling was used in conjunction with PCA and fuzzy c-means clustering for prediction of melt index in a polyethylene manufacturing process [46].

3. STATISTICAL LEARNING METHODS

In the recent years, developments in hardware technology has enabled collection, and storage of large amounts of data. Data can have variety of sources; scientific experiments and observational studies, market trends and various financial indicators, production facilities and industrial plants, and more recently online social networks are all sources from which data is often obtained. The buzzword, "big data", essentially represents this continual enthusiasm for collecting more and more data. The ever growing interest in data has been motivated by both the reduced costs of data collection, and the desire - deep grounded in human nature - for extracting meaning and knowledge from our observations in real life. Following advances in computational technology, people have come up with numerous different ways to analyze and learn from data. In this section, the aim is to provide some theoretical background on the concept of statistical learning, and mathematics of statistical methods within the scope of this thesis.

Broadly speaking, statistical learning methods aim to extract patterns, information, or trends from data. Throughout the years, learning methods have evolved to solve problems in various fields such as statistics, signal processing, data mining and AI research. Machine learning has emerged within the AI and computer science environment, whereas statistical learning has its roots in statistics. These fields often overlap, as both are essentially a set of different approaches to the same underlying problem, even though their starting points or purposes may differ. In machine learning scene, learning is classified into three groups: supervised, unsupervised and reinforcement. In supervised learning, which is also the subject of this thesis, it is assumed that labeled outputs (response variables, or targets) are a function of a set of inputs (or predictors), and it is aimed to learn this function, i.e. mapping, from given inputs to outputs. In unsupervised learning, however, there are no labeled outputs, and the aim is to make inferences from the inputs. A sequence of actions is aimed to be learned in reinforcement learning [21]

Regression is the general learning method in which the outputs of a supervised learning problem are quantitative, and the outputs are generated from a set of inputs. The presumed generative model is formally represented as follows:

$$y = f(x) + \varepsilon \tag{3.1}$$

Here, $f(x)$ is a mapping from input x to output y with $(x, y) \sim p(x, y) = p(y|x)p(x)$, since y is conditioned on x via Equation 3.1. The random error term ε is assumed to be independent of x, and zero mean Gaussian noise with variance σ_ε^2, so $\varepsilon \sim \mathcal{N}(0, \sigma_\varepsilon^2)$. The aim is to learn this mapping using a statistical learning method, i.e. to estimate the best approximate function from a set of infinitely many. Thus, a learning method (or "learner") takes the role of a middleman, whose job can be to extract knowledge for inference, and analysis from observations, or make accurate predictions on unseen data. Let $\mathcal{L}$ denote some learner to be used on the observed data $\mathcal{D}$; in most general sense, $\mathcal{L}$ defines a mapping from $\mathcal{D}$ to our estimate $\hat{f}$:

$$\mathcal{L} : \mathcal{D} \to \hat{f} \tag{3.2}$$

Once we obtain $\hat{f}$, outputs can be estimated from the inputs in the most general form as follows:

$$\hat{y} = \hat{f}(x) \tag{3.3}$$

There exists numerous way to approximate f; different terminologies and conventions have been adopted throughout the years depending on the domain of application, or the research field. In addition, there is no single true learning algorithm to solve this problem of function approximation once and for all. Each learner has its strengths and weaknesses, and its performance can vary with each application. Learning methods

have two main quantifiable properties: bias and variance.

$$bias(\hat{f}(x)) = E\{\hat{f}(x)\} - f(x) \tag{3.4}$$

$$V(\hat{f}(x)) = E\left\{\left[\hat{f}(x) - E\left\{\hat{f}(x)\right\}\right]^2\right\} \tag{3.5}$$

Here, E is the expectation, V is the variance operator. Bias is a measure of how far the predictions of a learner are from the correct values on general; simpler models with little flexibility tend to have high bias. Variance is a direct indication of the stability of predictions to small changes in the data; more complex models usually exhibit higher variance. Complexity, in this context, is proportional to the total number of parameters that needs to be estimated from data when approximating f with a learner. Simple models have smaller number of parameters since they make more assumptions about data and f, thereby restricting both the parameter and the function space. Learning is essentially an ill-posed problem; there are infinitely many solutions of Equation 3.1 for f in the function space. Think of the usual prediction setting; an estimate, $\hat{f}$ is obtained using some learner, $\mathcal{L}$ on training data $\mathcal{D}$. Let (x_0, y_0) denote new unseen observations, i.e., not included in $\mathcal{D}$, then y_0 is predicted as $\hat{f}(x_0)$. In the following derivation, all expectations are taken over the training set $\mathcal{D}$ and all future test points (x_0, y_0), i.e. average bias and variance over the whole predictor space (integrate Equation 3.4 and Equation 3.5 over x) are to be computed, hence the subscripts in E and V operators are dropped: $E \equiv E_{\mathcal{D},(x_0,y_0)}$, and $V \equiv V_{\mathcal{D},(x_0,y_0)}$. Mean squared error (MSE) of predictions can be computed as follows:

$$
\begin{aligned}
\mathrm{MSE}(\hat{y}_0) &= E^2\left\{\left[y_0 - \hat{f}(x_0)\right]^2\right\} = E\{y_0^2\} + E\{\hat{f}^2(x_0)\} - 2E\{y_0\hat{f}(x_0)\} \\
&= V(y_0) + E^2\{y_0\} + V(\hat{f}(x_0)) + E^2\{\hat{f}(x_0)\} - 2E\{y_0\hat{f}(x_0)\} \\
&= V(y_0) + E^2\{y_0\} + V(\hat{f}(x_0)) + E^2\{\hat{f}(x_0)\} - 2E\{\epsilon\}E\{\hat{f}(x_0)\} \\
&\qquad - 2E\{f(x_0)\}E\{\hat{f}(x_0)\} - 2cov(f(x_0)\hat{f}(x_0)) \\
&\vdots \\
&= \underbrace{V(\epsilon)}_{\sigma_\epsilon^2} + \underbrace{E\left\{\left[\hat{f}(x_0) - E\{\hat{f}(x_0)\}\right]^2\right\}}_{V(\hat{f})} + \underbrace{\left[f(x_0) - E\{\hat{f}(x_0)\}\right]^2}_{bias(\hat{f})^2} \qquad (3.6)
\end{aligned}
$$

Here, *cov* stands for covariance. Equation 3.6 is the famous "bias/variance decomposition" of MSE, it epitomizes the dilemma at the heart of statistical learning; we wish to exploit data to obtain general and flexible models, however, we could easily be "overfitting" to noise. On the other hand, we can avoid overfitting by introducing inductive bias into our models, and risking "underfitting". Moreover, in real life we will never be able to compute bias and variance exactly since underlying distributions are not known, and this dilemma more evident especially in situations where data is scarce [6]. Bias/variance trade-off is frequently visualized with variants of Figure 3.1; the left hand side represents simple models with high bias and low variance. As the model complexity increases from left left to right, bias declines and variance increases. The principle of "Occam's Razor" has a great influence on statistics; parsimonious models are preferred to more complex ones, unless there is strong justification for increasing complexity. Parsimonious models are desired not only for their simplicity and ease of interpretation, but also for their good generalization performance. When the ultimate aim is to make accurate predictions on unseen data, generalization is the keyword. Consider the extreme case: if we could quantify every second of real life and obtain all the data in the world, a highly complex model with millions of parameters would be more than welcome as it would be the only correct one. However, our observations are limited and data is scarce; hence it is way too easy to model noise in

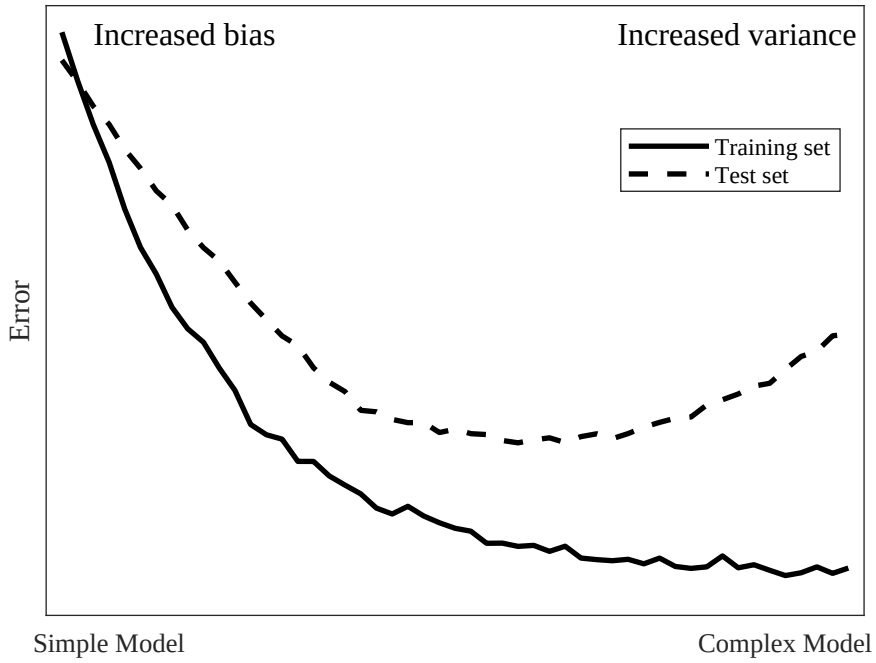

Figure 3.1. Bias/variance trade-off.

addition to patterns in data and overfit. Biased estimates are inherently simpler, as they are obtained by restricting the solution space. They can, however, underfit to the data. Decreasing bias, and allowing a larger hypothesis space would yield more flexible models capable of explaining more complicated relations within data. Major drawback of using more complex models is the increased variance, since a larger number of parameters need to estimated; each estimate brings in additional uncertainty. However in some applications, a biased estimator can be compensated by a greater decrease in variance, resulting in an overall decrease in the prediction error. Thus, it is presumed that a soft spot exists in between high variance and high bias, in which the model is flexible enough to account for the intricacies of the input-output mapping, but biased enough to avoid overfitting, and make stable predictions without modeling noise. We can solve this problem by first finding the best learner for the specific application and its corresponding parameters, which optimize the bias/variance trade-off. This topic will be further elaborated in Section 3.5.

Learners may be classified as linear and nonlinear models. In the context of this thesis, linearity of a model refers to the model parametrization; any model which can be expressed as a linear combination of parameters would belong in this group.

3.1. Linear Models

The simplest approach to estimate the relation between multiple inputs and a single response variable y in Equation 3.1 is to use a linear model. Switching from the random variable notation to matrix notation, the output is defined as:

$$\mathrm{y}_i = \hat{f}(\mathbf{x}_i) + \epsilon_i = \mathbf{x}_i\boldsymbol{\beta} + \epsilon_i \tag{3.7}$$

$$\hat{\mathrm{y}}_i = \mathbf{x}_i\boldsymbol{\beta} \tag{3.8}$$

Here, response and predictor variables are denoted as $\mathrm{y}_i \in \mathbb{R}$ and $\mathbf{x}_i \in \mathbb{R}^p, i = 1, 2, \ldots N$, hence, number of predictors and observations are p and N, respectively. $\hat{\mathrm{y}}_i$ is estimation (fitted value) of i^{th} output, and ϵ_i is assumed to be residuals, i.e. estimates of the random error:

$$cov(\epsilon_i, \epsilon_j) = \sigma_\epsilon^2 \delta_{ij} \tag{3.9}$$

$$cov(\epsilon_i, \mathbf{x}_i) = 0 \tag{3.10}$$

$$E\{\epsilon_i\} = 0, \quad \epsilon \sim \mathcal{N}(0, \sigma_\epsilon^2) \tag{3.11}$$

Here, δ is the Kronecker delta function. It should be noted that the above assumptions on ϵ may not hold, i.e. $bias(\hat{f}(x)) \neq 0$, since $\hat{f}$ is an estimate and $\hat{f} \neq f$. The whole training dataset D is defined from individual pairs of observations, $D \coloneqq \{(\mathbf{x}_i, \mathrm{y}_i)\}_{i=1}^N$. Thus, learning a linear mapping is equivalent to estimating the model coefficients. Without loss of generality, all data are mean-centered, and the traditional least squares (LS) solution to this problem are the coefficients which

minimize the sum of squared errors:

$$\mathrm{SS_R} = \sum_{i=1}^{N} (\mathrm{y}_i - \mathbf{x}_i\boldsymbol{\beta})^2 \tag{3.12}$$

Equation 3.12 represents the sum of squared errors, and is also called the least squares loss function. Since minimization is achieved on $\mathrm{SS_{RR}}$ without any constraints, LS yields unbiased estimates of $\boldsymbol{\beta}$ in Equation 3.8. Coefficients vector $\boldsymbol{\beta} \in \mathbb{R}^{\mathrm{p}}$ is estimated using an $N \times p$ training data (or design) matrix comprised of input variables $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \ldots \mathbf{x}_N]^{\mathrm{T}}$ and the corresponding vector of response values, $\mathbf{y}$. If the covariance of the design matrix ($\mathbf{X}^{\mathrm{T}}\mathbf{X}$) is non-singular, then there is a unique solution to this optimization problem, since the objective function is quadratic. The solution is given by:

$$\hat{\boldsymbol{\beta}}_{\mathrm{LS}} = (\mathbf{X}^{\mathrm{T}}\mathbf{X})^{-1}\mathbf{X}^{\mathrm{T}}\mathbf{y} \tag{3.13}$$

If the covariance of the design matrix is rank deficient, its inverse does not exist. This results from multicollinearity (or simply collinearity) among predictor variables; there exists some variable which can be predicted using a linear combination of the other variables. Even if predictors are not exactly collinear, high correlation between predictors yield LS estimates with high variance, hence the estimates can become unstable. One approach to handle collinearity is to work in a latent variable space with smaller dimensions than the original space spanned by the predictor variables. For that purpose, PLS extracts orthogonal components, $\mathbf{t}_{\mathrm{a}}$, and $\mathbf{u}_{\mathrm{a}}$ (for $\mathrm{a} = 1, 2, \ldots L$, where L is the number of components), which yield the maximum covariance between the response and the predictor variables [47]. PLS was first introduced by Wold in [48], and Wold and his son continued their work on PLS, which was initially used in the field of chemometrics [49], and consequently it has found its audience in chemical engineering. Starting from 1990s, PLS has been frequently used for prediction problems in general, and soft sensor design [50, 51]. Kano and Fujiwara, in their review, define

1. Input: $\mathbf{X}$, $\mathbf{y}$, and L, maximum number of components

2. Require: $L < rank(\mathbf{X})$; $\mathbf{X}$ and $\mathbf{y}$ are normalized

3. Initialize: $i = 1$; set $\mathbf{u}_0 \leftarrow \mathbf{y}$

while $i \leq$ L **do**

i. $\mathbf{v}_i \leftarrow \mathbf{X}^{\mathrm{T}}\mathbf{u}_{i-1}/(\mathbf{u}_{i-1}^{\mathrm{T}}\mathbf{u}_{i-1})$

ii. Scale $\mathbf{v}_i$ to be length one

iii. $\mathbf{t}_i \leftarrow \mathbf{X}\mathbf{v}_i$

iv. $\mathbf{p}_i \leftarrow \mathbf{t}_i^{\mathrm{T}}\mathbf{X}/(\mathbf{t}_i^{\mathrm{T}}\mathbf{t}_i)$

v. $\mathbf{t}_i \leftarrow \mathbf{t}_i/\|\mathbf{p}_i\|$; Scale $\mathbf{p}_i$ to be length one

vi. $\mathrm{b}_i \leftarrow \mathbf{y}^{\mathrm{T}}\mathbf{t}_i/(\mathbf{t}_i^{\mathrm{T}}\mathbf{t}_i)$

vii. $\mathbf{u}_i \leftarrow \mathbf{y}^{\mathrm{T}}\mathbf{b}_i/(\mathbf{b}_i^{\mathrm{T}}\mathbf{b}_i)$

viii. $\mathbf{X} \leftarrow \mathbf{X} - \mathbf{t}_i^{\mathrm{T}}\mathbf{p}_i$ and $\mathbf{y} \leftarrow \mathbf{y} - \mathrm{b}_i\mathbf{t}_i$

end while

return $\mathbf{P}$, $\mathbf{T}$, $\mathbf{V}$ $\mathbf{B}$ and $\mathbf{U}$

Figure 3.2. NIPALS Algorithm for PLS.

the history of soft sensor design in two parts: before and after PLS [22]. According to their surveys, PLS-based methods for soft sensor design still retain their popularity, even though many alternatives have emerged since 1990s.

Using NIPALS algorithm, which was first introduced by Fisher in 1920s in [52], a linear inner relation between X scores, $\mathbf{t}_{\mathrm{a}}$, the eigenvector corresponding to the maximum eigenvalue of $\mathbf{X}_{\mathrm{a}}^{\mathrm{T}}\mathbf{X}\mathbf{y}_{\mathrm{a}}^{\mathrm{T}}\mathbf{y}$, and the residual $\mathbf{y}$ ($\mathbf{y_a}$), from $\mathbf{y_{a+1}} \leftarrow \mathbf{y_a} - \mathrm{b_a}\mathbf{t}_{\mathrm{a}}$, is used to compute b_{a} [49]. While predictors are usually scaled to unit-variance, PLS can be employed with different scaling schemes [50]. Basic NIPALS algorithm is presented in Figure 3.2, where $\mathbf{T} = [\mathbf{t}_1, \mathbf{t}_2, ..., \mathbf{t}_L]$, $\mathbf{V} = [\mathbf{v}_1, \mathrm{v}_2, ..., \mathbf{v}_L]$, $\mathbf{P} = [\mathbf{p}_1, \mathbf{p}_2, ..., \mathbf{p}_L]$, $\mathbf{B} = diag[\mathrm{b}_1, \mathrm{b}_2, ..., \mathrm{b}_L]$ and $\mathbf{U} = [\mathbf{u}_1, \mathbf{u}_2, ..., \mathbf{u}_L]$

Despite being in the form of a linear model, PLS can be extended to capture nonlinearities among data. For instance, the inner relation can be replaced with a

quadratic function [53].

$$\mathbf{y}_{\mathrm{a}} = \mathrm{b}_{\mathrm{a},0} + \mathrm{b}_{\mathrm{a},1}\mathbf{t}_{\mathrm{a}} + \mathrm{b}_{\mathrm{a},2}\mathbf{t}_{\mathrm{a}}^2 \tag{3.14}$$

In kernel PLS, on the other hand, a nonlinear mapping from the original data space to the feature space using a (usually polynomial or Gaussian) kernel function is employed, while the inner relation between the components remains linear [54].

PLS has proven to be a powerful tool for dealing with multicollinearity in data, while maintaining accurate predictive performance. Unlike PCR, PLS can extract directions of large variance in the input space, which can best explain the variance in the outputs. PLS has also been suggested for identification of FIR models, and it was shown to give good results in both time and frequency domain [55].

Another way of dealing with high variance in parameter estimates is to place constraints on their magnitude. Ridge regression (RR) is one of the oldest biased estimation methods using a single parameter regularization [56]. Estimates of RR parameters ($\hat{\boldsymbol{\beta}}_{\mathrm{RR}}$) can be obtained via minimizing the sum of squared residuals subject to the constraint that $\boldsymbol{\beta}^{\mathbf{T}}\boldsymbol{\beta} \leq \mathrm{R}^2$, which can be equivalently stated as follows:

$$\hat{\boldsymbol{\beta}}_{\mathrm{RR}} = \arg\min_{\boldsymbol{\beta}} \left\{ \frac{1}{2}\sum_{i=1}^{N} \left(\mathrm{y}_i - \beta_0 - \sum_{j=1}^{\mathrm{p}} \mathrm{x}_{i,j}\beta_j \right)^2 + \lambda \sum_{j=1}^{\mathrm{p}} \beta_j^2 \right\} \tag{3.15}$$

By implementing an L_2 penalty, RR shrinks regression coefficients, and introduces bias [6]. Lasso, on the other hand employs L_1 penalty, it shrinks some of the regressions coefficients, and eliminates some completely [57]. Thus, it can be thought of as middle of the road between regularization and subset selection methods. The lasso estimates

are obtained using the following optimization equation:

$$\hat{\boldsymbol{\beta}}_{\text{Lasso}} = \arg\min_{\boldsymbol{\beta}} \left\{ \frac{1}{2} \sum_{i=1}^{N} \left(\mathrm{y}_i - \beta_0 - \sum_{j=1}^{\mathrm{p}} \mathrm{x}_{i,j} \beta_j \right)^2 + \lambda \sum_{j=1}^{\mathrm{p}} |\beta_j| \right\} \tag{3.16}$$

Lasso estimates may be efficiently determined using the least angle regression (LARS) path, in which regressors are added to the model in a stagewise manner as the penalty term is relaxed [58]. For both RR and Lasso, λ parameter needs to be defined to perform the optimization. In real-life applications, λ would not be known beforehand, and cross validation is usually used to tune this parameter, so that the obtained RR and Lasso models yield the best predictive performance.

From a Bayesian point of view, a prior on regression coefficients of PLS gives higher weights on eigendirections of predictors with larger eigenvalues, while ridge regression shows no particular preference for any direction [59]. Regularization (or biased regression) methods like PLS, RR and Lasso, in one interpretation, can be viewed as a means to reduce the function (or hypothesis) space, thereby decreasing the number of possible solutions one can achieve.

Relevance vector machine (RVM) has been introduced in the field of machine learning as a sparse solution for both regression and classification tasks. It is based on a probabilistic Bayesian framework, and it uses the same functional form as SVM, so this estimator was named relevance vector machine [60]. RVM is based on the linear form of regression problem, stated in Equation 3.8, but rewritten in a more general form:

$$\hat{f}(\mathbf{x}; \mathbf{w}) = \sum_{i=1}^{M} w_i \phi_i(\mathbf{x}) = \mathbf{w}^{\mathrm{T}} \boldsymbol{\Phi}(\mathbf{x}) \tag{3.17}$$

The output is the sum of M basis functions, $\mathbf{\Phi} = [\phi_1(\mathbf{x}), \phi_2(\mathbf{x}), ..., \phi_M(\mathbf{x})]^{\mathrm{T}}$, weighted with $\mathbf{w}$. In SVM, M is taken to be equal to N, the size of training set. Basis functions can be chosen either linear or nonlinear, and the regression problem is the same as that for linear models; one needs to estimate the "best" $\mathbf{w}$. During training, most of the weights are set to zero, and only the "relevant" basis functions are kept. SVM utilizes a kernel, $K(\mathbf{x}, \mathbf{x}_i)$ as the basis function, it is known to generalize fairly well and avoid overfitting as the final model is made up of a subset of the basis functions. Compared to SVM, RVM retains fewer parameters, and may have improved generalization capability [60]; some major drawbacks of SVM are:

- SVM is not set on a probabilistic framework; it yields only point estimates for regression, and hard rules for classification. In many applications, in addition to making predictions, it is also desired to estimate $p(y|x)$. The main advantage of a probabilistic approach is that one can estimate the "uncertainty" of predictions. Confidence intervals can be estimated after predictions with several bootstrap, or cross validation methods [61, 62]. RVM eliminates the need for this additional estimation step, it automatically provides a measure for prediction uncertainty due to its fully probabilistic framework.
- RVM is a more flexible model since (i) it does not require the basis functions to satisfy Mercer's condition, and (ii) any past experience, or knowledge specific to the problem at hand can be incorporated into the predictive model through prior probabilities.

Regression problem in Equation 3.1 is formulated as a probabilistic model:

$$\begin{aligned} \mathrm{y}_i &= \hat{f}(\mathbf{x}_i; \mathbf{w}) + \epsilon_i \\ \epsilon &\sim \mathcal{N}(0, \sigma^2) \\ y|x &\sim \mathcal{N}(y|\hat{f}(x), \sigma^2) \end{aligned} \tag{3.18}$$

Assuming that random noise ϵ_i is independent, the output y_i is Gaussian distributed with mean $\hat{f}(\mathbf{x}_i)$ and variance σ^2. For i.i.d. samples, y_i:

$$p(\mathbf{y}|\mathbf{w},\sigma^2) = (2\pi\sigma^2)^{(-N/2)}\exp\left\{-\frac{1}{2\sigma^2}\left\|\mathbf{y}-\mathbf{\Phi}\mathbf{w}\right\|^2\right\} \tag{3.19}$$

Here, $\mathbf{w} = [w_0, w_1, ...w_N]^{\mathrm{T}}$ and $\mathbf{\Phi} = [\phi(\mathbf{x}_1), \phi(\mathbf{x}_2), ..., \phi(\mathbf{x}_N)]^{\mathrm{T}}$. $\mathbf{\Phi}$ is the $N\times(N+1)$ design matrix with the general form of $\phi(\mathbf{x}_i) = [1, K(\mathrm{x}_i, \mathrm{x}_1), K(\mathrm{x}_i, \mathrm{x}_2), ..., K(\mathrm{x}_i, \mathrm{x}_N)]^{\mathrm{T}}$ as the basis functions. Note that conditioning on input vectors $\mathbf{x}_i$ is forgone, and the original notation used by Tipping [60] is employed for the parameters, so as to emphasize the differences from other linear models, described in previous sections.

Consequently, a prior distribution is defined for the parameters $\mathbf{w}$. This explicit constraint on the parameters has a role in learning similar to margins in SVM; both yield a penalty term in the error function, and can help fight against overfitting. However, in RVM this comes naturally within the Bayesian framework as more complex solutions, which are more likely to negatively affect the generalization capability, are preferred less and this is reflected in their prior distribution. This preference can be formulated using a zero-mean Gaussian prior on $\mathbf{w}$:

$$p(\mathbf{w}|\boldsymbol{\alpha}) = \prod_{i=0}^{N}\mathcal{N}(w_i|0,\alpha_i^{-1}) \tag{3.20}$$

Where, $\boldsymbol{\alpha}$ comprises of $N+1$ "hyperparameters" for each weight. The probabilistic model ends with the definition of hyperpriors' (or "hierarchical priors") over $\boldsymbol{\alpha}$ and the noise variance, σ^2:

$$p(\boldsymbol{\alpha}) = \prod_{i=0}^{N}\mathrm{Gamma}(\alpha_i|a,b) \tag{3.21}$$

$$p(\beta) = \mathrm{Gamma}(\beta|c,d) \tag{3.22}$$

Here, β is simply the inverse of σ^2, and it's often called the "precision", while $\boldsymbol{\alpha}$ and σ^2 are "scale" parameters in hierarchical models [63]. The following parametrization of Gamma distribution is used:

$$\text{Gamma}(\alpha_i|a,b) = \Gamma(a)^{-1} b^a \alpha_i^{a-1} e^{-b\alpha_i} \tag{3.23}$$

Where, $\Gamma(a) = \int_0^\infty t^{a-1} e^{-t} dt$. Uniform hyperpriors can be used by setting $a = b = c = d = 0$ in Equation 3.21 and 3.22.

Bayesian approach to prediction reduces the learning problem to explicit definitions of underlying assumptions through probabilistic models. For instance, the graphical model of RVM is drawn in Figure 3.3, in which the plate notation is adopted to represent N observations.

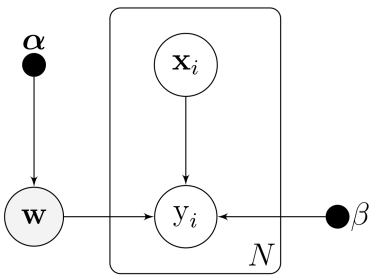

Figure 3.3. Graphical model of RVM.

Inference in RVM is made using the Bayes' rule. Denoting the measured value of a test point with the response variable y^*: Sparsity is achieved as prior distributions reflect our general preference of simple models over complex ones. Occam's razor, is exhibited naturally within the Bayesian framework [64]. "Automatic relevance determination" employs Bayesian inference in neural network models along the same lines; hyperparameters are used constrain NN weights, and only the relevant ones are retained. Probabilistic model introduces additional parameters which are then integrated out during training, hence they do not pose any problems in terms of complexity [65].

$$p(\mathbf{w}, \boldsymbol{\alpha}, \sigma^2|\mathbf{y}) = \frac{p(\mathbf{y}|\mathbf{w}, \boldsymbol{\alpha}, \sigma^2)p(\mathbf{w}, \boldsymbol{\alpha}, \sigma^2)}{p(\mathbf{y})} \tag{3.24}$$

$$p(\mathrm{y}^*|\mathbf{y}) = \int p(\mathrm{y}^*|\mathbf{w}, \boldsymbol{\alpha}, \sigma^2)p(\mathbf{w}, \boldsymbol{\alpha}, \sigma^2|\mathbf{y})d\mathbf{w}d\boldsymbol{\alpha}d\sigma^2 \tag{3.25}$$

In the above expressions, neither the posterior distribution of the weights, nor the posterior predictive distribution can be computed analytically, since the normalization constant in Equation 3.24 can't be determined analytically, so these distributions should be approximated instead. First, Equation 3.24 is decomposed:

$$p(\mathbf{w}, \boldsymbol{\alpha}, \sigma^2|\mathbf{y}) = p(\mathbf{w}|\mathbf{y}, \boldsymbol{\alpha}, \sigma^2)p(\boldsymbol{\alpha}, \sigma^2|\mathbf{y}) \tag{3.26}$$

The first term on the right hand side of Equation 3.26 is the posterior distribution of weights, and it does have an analytic solution; since its normalizing constant is a convolution of two Gaussians, but the resulting distribution is also Gaussian due to the linearity of Gaussians.

$$p(\mathbf{w}|\mathbf{y}, \boldsymbol{\alpha}, \sigma^2) = \frac{p(\mathbf{y}|\mathbf{w}, \sigma^2)p(\mathbf{w}|\boldsymbol{\alpha})}{p(\mathbf{y}|\boldsymbol{\alpha}, \sigma^2)} \tag{3.27}$$

$$= (2\pi)^{-(N+1)/2}|\boldsymbol{\Sigma}|^{-1/2}\exp\left\{-\frac{1}{2}(\mathbf{w} - \boldsymbol{\mu})^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}(\mathbf{w} - \boldsymbol{\mu})\right\} \tag{3.28}$$

$$\boldsymbol{\Sigma} = (\sigma^{-2}\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{\Phi} + \mathbf{A})^{-1} \tag{3.29}$$

$$\boldsymbol{\mu} = \sigma^{-2}\boldsymbol{\Sigma}\boldsymbol{\Phi}^{\mathrm{T}}\mathbf{y} \tag{3.30}$$

Here, $\mathbf{A} = diag(\alpha_0, \alpha_1, ..., \alpha_N)$. And the second term in Equation 3.26 is approximated by replacing the hyperparameter posterior distribution with a delta function at its mode:

$$\int p(y^*|\boldsymbol{\alpha}, \sigma^2)\delta(\boldsymbol{\alpha}_{\mathrm{MP}}, \sigma^2_{\mathrm{MP}})d\boldsymbol{\alpha}d\sigma^2 \approx \int p(y^*|\boldsymbol{\alpha}, \sigma^2)p(\boldsymbol{\alpha}, \sigma^2|\mathbf{y})d\boldsymbol{\alpha}d\sigma^2 \tag{3.31}$$

$$p(\boldsymbol{\alpha}, \sigma^2|\mathbf{y}) \propto p(\mathbf{y}|\boldsymbol{\alpha}, \sigma^2)p(\boldsymbol{\alpha})p(\sigma^2) \tag{3.32}$$

MP subscripts were used to denoted maximum aposteriori estimates.

Tipping asserts the validity of this approximation using his experiments [60], since it produces point estimates identical to those from the usual variational approach, called expectation maximization, at a much lower computational cost [66]. Using uniform hyperpriors further simplifies the calculations, and only the first term in Equation 3.32 needs to be maximized to obtain the MP estimates of the hyperparameters.

$$p(y^*|\boldsymbol{\alpha}, \sigma^2) = \int p(y^*|\mathbf{w}, \sigma^2) p(\mathbf{w}|\boldsymbol{\alpha}) d\mathbf{w} \tag{3.33}$$

$$= (2\pi)^{-N/2} |\sigma^2 \mathbf{I} + \boldsymbol{\Phi} \mathbf{A}^{-1} \boldsymbol{\Phi}^{\mathrm{T}}|^{-1/2} \exp\left\{ -\frac{1}{2} \mathbf{y}^{\mathrm{T}} (\sigma^2 \mathbf{I} + \boldsymbol{\Phi} \mathbf{A}^{-1} \boldsymbol{\Phi}^{\mathrm{T}})^{-1} \mathbf{y} \right\} \tag{3.34}$$

An iterative procedure is followed to maximize Equation 3.34 with respect to $\boldsymbol{\alpha}$ and σ^2; by differentiating Equation 3.34 and setting to zero, the optimum hyperparameters can be obtained by iteratively using the following equations:

$$\alpha_i^{new} = \frac{\gamma_i}{\mu_i^2} \tag{3.35}$$

$$(\sigma^2)^{new} = \frac{\|\mathbf{y} - \boldsymbol{\Phi}\boldsymbol{\mu}\|^2}{N - \sum_i \gamma_i} \tag{3.36}$$

Where, $\gamma_i \equiv 1 - \alpha_i \Sigma_{ii}$ are called the relevance factors, and Σ_{ii} is the i$^{\text{th}}$ diagonal of the posterior covariance matrix of weights in Equation 3.29.

A RVM model is learned as Equation 3.35 and Equation 3.36 are iterated simultaneously with Equations 3.28, 3.29 and 3.30 to update the posterior distribution of the weights, until convergence. During training, as some of the hyperparameters (α_i) go to infinity, the posterior, $p(w_i|\mathbf{y}, \boldsymbol{\alpha}, \sigma^2)$ becomes highly peaked at zero for the corresponding weights, all of which are then removed to achieve sparse solutions for $\mathbf{w}$.

Finally, the predictive distribution of a test point can be computed:

$$p(\mathrm{y}^*|\mathbf{y}, \boldsymbol{\alpha}_{MP}, \sigma^2_{MP}) = \int p(\mathrm{y}^*|\mathbf{w}, \sigma^2_{MP}) p(\mathbf{w}|\mathbf{y}, \boldsymbol{\alpha}_{MP}, \sigma^2_{MP}) d\mathbf{w} \tag{3.37}$$

$$= \mathcal{N}(\mathbf{y}^*|\hat{\mathrm{y}}^*, \hat{\sigma}^2_{\mathrm{y}^*}) \tag{3.38}$$

$$\hat{\mathrm{y}}^* = \boldsymbol{\mu}^{\mathrm{T}} \phi(\mathbf{x}^*) \tag{3.39}$$

$$\hat{\sigma}^2_{\mathrm{y}^*} = \sigma^2_{MP} + \phi(\mathbf{x}^*)^{\mathrm{T}} \boldsymbol{\Sigma} \phi(\mathbf{x}^*) \tag{3.40}$$

Here the predictive distribution mean ($\hat{\mathrm{y}}^*$) is the weighted sum of basis functions with the posterior mean of the weights. Perhaps unsurprisingly, variance of the predictive distribution ($\hat{\sigma}^2_{\mathrm{y}^*}$) comprises two terms: estimate of noise variance, and variance arising from the uncertainty in the estimates of weights. Therefore, one can obtain point estimates for the prediction computing the posterior mean from Equation 3.39, and use prediction variance to assess the uncertainty in predictions. Note that predictive variance of response variable in Equation 3.40 is also equivalent to prediction error variance ($V(\mathrm{y}^* - \hat{\mathrm{y}}^*)$) estimate in a frequentist framework [60].

RVM offers great advantages over conventional statistical learning methods, such as PCR, PLS, and SVR, for soft sensor applications. Due to its sparsity, not only it generalizes fairly well on unseen data, but it also helps combat collinearity among predictor variables by retaining only the relevant ones. Both properties make RVM a viable alternative in soft sensor design for industrial processes; it can handle the data rich, information poor nature of soft sensor inputs by removing redundancy, and show stable performance in the presence of multicollinearity. Moreover, the model parameters are optimized iteratively during training, hence there is no need for additional crossvalidatory parameter tuning step. Its good generalization property also makes it an aspiring contender for predictive problems in general, and the uncertainty in predictions can easily be assessed as prediction variance is automatically obtained during training within the Bayesian framework.

It is rather surprising to find out that relatively few applications of RVM exist in soft sensor design literature, unlike its Bayesian counterpart Gaussian process regression (GPR) [67,68]. In of the few studies, which used RVM on chemical processes, RVM has been shown to outperform PLS and LS-SVM models on the benchmark Tenesse Eastmen process [69]. In the same study, the authors compared prediction variance obtained from the RVM model with T^2 statistic of PCA, and the prediction variance appeared as a feasible alternative for process monitoring purposes, as it could identify when process went outside its normal operating conditions.

3.2. Nonlinear Models

Linear models are easy to parameterize, and - most of the time - not difficult to estimate; but they might be too biased for the application for which they are used. Hence, one may need to use nonlinear models to reduce bias to acceptable levels. In this section, we elaborate on nonlinear statistical learning methods employed in this thesis; first artificial neural networks are discussed, followed by instance based learning, a local approach to approximating functions. It should be noted that, other nonlinear modeling methods, such as SVR, regression trees, radial basis networks and neuro-fuzzy inference systems are also frequently used in the field of statistical learning [6].

3.2.1. Artificial Neural Networks

Inspired by the human brain, ANNs have received a great deal of attention in various fields, and proven to be successful at modeling complex non-linear systems. Chemical engineering processes are known to exhibit highly nonlinear characteristics, hence ANNs are among the most popular learning algorithms utilized in soft sensor design [70]. Historical process datasets usually spans days or even years of production, thus can end up being enormous, making ANNs convenient for modeling purposes [71, 72].

A neural network (NN) is a collection of artificial neurons (perceptrons), which receive and sum the weighted input signals, and determine outputs via threshold transfer functions, connected in a network fashion [21]. While single layered perceptron models, comprised of a single layer for the input and the output, can be used only to discriminate linearly separable data; multilayer perceptrons, which consist of at least one hidden layer, can be used to approximate any sigmoidal activation function. Thus, they are known as the "universal function approximators" [73]. The units in the hidden layers of an ANN contain activation functions, transforming the weighted some of the inputs received. Many activation functions have been proposed so far; linear, piecewise linear, hyperbolic tangent and sigmoid functions, and more recently, rectified linear units (ReLu) are common choices [74]. Retaining the basic properties of the perceptron, different architectures, specifically designed for the domain of application, have been proposed for ANNs throughout the years. Mathematically speaking, input layer units receive the data matrix, $\mathbf{X}$, passes the weighted sum of inputs through the sigmoid function, g_1. For the i^{th} observation ($i = 1, 2, ...N$), this can be formulated as follows:

$$z_{ik} = g_1\left(\sum_{j=1}^{p} w_{ijk}^{'} \mathrm{x}_{ij} + \theta_{ik}^{'}\right) \tag{3.41}$$

Here, x_{ij} is the j^{th} predictor of i^{th} observation, and z_{ik} denotes the k^{th} input to the consequent layer, in which the output is estimated by:

$$\hat{\mathrm{y}}_i = g_2\left(\sum_{k=1}^{K} w_{ijk} z_{ik} + \theta\right) \tag{3.42}$$

Here, K is the number of neurons in the hidden layer, while θ, and $w_{i,j}$ are the bias and weights, respectively. g_2 is a linear function, and the final output is the linear combination of individual outputs from K hidden units. Estimation error is the objective function aimed to minimize in an ANN; this is achieved through gradient

descent, in which the derivative of the objective function w.r.t. the weights need to be calculated. Backpropagation algorithm is used to efficiently calculate these derivatives; Equation 3.41 and Equation 3.42 define the forward pass in the algorithm. Current estimation error is fed back to each layer in the backward pass, and the network parameters are updated [75]. As with any algorithm built to minimize fitting error on the training set, ANNs suffer from the problem of overfitting to data. There are many approaches addressing this problem; early stopping, regularization and dropout have all been proven to improve generalization in ANNs [76]. In early stopping, a portion of the training data is reserved for validation, and the model error on this validation portion is computed at regular intervals during training. The training error gradually decreases until the end, whereas the validation error starts to increase beyond a certain point. At this point, it is assumed that the network starts to overfit to data and training is ceased [77].

Collinearity, which poses a serious problem to linear models, can also deteriorate the performance of ANNs, even though ANNs are free of ill conditioning. Redundancy introduces additional noise into the learning algorithm, resulting in highly unstable ANNs with increased prediction variance [8]. However, the usual craving for identifying relevant variables while minimizing redundancy prevails. Feature selection in ANNs does not only improve predictive accuracy, but it can also dramatically reduce training time, computational complexity and fight off against the curse of dimensionality. In addition, there has been a growing interest in interpreting ANNs, and learning from the models for knowledge discovery and data mining purposes [78].

Even though linear models are often shown to work well enough for soft sensor design, ANNs have also found many applications following its resurgence in the field of machine learning. Similar to the nonlinear PLS model with quadratic inner relations between latent variables, Qin and McAvoy have developed a neural net PLS (NNPLS), in which the inner relation is modeled using a MLP [28]. Feedforward networks, with a single hidden layer are usually preferred as they can approximate moderate to high nonlinearity in chemical processes well enough [79]. Other ANN architectures have

found only limited use in the literature; radial basis function networks were used to model a membrane separation process [80], stacked auto-encoders were proposed for variable selection in soft sensor models, and were shown to perform well on real industrial data [81]. In the last few years, deep learning, with its new found popularity in other domains, has been considered by some for soft sensor design. Shang *et al.* have asserted the advantage of using deeper structures for modeling complex chemical processes, and showed that the suggested deep neural network model had outperformed SVM on real data from a crude distillation unit [82].

3.2.2. Instance Based Learning

In most learning problems, the common practice is to train a single model on the entire dataset; presuming that model parameters are applicable to the whole domain. It should be noted that global models can still be used in an adaptive frame, updating the parameters with incoming data, as in recursive least squares. This way, data can be discarded, while keeping the model parameters. This global approach to learning, also known as distributed learning, may be useful for inferring general rules and abstractions, but may yield inferior performance for prediction, due to the high bias introduced by the global model.

Alternatively, in instance based learning, also known as lazy learning or just-in-time learning (JITL), a model is obtained only upon query; once the inputs are processed, the model is discarded and the observations are kept. This approach can have large storage requirements, so the name, memory-based learning, is also frequently used, and, as a natural extension, lazy learning algorithms employ local models. Local modeling, a fairly old idea, has its roots in nonparametric statistics; the predictive model, $\hat{f}$, does not have a fixed, predetermined form and is defined by the specific information acquired from data. The main philosophy behind local approach to learning is that the query point is best represented by a subset of the training set, and not all the observations. This subset, also called the neighborhood, is presumed to comprise samples most relevant to the query point.

In lazy learning there is no need to make global assumptions about the underlying distribution of data. Divide and conquer, is a recurring theme in local modeling; difficult problems can become easier to solve by breaking them down into smaller problems. JITL is particularly useful for approximating highly complex nonlinear input-output relationships with locally linear models. A major drawback of local learning is that the definition of local region itself starts to break down in high dimensions; as the number of input dimensions increase, observations become more spread out in the input-space hence for a constant number of nearest neighbors, a much bigger hypercube is occupied when p is larger. The infamous curse of dimensionality adversely effects local models, more than the global ones [6].

This framework for learning leads to the first question asked by the lazy learner: how should one define this relevance? The second problem that needs to be addressed is the degree of relevancy required for training observations to enter this subset; equivalently the number of observations contained in each subset. These observations are often called nearest neighbors. Instances within the subset can be weighted according their relevance using a weighting function, to give higher weights to relevant samples, while the effect of other samples on the model is diminished. This weighting function, or kernel function, can be determined from some predefined distance metric.

$$w_i(d_{ij}) = e^{-d_{ij}^2/\sigma^2} \tag{3.43}$$

Here w_i is the weight of the i^{th} instance in the model for predicting the j^{th} instance, and d_{ij} is distance between those two samples in some shape or form. The usual practice is to normalize the weights so they sum up to one. The kernel width is shown with σ, and it determines how fast the weights decay from the highest value to the lowest. Figure 3.4 shows the effect of kernel width on the spread of the weights.

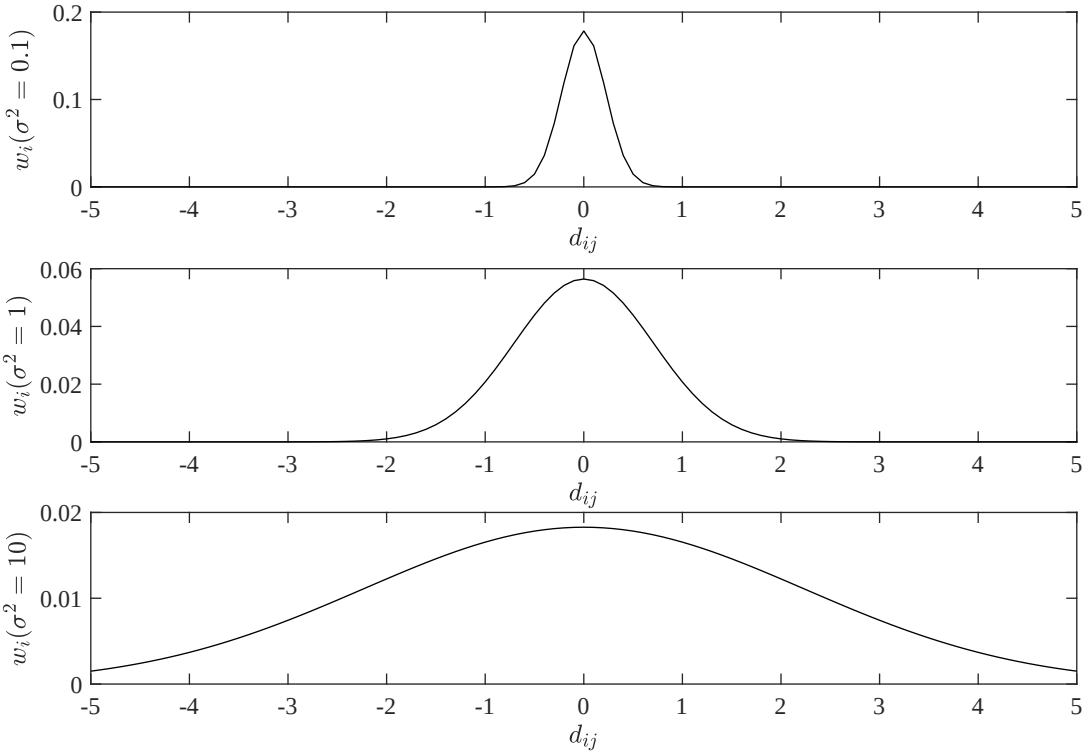

Figure 3.4. Effect of kernel width on the kernel function.

The most commonly used metric to measure instance relevancy is based on the Euclidean distance in the **X**-space.

$$d_e(\mathbf{x}_i, \mathbf{x}_j) = \sqrt{\sum_{k=1}^{p} (\mathrm{x}_{ik} - \mathrm{x}_{jk})} = \| \mathbf{x}_i - \mathbf{x}_j \|^2 \tag{3.44}$$

Locally weighted regression (LWR), initially proposed as a method for smoothing scatterplots, is among the oldest examples of local models, and it uses a tricubic weighting function [83].

$$w_i(d_{ij}) = (1 - |d_{ij}|^3)^3 \tag{3.45}$$

A weighted LS (WLS) model is built in the neighborhood of the query point, let $\mathbf{W} = diag(w_i)$ for $i = 1, 2, ...NS$ where NS is the neighborhood size, Equation 3.13 is modified to obtain the WLS estimates as follows:

$$\hat{\boldsymbol{\beta}}_{\text{WLS}} = (\mathbf{X}^{\text{T}}\mathbf{W}\mathbf{X})^{-1}\mathbf{X}^{\text{T}}\mathbf{W}\mathbf{y} \tag{3.46}$$

Bias/variance dilemma plays an important role in local modeling as well. By its nature local modeling has high variance hence the order (or the complexity) of the local model is usually maintained low. In addition, more assumptions can be made about the underlying relations in data when the model is fit to a small region, and the assumptions need not hold over the entire domain of application.

Similar to WLS, a locally weighted version of PLS (LWPLS) has also been proposed for predicting the active pharmaceutical ingredient (API) content in [84]. Weights are incorporated into the PLS model both when the input and output are scaled, and when the loadings are being calculated in the NIPALS algorithm.

The choice of similarity metric, equivalently the definition of local neighborhood, can be critical in the performance of JITL model. In the most strict sense, Euclidean distance is valid in the idealized case where inputs are Gaussian distributed, independent and have equal variance. Equal variance is achieved through normalization of inputs, however Euclidean distance is known to be quite sensitive to multicollinear predictors, a huge problem in data from industrial processes, and noise. For that reason, in addition to Euclidean distance, and its less assuming counterpart Mahalanobis distance, several techniques for measuring instance similarity have been proposed [85]. Angle measurement has been used in conjunction with the Euclidean distance in [37]. This way, both the direction of change in the in the input space, as well as their spatial positions of observations are taken into account. In correlation-based JIT (CoJIT) modeling, the distance metric is a weighted average of Q and T^2 statistics from PCA; Q statistic is the distance between the observation and the subspace spanned by the

principle components, indicating how well the observation is represented by the principle component (PC) space, and T^2 statistic is a measure of the variation in the PC subspace [38].

More sophisticated metrics for proximity have also been suggested in the literature. A similarity measure based on Gaussian mixture models (GMMs) was proposed to handle non-Gaussian data by approximating it with multiple local Gaussians [86]. The authors have exploited the divide and conquer strategy for the definition of similarity, the proposed metric was shown to perform better than Euclidean and Mahalanobis distance with LWPLS on Debutanizer column dataset. Again for non-Gaussian data, Xie and his collegues have suggested a similarity measure based on a weighted combination of support vector data description (SVDD), and Q statistic from PCA in [87]. The proposed similarity metric was implemented within the same modeling framework of CoJIT, and it had outperformed CoJIT method on the Sulfur recovery unit dataset in their work. An elastic net model was incorporated to determine relevant samples, and used locally weighted kernel PLS (LW-KPLS) to estimate quality variables in two industrial processes [88].

Another parameter of interest is the size of the local neighborhood: a small neighborhood size (NS) could very well approximate local characteristics of data, however it could also risk fitting to noise if the number of observations is too small to extract the signal. On the other hand, using a large neighborhood size could introduce bias and smooth out way too much over the data. In most applications, neighborhood size is determined via kfold CV, and is kept constant for all query points. This approach can be problematic and a variable neighborhood size would be more flexible and appropriate, considering the philosophy of local modeling. Just as a new model is trained for each query, the neighborhood size might as well be determined in a just in time manner. Instead of a single global CV step to tune NS, CV can be employed in a local manner for each query point. Jin *et al.* used validation error on the nearest neighbor of query to adaptively tune NS [89]. Gupta and her collegues introduced enclosing neighborhoods, which they define as an attempt to literally enclose the query points

in the convex hull of the neighborhood. Their work addresses the problem of adapting the neighborhood size to each query, and limits the variance of local modeling for color management [90].

3.3. Feature Selection

The problem of collinearity among predictor variables, resulting in redundancy, in soft sensor data was briefly discussed in previous sections. Feature selection is another way to address this problem by simply detecting and removing them, and then learning a model using a subset of the original dataset. Even in cases where data has negligible redundancy, one could benefit greatly from feature selection for numerous reasons: (i) reducing the number of inputs can increase efficiency by eliminating variables which provide little or no information on the output, and reduce computational cost, (ii) following Occam's Razor, opting for parsimonious models with few parameters is quite attractive, both for improved interpretability, and predictive purposes, and (iii) numerical and computational problems caused by a large p, small N situation, i.e. when data is scare, and we have too many inputs to handle, can be alleviated [91].

Addition of parameters to a model will never increase fitting error, however that does not necessarily mean our model would explain the pattern any better, nor does it indicate that predictions on unseen data would be improved. On the contrary, variance of the model increases with each parameter estimate. Similar to regularization methods, feature selection also favors increased bias for the sake of reduced variance in estimations [6]. Feature selection improves the performance of most supervised learning algorithms, provided that it is based on a general, systematic procedure, and not some ad-hoc approach. Some may view feature selection as a part of data pre-processing step, as more often than not, it is performed manually by experts based on prior knowledge of the process, or some rules of thumb. However, to incorporate feature selection with any supervised training method properly, one must make sure that feature selection, and model training steps are evaluated (or validated) on separate data [92]. Reusing validation data can lead to subjective bias.

Feature selection algorithms can be broadly classified into three: filter, wrapper and, embedded. Filter algorithms aim to extract relevant predictors from the input data, independent of the performance of the estimator, thus have the potential to be more biased [93]. The wrapper approach, on the other hand, ranks variables based on their performance using the base model. If training error is taken into account during feature selection, the algorithm can end up overfitting the data; hence, CV is often used to estimate prediction error [93]. In embedded methods, feature selection is performed while the predictive model is being trained. Penalized regression methods, such as Lasso, and RVM are some examples for embedded feature selection.

Regardless of the algorithm, result of feature selection highly depends upon the choice of search strategy [91]. One could perform an exhaustive search, going through every combination of predictors to find the optimum subset. However, that is extremely infeasible even if p is small, and impossible as p increases. The simplest alternative to exhaustive search is sequentially adding or removing predictors, based on some performance metric. For instance, in stepwise regression, predictors are evaluated according to their significance in the LS model. Forward selection is a special case of stepwise regression, in which the most significant predictor is added at each step, as long as it improves the fit. Similarly, in backward selection the least significant predictor is removed [6]. All LS based relevance metrics measure approximately the same quantity, correlation between input and output variables. Mutual information (MI), is also frequently used as it can model the nonlinear relationship among data. Chi-square test, a popular one in social sciences, is a non parametric approach to determine whether two variables are dependent [6].

In the most general sense, let X and Y denote two random variables, of which x, and y are realizations in the same order.

$$\hat{\rho}(X,Y) = \frac{\sum (x-\bar{x})(y-\bar{y})}{\sqrt{\sum (x-\bar{x})^2 \sum (y-\bar{y})^2}} \tag{3.47}$$

$$\hat{I}(X,Y) = \iint \hat{p}(x,y) \log \frac{\hat{p}(x,y)}{\hat{p}(x)\hat{p}(x)} dxdy \tag{3.48}$$

Here $\hat{\rho}$, $\hat{I}$, and $\hat{p}$ are estimates of correlation coefficient, MI, and probability density function, respectively. $\bar{x}$ and $\bar{y}$ are sample means of variables X and Y.

Several measures based on information theory criteria exist for linearly parametrized models. The general idea is to use model likelihood with bias correction to estimate its generalization performance. Akaike's information criterion (AIC) defines bias as p/N, therefore it penalizes the likelihood using the number of predictors in the model. Similarly, Bayesian information criterion (BIC) compares the posterior probability of models for feature selection, it is of the same form as AIC with a larger penalty term for the number of features. Both metrics are simple yet effective measure for feature selection, however they are valid only in the case case of maximum likelihood estimation, i.e. least squares [94]. Alternatively, resampling methods such as cross validation, bootstrap and jackknife can be used to estimate bias [95].

Souza *et al.*, used MI to detect both the relevant variables and their appropriate time lags for a LS-SVM model [96]. Same authors have also proposed a multilayer perceptron based filter method for feature selection, and it was shown to be superior to other conventional methods on both simulated data and water treatment process dataset [97]. Martens and Martens proposed jackknife resampling to estimate the variance of PLS coefficients obtained during cross validation, and used t-test to eliminate useless variables in the model [98]. Other PLS based variable selection methods, such as PLS-Beta, variable influence on projection, and the selectivity ratio have also been proposed for soft sensor design; Kano and Fujiwara provide a brief yet effective summary of such approaches [22].

Industrial data is prone to contain both unnecessary and redundant variables, and the methods described until now seek for only the relevant variables, hence they do not address the critical problem of redundancy. The need to reduce redundancy has been raised as an issue in different contexts, and several methods have been proposed [99]. Redundancy can be determined by Euclidean distance, or any of the metrics already available for measuring relevance can also be used to detect redundant inputs.

Minimum redundancy - maximum relevance (MRMR) approach, as suggested by Ding and Peng for variable selection on microarray gene expression data, aims to find most relevant predictors, while simultaneously keeping redundancy at minimum [100].

3.4. Ensemble Modeling

Field of statistical learning is rife with learning algorithms, some of which have been discussed so far in this thesis, and they are simply the tip of the iceberg. However, the growing body of research in this field suggests that there is no single method to solve all of our problems. Each learner has its own niche, and can turn out to be useful in some domain while it is inferior in others. The "No Free Lunch Theorem", originated in Wolpert and Macready's work, epitomizes this situation; it states that all search algorithms in optimization can perform well on some optimization problem [101], and one may extend this finding to statistical learning algorithms. Each learning algorithm comes with its own baggage of model structures, assumptions and parameters; all of these lead to different outcomes on different sections of problems. A single learner, even if finely-tuned on data, can have inferior performance in some cases; a different learner can perhaps compensate for the weaknesses of the former learner. This is the motivation for ensemble learning: in contrast to a winner takes all approach in most supervised learning tasks, an ensemble of learners is favored.

Following the same notation, (x_0, y_0) denotes unseen observations and we wish to predict y_0 using an ensemble of M estimates, $\hat{f}_m$ with the corresponding hyperparameters θ_m, all of which were obtained by training learners, $\mathcal{L}_m$ on $\mathcal{D}_m$ for $m = 1, 2, ...M$, in the most general sense. For the sake of simplicity, the ensemble output, $\hat{f}^{\text{ENS}}$ will be defined as the average of all M outputs as follows:

$$\hat{f}_m \coloneqq \hat{f}(\mathcal{D}_m|\mathcal{L}_m) \tag{3.49}$$

$$\hat{y}_0^{\text{ENS}} = \hat{f}^{\text{ENS}}(x_0) = \frac{1}{M}\sum_{m=1}^{M} \hat{f}_m(x_0) \tag{3.50}$$

As in the case for single learners, let's start from the beginning by decomposing MSE into bias and variance components for an ensemble of learners. MSE of a single learner was shown to be equal to random error variance, prediction variance and the squared bias summed up in Equation 3.6. The same equality can be derived for the ensemble model:

$$\begin{aligned}
\mathrm{MSE}(\hat{y}_0) &= E\left\{\left[y_0 - \hat{f}^{\mathrm{ENS}}(x_0)\right]^2\right\} \\
&= \sigma_\epsilon^2 + \left[y_0 - E\{\hat{f}^{\mathrm{ENS}}(x_0)\}\right]^2 + E\left\{\left[\hat{f}^{\mathrm{ENS}}(x_0) - E\{\hat{f}^{\mathrm{ENS}}(x_0)\}\right]^2\right\} \\
&= \sigma_\epsilon^2 + \left[\frac{1}{M}\sum_{m=1}^{M}\left(y_0 - \hat{f}_m(x_0)\right)\right]^2 + \frac{1}{M^2}\sum_{m=1}^{M} E\left\{\left[\hat{f}_m(x_0) - E\{\hat{f}_m(x_0)\}\right]^2\right\} \\
&\quad + \frac{1}{M^2}\sum_{m=1}^{M}\sum_{\substack{n=1 \\ n\neq m}}^{M} E\left\{\left[\hat{f}_m(x_0) - E\{\hat{f}_m(x_0)\}\right]\left[\hat{f}_n(x_0) - E\{\hat{f}_n(x_0)\}\right]\right\} \\
&= \sigma_\epsilon^2 + \frac{1}{M^2} bias^2(\hat{f}_m) + \frac{1}{M^2} V(\hat{f}_m) + \frac{1}{M^2} cov(\hat{f}_m)
\end{aligned} \tag{3.51}$$

Equation 3.51 is quite informative as it highlights one of the main motivations behind model averaging: it can be exploited for reducing the variance of a learner since the third term in Equation 3.51 is scaled by $1/M^2$. Ensemble learning is frequently used to stabilize learners, such as NNs, decision trees and methods incorporating feature selection, which can be sensitive to small changes in data [102]. In addition, it states that MSE of an ensemble model depends on the bias and variance of the individual learners, and - perhaps more importantly - the covariance among learners. Here, covariance can take both positive and negative values, and it can be viewed as a measure of ensemble diversity, a crucial point in ensemble learning. It can be concluded that ensemble models do not necessarily reduce MSE, especially if the ensemble members (also called base learners) are not sufficiently different from each other, the resulting model can have even higher MSE of prediction than a single learner [103].

Ensemble learning can be broken down into two separate tasks:

(i) Obtaining a corpus of different models,
(ii) Combining the outputs of these learners.

The first task can be achieved in multiple ways, for instance, different models can be obtained by simply using different learning methods ($\mathcal{L}_m \neq \mathcal{L}_n$). In this manner, the diversity in model structures and parameters can be exploited. This approach has won many competitions in Kaggle, a platform for competitive predictive modeling. Another way the base learners can be trained is by fixing the learning algorithm, but using different hyperparameters ($\theta_m \neq \theta_n$). Bayesian methods accomplish this by the very definition of Bayesian learning paradigm; imposing a prior distribution on models yields a weighted average of outputs from all settings of model parameters [63]. Similarly, an ensemble of NN models can be generated by training multiple NN models all initialized with different weights. Decision tree ensembles can also be formed by randomizing the starting points of each member [104]. A different method to obtain different base learners is to train them on different sets ($\mathcal{D}_m \neq \mathcal{D}_n$). This can be achieved by means of bootstrap sampling the original dataset repeatedly, as in "bagging", partitioning the data into multiple folds, as in "crogging", or training models in a sequential manner, as in "boosting" [102, 105, 106].

The remaining issue is then how these individual learners should be combined. The easiest choice is to take the mean (or its robust counterpart, median) of all outputs. In the least assuming way, a new mapping, g, which takes the outputs of ensemble members as input, and returns the final ensemble prediction, is defined: $\hat{f}^{\text{ENS}} := g(\hat{f}_1, \hat{f}_2, ..., \hat{f}_M)$. This formulation, called stacked generalization, suggests the use of an additional learner to obtain the final predictions; g should learn the weaknesses and strengths of each learner to combine them in the most useful way. Wolpert has proposed using kfold CV to estimate biases and variances of base learners [107].

Bayesian model averaging (BMA) is an extension of Bayesian paradigm for making inferences on model selection and combination. Given observations $\mathcal{D}$, for a single learner, $\mathcal{L}_m$ with the corresponding parameters, θ_m, the marginal likelihood is defined as: $p(\mathcal{D}|\mathcal{L}_m) = \int p(\mathcal{D}|\theta_m, \mathcal{L}_m)p(\theta_m|\mathcal{L}_m)d\theta_m$. For $m = 1, 2, ...M$ learners, inference can be made through application of Bayes rule. For instance, the posterior distribution of some unseen observation, y_0 can be obtained as:

$$p(\mathcal{L}_m|\mathcal{D}) = \frac{p(\mathcal{D}|\mathcal{L}_m)p(\mathcal{L}_m)}{\sum_{i=1}^{M} p(\mathcal{D}|\mathcal{L}_i)p(\mathcal{L}_i)} \tag{3.52}$$

$$p(y_0|\mathcal{D}) = \sum_{m=1}^{M} p(y_0|\mathcal{D}, \mathcal{L}_m)p(\mathcal{L}_m|\mathcal{D}) \tag{3.53}$$

$$\hat{y}_0^{\text{ENS}} = \sum_{m=1}^{M} \hat{y}_{0,m}p(\mathcal{L}_m|\mathcal{D}) \tag{3.54}$$

Equation 3.54 shows that final estimate is basically a weighted average of individual point estimates from model outputs, with the model posterior probabilities as the weights. As it is the case with all Bayesian methods, BMA requires nontrivial computation, except for simple settings. More often than not, model likelihoods do not have a closed form solution [108]. In the absence of model posteriors, one could estimate them from data according to the needs of the particular application. For instance in prediction, validation error obtained from the training set with an unbiased method is presumed to be a good estimate of a learner's predictive performance. Thus, the likelihood in Equation 3.54 is often taken as inversely proportional to validation error [109].

$$p(\mathcal{D}|\mathcal{L}_m) \propto 1/\widehat{\text{PE}}_m \tag{3.55}$$

$$p(\mathcal{D}|\mathcal{L}_m) = \frac{\text{VE}_m^{-1}}{\sum_{i=1}^{M} \text{VE}_i^{-1}} \tag{3.56}$$

Here, $\widehat{\text{PE}}_m$ denotes an estimate of the prediction error of $\mathcal{L}_m$, for which the validation error, VE_m, is used. Consequently, Equation 3.56 can be used with Equation

3.52, and the final predictions can be obtained using Equation 3.54. Estimation of prediction error, and validation approaches for soft sensor design in general are discussed in the following section.

3.5. Model Selection and Parameter Tuning

Given a learner, $\mathcal{L}$, statistical learning, can be viewed as estimation of the model parameters with respect to various loss functions. Learning, as defined in Equation 3.2, can be expressed in more detail with the introduction of parameters:

$$\mathcal{D} \xrightarrow{\mathcal{L}_{\theta_{\mathrm{H}}}} \hat{f}_{\theta_{\mathrm{H}}}(\theta|\mathcal{D})$$

Here, θ are the model parameters learned during training on data, and θ_{H} are the hyperparameters. These two terms are often used interchangeably in the literature, and can have different meanings depending on the field. In this context, any model property which is set prior to the training step, and effects the learning process is called hyperparameter. Hyperparameters may or may not change during training; often in Bayesian statistics hyperparameters are learned unless they are assumed to be constant [63]. Parameters, on the other hand, have to be learned from data. For instance, λ is a hyperparameter of Lasso objective function in Equation 3.16, and $\hat{\boldsymbol{\beta}}_{\mathrm{Lasso}}$ is a parameter which is obtained from data. Both features play crucial role in the final prediction performance of a learner, hence should be given utmost importance and optimized properly. For the sake of simplicity, both terms will be referred to as model parameters hereafter.

Let θ denote some parameter settings of a learner, and $\theta \in \Theta$, where Θ is a set of all settings possible. Given training data, we estimate $\hat{f}(\theta) \equiv \hat{f}_\theta$ with $\hat{f}_\theta \in \widehat{F}$, and $\widehat{F}$ is a set of all possible model and parameter estimates which can be obtained using the learner $\mathcal{L}$. As a consequence of this formulation, parameter tuning and model selection can be viewed as two sides of the same coin: each parameter setting defines a new

model, and we wish to find the optimum θ which leads to $\hat{f}_\theta$. Perhaps, model selection is the more enveloping of the two since choosing a learner can also be classified into this category. Feature selection is also a subset of the more general model selection problem.

This brings us to the real issue at the heart of statistical learning: how should we decide on the parameters settings? Equivalently, which model estimate should we opt for? In predictive modeling, the ultimate aim is to achieve minimum prediction error on new, unobserved data. Thus, if the expected prediction error can be estimated during training, then it would be used for any kind of model selection problem. It is strongly advised against interpreting training error as any measure of generalization performance. The rule of thumb is that data is used for only one task; one simply does not validate on training data as the resulting assessment would be too optimistic. Some analytic expressions have been proposed for linear regression applications to use in sample estimates to evaluate out of sample error, and tune model complexity by estimating the amount of optimism obtained from training data. AIC and BIC, both of which have been previously discussed for feature selection, perform model selection using in sample statistics. Both metrics employ some penalty on number of parameters hence prefer more parsimonious models [6].

On the other hand, CV and bootstrap both directly aim to estimate out of sample error, i.e. generalization performance, by imitating the prediction step during training. CV, an intuitive solution to this problem, has thus become one of the most popular methods for model selection. The idea is to divide the training dataset into two parts, referred to as calibration and validation, respectively to avoid confusion with test data; and construct a model using only one part to validate on the other part of the data. In the most general case, data is first shuffled and then divided into k parts, called folds. Then each fold is held out in turn for validation, while the other folds are used to build a model. In the repeated case, once each fold is validated the entire training data is reshuffled and the whole partitioning-validating procedure is repeated for R times in total. Mosteller and Tukey have likened this approach to “squeezing data dry” [110].

In leave-one-out CV (LOOCV), a special case when k $= N$, only one sample is held out at a time; resulting in mostly overlapping calibration sets at each iteration, prediction error estimates with high variance. Even though LOOCV has the smallest bias since almost all the training data is used at each step, choosing moderate k values ($\sim 5-10$) reduces variance of the estimates [111]. Values of k and R can change from application to application, and largely depend on the total number of observations.

It should be noted that CV is a viable choice for model selection and evaluation provided that it is used appropriately. The pitfalls in validating embedded feature selection methods was discussed briefly in Section 3.3. The central point in designing a CV procedure is that one should make sure each observation is used for one task only [92].

With all theoretical considerations in mind, the procedure for parameter tuning is rather straightforward. In kfold CV with R repetitions, we generate k pairs of mutually exclusive calibration and validation sets R times in total. Let $D^{\mathrm{r}}_{-i} = \{\mathbf{X}^{\mathrm{r}}_{-i}, \mathbf{y}^{\mathrm{r}}_{-i}\}$ and $D^{\mathrm{r}}_{i} = \{\mathbf{X}^{\mathrm{r}}_{i}, \mathbf{y}^{\mathrm{r}}_{i}\}$; D^{r}_{i} represents the set of observations in the i^{th} fold, and D^{r}_{-i} is comprised of the training observations excluding the i^{th} fold. D^{r}_{-i} and D^{r}_{i} are often referred to as the calibration and validation sets at r^{th} repetition obtained from the shuffled training set D^{r}, thus, $D^{\mathrm{r}} = D^{\mathrm{r}}_{i} \cup D^{\mathrm{r}}_{-i}$. The average CV error (CVE) can be computed as follows:

$$\hat{f}^{\mathrm{r}}_{i,\theta} \coloneqq \hat{f}(D^{r}_{-i}; \theta | \mathcal{L}) \tag{3.57}$$

$$\hat{\mathbf{y}}^{\mathrm{r}}_{i}(\theta) = \hat{f}^{\mathrm{r}}_{i,\theta}(\mathbf{X}^{\mathrm{r}}_{i}) \tag{3.58}$$

$$\overline{\mathrm{CVE}}(\theta) = \frac{1}{\mathrm{R}} \frac{1}{\mathrm{k}} \sum_{\mathrm{r}=1}^{\mathrm{R}} \sum_{i=1}^{\mathrm{k}} \left[\mathbf{y}^{\mathrm{r}}_{i,val} - \hat{\mathbf{y}}^{\mathrm{r}}_{i,val}(\theta) \right]^2 \tag{3.59}$$

Here, $\hat{f}^{\mathrm{r}}_{i,\theta}$ is a model estimated during r^{th} repetition on i^{th} calibration fold, D^{r}_{-i}, with the parameters θ. At the end of this procedure, an average CV error of $\overline{\mathrm{CVE}}(\theta)$ is obtained for θ. Consequently, we can select the "optimum" parameter $\theta \in \Theta$ as the

one which minimizes the average CVE.

$$\theta^* \coloneqq \underset{\theta \in \Theta}{\operatorname{argmin}}\, \overline{\mathrm{CVE}}(\theta) \tag{3.60}$$

Figure 3.5 shows how the number of components in a PLS model can be optimized. Lowest $\overline{\mathrm{CVE}}$ (obtained from 10fold CV with 20 repetitions) is achieved with a PLS model of five components (shown with a circle). The dotted line in Figure 3.5a corresponds to the limit of one standard error (1 SE), suggested by Breiman [112]. In 1 SE rule, the most parsimonious model within 1 SE of the minimum CVE is selected as the optimum model, hence four PLS components seem sufficient.

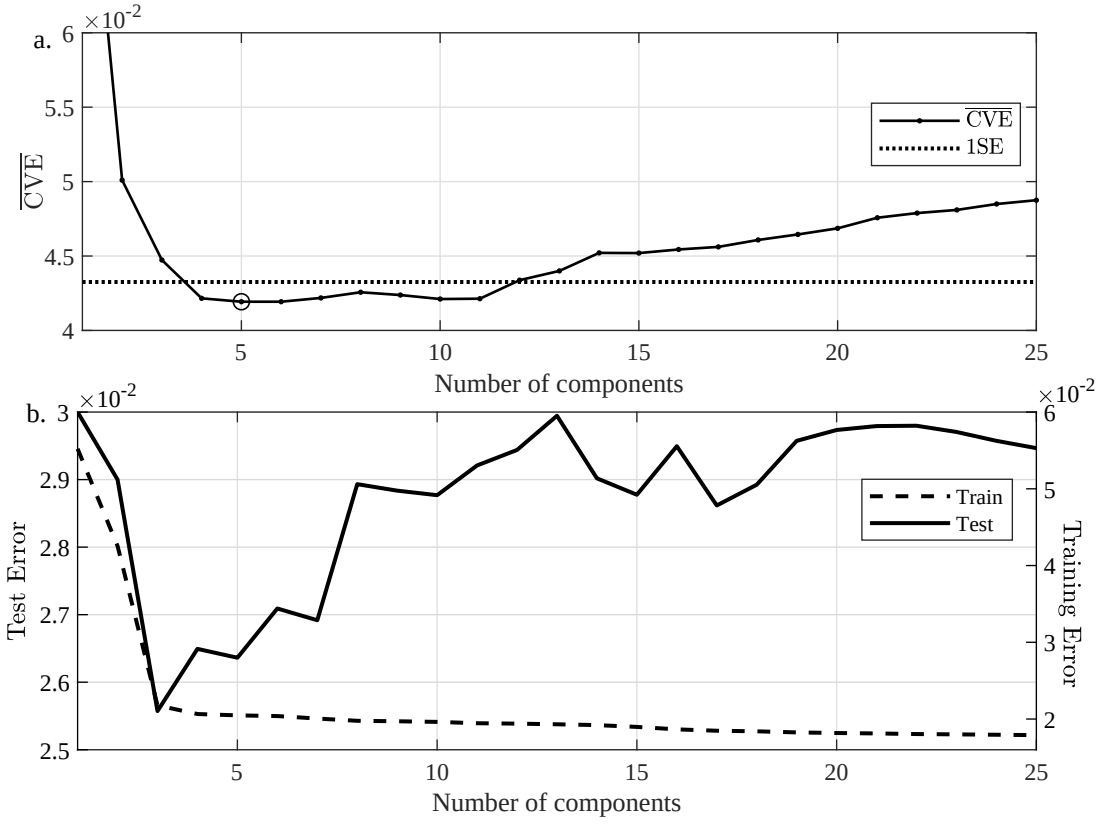

Figure 3.5. Optimizing the number of components in a PLS model via kfold CV.

Training and test errors corresponding to each PLS model can be seen in Figure 3.5b; as expected, the training error continually decreases with the increased model complexity, quantified by the number of latent variables in this case, and the prediction error starts to increase beyond a certain point. This is the bias/variance tradeoff in action.

Proper assessment of prediction performance, and parameter tuning, unfortunately, have not been employed extensively in chemical engineering literature. While recent advances in the field of statistical learning have found their way into soft sensor design applications as well, many studies still lack the rigor of the well designed experiments, and the scarcity of benchmark datasets for soft sensor design is an important issues which should also be addressed as it would encourage overall a more objective approach to model assessment [41].

The preceding analysis of model selection and assessment was made with the assumption of static (or steady state) data, i.e. observations have no temporal relevance. Soft sensor data from industrial processes, on the other hand, are in the form of data streams; soft sensor design is essentially a time-series forecasting problem. Most applications in machine learning on the related subjects of model selection, validation and assessment are confined to steady state data, and there is a limited amount of research on model evaluation in time series problems.

The propensity to use CV mainly stems from the fact that it can make use of the entire data, and by repeated application of CV variance in model selection can be greatly reduced. However, neither of these properties of CV is applicable in time series forecasting. Shuffling data makes no sense when the order of samples do have a meaning, and partitioning training set into mutually exclusive folds does not necessarily yield independent estimates of prediction error since autocorrelation may exist between observations. The usual approach is then to use out-of-sample evaluation (also called last block or holdout validation) in which the last portion of data is held out during training and reserved for validation [113].

Time series data by its nature can have additional issues affecting the parameter tuning procedure. For instance, in some cases the underlying input-output relationships in data might be evolving over time, hence the i.i.d. assumption no longer holds. Instead of last block validation, a "rolling-origin" approach would be more suited nonstationary data which exhibits time varying properties. In rolling-origin validation methods, validation points are appended to the calibration set once they are predicted, and the model trained on the calibration may or may not be updated using the newly appended validation point [114]. This approach is also called prequential, as in predictive sequential, it was suggested by Gama as an alternative to the batchwise holdout method for parameter tuning and model evaluation in data streams [115]. Model selection and assessment for nonstationary time series data with drift is discussed in more detail in the next section.

4. LEARNING FROM DATA STREAMS

In most real life applications of statistical learning methods, instead of the usual training and test steps, both of which comprise of a fixed number of observations, data is acquired in the form of streams. One could view data streams as test sets of infinite size resulting from emerging technologies which have enabled continuous feeding of data from sensory measurements in large amounts. Change with respect to time, i.e. nonstationarity is an inevitable feature of data streams hence predictive modeling within the context of a changing environment should be able to deal with nonstationary data evolving in time. Nowadays, extracting information from data streams has become the subject of interest in many fields, such as speech recognition, text classification, natural language processing, and anomaly detection [116].

Soft sensors for industrial processes have to be developed within the same framework; process measurements, collected via numerous sensors, arrive in the form of streams in time, hence must be processed sequentially for real-time estimation. New problems emerge for learning in this scenario. In contrast to the static case, training set expands with each new measurement; not only do training samples change, but the underlying function, or the concept, which we always presume that exists between the input and the output variables, and we wish to estimate, can also change in time. Changes in the underlying functional relationship in time are generally known as concept drift [26].

While concept drift has been discussed in detail in the literature, a unified terminology or a taxonomy does not exist, unfortunately. Formally, concept drift is defined as a change in distribution of data:

$$\exists\, x, y \text{ s.t. } p^{t_0}(x, y) \neq p^{t_1}(x, y) \tag{4.1}$$

Equation 4.1 defines concept drift from time t_0 to t_1. Thus, the functional relationship in Equation 3.1 may also change in time, and f becomes a function of not only x but also the time variable, t, $f(x,t) \equiv f^t(x)$.

More specifically, changes in $p(y|x)$, the conditional distribution of the output given the input, are referred to as real concept drift in the literature [116]. Changes in $p(x)$ that do not necessarily affect the conditional distribution, on the other hand, are called virtual concept drift, and thus are presumed to occur due to insufficiency in input data, and not from real drift in data [117]. In the current thesis, we follow the same convention to differentiate between two types of drift, real and virtual.

Concept drifts can be encountered in several different forms: (i) it can be an abrupt change, as in sensor failure or hardware replacement, (ii) incremental or gradual changes are observed with deterioration in process equipment such as fouling, or catalyst deactivation, and (iii) recurring changes can occur when process operates back and forth between different operating conditions, as in switching from one feed supplier to another, or regular seasonal changes. Further criteria, such as predictability, frequency and severity, may also be used to analyze and classify various types of concept drift [40].

One of the central issues in soft sensor design is its maintenance in the presence of concept drift. Data from process industries exhibit time varying characteristics as mentioned above, and changes in the process dynamics can severely effect the performance of soft sensor models. According to surveys on soft sensor applications in Japan, decline in performance and accuracy in time as the process dynamics change was among the most common problems [22]. Several solutions for soft sensor adaptation have been proposed [25], but, there still remain issues which are unsolved, or can be improved upon.

4.1. Adaptive Sample Selection (Windowing) for the Training Set

One cannot rely on a single learning model trained offline to keep soft sensors up to date. Furthermore, predictions need to be provided in real-time, thus learning algorithms should be able to adapt to new concepts within certain time and memory constraints. In the field of machine learning, this scenario for modeling is known as online (or incremental) learning; it differs from offline learning, in which the training data is treated as static, and once the model parameters are estimated offline they are fixed. Offline algorithms implicitly assume a static training set, comprising independent and identically distributed data generated from a stationary distribution $p(\mathcal{D})$, where $\mathcal{D} \coloneqq \{\mathbf{X}, \mathbf{y}\}$ [76]. These assumptions, however, generally do not hold for data streams, which are inherently time dependent. Online approach to learning can be particularly useful in dealing with non-stationary data, and can process data in real time in both batches or on sample by sample basis.

Adaptive learning algorithms, utilizing online learning models are developed to handle concept drift in data streams. Figure 4.1 shows main approaches in developing adaptive algorithms for handling concept drift using a single learner. Each row corresponds to a different adaptive learning method, and each column refers to data points sampled at successive time instants. Circles and squares are used to represent the difference in training data with respect to concept homogeneity, i.e. the first three data points and the last three (excluding the farthermost right circle) are sampled under the rule of the same concept, while the three squares in the middle belong to another concept. The rectangles formed by the solid line would consists of data, which only belong to the circle concept, whereas that formed by dashed lines comprises of heterogeneous data consisting of circle and square concepts. The current test point is filled with gray and it belongs to the circle concept, and the observations included in the training set selected for prediction in each method are shown with double lines. Moving window approach (MW) at the top selects the most recent three training samples as they belong to the circle concept of current query point. The recursive method (middle) updates its model with each new observation, hence it uses a model estimated

from a training set which includes square concepts as well. JITL (at the bottom), on the other hand, reveres spatial similarity and it uses all the past circle training samples.

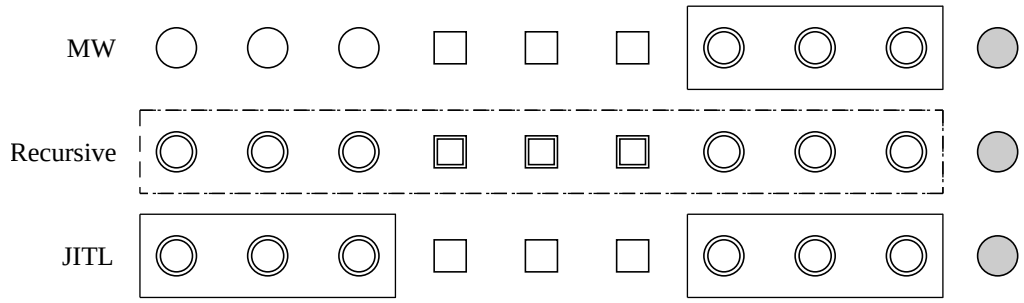

Figure 4.1. Common approaches to handle concept drift.

These three approaches have their strength and weaknesses. For instance, recursive algorithms are quite stable in the presence of outliers, however, they are slow to react to abrupt changes and can suffer from inertia brought in by old data which does not necessarily relate to the new concept. Moving window approaches can adapt more quickly than recursive ones, but the choice of window size plays a crucial role in achieving decent adaptation performance, even more so than the forgetting factor in recursive methods since the window defines a hard boundary. In addition, it is based on the assumption that most recent data are more relevant to the current concept, hence it is a lot more sensitive to noise and outliers and the model estimates tend to be less stable. JITL is similar to the moving window approach since both advocate use of only a subset of data, but JITL does not necessarily rely on temporal relevance. Similarity can be measured by any means; more often than not, spatial distance in input space is used to define relevance. JITL can address abrupt changes fairly well, especially in the presence of recurring drifts.

In addition to the methods stated above, ensemble learning has found many applications in drifting data scenarios [39]. With ensemble learning, a collection of submodels is maintained and combined according to the current concept. Ensemble techniques are suggested to address the transition periods during a concept change as

each member may be utilized to model a different aspect of the transition. Ensemble methods suffer from high complexity as there are many aspects to consider when designing an ensemble learning algorithm for online prediction, in addition to the regular problem of maintaining ensemble diversity. Should we expand the ensemble members with new data, create new members, or do both? Should we keep the older learners even if they make worse predictions on newer data thinking that some day they will prove useful? Even if a decent forgetting/learning strategy is determined, one needs to figure out the best way to combine the ensemble models.

4.2. Adaptive Mechanisms in Adaptive Learning Algorithms

Generally speaking, all adaptive learners are made up of the following three dimensions:

(i) A mechanism to detect concept drift.
(ii) An updating scheme for adapting the predictive model parameters.
(iii) A forgetting mechanism for removing older information.

Dimension (i) of an adaptive learner can be either explicit, implicit or nonexistent; one could view online learning algorithms as implicit concept change detectors since they provide the basic framework for continuously updating the model parameters for each observation. Moreover, online learners naturally occupy dimension (ii) as incoming data is incorporated when parameters are updated incrementally. An alternative would be to retrain the model. In addition, in JITL, model parameters are estimated only on demand; thus it does not require an additional adaptation scheme as it trains models from scratch for each query.

In the absence of an explicit drift detection mechanism, learning algorithms perform blind updating. For instance, in a moving window setting, if the window size is kept constant then the model updates would be performed at a fixed rate. Thus, when concept change occurs, the adaptation rate could be too slow, or when the process is

relatively stable updates would be too frequent, and could lead to unstable parameter estimates. One could perform implicit drift detection by adaptively changing hyperparameters such as the window size and forgetting factor, instead of keeping them fixed. Gama *et al.* refer to this as an informed updating strategy since the adaptive algorithm reacts by changing the window size [116]. Explicit detection, on the other hand, in general yields better predictive performance. There are broadly two approaches to change detection: mechanisms based on statistical process control concepts, and detection by conducting significance tests on different windows of data. Generally speaking, detection strategies based on statistical process control techniques are known to provide more precise estimates of the location of concept change [118].

Dimension (iii) gains importance in applications with strict memory requirements; unless there is any shortage of memory, removal of old samples from the database is less of a concern, however it can be used as a method to eliminate redundant information and increase the efficiency of the algorithm. In a MW setting, a hard threshold is employed to forget old knowledge, i.e. observations are either in or out of the window. The usual approach is to employ predictions on a landmark moving window basis; in which the landmark window grows with each instance received until maximum memory storage is reached, and it encapsulates the smaller local moving window of recent samples. In ensemble learning, however, more attention should be paid to dimensions (ii) and (iii) in adaptive algorithms. Forgetting old knowledge, and absorbing new one become more tricky when an ensemble of models exists, as more questions need to be answered due to increased complexity of the modelling setting. The stability/plasticity dilemma again plays a crucial role here.

Divide and conquer is a frequently exploited approach for adaptive modeling in chemical engineering applications since industrial processes in general are run in multiple operating modes. In most studies, the first step is to partition the historical database into multiple regions by any means before test points arrive. Some methods opt for a MW, or a JITL strategy to select a local region for prediction based on temporal or spatial relevance, respectively. Then, each test point is predicted using

either a single local region or an ensemble of regions, followed by updating local model parameters, and inclusion of the test point into the database. The majority of the proposed adaptive algorithms is based on this structure, and may differ only in small details.

In correlation based just in time modeling (CoJIT), localization was employed on a moving window basis to obtain multiple nonoverlapping regions comprised of consecutive observations. They define a dissimilarity metric based on Q and T^2 statistics from PCA to select the best region for predicting query point. The authors suggest using PCR as the base learner, and CoJIT was shown to outperform RPLS in a CSTR simulation and real industrial data [38].

Kadlec and Gabrys proposed a localization procedure, alternative to k-means, which does not require the number of clusters to be preset, in their highly influential paper on adaptive soft sensor design [41]. Starting with an initial window of size n^{init} comprised of subsequent observations, a PLS model is trained; then the window is slided by one sample, and the observations in this shifted window are predicted using the initial PLS model. A t-test is performed on the mean values of residuals and prediction errors of the models obtained from consequent windows, and this procedure is repeated until a significant result is obtained, thereupon the first local region, or receptive field as they had named it, is determined. In the end, multiple overlapping regions are obtained, and separate PLS models are fit to each region. Throughout their incremental local learning soft sensing (ILLSA) algorithm, prediction accuracies of these regions are monitored via a performance index called local experts descriptor, which is essentially a two dimensional density estimation of prediction errors of the regions on previous observations. Predictions from local regions are combined based on a Bayesian model averaging using these descriptors in conjunction with their output for the query point, thereby giving higher weights to regions which have made similar predictions in the past and achieved high accuracy. Local models were updated with incoming data using RPLS, and the algorithm parameters were determined via CV. ILLSA was shown to outperform RPLS in catalyst deactivation dataset. Even though

local PLS models are updated with new data, authors did not consider removing older redundant regions or creating new regions with new concepts; localization step was performed only once in an offline manner. It should be noted that this approach to define different concepts is expected to have low power since it is based solely on a comparison of mean values of residuals and prediction errors. Moreover, repeated application of t-test during this step is prone to type-I error, and it is questionable to perform t-test in the first place on subsequent, overlapping windows of observations which exhibit autocorrelation.

Kaneko and Funatsu proposed an ensemble of LWPLS models (ELWPLS); offline localization step was similar to that in ILLSA, but based on an error threshold instead of a t-test [119]. Consequently, LWPLS models are constructed only in local regions which encapsulate the query point, and the final prediction is obtained as the weighted sum of outputs of each LWPLS models, in which the weights are assigned to be inversely proportional to the previous prediction errors of the subsets. ELWPLS performed better than conventional PLS, as well as MW and JITL variants of PLS on debutanizer column dataset.

The localized and adaptive RPLS (LARPLS) algorithm, as proposed by Ni el at., also starts by an offline localization step in which the training data is divided into windows comprised of consecutive observations multiple times, and separate PLS models are fit [120]. The region with lowest prediction error on the previous sample is selected and extended to include the test point after each prediction, and the model parameters are updated using RPLS with a forgetting factor. LARPLS aims to address two main issues. First, the forgetting factor in RPLS is determined adaptively as a function of the most recent prediction error, instead of keeping it constant at an arbitrarily fixed value which could potentially introduce bias into the model. Second, LARPLS explicitly detects concept change by comparing the most recent absolute relative prediction error to a preset threshold, and relocalizes the regions in the historical dataset. While LARPLS can handle the problem of determining forgetting factor, it introduces a new parameter: the threshold for adapting which needs to be tuned. A systematic approach

for tuning was not suggested, and this threshold was determined by trial and error.

LARPLS was improved upon with localized adaptive soft sensor (LASS) algorithm in [34]. LASS employs the same localization procedure as LARPLS; the previous test point is used as a validation sample to determine the best local region, and its size for the current prediction. LASS differs from LARPLS in two main aspects: First, once a test point is predicted and appended to the current local region, the oldest sample is removed to maintain a constant window size, i.e. LASS uses a MW approach. Second LASS can adaptively determine the relocalization threshold. LASS was shown to perform better than both RPLS and LARPLS on three chemical engineering processes: a propylene polymerization process, the catalyst activity dataset and a CSTR process.

Alternatively, unsupervised clustering methods, such as kmeans and Gaussian mixture models (GMMs) can be used for localization. GMMs were utilized for creating ensembles of GPR models, and a cautious model updating strategy was adopted by checking model variance for extremely small values throughout the operation to avoid numerical instabilities, GPR models were used to estimate model variance from data [121]. This GMM based adaptive soft sensor was shown to perform better than RPLS on Tennessee Eastman process, and two real life industrial datasets.

4.3. Adaptive Implementations of Various Learners

Recursive LS (RLS) RPLS, and stochastic gradient descent are among the most popular examples of online learning algorithms. RPLS is based on a recursive updating mechanism for weights in PLS models [27]. In RPLS, new information can be incorporated recursively in batches, or with a single observation; two separate PLS models can also be combined within the same formulation. RPLS has the advantage of integrating a forgetting factor which can be used to age (or downweight) old knowledge while absorbing the new one, hence adapting to the changing environment. For a single

observation, PLS model can be updated by applying PLS to the new matrices:

$$\mathbf{X}^{new} = \begin{bmatrix} \lambda_f \mathbf{P}^{\mathrm{T}} \\ \mathbf{x}_i \end{bmatrix} \text{ and } \mathbf{y}^{new} = \begin{bmatrix} \lambda_f \mathbf{B}\mathbf{Q}^{\mathrm{T}} \\ \mathrm{y}_i \end{bmatrix}$$

Here λ_f is the forgetting factor, and $0 < \lambda_f < 1$. At one extreme, taking $\lambda_f = 1$ corresponds to building a global PLS model on the entire historical dataset, i.e. blindly updating the existing model with each new sample, and $\lambda_f = 0$ corresponds to solely using the most recent sample for prediction. An optimum middle ground exists, where novel information is given enough weight, and the effect of older knowledge is downgraded adequately. Selection of λ_f is the embodiment of stability/plasticity dilemma. In any online learning algorithm, plasticity is required to incorporate novel information, and stable learners can handle noise and avoid forgetting old knowledge entirely, which is called catastrophic forgetting [122].

Another RPLS model was developed based on a two updating mechanisms; a bias correction step was introduces, PLS models were trained on a MW basis and old information was retained by keeping a running average of the mean, and recursively updating the variance [33].

$$\bar{\mathbf{x}}_{k+1} = \frac{W-1}{W}\bar{\mathbf{x}}_k + \frac{1}{W}\mathbf{x}_{k+1} \tag{4.2}$$

$$\sigma^2_{k+1} = \frac{W-2}{W-1}\sigma^2_k + \frac{1}{W-1}\left(\mathbf{x}_{k+1} - \bar{\mathbf{x}}_{k+1}\right)^2 \tag{4.3}$$

$\bar{\mathbf{x}}_{k+1}$ and σ^2_{k+1} correspond to the mean and variance at $(k+1)^{\text{th}}$ point. Equation 4.2 and Equation 4.3 ensure that older information from the historical dataset is also utilized in within the MW framework. Bias correction was employed using an exponential weighted moving average (EWMA) filter on the past prediction errors. Dual updating mechanism in the algorithm helps reduce variance as the PLS weights are updated at fixed time intervals and not for every instance, while the output bias is corrected for the rest of the predictions. It was asserted that recursively updating PLS parameters helps in quick adaptation of model, and the bias correction can track slow changes in

the process dynamics. Proposed model was shown to outperform conventional variants of PLS, global, dynamic and recursive. This approach, however, does not incorporate any mechanism for detecting concept drift; and the frequency of PLS model updates was set arbitrarily.

A different version of adaptive PLS modeling for online prediction is locally weighted PLS (LWPLS) [84]. Similar to local modeling approach in LWR, instances can be weighted according to some predefined similarity and/or distance measure in PLS. Both RPLS and LWPLS, thus, give higher importance to certain instances when estimating model parameters; in RPLS it is assumed that more recent observations are more relevant to the current data point, and in LWPLS the relevancy can be any measure of similarity, which is the proximity of observations in the input space. Unlike RPLS, the local structure in LWPLS can address nonlinearity as well as changing concepts. Kim *et al.* have applied LWPLS with Euclidean distance to measure dissimilarity, to a pharmaceutical process, to predict the active pharmaceutical ingredient (API) in granules, and reported that improvements had been observed in predictive performance compared to the conventional PLS.

Recursive versions of other statistical learning methods have been developed as well. For instance, Cauwenberghs and Poggio have described an incremental online SVM (OSVM) which can also be reversed for decremental learning, i.e. forgetting for classification [29]. Their incremental method had been further extended for regression in accurate online support vector regression (AOSVR), and it was shown to yield exact results as those of batch SVR [123]. Extreme learning machine (ELM) has been proposed as an alternative to backpropagation for training NNs with a single hidden layer; unlike gradient descent algorithm, weights are determined analytically and hence NN models can be learned at much higher speeds enabling online implementation [124]. Both OSVM and ELM have found many applications in soft sensor design [32, 43, 125, 126].

While development of learning algorithms play a crucial role for prediction in data streams, model assessment and evaluation of predictive performance of adaptive algorithms is another issue of almost equal importance. A systematic approach is required to assess predictive algorithms with data evolving in time. Several cross validation and bootstrap methods exist for evaluating static learners, and their results are often viewed as reliable objective estimates of performance. However, there is no such technique agreed upon to apply in streams of data. In addition, the nonstationary characteristics of data streams violate the assumption of i.i.d. samples, and the temporal relevance in time series data should be preserved hence random permutations or shuffling of observations are not allowed.

Evaluation on a rolling forecasting origin was suggested by Bergmeir and Benitez. to evaluate forecasting methods for time series data in [114]. In the rolling origin setting, test points are appended to the training set, and the model may or may not be retrained on this new sample. Gama suggests use of sequential analysis for evaluation, they define the predictive sequential (or prequential) method in [115]. Prequential analysis is also based on a rolling origin, in addition it allows the learner to adapt as new test data arrive. In this thesis, we also adopt the use of prequential analysis for evaluating the proposed adaptive learning algorithms.

5. SOFT SENSOR DATA

In order to assess the performance of a learning algorithm proposed for soft sensor design, an appropriate experimental strategy along with appropriate data is needed. In the current study, various statistical learning tools are employed, and evaluated in experiments on three different datasets, of which two are synthetic, and one is from the industry, i.e. real data; and these datasets are discussed from mechanistic and statistical aspects in this chapter.

5.1. Synthetic Data

The use of synthetic data in machine learning experiments allows one to make assessments which can answer more specific questions. Synthetic data, while being required to imitate a real life process in order to extrapolate the results of the experiment to real world, is essentially free of unknown disturbances which are often encountered in real life. Here, two synthetic datasets are generated from simulations in SimSci PROII^{TM} software and MATLAB^{TM} Simulink; the first one is based on a steady state model of a distillation column, and the second one is a dynamic simulation of a continuous stirred tank (CSTR) process.

5.1.1. Steady State Crude Distillation Unit Simulation

Petroleum refining begins when crude oil enters the refinery, and it is first separated into its fractions in a crude oil distillation unit (CDU); kerosene, diesel and naphta are among the usual products of a CDU. CDUs, usually comprised of an atmospheric and a vacuum distillation column, are possibly the most crucial step in refineries. Crude oil is useless on its own, and it should be purified as its fractions are highly valuable and make up large portion of the profits of a petroleum plant, hence the capacity of a refinery is often defined based on its CDU output [127].

Petroleum products are usually made up of many components, thus they are specified using property measures other than composition or density. API gravity (or specific gravity) is a density measure developed for petroleum products; it is inversely proportional to the liquid's density and is defined relative to the density of water. In addition, T95 values, the temperature at which 95% of oil evaporates, are among the most important properties of a fuel, it is frequently used as an indicator for the quality of a petroleum liquid. Laboratory tests for determining such properties are usually conducted w.r.t. the standards defined by American Society for Testing and Materials (ASTM).

SimSci PROII^{TM} is a software for steady-state chemical process simulations, and it is often used for monitoring, optimization, and troubleshooting existing processes as well as testing new process designs [128]. A CDU comprised of a preheater, three pumparounds, overhead and bottom streams, and three side stream with strippers was constructed using PROII (see Figure 5.1). At nominal operating conditions, crude oil with API gravity of 29.5 enters the CDU at 390 °C, 2 atm and 13.5 m^3/min. The quality variables are taken to be the T95 (ASTM – D86) values of three outlet streams: kerosene, diesel and gas oil. Laboratory measurements are required to determine T95 values, and these measurements can be obtained every 8-24 hours or so in the industry; thus it is aimed to design a soft sensor for predicting these key variables.

Two different types of perturbations were applied to 16 input variables (IVs) selected from the process; e.g. crude inlet temperature and flowrate, reflux ratio, and distillate temperature; these IVs are listen in Table 5.1. For both types of perturbation, two datasets were created: interpolation and extrapolation, as denoted by the subscripts int and ext, respectively. The purpose of interpolation datasets is to evaluate the performance of predictive models on operating conditions which are more or less similar to the training data. On the other hand, extrapolation dataset will be used to assess the performance of estimators on data outside of the region defined by the training data, i.e. to emulate virtual concept drifts.

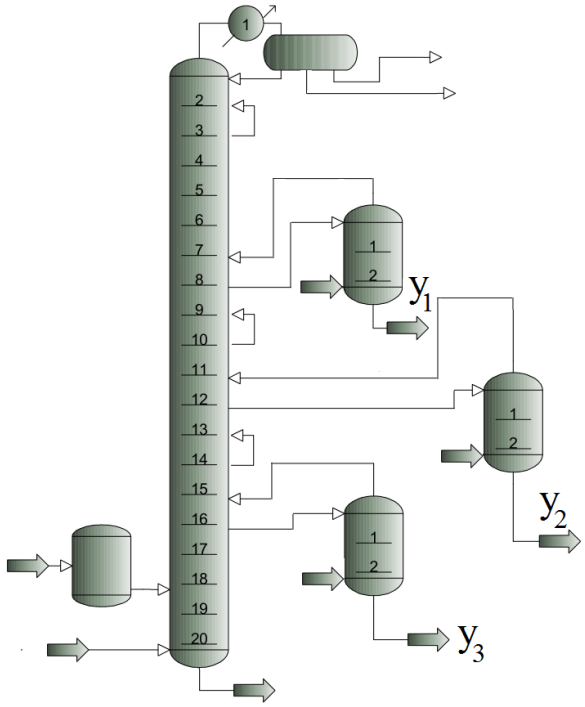

Figure 5.1. Distillation process flowsheet drawn using PROII software.

Table 5.1. Input variables to CDU simulation.

IV	**Description**	**IV**	**Description**
1	Crude inlet flowrate	**9**	Gas oil flowrate
2	Crude inlet temperature	**10**	Stripper 1 inlet flowrate
3	Inlet water flowrate	**11**	Stripper 2 inlet flowrate
4	Reflux ratio	**12**	Stripper 3 inlet flowrate
5	Condenser Duty	**13**	Steam temperature
6	Distillate temperature	**14**	Pumparound 1 temperature
7	Kerosene flowrate	**15**	Pumparound 2 temperature
8	Diesel flowrate	**16**	Pumparound 3 temperature

The first perturbation scheme is constructed using a fractional factorial design on 16 IVs (2^{16-10}), followed by addition of random numbers distributed proportional to the magnitude of each IV. The resulting simulation input-output data is named $DS_{1,int}$. Following the same procedure, $DS_{1,ext}$ was generated using random perturbations of twice the standard deviation of those in the interpolation set. This scheme was employed multiple times to obtain sufficient number of converged simulations. Figure 5.2 shows the distributions of IV3 and IV6, top tray temperature (one of the simulation outputs) in interpolation to extrapolation sets. Bi-modal distributions seen in $DS_{1,ext}$ shows that virtual drift is real life processes are mimicked successfully.

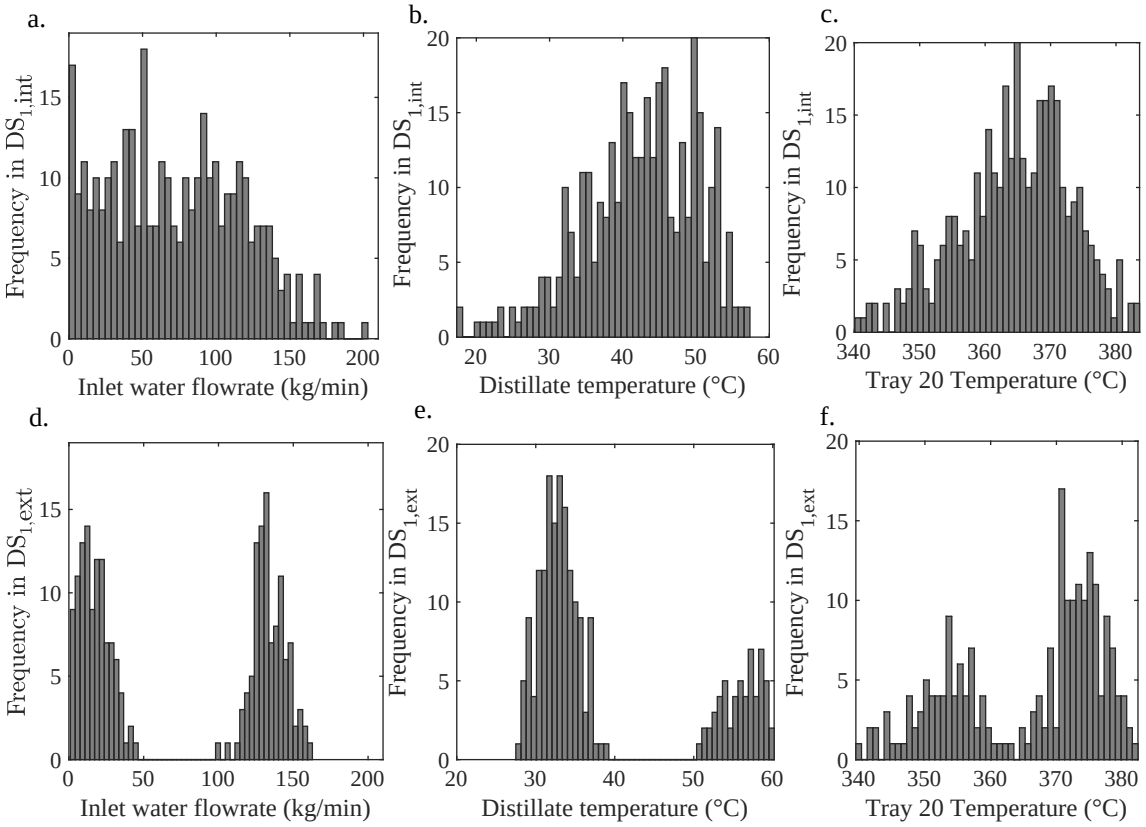

Figure 5.2. Histograms of IV3, IV6, and simulation output of one process variable (top tray temperature) from interpolation (a, b and c) and extrapolation (d, e and f) parts of DS_1.

Second perturbation scheme is essentially a generative model; inputs of interpolation dataset, $DS_{2,int}$, were generated from a model with five latent variables, whereas

the extrapolation inputs for data in $DS_{2,ext}$ were obtained from a six latent variable model, orthogonal to the first five. In this scheme, a much smaller portion of the input space is spanned; and interpolation and extrapolation sets have different perturbed modes. Thus, it is expected that T95 values in $DS_{2,ext}$ would be more difficult to predict.

75 process variables, such as tray temperatures, pressures and flowrates, were exported from the simulations in addition to the simulation inputs. Once the constant outputs were removed and the simulation inputs were added, 70 variables were retained in total.

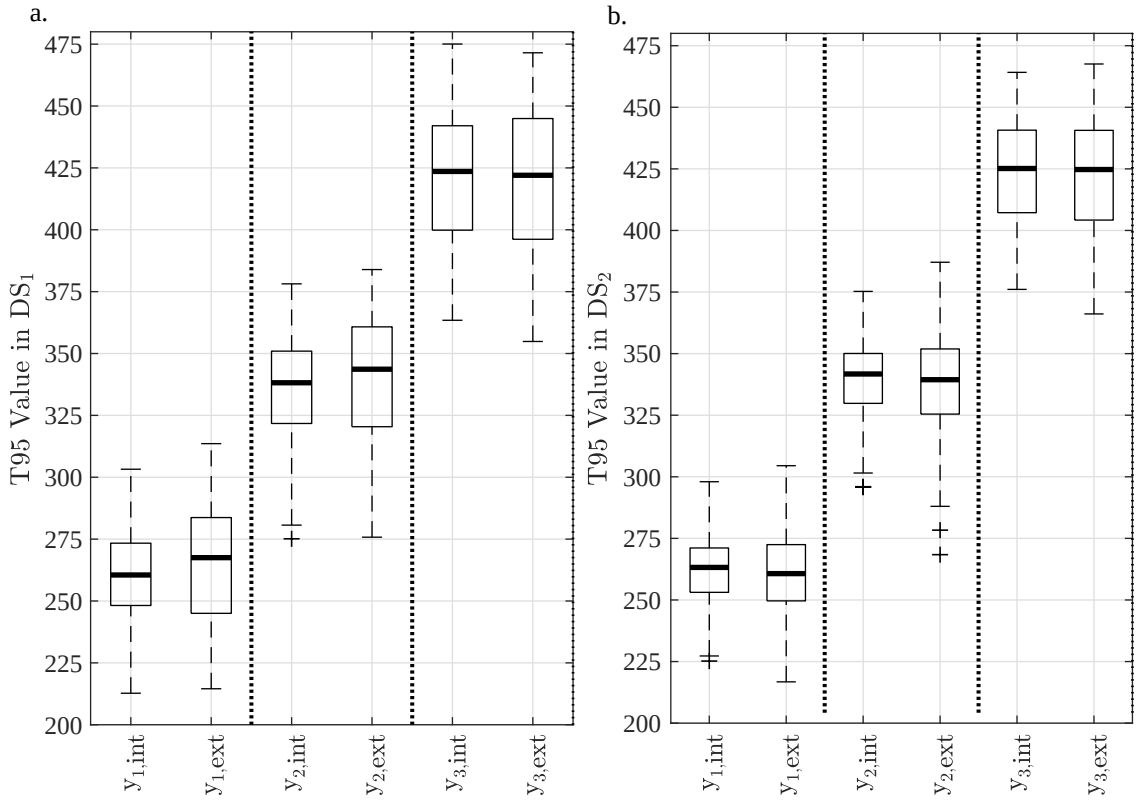

Figure 5.3. Boxplots of response variables, y_1, y_2 and y_3, are separated using dotted lines, and the interpolation and extrapolation data are paired from both DS_1 (a) and DS_2 (b).

Key variables of this process, the T95 values of three streams, kerosene, diesel and gas oil were designated as the response variables (shown with subscripts 1, 2 and 3, respectively) and the rest of the variables were used as predictors for the soft sensor estimators. Response variables from interpolation and extrapolation parts of both datasets, DS_1 and DS_2 are shown in Figure 5.3. For all response variables, extrapolation sets seem to cover a wider range of T95 values, this is consistent with the application aim of extrapolation sets. Thus, each dataset is comprised of 300 observations and 70 predictors. As a final step before modeling, Gaussian random noise with zero mean and 2.5 standard deviation was added to the response variables to emulate the repeatability of T95 laboratory measurements in real life, whereas the predictor matrix remains noise free. The regression problem in this dataset is to estimate $\hat{f}$ from input matrix of steady state observations, $\mathbf{X}$ and the corresponding response vector y_i, for $i = 1, 2$ and 3.

$$\hat{\mathrm{y}}_i = \hat{f}(\mathbf{X})$$

5.1.2. Dynamic CSTR Simulation

An isothermal CSTR process, with an external heat exchanger unit, was simulated using MATLAB™ Simulink. An exothermic first order reaction, A $\rightarrow$ B, takes place inside a reactor [129]. Two streams are fed to the reactor, reactant, and solvent streams denoted as $\mathrm{F_A}$ and $\mathrm{F_S}$, respectively. Isothermal operation is achieved via a temperature controller, which manipulates the coolant flowrate, $\mathrm{F_C}$, and the reactor height is controlled with a level controller which sets the reactor effluent flowrate, $\mathrm{F_{out}}$. In order to make the simulated system similar to an industrial process, and evaluate the proposed statistical methods in the current thesis within a more rigorous experimental setting, the CSTR system is extended to include a jacketed heat exchanger unit in which the solvent stream is heated before it enters the reactor.

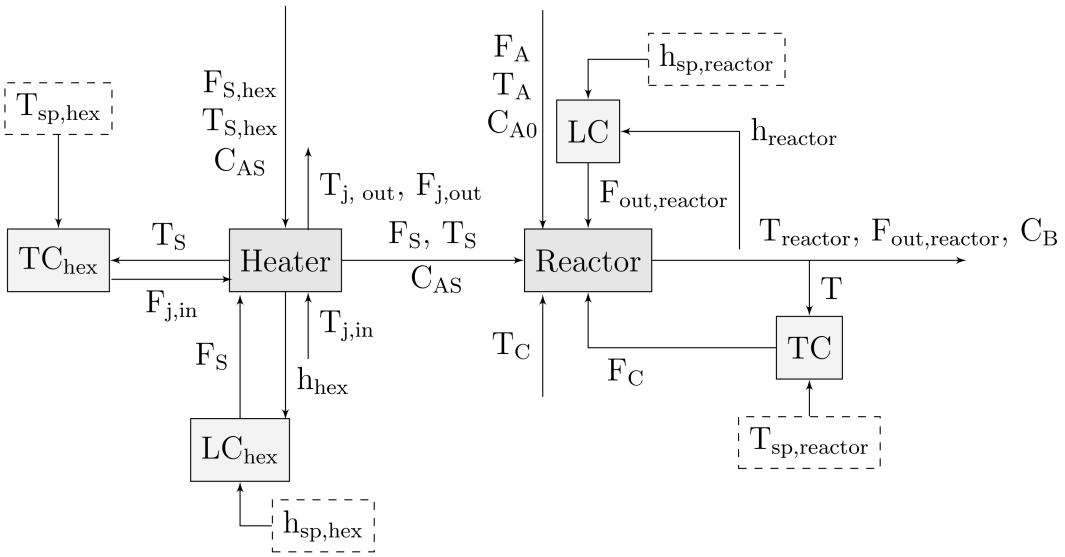

Figure 5.4. Flow diagram of the reactor system simulated in Simulink.

Temperature of the solvent flow into the reactor is controlled by manipulating the flowrate of the jacket fluid, and the liquid height inside the heat exchanger is controlled with a level controller which manipulates the outlet flowrate (Figure 5.4).

Here, the subscripts hex and reactor are used to differentiate between variables contained in two processes, the heater and reactor, respectively; for process variables shared between these two units the subscript is dropped. TC and LC are temperature and level controllers in the reactor unit, whereas TC_{hex} and FC_{hex} are for the heater. Set points of controllers are denoted with the subscript, sp, and drawn with dashed lines. Reactor unit was modeled with the following material and energy balances from Yoon and MacGregor [129]:

$$\frac{dC_A}{dt} = \frac{F}{V}C_{A0} - \frac{F}{V}C_A - k_0 e^{-\frac{E}{RT}}C_A \tag{5.1}$$

$$V\rho_L C_{p,L}\frac{dT}{dt} = \rho_L C_{p,L} F(T_0 - T) - \frac{aF_C^{b+1}}{F_C + aF_C^b/2\rho_L C_{P,L}} + (T - T_{C,\,in}) - \Delta H_{rxn} V - k_0 e^{-\frac{E}{RT}}C_A \tag{5.2}$$

Mass and energy balance equations of the heat exchanger model are as follows:

$$q = UA(T_{j,out} - T_{out}) \tag{5.3}$$

$$\rho_S C_{p,S} V_S \frac{dT_{out}}{dt} = F_{in}(T_{in} - T_{out}) + q \tag{5.4}$$

$$\rho_j C_{p,j} V_j \frac{dT_{j,out}}{dt} = F_j(T_{j,in} - T_{j,out}) - q \tag{5.5}$$

$$A_{cont} \frac{dF_{out}}{dt} = F_{in} - F_{out} \tag{5.6}$$

U_{hex}, and A_{hex} are heat transfer coefficient and area, respectively. ρ, V, C_p and F are density, volume, heat capacity and flowrate, and the subscripts L, S, and j denote the process, solvent and the heating fluids, respectively. Settings for physical constants in material and energy balances, and other simulations parameters are listed in Table 5.3. Controller settings used in the simulations can be found in Table 5.2.

It is desired to predict the outlet concentration of the reactor product, C_B. In total, 19 process variables, such as inlet flowrate, reactor height, and temperature, are sampled every 10 minutes, whereas the response variable is measured once in every 12 hours according to our scenario. Process variables, and their nominal operating conditions are tabulated in Table 5.4; Gaussian measurement noise with zero mean and variance σ_e^2 was added to all variables to mimic real life process measurements.

Table 5.2. Controller settings for the CSTR simulation.

Controller	**Proportional**	**Integral**	**Range**
TC	-0.3	-0.1	[-0.9, 9]
LC	-8	-10	[-14.5, 55]
TC_{hex}	-0.5	-0.1	[-0.7, 2.2]
LC_{hex}	-0.01	-0.01	[-0.85, 1.1]

Table 5.3. Physical constants and parameter settings for the simulation.

Parameter	**Description**	**Value**
$A_{reactor}$	Base area of reactor	3 m^2
ρ_L	Density of process fluid	1×10^6
E/R	Activation energy term	8.3301×10^3 K
$C_{p,L}$	Heat capacity of process fluid	1 cal/gK
a	-	1.678×10^6
b	-	0.5
k_0	Rate constant	1×10^{10} m^3/kmol.min
ΔH_{rxn}	Enthalpy of reaction	-1.3×10^7 cal/kmol
ρ_S	Density of solvent stream	1×10^3 kg/m^3
$C_{p,S}$	Heat capacity of solvent stream	1×10^3 cal/gK
ρ_j	Density of jacket fluid	0.83×10^3 kg/m^3
$C_{p,j}$	Heat capacity of jacket fluid	0.5×10^3 cal/gK
A_{cont}	Base area of the container	1 m^2

Table 5.4. Process variables of the CSTR simulation.

Variable	**Description**	**Nominal Value**	$\sigma_{\mathbf{e}}^{\mathbf{2}}$
$F_{out,\ reactor}$	Outlet flow rate	1 m^3/min	1×10^{-4}
$h_{sp,\ reactor}$	Height set point	3 m	-
$h_{reactor}$	Height	3 m	1×10^{-4}
F_C	Coolant flow rate	15 m^3/min	1×10^{-2}
$T_{sp,\ reactor}$	Temperature set point	370 K	-
$T_{reactor}$	Temperature	370 K	4×10^{-4}
T_C	Coolant temperature	365 K	2.5×10^{-3}
F_A	Reactant inlet flow rate	0.1 m^3/min	1×10^{-6}
T_A	Reactant inlet temperature	370 K	1×10^{-2}
F_S	Solvent inlet flow rate	0.9 m^3/min	2×10^{-6}
$h_{sp,\ hex}$	Height set point	1.2 m	-
h_{hex}	Height	1.2 m	5×10^{-6}
$F_{j,\ in}$	Jacket fluid flow rate	0.77 m^3/min	5×10^{-6}
$T_{sp,\ hex}$	Solvent temperature set point	370 K	-
T_S	Solvent temperature	370 K	1×10^{-2}
$T_{j,\ in}$	Jacket fluid inlet temperature	460 K	4×10^{-6}
$T_{j,\ out}$	Jacket fluid outlet temperature	415 K	4×10^{-4}
$F_{S,\ hex}$	Solvent inlet flow rate	0.9 m^3/min	4×10^{-6}
$T_{S,\ hex}$	Solvent inlet temperature	342 K	4×10^{-6}
C_B	Outlet product concentration	1.86 kmol/m^3	2.5×10^{-5}

All input variables, except for the inlet concentration of A in the reactant stream, were generated based on the following stochastic first order autoregressive (AR) dynamics [129]:

$$u_t = \phi_u u_{t-1} + a_u \tag{5.7}$$

Here, u stands for input variable, and the subscripts correspond to its time stamp. ϕ is the AR coefficient, and a_u is the random disturbance term, specific to each input. Disturbances in the controller setpoints were modeled as Bernoilli trials with succes probability p_u at each time t. Simulation parameters for the input variables are tabulated in Table 5.5. Note that AR coefficients of simulation inputs are close to unity, input space is thoroughly spanned when such large coefficients are used. This results in close-to-nonstationary dynamics in input variables, and is used to imitate virtual concept drift which is present in real process data. Trajectories of representative process variables, obtained as a result of the employed disturbance scenarios and controller settings are shown in Figure 5.5.

Table 5.5. Simulation parameters of input variables.

Input Variable	$\boldsymbol{\phi_u}$	$\boldsymbol{\sigma_u}$	$\boldsymbol{p}$
$h_{sp,\ reactor}$	0.8500	0.2000	-
$T_{sp,\ reactor}$	0.6000	0.5000	0.9995
T_C	0.9990	0.0100	-
F_A	0.9900	0.0020	-
T_A	0.9995	0.1000	-
$h_{sp,\ hex}$	0.8500	0.0500	0.999
$T_{sp,\ hex}$	0.8000	8.5000	0.9995
$T_{j,\ in}$	0.9990	0.2375	-
$F_{S,\ hex}$	0.9000	0.0190	-
$T_{S,\ hex}$	0.9990	0.2375	-

Each simulation run lasts 350 virtual days, corresponding to approximately one year of operation. Response variable, C_B is sampled once in every 12 hours; hence each run yields 700 labeled samples, 300 of which were designated training set, and the last 400 samples were reserved for testing. The input matrix, $\mathbf{X}$, comprises of 19 process variables tabulated (Table 5.4), and their n additional lagged values.

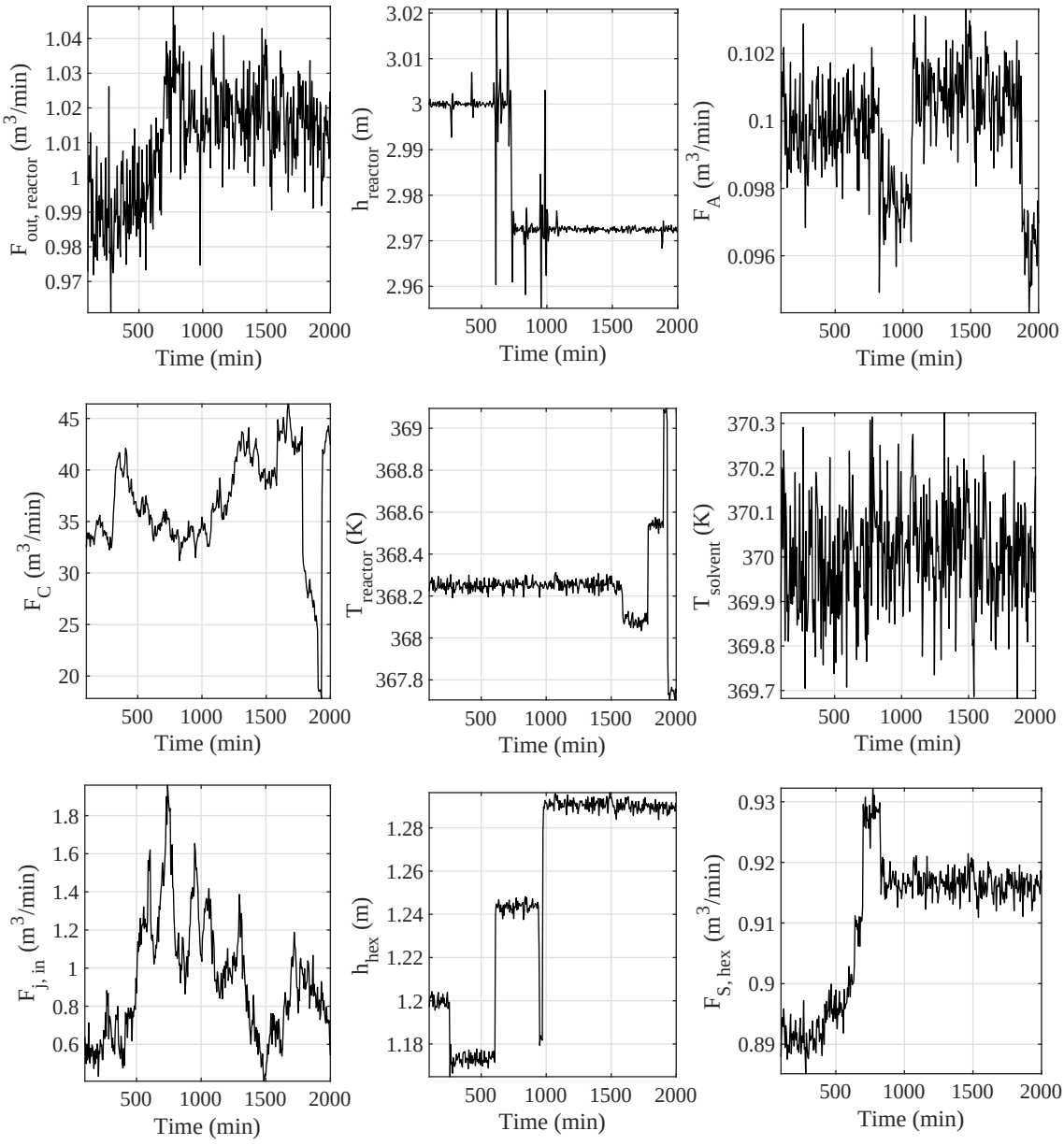

Figure 5.5. Example trajectories of various process variables with controlled variables in the middle, corresponding manipulated variables and disturbances on the left and right hand side, respectively.

Let $\mathbf{X}_{base}(t)$ denote the 300×19 matrix of process variables, associated with the output variable sampled at time t, and TS_x is the sampling time of process variables, set as 10 minutes in this case. Note that only some process variables, such as temperature and level set points, are sampled at the same time, t, as the output. Then the model input with n lags is defined as $\mathbf{X}^{(n)}(t) \coloneqq [\mathbf{X}_{base}(t), \mathbf{X}_{base}(t - TS_x), ..., \mathbf{X}_{base}(t - nTS_x)]$. Thus the usual regression problem can be restated as an $(n+1)^{th}$ FIR model, emphasizing the temporal relationship in dynamic modeling as follows:

$$\hat{y}(t) = \hat{f}(\mathbf{X}^{(n)}(t)) \tag{5.8}$$

Five different concept drift (CD) scenarios are realized on the CSTR system. Sets of disturbance models differ in the way the input variable C_{A0} was simulated, recall that C_{A0} is not included in the set of predictor variables for soft sensor modeling. The first set, denoted as S0, is the default operation scheme in which C_{A0} remains essentially constant at its nominal operating point, 19.1 kmol/m^3. The second (S1) and third (S2) set of disturbances were designed so as to resemble abrupt changes, which occur in real life process operations. For the second set, C_{A0} was modeled as a collection of step inputs with varying magnitudes of small sizes. S1.1 and S1.2 correspond to cases in which the frequency of step inputs is slow and fast, respectively. In the third model, C_{A0} was again simulated as multiple steps inputs, all of which have the same size of large magnitudes. Similarly, S2.1 and S2.2 denote simulations in which C_{A0} is perturbed at two different frequencies. In the last the simulation setting (S3), the aim was to simulate gradual changes in process inputs. Hence, C_{A0} was modeled as a ramp change. Both in S3.1 and S3.2, the slope of the ramps remains constant, S3.1 is made up of ramp inputs of varying lengths only, and S3.2 is a mixture of ramps with occasional step changes. On the other hand, in S3.3 and S3.4 both the sign and the magnitude of the slope varies along the input trajectory; and the frequency of ramp change is faster in S3.4 than that in S3.3. Concept drift models (CDMs) are listed in Table 5.6, along with the corresponding descriptions of various types of real concept drifts they emulate. Virtual concept drift, on the other hand, is present in all scenarios

since input variables were generated from autoregressive processes. Representative C_{A0} trajectories for different concept drift scenarios can be seen in Figure 5.6.

Table 5.6. Different concept drift scenarios simulated using the CSTR system.

Number	**Type of Real Concept Drift**
S0	-
S1.1	Abrupt changes at multiple operating points with low frequency
S1.2	Abrupt changes at multiple operating points with high frequency
S2.1	Abrupt changes between two random operating points with low frequency
S2.2	Abrupt changes between two random operating points with high frequency
S3.1	Gradual changes between two random operating points
S3.2	Gradual changes mixed with abrupt shifts of operating points at moderate frequency
S3.3	Gradual changes with abrupt shifts of low frequency in the direction of change
S3.4	Gradual changes with abrupt shifts of high frequency in the direction of change

5.2. Real Process Data

Synthetic data is quite useful for elucidating predictive methods, and analyzing results in a systematic approach, but proposed novel learning algorithms should be tested on real data to assess their predictive performance since the suggested method is ultimately aimed to be used in a real life setting. The final dataset used in evaluating

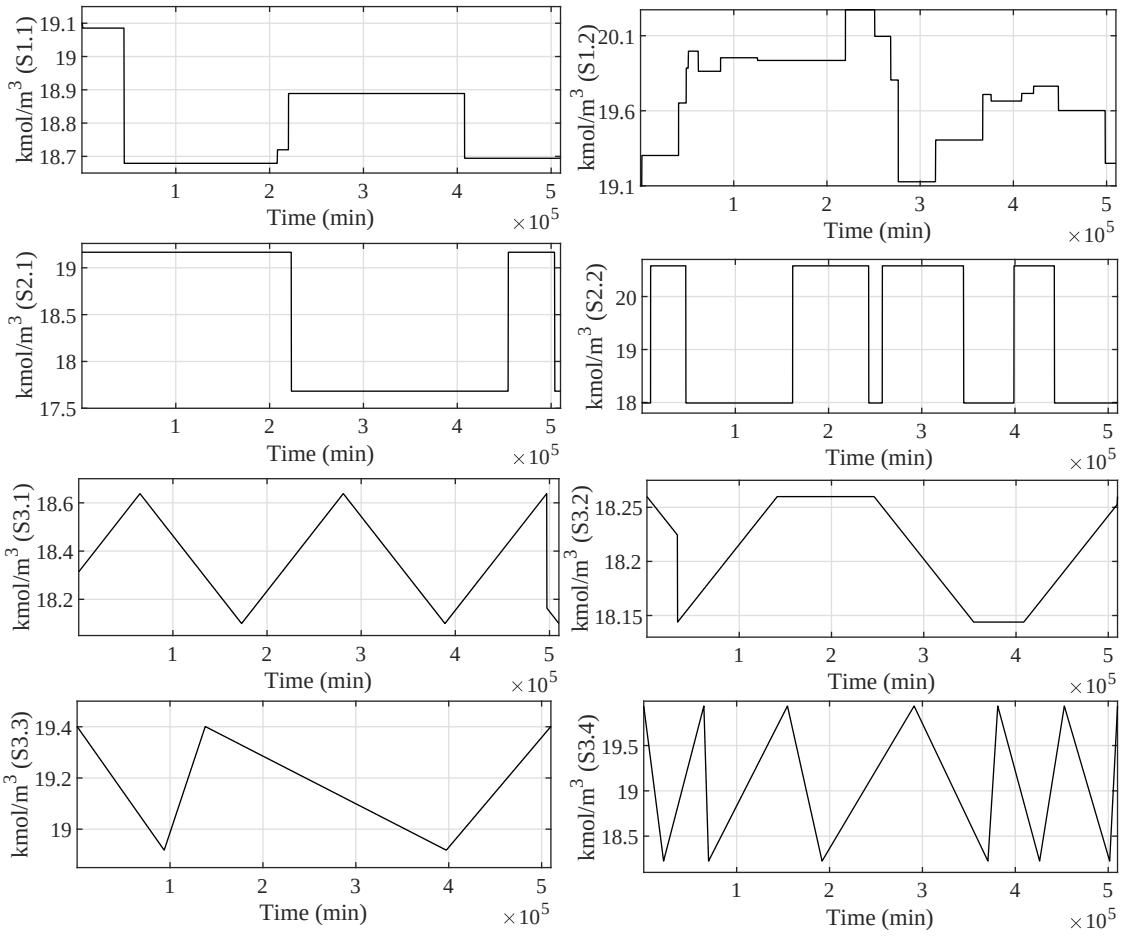

Figure 5.6. Trajectories of C_{A0} from different concept drift scenarios.

statistical learning methods is the debutanizer column dataset, provided by Fortuna *et al.* [2], from a real industrial distillation process, as described below.

5.2.1. Debutanizer Column

The debutanizer column is one of the units in a desulfurization process, along with a deisopentanizer column; bottoms product from the debutanizer column is fed to the deisopentanizer column. Two gas chromatographs are placed, one on the LP gas splitter and the other on the deisopentanizer column (Figure 5.7). There are two main concerns in the process: (i) to obtain maximum amount of stabilized gasoline (C5) from the overheads of the LP has splitter, and (ii) to keep the butane (C4) fed to the naphta splitter from the debutanizer column bottoms at minimum. Two soft sensors, N1 and N2, are needed to address these issues. In this thesis, only the first issue, i.e. prediction of the butane content, is of concern.

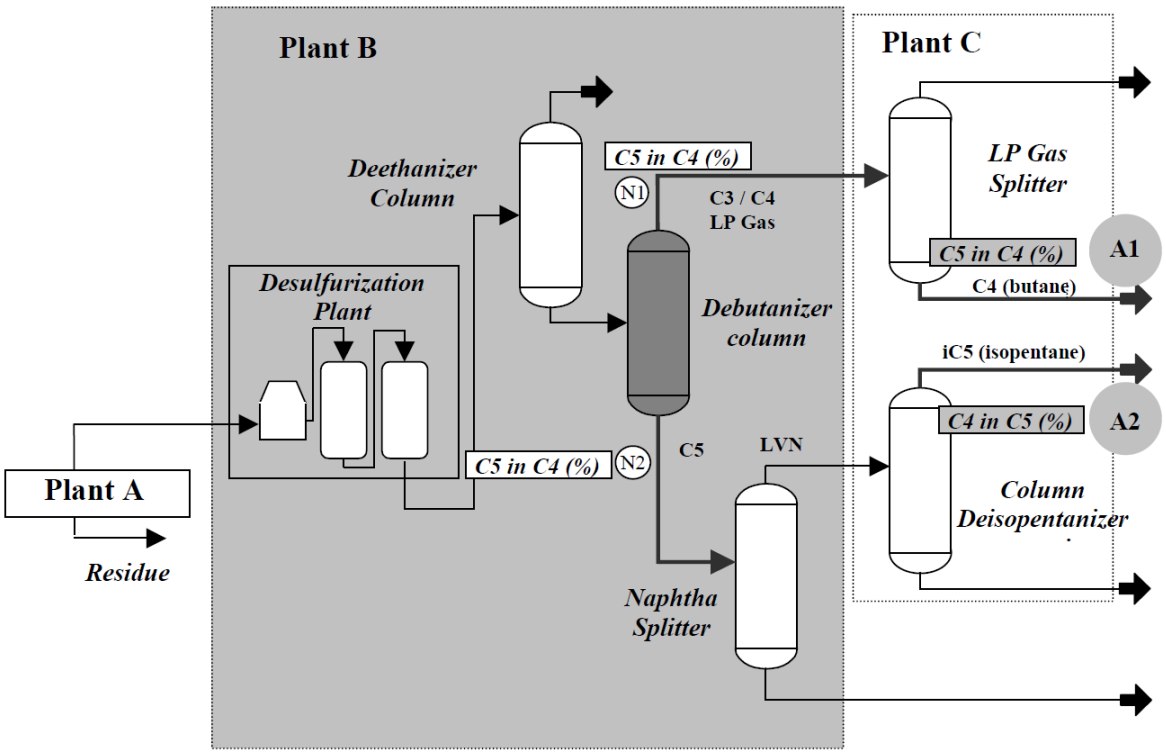

Figure 5.7. Flowsheet of the overall process, including the debutanizer and the immediate upstream and downstream units [2].

In Figure 5.7, A2 is placed on the overhead products of the deisopentanizer column to measure the bottoms concentration of butane. It is presumed that all butane detected in the deisopentanizer overhead stream comes from the debutanizer bottoms effluent, designated as the quality variable. Butane concentration can be measured with a delay of 45 minutes, since it takes about 30 minutes for butane content to reach the overhead stream, and there is an additional time delay of 15 minutes to make a measurement using chromatography. Thus, it is desired to place a soft sensor in N2 to provide online predictions of butane concentration for control and monitoring purposes.

The debutanizer column is shown in more detail in Figure 5.8; gray colored boxes with labels correspond to seven process variables, selected by Fortuna *et al.* [2] as inputs for the soft sensor, and all variables are measured with a sampling time of 6 minutes. Table 5.7 lists the process variables along with their names.

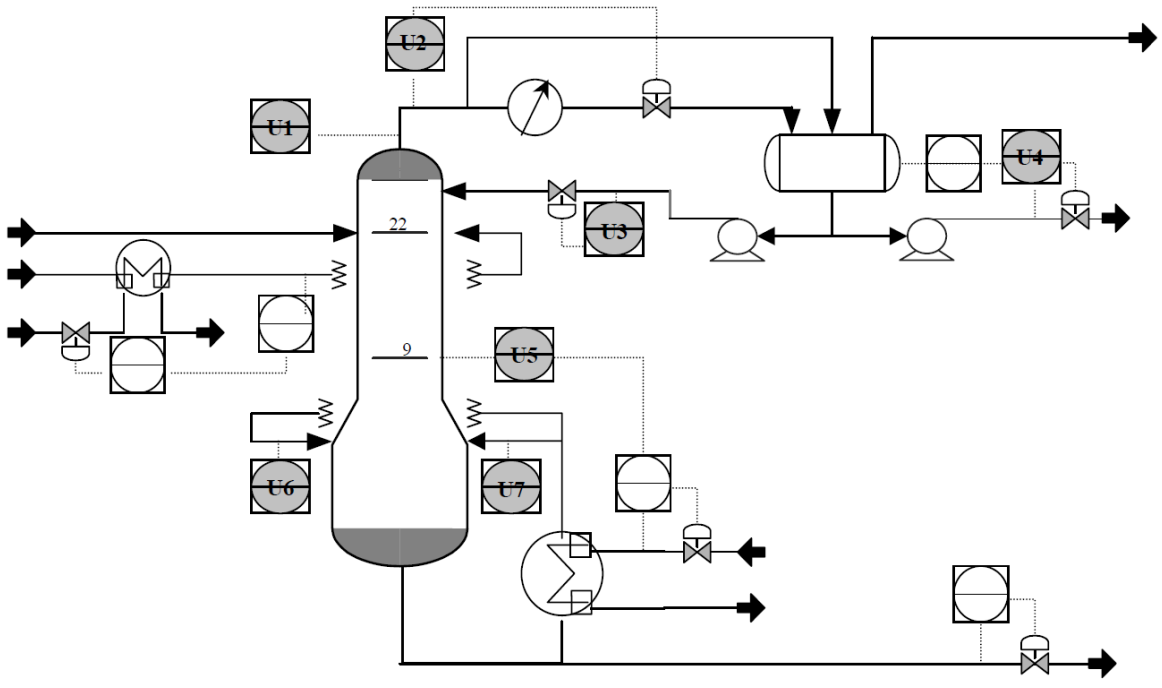

Figure 5.8. A detailed diagram of the debutanizer column, and the process variables measured [2].

Table 5.7. Process variables of the debutanizer column.

Variable	**Name**
1	Top tempeature
2	Top pressure
3	Reflux flow
4	Flow to next process
5	6^{th} tray temperature
6	Bottom temperature
7	Bottom temperature

Dataset consists of 2394 measurements, seven input variables and one output variable. No additional preprocessing was performed on the data, since it had already been analyzed, and processed for outliers by the author [2]. There is high collinearity among variables 6 and 7, as indicated by the correlation map in Figure 5.9; measurements of these two temperature values are located in close proximity (Figure 5.8).

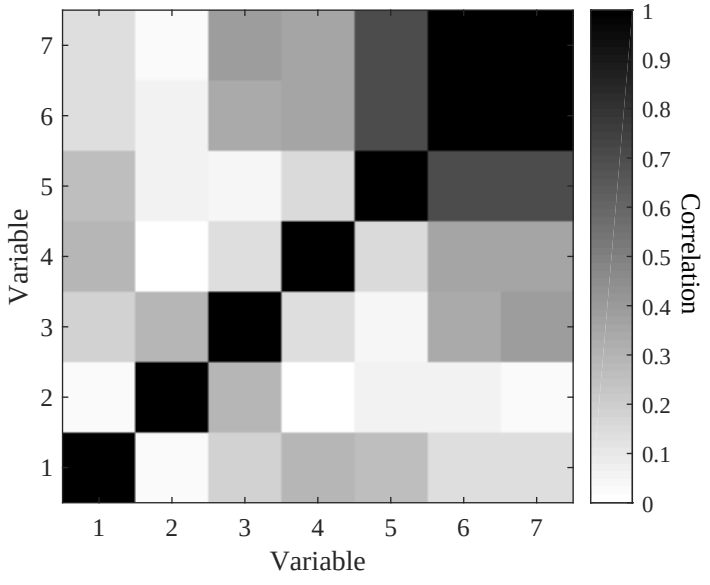

Figure 5.9. Correlation map of process variables determined from all measurements.

Trajectories of the process variables reveal the nonstationarity of the process, e.g. level and frequency content of measurements of process variables 1, 2, 4 and 6 show significant variation during the course of data collection period in Figure 5.10.

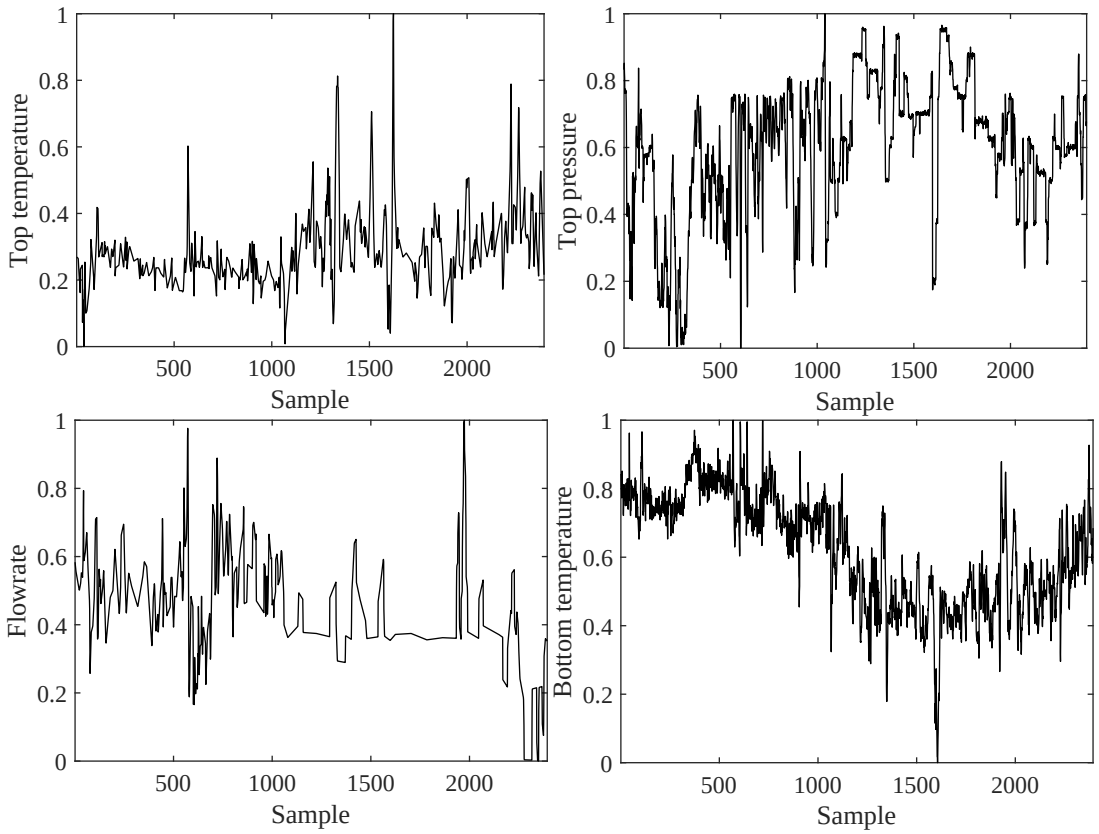

Figure 5.10. Trajectories of some example process variables.

6. RESULTS

In this chapter, we propose and elucidate various soft sensor design strategies in three sections. Each section covers a different issue, significant in soft sensor design, but it should also be noted that these issues are not entirely mutually exclusive. In the first section, a systematic approach for statistical modeling of a steady state process with a large number of collinear predictors is employed using various feature selection methods in conjunction with linear and nonlinear models. In the second section, two different statistical modeling methods, PLS and RVM, are compared under batch learning, and an online prediction scenario in the presence of concept drift. It will also be shown that RVM is a viable alternative to PLS for dynamic process modeling due its sparse nature, and superior predictive performance. The last section is devoted entirely to adaptive learning for soft sensor maintenance; and we propose multiple novel adaptive mechanisms to address the need for soft sensor maintenance.

6.1. Soft Sensor Design for a Steady State Crude Distillation Unit

In this section, it is aimed to design a soft sensor for a CDU at steady state operating conditions, hence data ($DS_{1,int}$, $DS_{1,ext}$, $DS_{2,int}$ and $DS_{2,ext}$) from Section 5.1.1 is to be used. It should be noted that the subscripts 1 and 2 indicate the generative model of data, while subscripts int and ext correspond to interpolation and extrapolation predictions, respectively. The interpolation scenario is presumed to mimic prediction when the process is operating at regular conditions, the extrapolation set, on the other hand, imitates a virtual drift scenario.

A total of 37 predictive models, combining various feature selection, and linear/nonlinear modeling tools outlined in Section 3 were constructed. The entire list of learners (or estimators), and the number notation which is used to differentiate the models are shown in Table 6.1. Models were trained on both the original predictor matrix, and an extended version augmented with squares of predictors, denoted by

the subscript "SqrPred" throughout this section. The only preprocessing done prior training models was autoscaling, and removing constant columns of data, thus the dimension of input matrix is 300×70 if only the first order predictors are used, and 300×140 with the squared terms added.

Table 6.1. Estimators for the steady state CDU.

Model	**Model**	**Model**	**Model**
1	Least squares	7.1	RVM
2.1	Ridge regression	7.2	$\text{RVM}_{\text{SqrPred}}$
2.2	$\text{Ridge regression}_{\text{SqrPred}}$	8.1	Lasso
3.1	Bayesian LR	8.2	$\text{Lasso}_{\text{SqrPred}}$
3.2	$\text{Bayesian LR}_{\text{SqrPred}}$	9.1	$\text{PLS}_{\text{QuadIn}}$
4.1	PLS	9.2	$\text{FS}^{\text{cal}} + \text{PLS}_{\text{QuadIn}}$
4.2	$\text{PLS}_{\text{SqrPred}}$	9.3	$\text{Red}_{\text{MI}} + \text{PLS}_{\text{QuadIn}}$
4.3	PLS_{Sc}	10	Kernel PLS
4.4	$\text{PLS}_{\text{SqrPred,Sc}}$	11.1	ANN
4.5	PLS_{Cr}	11.2	ANN_{Cr}
4.6	$\text{PLS}_{\text{SqrPred,Cr}}$	12.1	FS + ANN
5.1	FS^{cal} + PLS	12.2	$\text{FS} + \text{ANN}_{\text{Cr}}$
5.2	$\text{FS}^{\text{cal}} + \text{PLS}_{\text{SqrPred}}$	12.3	$\text{Red}_{\text{corr}} + \text{ANN}$
5.3	FS^{val} + PLS	12.4	$\text{Red}_{\text{corr}} + \text{ANN}_{\text{Cr}}$
5.4	Red_{corr} + PLS	12.5	$\text{Red}_{\text{MI}} + \text{ANN}$
5.5	$\text{Red}_{\text{corr}} + \text{PLS}_{\text{SqrPred}}$	12.6	$\text{Red}_{\text{MI}} + \text{ANN}_{\text{Cr}}$
6.1	JK + PLS	13	$\text{FS}^{\text{cal}} + \text{RVM}_{\text{local}}$
6.2	JK + $\text{PLS}_{\text{SqrPred}}$		
6.3	$\text{JK} + \text{PLS}_{\text{Cr}}$		
6.4	$\text{JK} + \text{PLS}_{\text{SqrPred,Cr}}$		

In Table 6.1, the first column gives an approximate classification of the learners, i.e., a separation into "family" of learners. Models 1-4 correspond to batch modeling techniques, in which full input matrix is used; models 5-6 are the variants of PLS used in conjunction with different feature selection methods, and models 7-8 are sparse models based on more recent regression techniques. From another perspective, models 1-4, 7 and 8 are grouped as linearly parametrized learners, while models 9-12 are nonlinear learners constructed using nonlinear PLS and ANN methods. The final learner, model 13, stands out as the sole local learner with an adaptation mechanism in this list.

A parameter tuning step is incorporated within the training phase of all models; number of components in latent variable models, hidden units in NNs and number of predictors in learners in which feature selection is incorporated, are among examples of parameters tuned during this tuning step. All parameters are optimized using 10fold CV with 20 repetitions; model parameters and their corresponding ranges used during CV for tuning are tabulated in Table 6.2. Along with conventional PLS models,

Table 6.2. Model parameters and their ranges in which CV was performed with a grid search.

Model	**Type of Model Parameter**	**Range**
2	Regularization	$[10^{-2}, 1]$
4, 6, 9	L	[1, 25]
5	L and p	[1, 25] and [1, 30]
8	Regularization	$[10^{-6}, 1]$
10	Kernel parameter and L	$[10^{-2}, 10]$ and [1 25]
11	K	[1, 10]
12	K and p	[1, 5] and [1, 30]
13	p_{N} and NS	[1, 10] and [10, 200]

a number of variants of PLS is utilized. The subscript Sc in PLS indicates that a different scaling scheme, in which the variance of input columns are estimated from

X residuals obtained during the NIPALS algorithm for scaling, is employed [51]. In PLS_{Cr}, an ensemble PLS model is constructed by training base learners in the ensemble on different sets of data generated via crogging (cross validation averaging) [105]. In crogging, entire training data is partitioned into mutually exclusive sets in which separate base models are trained. Since kfold CV was already used for parameter tuning in the current study, crogging was just an additional step of averaging all the models estimated from separate calibration sets.

A backward selection method based on jack-knifed parameter estimates in PLS is used in JK+PLS [98]. A t-test, with significance $\alpha = 0.1$, is applied to all regression coefficients, and insignificant variables are eliminated in this method. To ease computation, for $p > 30$ instead of removing variables one by one, the number of variables removed at each iteration starts at 20% of p, and decreases exponentially until unity.

Various feature selection algorithms are used with PLS and ANN; FS^{cal} is performed by inserting the predictor which has the highest partial correlation with the response variable in a stepwise manner in the calibration set of the inner CV loop. In FS^{val}, on the other hand, calibration set is further partitioned to obtain a validation part within the same set; predictors are included with respect to the MSE of corresponding LS estimators on the validation set. The subscripts, cal and val, thus differentiate between two different metrics used for determining a search route, fitting errors and validation errors respectively. Initially, backward selection was also applied in conjunction with PLS models; however, its performance was inferior to other PLS models with feature selection, and so it was not pursued.

MRMR approach to feature selection is employed in two different formulations with the following criteria for predictors, $i = 1, 2, \ldots, p$:

$$\max RC_i, \quad RC_i = \frac{\rho_i}{VIF_i^{0.75}} \tag{6.1}$$

$$\max RM_i, \quad RM_i = \frac{I_i}{VIF_i^{0.75}} \tag{6.2}$$

Here, ρ_i is the linear correlation coefficient, and I_i is the mutual information of the i^{th} predictor ($\mathbf{x}_i$) with $\mathbf{y}$, and VIF_i is the variance inflation factor of the same predictor computed as follows:

$$VIF_i = \frac{1}{\mathrm{R}^2_{-i}} \tag{6.3}$$

Here, R^2_{-i} is the R^2 value when the i^{th} predictor is regressed against all other predictors. VIF_i and I_i are raised to the power of 0.75 to decrease their weight relative to that of the correlation term in the numerator in Equation 6.1 and Equation 6.2, in which feature selection is employed, for all predictors satisfying the condition $VIF < 1,000$.

In RVM, a Gaussian likelihood is used as the model, all termination criterion used during iterative training procedure (changes in the logarithm of precision of coefficients, noise precision, and relevance factors) are set to 1×10^{-3}.

For $\mathrm{PLS}_{\mathrm{QuadIn}}$, a nonlinear PLS model with at most second order inner relations is formed. At each iteration of the NIPALS algorithm, a partial t-test on $\mathrm{b}_{\mathrm{a},2}$ in Equation 3.14 is conducted at significance $\alpha = 0.05$ to prevent inflation of variance resulting from additional parameters. With the first occurrence of a negative test result, the second order term is removed, and a linear inner relation is retained in subsequent iterations, since nonlinear relations with a significant effect on bias are expected to be manifested in low indexed components.

In Kernel PLS, mapping from original input variables to the latent variables is nonlinear, while the inner relation between the latent variables, input and the target scores, remain linear. Gaussian kernel is used for projecting inputs to a new feature space, and this mapping is applied within the NIPALS algorithm [54].

All ANN models in this section are essentially multilayer perceptrons (MLPs) using a single hidden layer and sigmoidal activation function. During the MLP training phase, early stopping is applied using a 80/20 partition; training is stopped once

it is observed that validation error has increased for 50 epochs. In ANN models with crogging, ensembles of ANN models with same number of neurons, but different initial values of weight parameters on different but overlapping calibration datasets were averaged over. This technique is theoretically based on the bias/variance decomposition, and has been suggested to lower prediction errors as it decreases variance of parameter estimates [130].

All models were optimized offline; and the test data were predicted in a batch manner, except for the model 13. In FS + $\text{RVM}_{\text{local}}$, the test set was treated as a data stream; sample-based predictions were made and labeled test points were appended to the training set. Thus, the model parameters, optimum neighborhood size and the number of neighborhood variables were determined offline on the original, unextended training set, but the local model was retrained for each query point.

Predictive performance of estimators were assessed using repeated double CV (rdCV) [131]. The inner loop was used for parameter tuning, and the outer loop, which consists of identical partitioning of observations for all models, was reserved for determining unbiased prediction error (PE) estimates ($\widehat{\text{PE}}$). The root mean squared error (RMSE) of predictions on the test set, defined as in Equation 6.4, were used to estimate PE.

$$\text{RMSE} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(\text{y}_i - \hat{\text{y}}_i)^2} \tag{6.4}$$

The outer loop was partitioned into four folds and repeated 10-15 times, yielding 50 RMSE values from test sets with a constant size of 75 observations, which were then averaged to obtain $\widehat{\text{PE}}_i$ in which i denotes three different response variables, y_1, y_2 and y_3. Average of $\widehat{\text{PE}}_i$s over these variables yields avg($\widehat{\text{PE}}$), an estimate of the expected PE ($E\{\widehat{\text{PE}}\}$). The "worst case scenario" criterion, defined as avg$\widehat{\text{PE}}$ + 2SD$\widehat{\text{PE}}$, where SD stands for standard deviation, was used to evaluate estimators based on their worst prediction performance observed in different partitions of the data. $\text{DS}_{1,\text{ext}}$ and $\text{DS}_{2,\text{ext}}$,

were reserved solely for testing; none of the models were retrained on these data, hence estimators from $\mathrm{DS}_{1,\mathrm{int}}$ and $\mathrm{DS}_{2,\mathrm{int}}$ were used. Furthermore, two different training sets were used; one with 50 and another with 225 observations, corresponding to low and high training set sizes respectively.

Two metrics are defined to evaluate predictive efficiencies of models as follows:

$$\mathrm{Eff}_i(k,l) = \frac{\min\limits_{j\in S_P}\left(\mathrm{avg}\widehat{\mathrm{PE}}_j(k,\{N=225,\mathrm{D}_{int}\})\right)}{\mathrm{avg}\widehat{\mathrm{PE}}_i(k,l)} \tag{6.5}$$

$$\mathrm{Eff}'_i(k,l) = \frac{\min\limits_{j\in S_P}\left(\mathrm{avg}\widehat{\mathrm{PE}}_j + 2\mathrm{SD}\widehat{\mathrm{PE}}_j(k,\{N=225,\mathrm{D}_{int}\})\right)}{\mathrm{avg}\widehat{\mathrm{PE}}_i + 2\mathrm{SD}\widehat{\mathrm{PE}}_i(k,l)} \tag{6.6}$$

Here S_P denotes the set of predictive models listed in Table 6.1. k is an element of S_K which is the set of two different modes of simulations (different schemes of perturbation to input variables), and l is an element of S_L which contains training sets of two sizes, 50 and 225; $i = 1, 2, ...37$ represents the models in Table 6.1. An $\mathrm{Eff}_i(k,l)$ value close to unity indicates that the i^{th} predictor is among the best for the dataset (k,l), since its prediction performance is on par with the best predictor for $N = 225$ on $\mathrm{D}_{\mathrm{int}}$. Consequently, an overall and a worst case scenario evaluation can be done by averaging, and minimizing $\mathrm{Eff}_i(k,l)$ over S_K to obtain $\mathrm{avgEff}_i(l)$, and $\mathrm{minEff}_i(l)$, respectively. These two measures of efficiency were used as direct indicators of the relative generalization capability of the different predictive models on the CDU simulation under different operating conditions.

6.1.1. Comparison of Linear Models and Variants of PLS

All modes of the process are perturbed in DS_1, i.e. the input space is well-spanned thus the prediction problem is a relatively easy one, in which only process nonlinearities can be challenging to deal with for the conventional linear models. In Figure 6.1 the top two plots show interpolation results for $N = 225$ and $N = 50$,

similarly the extrapolation performances are plotted in bottom two figures for $N = 225$, and $N = 50$, respectively. Bars on the left and right correspond to two definitions of efficiency as in Equation 6.5 and 6.6. Inclusion of squared terms in the model has adverse effects on all learners, with PLS (model 4) and RR (model 2) being affected the most.

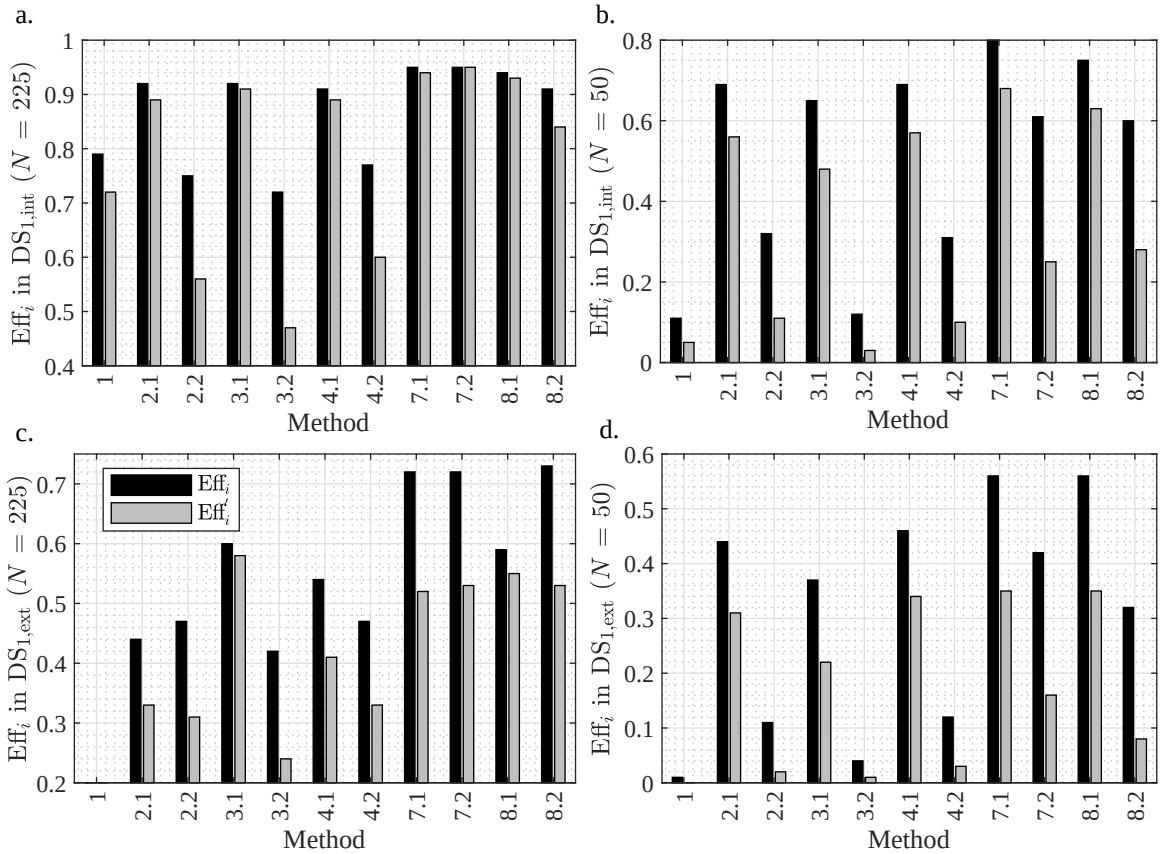

Figure 6.1. Interpolation (a, b), and extrapolation (c, d) efficiencies, Eff_i (black bars) and Eff'_i (gray bars) of linear models in DS_1.

However, RVM and Lasso (models 7 and 8, respectively), both of which perform discrete feature selection during training, appear to combat this problem much better than other linear models in general. Moreover, performance of Lasso model is drastically improved by adding squared predictors in Figure 6.1c. It is seen that prediction efficiencies drop down to 0.5-0.6, thus high collinearity among predictors cannot be handled by shrinking parameters (RR in model 2.1 and 2.2), or feature extraction

(PLS models 4-5), especially when the size of the training set is much lower than the total number of predictors (Figures 6.1b and 6.1d), but sparse regression techniques yield superior performance in an extrapolation scenario.

In the second simulation scenario (DS_2), performance of both RR and PLS are on par with those of RVM and Lasso (Figure 6.2). It is also seen that prediction efficiencies of all learners (without using squared predictors) interpolating in a latent subspace is much higher than interpolation in the whole input space. While all learners are affected by sample size, RVM and Lasso seem to be affected less, compared to PLS and RR. It should also be noted that Lasso is the only learner which has increased its efficiency using squared predictors.

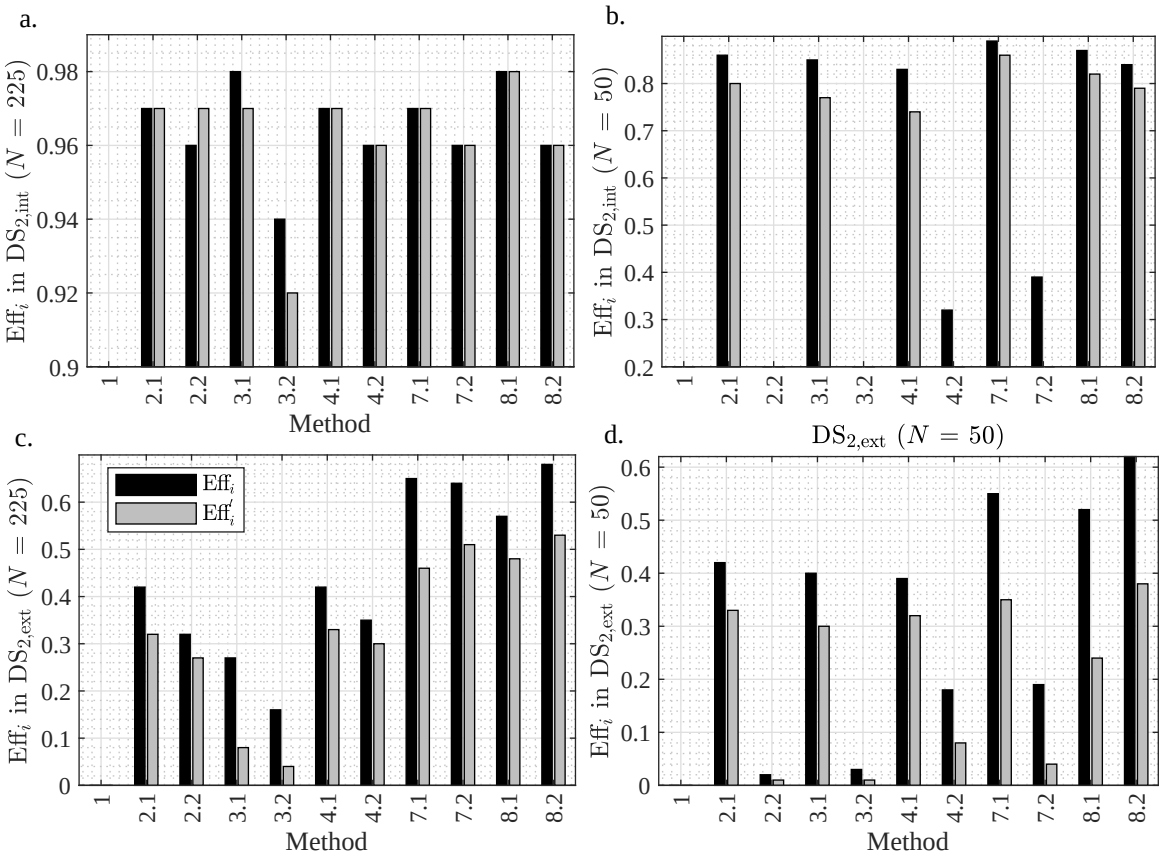

Figure 6.2. Interpolation (a, b), and extrapolation (c, d) efficiencies, Eff_i (black bars) and Eff'_i (gray bars) of linear models in DS_2.

Performance of PLS variant methods are shown in Figure 6.3 in which vertical lines are drawn to visually separate PLS models as conventional PLS and its variants, PLS models in conjunction with feature selection, and nonlinear versions of PLS from left to right. Neither scaling, nor crogging PLS models provide any major improvements in this case as they increase the efficiency only marginally over conventional PLS models, if any at all. Crogging has almost no benefit over other PLS variants; conventional PLS, being an already stable estimator, shows no further improvement with model averaging. In JK + PLS, on the other hand, the efficiencies are way lower, especially at small training sizes. Backward elimination procedure, as applied in JK + PLS, is usually not recommended for problems like this, i.e. when $N < p$ [6]. When feature selection starts with the full model, there is simply too many predictors, and not enough data to estimate their coefficients properly.

Feature selection was incorporated in models shown in the mid section of both plots (model 5 in Figure 6.3), and in the nonlinear section (model 9 in Figure 6.3), all of which stand out as the best performing PLS based models. An additional inner loop for determining feature selection trajectory through CV is necessary in FS^{val}; however, it can be concluded that this approach does not improve the predictive performance of the final model. In addition, further partitioning of data to smaller size might have a negative effect as seen in the efficiencies of FS^{val} as opposed to those of FS^{cal}. Inclusion of a redundancy check, especially when the training set is small, yields the best predictive performances overall. Both of the nonlinear PLS models, $\text{PLS}_{\text{QuadIn}}$ and Kernel PLS, are inefficient in the absence of any feature selection, and provide no real benefit over the conventional PLS. Using MI in conjunction with a redundancy check in $\text{PLS}_{\text{QuadIn}}$ results in the highest efficiencies, overall.

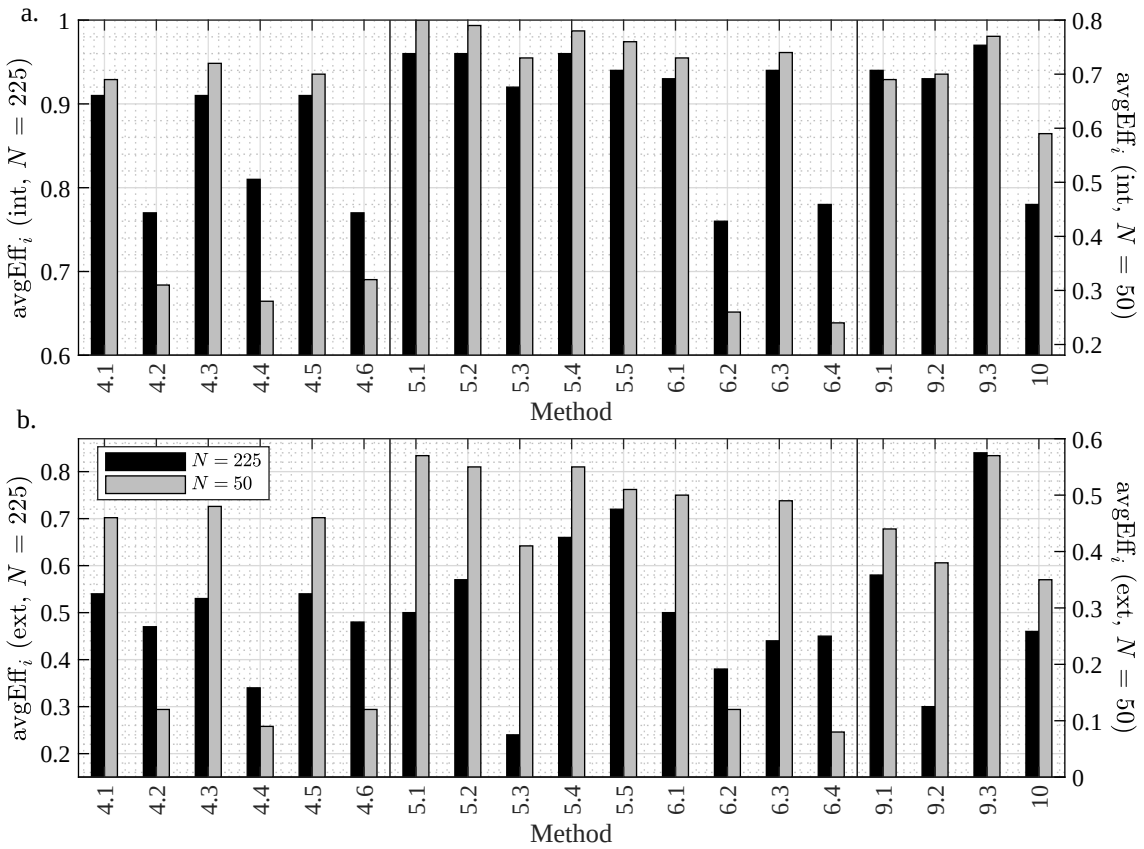

Figure 6.3. Average of efficiencies, avgEff_i, of PLS variants in $\text{DS}_{\text{1-2,int}}$ (a), and $\text{DS}_{\text{1-2,ext}}$ (b) for $N = 225$ (left axis) and $N = 50$ (right axis).

Optimum number of parameters, i.e. PLS components and predictors, are averaged over all results, and plotted in Figure 6.4. Number of components selected lies in a rather narrow range among models, however, number of predictors retained varies significantly. JK + PLS, being a backward elimination algorithm, prefers a larger number of predictors in the model, almost the same as the number of observation when $N = 50$. FS, on the other hand, yields the most parsimonious models, which largely explains its high efficiency when the training set is small. Redundancy check successfully reduces variance in parameter estimates as it addresses the problem of collinearity among predictors directly; that way a larger portion of the inputs can be maintained within the model without the unnecessary inflation of variance. It may also be noted that feature selection methods, generally yield PLS models with smaller number of features and

PLS components, explaining the lower prediction accuracy of models obtained using $N = 50$ training sets. The unacceptably low prediction performance of Kernel PLS (model 10 in Figure 6.3) shows that feature selection is required so that Kernel PLS can be employed on data with high collinearities.

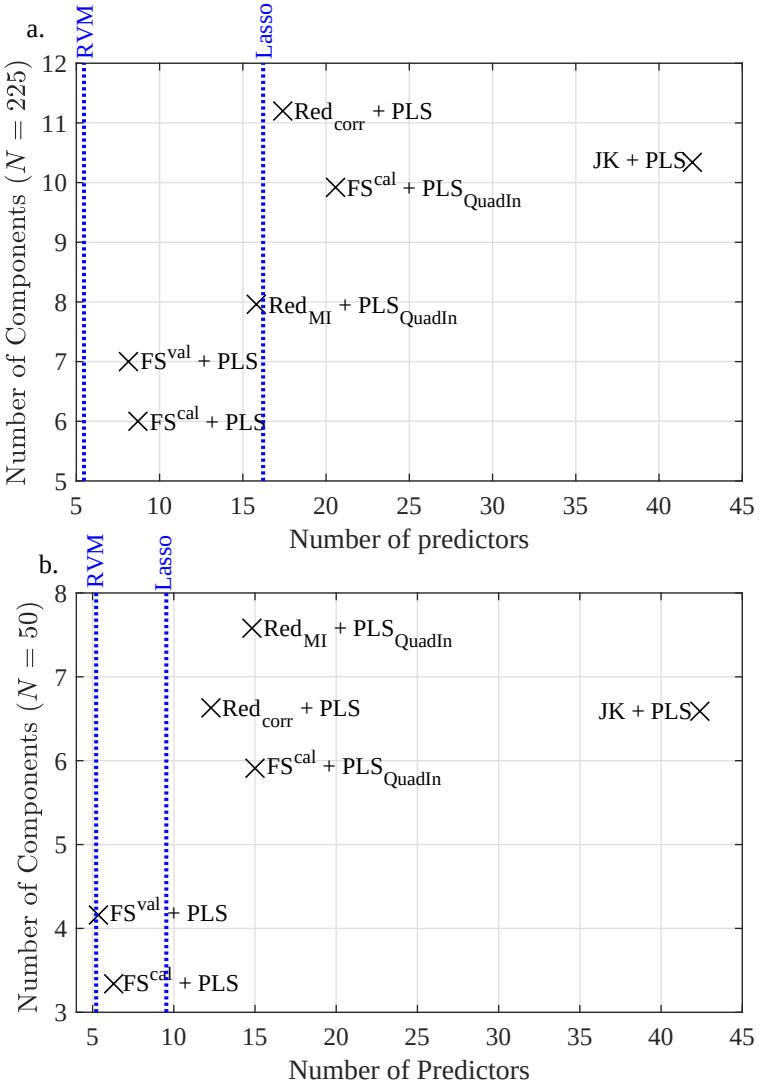

Figure 6.4. Optimum parameters in PLS based methods, averaged over three response variables and two simulation scenarios, for the training set of size 225 (a) and 50 (b).

Opting for a quadratic inner relation in the PLS model results in excess complexity, and lower efficiency (Figure 6.3), however since Red_{MI} retains only one third

of the predictors (Figure 6.4), this complexity can prove to be useful as it exploits additional predictor-response variable information than the linear approach to PLS (compare models 5.1 to 9.2 and 9.3 in Figure 6.3). RVM and Lasso, both of which perform feature selection during training, were also plotted in dotted lines so as to give an idea where the PLS models stand in comparison with other methods.

6.1.2. Comparison of Nonlinear Models

ANNs are known to yield unstable estimates, especially when training set size is small and/or there exist collinearities among predictors. As a result, the avgEff of the conventional ANN model (model 11), when implemented without any feature selection or model averaging, lags significantly behind RVM and Lasso, due to its occasional very low performance (Figure 6.5). Crogging in ANN (model 11.2) leads to an almost 100% increase in the prediction efficiency, while performing predictor selection in conjunction with an ANN model (models 12.1-12.6) results in a even greater improvements in terms of prediction performance. Feature selection in ANNs reduces the need for large datasets, as the interpolation efficiency of FS + ANN surpasses those of RVM and Lasso for $N = 50$; when training set is small, feature selection can prove useful in ANN modeling on both simulation types, and within two prediction scenarios (Figure 6.5a and b, for interpolation and extrapolation, respectively). Optimum model parameters from training sets of two different sizes can be seen observed on the right hand side in Figure 6.5. Similar conclusions to those in PLS based models can be drawn in terms of optimum parameters; FS can handle smaller training set sizes, and maintain high efficiencies even in extrapolation sets by retaining a smaller number of predictors for both, $N = 225$ and $N = 50$ (Figure 6.5c and d, respectively). Average of optimum number of hidden units is between one and two in all ANN models incorporating feature selection, and less than five for ANN with the full set of predictors. This indicates that although nonlinear models, such as ANN and $\text{PLS}_{\text{QuadIn}}$, outperform linear models in prediction, the CDU system does not exhibit major nonlinear behavior; thus only a feedforward network with a single hidden layer, and only few neurons can easily model this process.

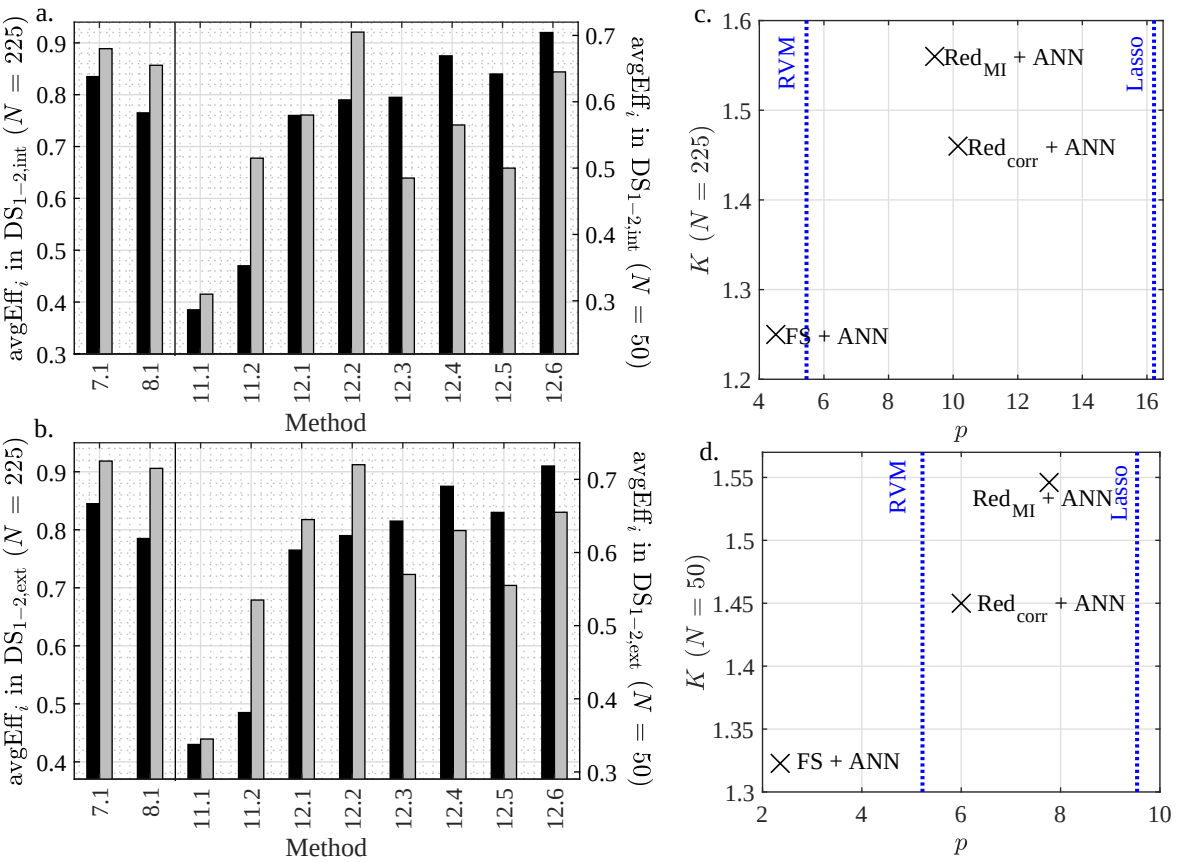

Figure 6.5. Overall efficiencies of ANN based models, in comparison to RVM and Lasso, separated with a solid vertical line (interpolation and extrapolation efficiencies for $N = 225$ in a and b, respectively), and optimum model parameters for two training set sizes (c, d).

For local modeling, RVM was chosen as the base model as its predictive performance in CDU has been stellar so far, and its training time is among the lowest of all estimators. It can be concluded that optimizing local regression parameters in FS + $\text{RVM}_{\text{Local}}$, in terms of size of the neighborhood and predictor space, contributes positively to performance of the local modeling scheme. It is not surprising that both on average and in the worst case scenario, FS^{cal} + $\text{RVM}_{\text{Local}}$ stands out with its performance as seen in Table 6.3 since query points are appended to the training set once predicted; giving a chance for the model to, in a sense, adapt to the changing char-

acteristics of the process even in the absence of time dependent data. As mentioned before, performance of local models can deteriorate in high dimensions. Thus, performing feature selection to reduce the dimension of neighborhood space, i.e. the dimension of input space in which the nearest observations are selected followed by a local RVM model yields the best predictive performance among all other learners. The optimum number of predictors retained in the neighborhood space varied from two to six.

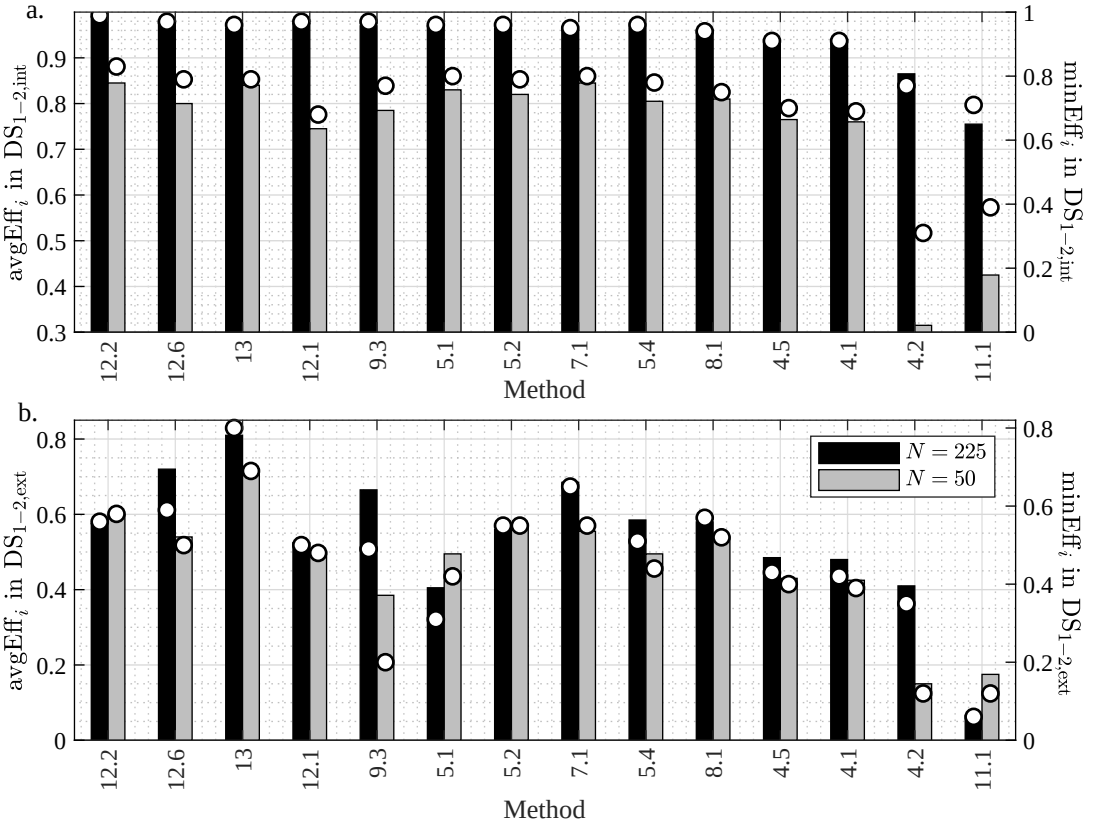

Figure 6.6. Average and minimum efficiencies of select methods, dark and gray colored bars are avgEff$_i$ in training sets $N = 225$ and $N = 50$, respectively (left axes), and the white circles represent the corresponding minEff$_i$ (right axes).

In addition to adapting to new operating conditions, RVM$_{\text{Local}}$ can handle nonlinear data by using local linear models to approximate global nonlinear relations. In interpolation, however, efficiency of RVM$_{\text{Local}}$ is marginally exceeded by global ANN methods (Figure 6.6a).

Table 6.3. Prediction errors on average, and worst case scenario of all models in this section, subscript "1" is placed on three of the best performing models.

Model	**avg$\widehat{\text{PE}}$**	**avg$\widehat{\text{PE}}$+2SD$\widehat{\text{PE}}$**	**Model**	**avg$\widehat{\text{PE}}$**	**avg$\widehat{\text{PE}}$+2SD$\widehat{\text{PE}}$**
1	>100	> 100	6.4	12.17	63.61
2.1	6.55	13.56	7.1^1	3.6	6.14
2.2	22.68	96.51	7.2	5.57	29.64
3.1	5.61	16.78	8.1	4.79	12.17
3.2	29.36	> 100	8.2	4.19	14.01
4.1	5.11	8.98	9.1	9.90	42.23
4.2	10.41	42.83	9.2	15.44	> 100
4.3	5.20	9.74	9.3	4.77	14.65
4.4	9.97	36.78	10	5.87	8.45
4.5	5.12	9.17	11.1	18.38	71.88
4.6	10.29	43.03	11.2	12.87	38.77
5.1	5.40	16.12	12.1	4.23	11.75
5.2	4.17	10.52	12.2	3.69	7.13
5.3	> 100	> 100	12.3	4.67	12.72
5.4	4.07	7.140	12.4	3.79	6.38
5.5	4.20	10.13	12.5	4.61	12.76
6.1	5.87	16.30	12.6^1	3.64	5.88
6.2	8.80	35.08	13^1	3.13	4.21
6.3	6.06	16.90			

One could say that the data exhibits only weak nonlinearities which can easily be handled using ANN, and there is no need for local modeling. It should be noted that inclusion of test points into the training set after prediction has led to a significant increase in extrapolation performance as $\text{RVM}_{\text{Local}}$ stands out in terms of both the general and worst case scenario efficiencies in Figure 6.6b. This result was expected

within the virtual drift scenario implemented in this dataset, as more and more test points are added during predictions, $\text{RVM}_{\text{Local}}$ is able to search for nearest neighbors which belong to the new concept, and thus yield more accurate predictions.

In order to have a sense of absolute prediction performances of models in addition to their relative comparisons, avg$\widehat{\text{PE}}$ and avg$\widehat{\text{PE}}$ + 2SD$\widehat{\text{PE}}$ are tabulated id Table 6.3. It should be noted that LS (model 1), Bayesian $\text{LR}_{\text{SqrPred}}$ (model 3.2), FS^{val} + PLS and FS^{cal} + $\text{PLS}_{\text{QuadIn}}$ have unacceptably high RMSEs, well above 100s.

6.2. Soft Sensor Performance of PLS and RVM on Dynamic Data

PLS is among the most popular choices for soft sensor design, it has many features which encourage its use with industrial data: First, it performs supervised dimensionality reduction through feature extraction, hence it can handle collinearities in predictors as it is often encountered in process measurements. Second, when process operates in a narrow region, i.e. when there is only small to moderate nonlinearity, PLS provides decent prediction performance, while it can easily be improved via incremental learning for soft sensor maintenance in the presence of concept drift, since it has a simple, linear parametrization. Last, due to its relative low complexity, PLS model parameters can be interpreted, and additional knowledge can be extracted from the corresponding loading and score vectors. On the other hand, PLS can have a number of shortcomings which motivate the use of different learners [59]. For instance, as seen in Section 6.1, RVM surpassed PLS in prediction performance on steady state data. In this section, these two methods will be further investigated on dynamic data under various simulation conditions and real process data, within batch and online modeling frameworks.

Synthetic data is generated from the dynamic CSTR simulation from Section 5.1.2, and the debutanizer column dataset from Section 5.2.1. This section is divided into two parts: (i) batch prediction performances of PLS and RVM are assessed with a global approach to modeling, (ii) these two methods are evaluated for online prediction in a data stream scenario, in which concept drift is present.

6.2.1. Comparison of PLS and RVM Methods for Batch Modeling

In a batch modeling scenario, PLS and RVM methods are compared considering multiple lags of process measurements as input for the soft sensor model. From synthetic data, disturbance model S0 is used to mimic processes under solely virtual drift, comprised of 20 runs in which the inlet concentration of species A remains constant, while all other measured disturbances are changed stochastically. The input matrix is constructed as follows:

$$\mathbf{X}^{(n)}(t) = [\mathbf{X}_{base}(t), \mathbf{X}_{base}(t - TS_x), ...\mathbf{X}_{base}(t - nTS_x)]$$

Process dynamics are represented using a FIR model, with lagged terms. Here, $\mathbf{X}_{base}(t)$ is the input matrix of 700×19 dimensions, with each column corresponding to a measured process variable, and n denotes the number of lagged terms included in the model input. Time variable is represented by t, and TS_x denotes the sampling interval of predictor variables. Global PLS and RVM models were constructed separately on input matrices with varying number of lagged terms for $n = 0, 2, ..., 8$; the ranges of L in parameter tuning for PLS were taken to be $[1, \min(25, rank(\mathbf{X}^{(n)}(t)))]$, here 25 is a practical upper limit which is employed throughout this section for L. Using 300/400 partitioning on simulation data, and the RMSEs of predictions were averaged over 20 runs to obtain $\widehat{\mathrm{PE}}$.

$$\mathrm{RMSE}_r = \sqrt{\frac{1}{N}\sum_{i=1}^{N}\left(\mathrm{y}_{ri} - \hat{\mathrm{y}}_{ri}\right)^2} \tag{6.7}$$

$$\widehat{\mathrm{PE}} = \frac{1}{\mathrm{R}}\sum_{r=1}^{\mathrm{R}}\mathrm{RMSE}_r \tag{6.8}$$

Here, RMSE_r is the RMSE obtained from predictions in r^{th} run, with $\mathrm{R} = 20$ being the total number of simulation runs.

For both methods, $\widehat{\text{PE}}$ declines as the number of lagged terms in the model is increased until a certain point (Figure 6.7a), PLS and RVM can both handle small N, large p problems fairly well. Additional lags of model input both increases p, and introduces excess collinearity as process variables are highly auto correlated; thereby making it more challenging to construct stable models. Beyond four lags, in which the predictor matrix is of dimensions 300×95, predictions of PLS model seem to have large bias since the optimum number of PLS components determined via CV varies only slightly even when higher order FIR models are constructed (squares in Figure 6.7b). RVM, on the other hand, consistently yields smaller $\widehat{\text{PE}}$, and the deterioration of RVM predictions for larger number of time lags seems to be due to the increased variance, since a large number of predictors is retained in the RVM model (circles in Figure 6.7b). Note that feature selection embedded within the RVM model is a discrete one, predictors are either in or out, meaning that it is more susceptible to the problem of variance compared to the continuous variable selection performed by PLS. It is interesting to note that PLS and RVM tend to favor models of more or less same dimensions (compare squares to circles in Figure 6.7b), it can be presumed that there exists an inherent number of dimensions, smaller than that of the original input space for this data. Figure 6.7c shows the distribution of frequency of lagged terms chosen via batch learning by a RVM model using all eight of the lags in the input matrix as it retains only the relevant predictors for 20 simulation runs. The x-axis in Figure 6.7c denotes the lag to which the predictors correspond; darker colored cells represent higher selection frequencies, whereas lighter colors indicate the lower ones.

It appears that RVM tends to favor $\sim$2-3 lagged predictors, anything beyond that brings no further advantage as the $\widehat{\text{PE}}$ line flattens out in Figure 6.7a as well. This holds true for all 20 runs, meaning that predictors chosen by RVM vary little to none from run to run; the embedded feature selection performed by RVM does not exhibit high variance over simulations initialized differently.

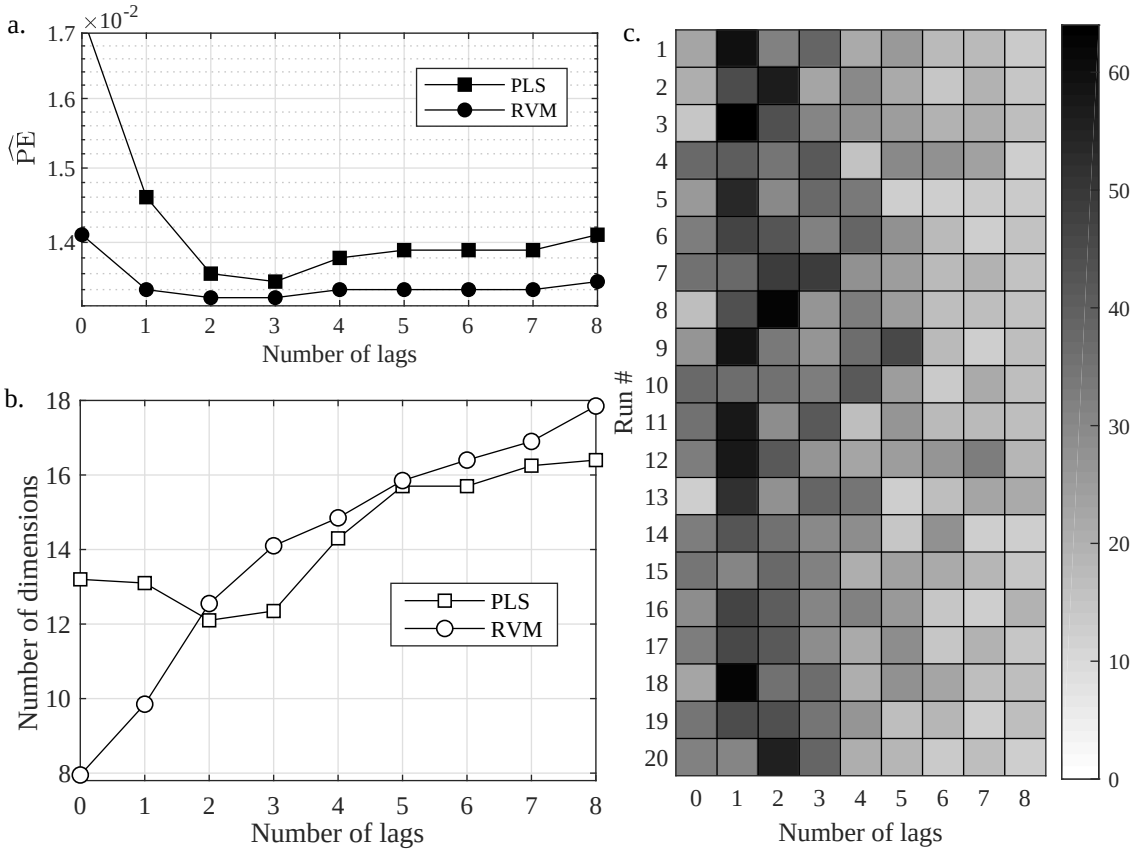

Figure 6.7. Comparison of global PLS and RVM models when lagged predictors are taken as inputs.

Even in the presence of multicollinearity resulting from an excess number of lagged predictors, RVM can select a small set of relevant predictors and yield stable predictive performance, better than that of PLS.

In order to evaluate the batch learning performance of both methods on real dynamic data, debutanizer column dataset is used from Section 5.2.1. The whole dataset was divided into four parts while preserving the temporal order of the observations, presuming that process operates in a narrower range of conditions, thus regions, which are presumed to be free of concept change, of similar sizes (first three parts comprise 500 observations, whereas the last one has 862) where it would be possible to directly compare these two methods are obtained. Each part was further partitioned into train-

ing/test sets, keeping the size of all training sets constant at 300. $\mathbf{X}_{base}(t)$, in this case, corresponds to a 500×7 matrix; lagged terms are all equally distant ($\sim$ 12 minutes) in the time domain, and the sampling time of the response variables is $\sim$ 45 minutes. Let $n = 0$ represent $\mathbf{X}_{base}(t)$, the maximum value of n is taken to be 16, resulting in a matrix of dimensions 500×119 for $\mathbf{X}^{(16)}(t)$. The same procedure for tuning parameters and model construction as in experiments on synthetic data are applied, however, in this case only one RMSE is obtained from each section.

Similar to the results from simulation data, RMSEs of RVM model are lower than those of PLS over a wide range of models (Figure 6.8). In addition, the gap between PLS and RVM widens as the number of lagged predictors in the model increases; it can be concluded that opting for a model with larger number of terms can improve predictive performance of RVM even further. Averaging both the RMSEs and the number of dimensions over different regions of data, it appears that RVM tends to retain a larger number of predictors in the model, hence yields a more complex model by standard definitions compared to PLS (Figure 6.9b). However, this does not necessarily result in a decline in predictive performance, on the contrary, it actually improves predictions as the gap between RMSEs from PLS and RVM learners (Figure 6.9a) widens, showing that RVM is more successful in reducing model bias. These observations are consistent with those that were made in simulated data (Figures 6.7b and 6.7c). Furthermore, the shape of RMSEs in Figure 6.9a is typical in prediction; there exists an optimum number of lags for which minimum RMSE is attained, meaning that number of lags is essentially a hyperparameter, which can be optimized via a crossvalidatory approach to set the number of features automatically. A steady increase in number of dimensions is observed with RVM, whereas PLS stabilizes at optimum dimensions four times that of the PLS model when $n = 0$, hence it would be easier to find this optimum number of features via CV in RVM than PLS.

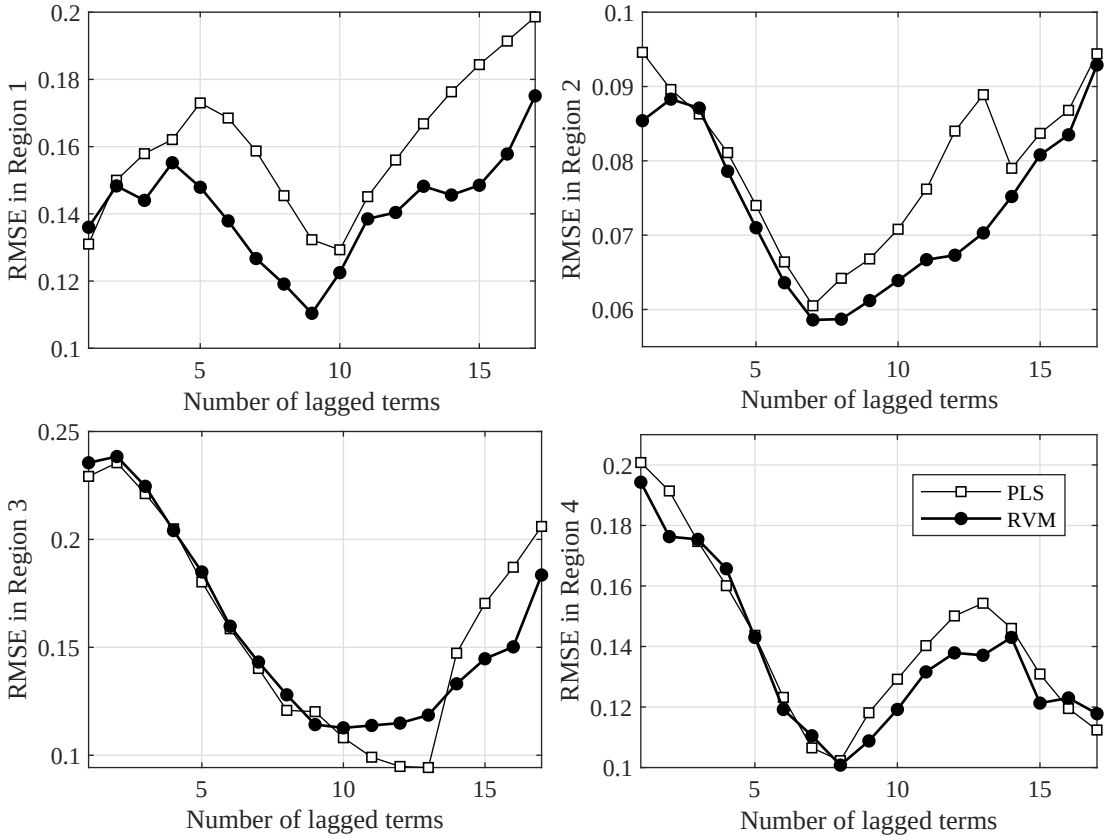

Figure 6.8. RMSEs in different portions of real data, the number above each figure indicates the segment of data for which training/prediction is performed.

Determining a convenient number of previous lags to be included in a predictive model has not been extensively investigated in the soft sensor design literature. It is essentially a bias/variance problem; considering only the most recent measurements can result in biased parameter estimates, whereas inclusion of a large number of previous measurements increases variance, but may improve predictive performance. CV (10fold with 20 repetitions) was performed to determine the optimum order of the FIR model in both PLS and RVM. In PLS, CV is initially used to optimize the number of components; thus for each input matrix of different sizes an optimum PLS model and its corresponding CVE is obtained.

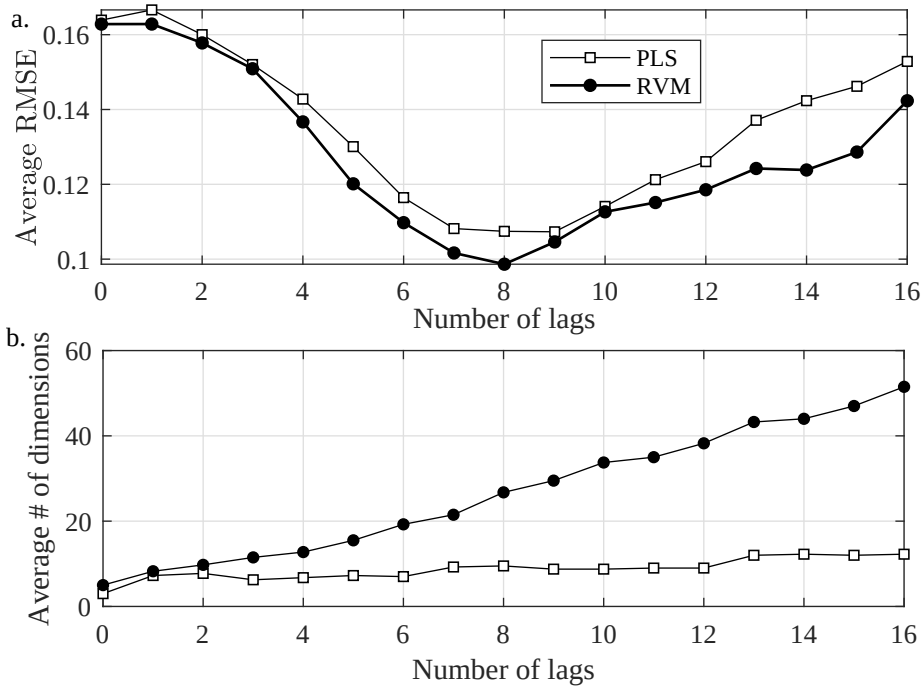

Figure 6.9. RMSE and dimensions of global PLS and RVM.

For $n = 0, 1, ...16$, the input matrix and the accompanying PLS model which minimizes the CVE is selected as the optimum. In RVM, on the other hand CV was used directly for the purpose of the selecting the number of lagged terms that should be included in the input matrix; training data is partitioned for CV in the usual manner, RVM models with input matrices made up of different number of previous lagged predictors are trained and the input matrix, i.e. the number of lagged terms, which minimizes the CVE is selected. It should be noted that feature selection performed within the training step of RVM, independent of the CV loop for input matrix selection, works against the addition of lagged terms. RVM training is essentially the step determining the effective number of parameters, the complexity of model in this case. Table 6.4 shows that tuning the number of lags via CV yields RVM models with higher (32%) predictive accuracy than PLS in two of the sets, while PLS is the winner in only one set.

Table 6.4. Optimum number of lags, n, determined via 10fold CV in PLS and RVM, and the corresponding RMSEs for each set of data.

	RMSE		**Optimum n**	
Model/Set	**PLS**	**RVM**	**PLS**	**RVM**
1	0.176	0.140	13	12
2	0.094	0.093	16	16
3	0.206	0.150	16	15
4	0.112	0.137	16	12

6.2.2. Comparison of PLS and RVM Methods for Online Prediction

In real life applications of soft sensors, data comes in the form of streams and soft sensors are expected to provide prediction in real time, not on a batch basis. The simplest way to adapt to changing data is to utilize a moving window framework for prediction. Initially, the size of the moving window, W, is set; once a test point is predicted, it is appended to the window, and the oldest observation in the window is discarded, thus keeping the size of the window constant at W throughout. In the current section, performance of PLS and RVM methods are compared within this MW framework. It should be noted that MW may not necessarily be the "ideal" method for adaptive learning, but it is one of the most frequently used ones. The aim in this section is not to find the best adaptive method, but to compare the performance of two learners under identical adaptive learning schemes.

For the experiment on synthetic data, disturbance models with real concept drift, i.e. S1.1, S1.2, S2.1, S2.2, S3.1, S3.2, S3.3 and S3.4 are selected from the dynamic CSTR simulations. Two previous time lags of predictors are used in all MW models, based on preliminary studies, and W is varied from 30 to 60 with increments of 10. Hence, input matrices $\mathbf{X}$ of dimensions 700×57 ($n = 2$) are constructed for each run. It should be

noted that for W's smaller than 30, the modeling problem is much too ill-conditioned, since $N \ll p$, and this could have potentially masked the effect of opting for a PLS or RVM based MW model. Furthermore, $\widehat{\text{PE}}$s tend to stabilize, and remain more or less around the same value beyond a certain W, and it is unnecessary to include higher Ws in this range.

Two different MW PLS models are considered (Table 6.5); the first one, denoted as PLS_{MW}, is constructed from the input matrix with two previous lagged terms, whereas in the second one, $\text{PLS}_{\text{MW}}^{\text{opt}}$ feature selection is performed as a preprocessing step, and 10 predictors (out of 19) most correlated with the response variable are selected from the base input matrix. Second model is introduced in order to see the performance of a PLS model constructed from preselected inputs, i.e. the aim is to mimic the embedded feature selection of RVM by performing filter feature selection on PLS. A further grouping of PLS models is employed based on the method of optimum component selection, either using the training set once for optimization, or re-optimizing this parameter for each MW (Table 6.5). TS and W in parentheses are used to distinguish between these two modes: 10fold CV with 20 repetitions is performed in an offline manner once on the initial training set (TS) to determine the optimum number of components, L, for all future test points, and a new optimum is found, for each query point using only the observations in the MW, respectively. The four PLS models constructed are summarized in Table 6.5.

Table 6.5. Number of inputs used in the models, and the CV method used for optimizing L in PLS_{MW} models.

Model	$\boldsymbol{p}$	**CV**
$\mathbf{PLS}_{\text{MW}}$ **(TS)**	57	Offline on TS
$\mathbf{PLS}_{\text{MW}}^{\text{opt}}$ **(TS)**	10	Offline on TS
$\mathbf{PLS}_{\text{MW}}$ **(W)**	57	Online on MW
$\mathbf{PLS}_{\text{MW}}^{\text{opt}}$ **(W)**	10	Online on MW

Prediction performance of RVM exceeds those of PLS models in most of the cases (Figure 6.10), consistent with the results obtained for batch learning. Furthermore, $\widehat{\text{PE}}$s of RVM are likely to remain constant across a range of Ws. PLS models can also be compared with respect to feature selection on/off and constant/adaptive component number selection method. Implementing an online parameter tuning scheme with PLS yields overall better prediction results; online parameter tuning can be considered as an adaptation mechanism, it serves the same purpose as opting for a MW framework instead of a static global model. Each test point is entitled to its own optimum model, and thus is treated as a new concept rather than relying on a single setting and assuming that it applies to the entire test set. PLS_{MW} (W) consistently performs better than PLS_{MW} (TS), albeit less drastically in disturbance models where the frequency of concept drift is smaller (compare the plots on the left column, to those on the right in Figure 6.10). It can be speculated that performing CV at every step proves to be particularly useful when concept drift is encountered frequently, decreasing the bias; in the absence of frequent concept change, changing the model at every single query point may lead to an excessive increase in variance. This suggests that each concept drift scenario has its own optimum level of adaptation depending on the input dynamics. It should also be noted that advantage of PLS_{MW} (W) is lost in ramp CDMs for $\text{W} > 50$, and for $\text{W} > 60$ in step CDMs. Furthermore, feature selection in a MW setting does not seem to be advantageous for PLS predictions. Note that in disturbance models S1.2, S2.1 and S2.2, prediction errors of two PLS based MW models (PLS_{MW}(TS) and PLS_{MW}(W))were too large and hence left outside the y-axis limits in Figure 6.10.

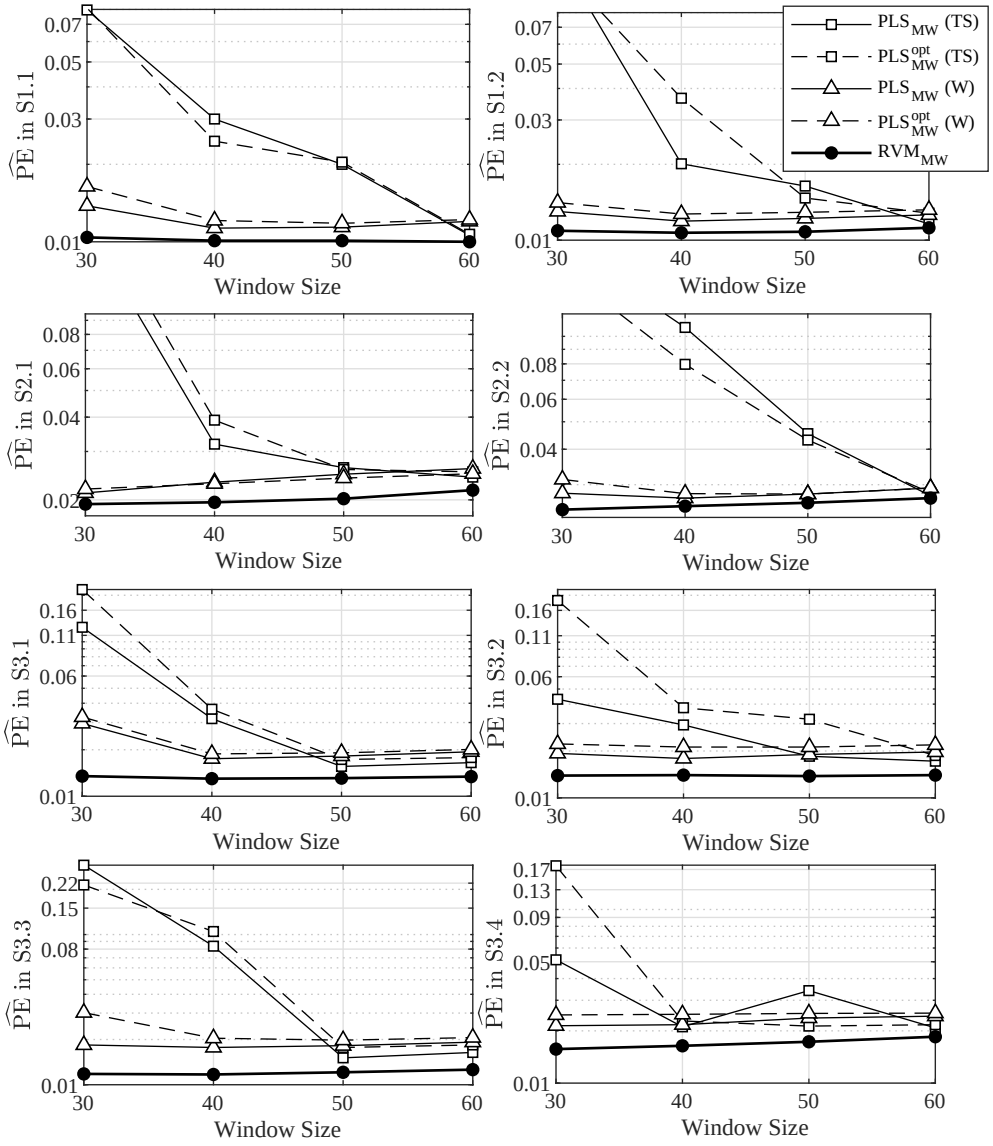

Figure 6.10. $\widehat{\text{PE}}$s of MW PLS and RVM models at different window sizes, frequency of concept change in concept drift scenarios on the left column are lower than those on the right.

The comparison made above is repeated on debutanizer column data; entire debutanizer column dataset is divided into training, and test sets of sizes 1000, and 1394, respectively. This division of dataset is also maintained in the following section, hence the results from this section can easily be compared to those. Input matrices are constructed using 0, 2, 4, 6, 8 and 12 previously lagged terms. The comparisons are based

on two different ranges of Ws: large W values in the interval [200, 300], and Ws in [10, 100], respectively. Motivation for conducting separate analyses lies in the instability of predictive models for small window sizes, as it is discussed below.

RMSEs of PLS and RVM models using MWs of sizes W $\in \{200, 210, ..., 300\}$ are shown in Figure 6.11. While RVM again yields superior performance, PLS_{MW} is shown to be improved by employing an online parameter tuning approach. Interestingly, when only few lagged terms are included in the model input matrix, offline parameter tuning has better predictive performance than the online one (RMSE plots for 0-4 lags in Figure 6.11). However, when the input dimension is increased the opposite is observed, and online CV appears to be a better choice.

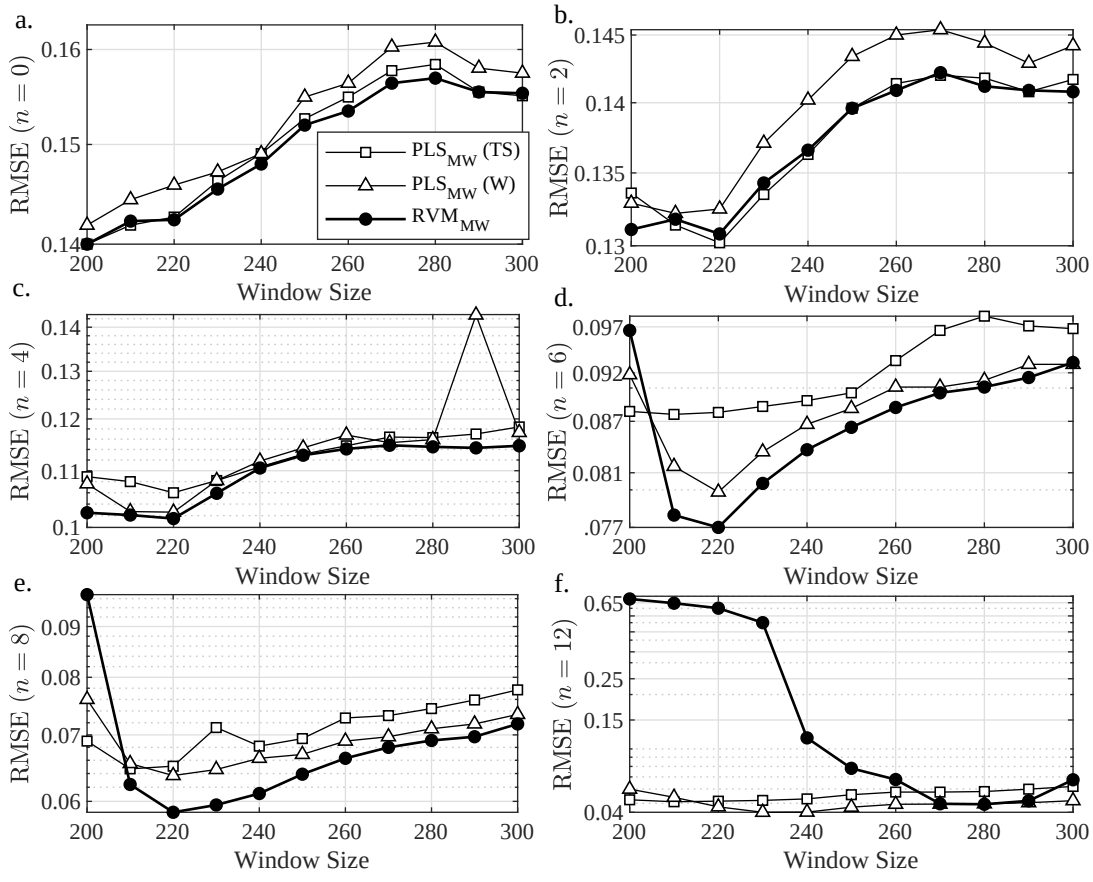

Figure 6.11. RMSEs for W in the range [200, 300].

This may be due to the bias/variance trade-off: variance may be the dominating factor for predictors containing only small time lags, hence offline optimization of number of components yields better predictive performance, but reducing bias becomes more important as more time lags are taken into consideration.

It should be noted that while predictive performance of RVM_{MW} surpasses those of both PLS_{MW} models, RVM can yield excessive prediction errors at $n = 12$ and for $W < 270$ (Figure 6.11f). It is presumed that this is caused by numerical issues during RVM training; especially the ill conditioning of Hessian matrix can lead to such erroneous predictions (see Figure 6.12a for a representative example of some unstable predictions of RVM) [132]. In the presence of extreme collinearity, as it is the case when previous lags are introduced, Hessian matrix can become semi positive definite, thereby making it difficult to find its inverse via Cholesky decomposition. In order to bypass this problem, Hessian matrix is perturbed slightly to obtain the nearest positive definite matrix. However, this is only a temporary fix implemented solely to proceed computation, hence numerical instability in the resulting parameter estimates is unavoidable and gross errors may be obtained in some regions of the test set (Figure 6.12a). It is expected that this behavior of RVM will manifest itself in the norm of regression coefficients estimated form data. Parameter estimates from the regions in which gross prediction errors are obtained (observations 900-910 in Figure 6.12a) appear to be extremely large (peak around observations 900-910 in Figure 6.12c). This result is presumed to be due to RVM training since predictions of PLS_{MW} on the same observations are acceptable (Figure 6.12b).

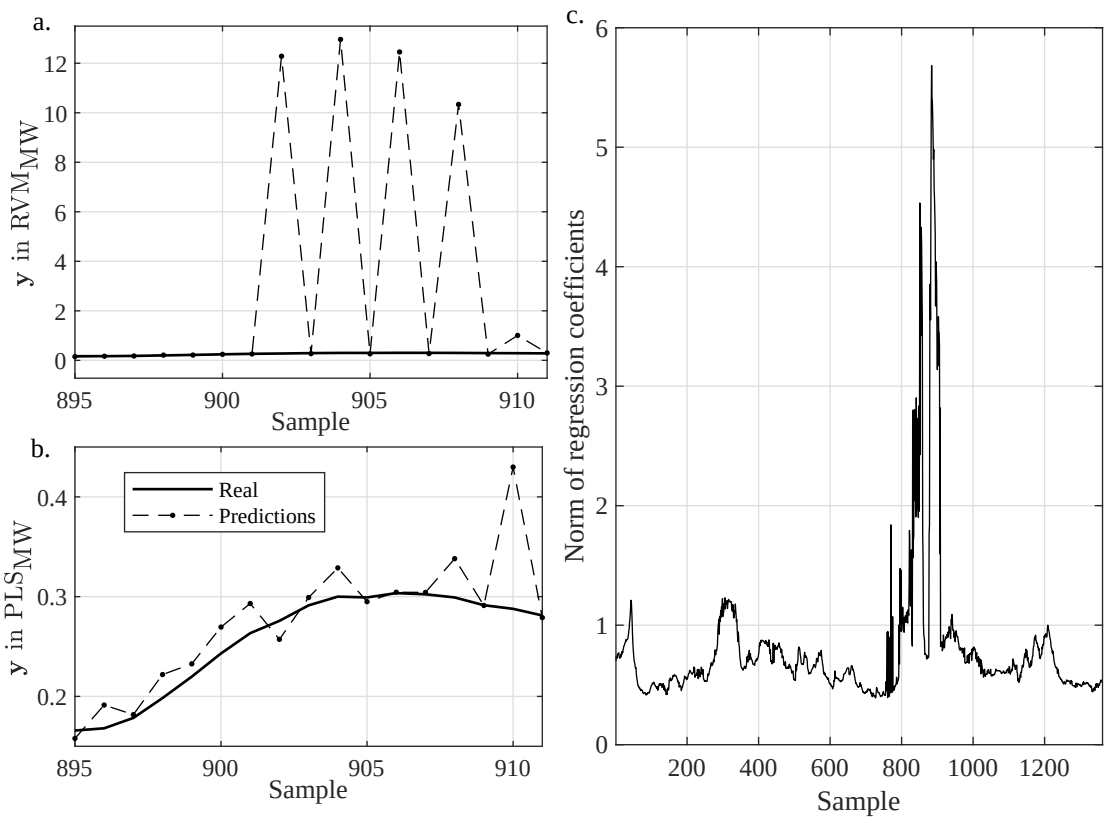

Figure 6.12. Gross prediction errors in $\mathrm{RVM_{MW}}$ (a) in comparison to stable estimates of $\mathrm{PLS_{MW}}$ (b) using W = 210 and $n = 12$, along with the norm of regression coefficient estimates obtained from $\mathrm{RVM_{MW}}$ (c).

Next PLS and RVM methods are compared using MW of size W $\in \{10, 20, ...100\}$ (Figure 6.13). In the most extreme case, when W = 10 and 12 lagged terms are included in the model, the input matrix in a MW is of size 10×119; an ill posed situation, hence unstable predictions with exceptionally high errors are to be expected regardless of the learning method used. In order to bypass this, and observe performances of two modeling methods without this numeric instability masking the results, RMSEs are computed at the 99th percentile. While PLS is known to handle collinearity, and stabilize parameter estimates fairly well, in such extreme scenarios of $N \ll p$ its performance can deteriorate due to computational reasons [133]. In our work, this problem is of major concern especially when CV is performed on the current

window in PLS_{MW}(W) since the number of rows is far exceeded by the the number of columns. Except for a few Ws, performance of RVM_{MW} is superior to PLS based MW models at smaller Ws (Figure 6.13). This differences in prediction accuracies is more pronounced for models with larger number of predictors as it was expected (see subplots for 6-12 lags in Figure 6.13). Previously in Section 6.1, discrete feature selection was found to be more suited for handling multicollinearity and redundancy than feature extraction especially when the number of training observations is small, and a similar conclusion can be drawn here. However, when W is too small it is more likely that we encounter certain process variables which remain almost constant within the corresponding MW. In this situation, RVM would not include this variable since an almost constant predictor does not contribute to explaining the variation in the response variable. This may occasionally deteriorate prediction accuracy, albeit rarely (see Figure 6.13 at W = 40 and W = 70). Thus, PLS based models may perform better since they would still extract some information from this constant predictor. It is observed that while this issue has a larger influence on RVM_{MW}, PLS_{MW} models may also yield high prediction errors, albeit less frequently. For that reason, we have opted for 99^{th} percentile RMSEs in Figure 6.13. RMSE's can be found in Appendix A. For both small and large values of W, we have observed that prediction accuracies of PLS_{MW}(W) are on the same level as those of PLS_{MW}(TS), if not worse, when the number of inputs is small; we believe this to be due to the narrow range of PLS components allowable (compare Figure 6.14a to c for W = 50, and Figure 6.14c to d for W = 200). The fast decay in autocorrelation function of model input with zero time lags shows that change in consequent optimum Ls is closer to random (Figure 6.14e), when a large number of lagged predictors are included in the model, on other hand, this decay is much slower suggesting an underlying trend in optimum L's (Figure 6.14f). Even though the "real" value of optimum L is unknown, the existence of a slight trend is more plausible than a random change in consecutive optimums. At larger values of n, ($n > 4$) and W (W > 50) it is possible to achieve some reduction in bias by optimizing L for each query (Figure 6.13). At small n, however, MSE is dominated by variance and optimizing L for each query does not reduce bias in such narrow range of components, and one should opt for constant L rather than change it at each observation.

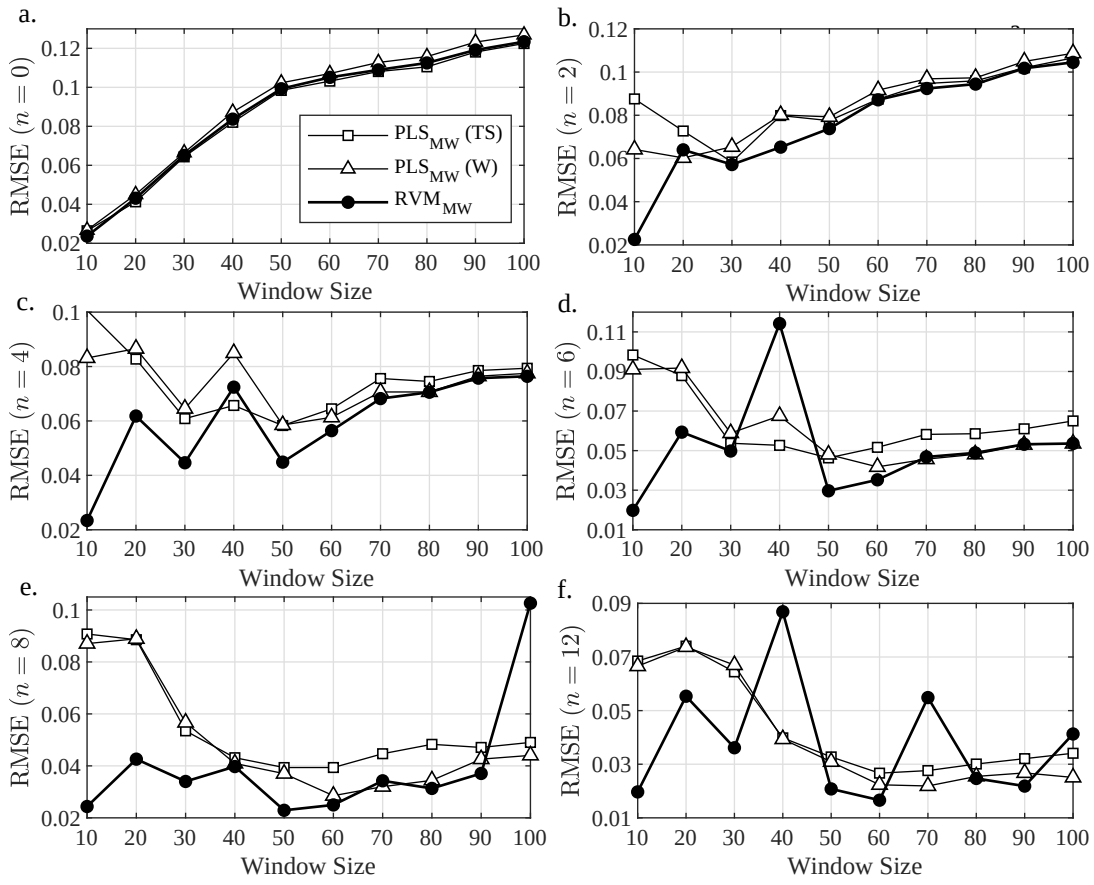

Figure 6.13. 99^{th} percentile RMSEs for W in the range [10, 100].

It should be emphasized that for smaller values of W in PLS_{MW}(W) 10fold CV is performed using small number of training observations, it essentially reduces to LOOCV for the smallest W (W = 10), thereby increasing variance.

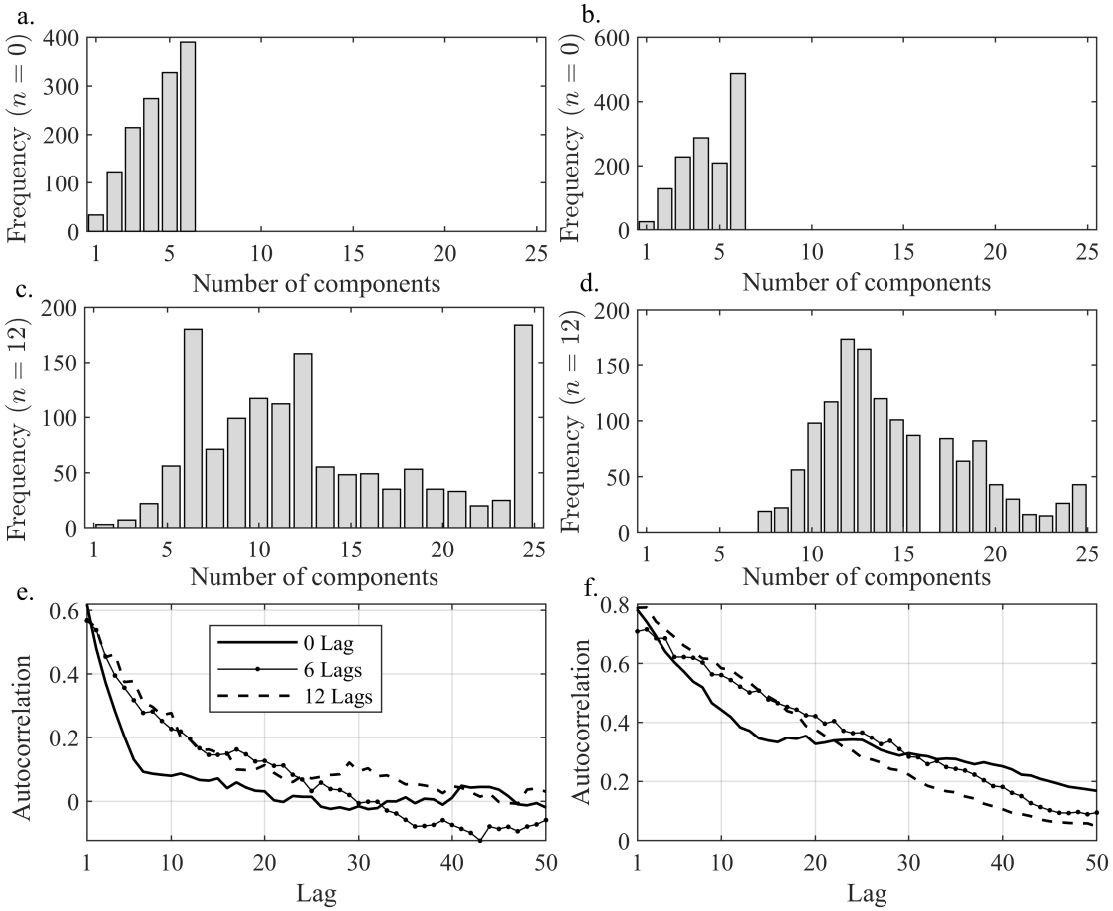

Figure 6.14. Distributions of optimum L determined for each query point during testing for $n = 0$ (a, b), $n = 12$ (c, d) and autocorrelation functions (e, f), subplots on the left (a, c, e) are obtained for W = 50, and the right (b, d, f) for W = 200.

It should also be noted that RMSE is not the only criteria for comparison; parameter tuning step in both PLS_{MW} models is of significant computational cost, especially in PLS_{MW} (W) in which 10fold CV is performed for each query point. Even for a relative small-sized input matrix of dimensions 140×35 the difference in computation time can be observed between these three models; computational burden of PLS_{MW}(TS) is

the smallest, followed by that of RVM_{MW} and $\text{PLS}_{\text{MW}}(\text{W})$, each ordered by a difference of one order of magnitude in terms of average time elapsed in a test set.

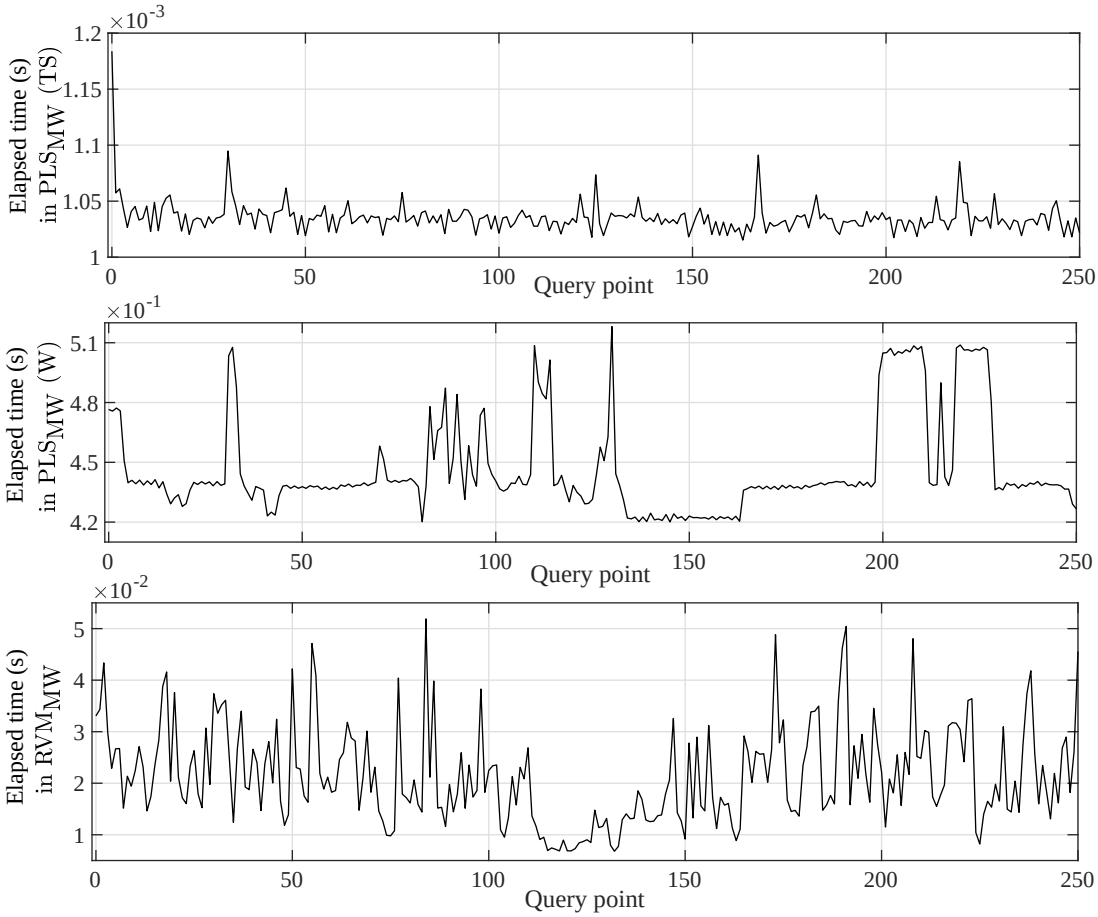

Figure 6.15. Elapsed time in seconds for each query: predictions from when W = 140 and 4 lagged predictors are included in the model are presented, average time elapsed in predictions is 0.001, 0.447 and 0.021 seconds in the order of subplots.

RVM_{MW} provides better predictive performance than PLS based MW models; while accuracy of $\text{PLS}_{\text{MW}}(\text{W})$ can be comparable to that of RVM_{MW} in some cases. However, the excess burden in computation is not warranted as a single prediction via RVM is $\sim$ 10 times faster than PLS with online CV. However, since embedded feature selection in RVM can sometimes decrease prediction accuracy, we suggest taking cautionary actions to avoid this rare problem in RVM when a MW based strategy is

used for prediction (see the next section on adaptive soft sensor design).

6.3. Adaptive Soft Sensor Design

The final issue we address in soft sensor design for industrial processes is maintenance of good predictive performance in the presence of concept drift. In order to handle concept changes in data streams, adaptive learning algorithms are needed. Even though several algorithms have been proposed in the literature of soft sensor design, there are still open issues, and shortcomings of learning methods. In this section we develop various adaptive (AD) learning methods, and describe different manners in which they can be applied. It should be noted that novel adaptive methods suggested in the current study are designated with AD, while other methods are referred with their traditional names used in the literature. One could think of these ADs as the building blocks of a larger, encapsulating algorithmic framework, aiming to address different problems in learning from data streams, while analyzing shortcomings of different approaches for developing adaptive algorithms.

6.3.1. Adaptive Mechanisms

A slightly different notation will be adopted hereinafter for the sake of simplicity. Staying true to the initial definitions in Chapter 3 for input and output, $\mathbf{X}$, and $\mathbf{y}$, following definition is used.

$$D_\tau \coloneqq \{(\mathbf{x}_t, \mathrm{y}_t)\}, t \in \tau \tag{6.9}$$

Here, τ is a set of integer values, corresponding to the indices of ordered pairs of observations, $(\mathbf{x}_t, \mathrm{y}_t)$. D_τ is thus a subset of the entire historical dataset D of size N for $|\tau| \leq N$. For instance, when a learner, $\mathcal{L}$ is trained on a MW basis to predict k^{th}

query point, the corresponding training set will be:

$$D_{\mathrm{MW}} = \{(\mathbf{x}_t, \mathrm{y}_t)\} \tag{6.10}$$

$$\tau_{\mathrm{MW}} = \{t | t \in \mathbb{Z}, N_0 + k - \mathrm{W} \leq t \leq N_0 + k - 1\} \tag{6.11}$$

$$\mathrm{W} \coloneqq |\tau_{\mathrm{MW}}| \tag{6.12}$$

Here, N_0 denotes the the origin of evaluation, i.e. $N_0 = N - k + 1$, and τ_{MW} comprises time ordered indices of observations within the most recent window of k^{th} of size W. Only RVM will be employed as the base learner $\mathcal{L}$ throughout this section, and the following simplification may be used:

$$\hat{f}(D_{\mathrm{MW}} \mid \mathcal{L} = \mathrm{RVM}) = \hat{f}(D_{\mathrm{MW}}) \equiv \hat{f}_{\mathrm{MW}}$$

$\hat{f}_{\mathrm{MW}}$ is then a RVM model trained using observations in D_{MW}. RVM training yields posterior estimates for the weights, $\mathbf{w} \in \mathbb{R}^p$, some elements of which are zero; we use p' to denote the number of nonzero elements retained in the model, and p' can be referred to as the "effective" number of predictors. A RVM model yields, along with point estimates of the regression parameters, model error variance estimate $\hat{\sigma}^2$, and prediction error variance $\hat{\sigma}^2_{\mathrm{PE}}$ (for a given test point). For the tuple of k^{th} test observations, $(\mathbf{x}'_k, \mathrm{y}'_k)$, response can be predicted as:

$$\hat{\mathrm{y}}'_k = \hat{f}(D_{\mathrm{MW}}, \mathbf{x}'_k) = \hat{f}_{\mathrm{MW}}(\mathbf{x}'_k) = \mathbf{x}'_k \mathbf{w} \tag{6.13}$$

Test points are denoted with a prime in order to distinguish them from the initial training set, k^{th} test point, then, corresponds to $N_0 + k^{\mathrm{th}}$ observation from the beginning where N_0 is the size of the initial training set. Herein, the name "query" is reserved for these test points only.

The proposed ADs in the current thesis are based on a MW approach, and can be further grouped into two main categories (AD1 and AD2) according to their algorithmic complexity:

- MW
- AD1: Moving window with adaptive window size
- AD2: Moving window and JITL ensemble

6.3.1.1. MW. Moving window approach in adaptive modeling, being the most intuitive one, is frequently used to handle time varying data. It is based on the same philosophy as JITL; once a query point arrives, a predictive model is trained on only the most relevant subset of the whole training set. JITL differs in the definition of "relevance", as it adopts a spatial measure for similarity which often neglects the time stamp of observations. In a MW, however, all observations are ordered and the relevance is determined based on temporal proximity to the query point. Thus, as new samples are acquired and appended to the dataset, the window moves (or slides) to accommodate the new sample. As an example, moving windows of 300^{th} and 500^{th} query points are shown in Figure 6.16, with the corresponding local models $\hat{f}^{300}$ and $\hat{f}^{500}$. It should be noted that windows are slided by one sample, and the models are trained on a sample basis, not on batches. In application of MW method with constant W, optimum W is unknown. Thus, we suggest using rolling origin validation to determine W specific for each problem. The last 65% of training data is reserved for validation, but this percentage may vary depending on how large the dataset is. It is desired to have as many validation points as possible, in order to reduce the variance of prediction error estimates, so, keeping the size of the calibration set large enough to contain observations corresponding to the largest window size should suffice.

In most soft sensor applications with MWs, window size is kept constant. While this approach may sound appealing since it simplifies the algorithm as window size is set only once in the beginning, it implicitly assumes that the target concept changes in small magnitudes and are slower than the approximate convergence of the testing

learning curve. This assumption is rarely justified in real life as different types of concept drifts can be manifested all at the same time, and it is practically impossible to detect the type of concept drift in industrial processes. Furthermore, the rate of concept changes might also change with respect to time. Hence, a sounder approach - adaptive to its core - would be to adapt the window size during online prediction. A variable moving window size can also address the "stability/plasticity dilemma"; when concept change is abrupt a small window size should be preferred so as to give larger emphasis on most recent data, and in the case of more gradual changes, or stable concepts, a larger window size should be maintained [25].

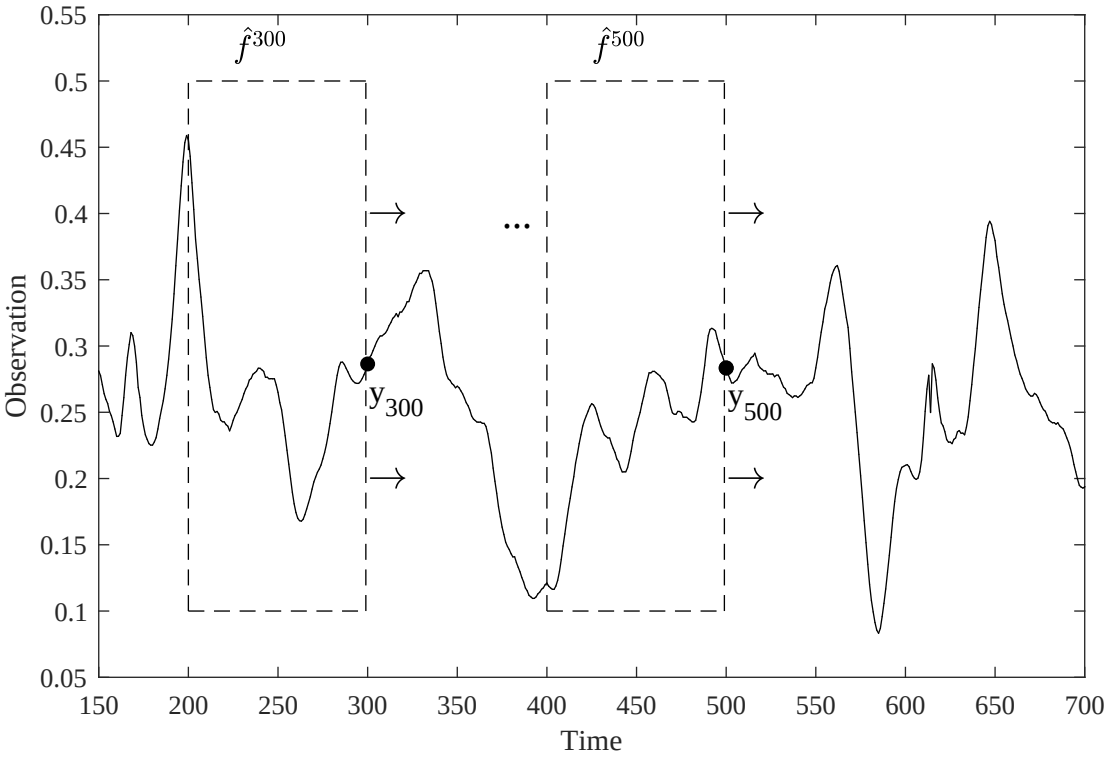

Figure 6.16. Online prediction using a moving window with constant W = 100.

6.3.1.2. AD1: MW with Adaptive Window Size. A MW with variable W is adopted to track process dynamics more successfully, i.e. moving window size may be tuned by identifying operating regions in which the target concept is presumed to be homogeneous within the MW, and stable parameter estimations can be made. If the window

size is too large, then it would include observations from the previous concept, hence degrading the benefits obtained from the MW technique. If, on the other hand, the window size is too small, then leaving out essential information about the current concept may lead to fitting to noise. Adapting window size according to process dynamics is an informed model adaptation strategy, since it implicitly seeks to recognize concept change [118]. In addition, one should also consider computational costs since an ideal adaptive mechanism should fulfill this task in real time so online predictions can be made. Here three adaptation mechanisms are suggested for online tuning of MW size.

- AD1A: Exponentially increase window size based on the Mahalanobis distance of the query point with respect to the training in the MW (Check_{M} algorithm),
- AD1B: Use R^2_{adj} statistic to compare models obtained from different sizes of windows (Check_{R^2} algorithm),
- AD1C: Conduct an F-test on successive, $\hat{\sigma}^2$ statistics, and change W accordingly (Check_{s} algorithm).

Mahalanobis distance, d^2_{M}, is a frequently used distance metric for outlier detection, and in AD1A, d^2_{M} of the k^{th} query point, $\mathbf{x}'_k$, to observations in its window, $\mathbf{X}_{\text{MW}}$ is computed as follows:

$$d^2_{\text{M}}(\mathbf{X}_{\text{MW}}) = \left(\mathbf{x}'_k - \overline{\mathbf{X}}_{\text{MW}}\right) \mathbf{S}^{-1}_{\text{MW}} \left(\mathbf{x}_k - \overline{\mathbf{X}}_{\text{MW}}\right) \tag{6.14}$$

Here, $\overline{\mathbf{X}}_{\text{MW}}$ is the mean, and $\mathbf{S}_{\text{MW}}$ is the covariance matrix of the observations in the current window. The idea is to determine whether the current window is a viable training set to predict query point; this holds if $\mathbf{x}'_k$ "belongs" to its predetermined window. Mathematically speaking, $\mathbf{x}'_k$ should be close enough to the center of ellipsoid defined by distribution of observations within the window. d^2_{M} follows Chi-square distribution with p degrees of freedom for input of size N with p predictors for known mean and variance matrices. When mean and covariance matrices are estimated from the sample, d^2_{M} follows F-distribution and upper confidence limit for d^2_{M} at a significance

level α can be computed as [134]:

$$\mathrm{UB}_{\alpha} = \frac{p(\mathrm{W}-1)(\mathrm{W}+1)}{\mathrm{W}(\mathrm{W}-p)}\mathcal{F}_{p,\mathrm{W}-p,\alpha} \tag{6.15}$$

Here, $\mathcal{F}_{p,\mathrm{W}-p,\alpha}$ denotes F-distribution with p and $\mathrm{W}-p$ degrees of freedom. A positive test result, i.e. $d^2_{\mathrm{M},k} > \mathrm{UB}_{\alpha}$, at k^{th} query point will be denoted as $I_{\mathrm{M}} = 1$ from hereafter.

The motivation for using Mahalanobis distance lies in the fact that it defines a elliptical distribution as it takes the covariance structure within observations into account, unlike Euclidean distance in which it is assumed that observations form a sphere. For instance, two dimensional correlated data is generated from standard normal distribution and plotted in Figure 6.17c; the usual $\pm 3\sigma$ lines fail to detect outliers shown in squares in Figure 6.17a and 6.17b since they neglect correlation between x_1 and x_2. However, they can be identified visually from Figure 6.17c in $x_1 - x_2$ space, and the Mahalanobis distances of these points fall outside the $99^{\mathrm{th}}\%$ confidence limit obtained using Equation 6.15.

In AD1A we define two significance levels, α_1, and α_2, satisfying $\alpha_1 > \alpha_2$, and confidence bounds corresponding to these significance levels should be calculated using Equation 6.15. Significance levels are taken as $\alpha_1 = 0.01$, and $\alpha_2 = 0.005$ in this thesis. When the k^{th} input is received, its Mahalanobis distance to the current D_{MW} is calculated, and AD1A mechanism is invoked only if this distance exceeds the upper bound in Equation 6.15. If the distance is outside the limit, window size is exponentially increased by δ until a window size within the limits is obtained. For instance, for $\delta = 0.2$ (the value used in this study) and $\mathrm{W} = 50$, $\mathrm{W} = (1 + 0.2) \times 50 = 60$ is used to calculate d^2_{M} (Equation 6.14), and W is incremented by a factor of 1.2 until $d^2_{\mathrm{M}} < \mathrm{UB}_{\alpha_1}$ is satisfied. If none of the MWs satisfies this criterion, calculations are repeated at a smaller significance level (α_2). In the case of another failure to determine a convenient window, AD1A settles for the window size corresponding to the smallest Mahalanobis

distance.

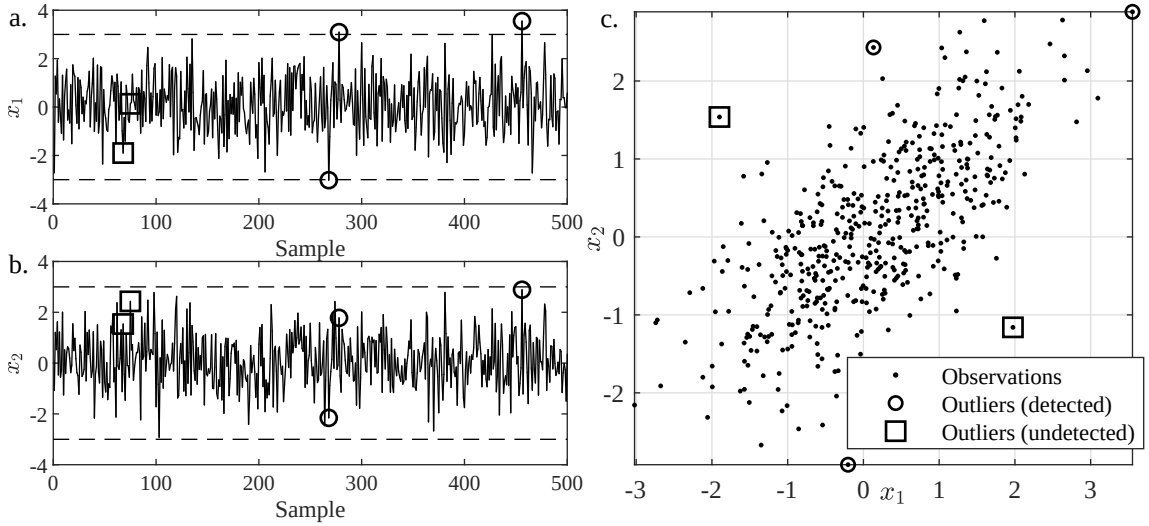

Figure 6.17. Mahalanobis distance to detect outliers in two dimensions, circled dots represent the outliers detected, and dots with squares are the outliers undetected in one dimension with 3σ rule.

One advantage of using only the spatial distance in the input space is that anticipatory action can be taken at time of k^{th} instance, even if the output has not been received yet. In cases of laboratory equipment failure, or sensor downtime when the current output is not received yet, AD1A can still be used since it is independent of the output value. One could argue that neglecting information about the response variable would deteriorate the predictive performance, however, adaptation in AD1A is realized in an informed manner, since Mahalanobis distance is a viable, albeit indirect, indicator of quality of the model estimated in the current MW. As an example, prediction trajectories using MW method and AD1A with W = 20 on the debutanizer column data are shown in Figure 6.18, in which elliptic region on the right in subplot a (observation number 1047) shows that absolute prediction errors using solely MW is above 1.0 (dashed lines in Figure 6.18a), which is obviously an outlier since most of the prediction errors lie between $0.01 - 0.2$. This gross error is reduced to a reasonable value of ~ 0.15 when AD1A is used instead (solid line with markers in

Figure 6.18a). On the other hand, there may be a downside of this method, causing the moderate-magnitude prediction errors increase, e.g. the circled region on the left contain a number of test samples with higher PEs due to AD1A. Performing a hypothesis test at every step is likely to inflate type-I error rate, thus window size may be unnecessarily increased and prediction accuracy may be degraded for test points with "false positive" results (observations 1004, 1011, 1013, 1016, 1020 in Figure 6.18a). Nevertheless, this slight degradation of prediction accuracy is much smaller compared to the improvement achieved with gross errors, justifying the use of AD1A method.

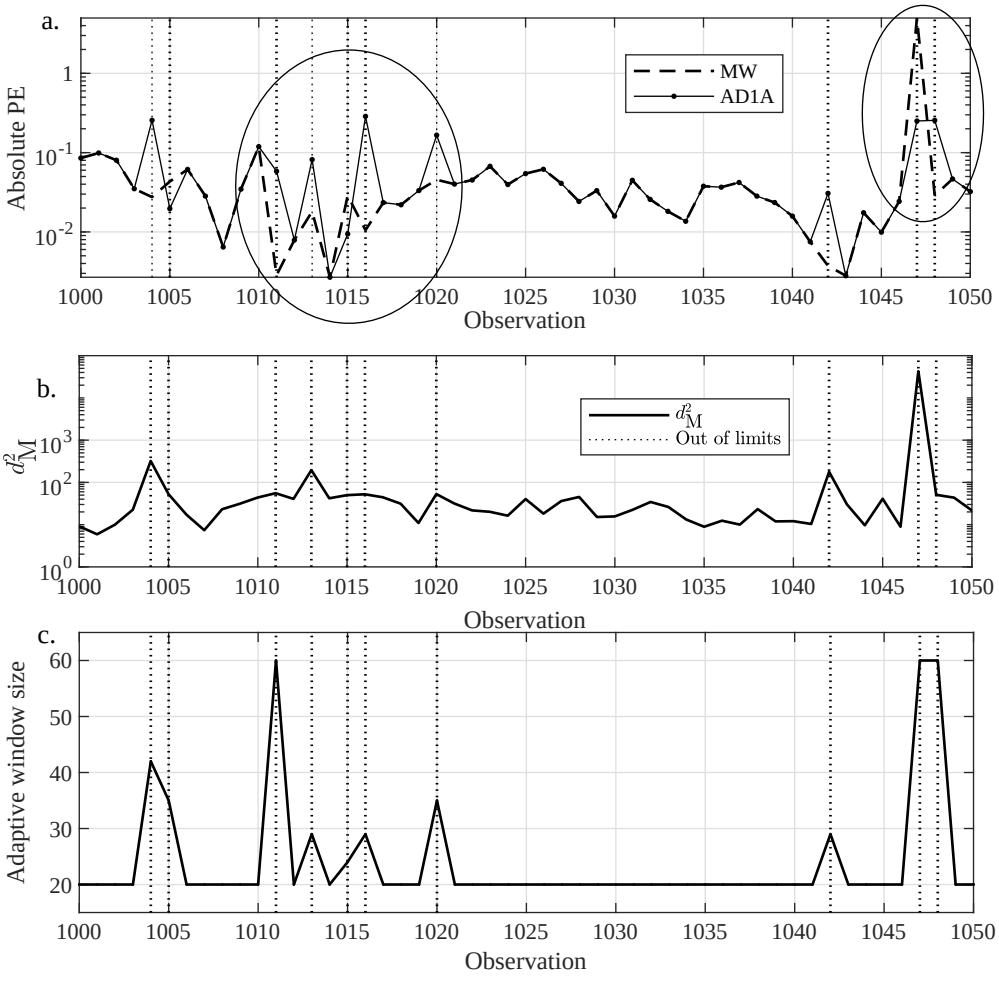

Figure 6.18. An example of Mahalanobis distances computed for each observation, and the absolute prediction errors using from MW and AD1A using W = 20.

In addition, models built on regions, in which the query point was frequently detected to be outside of the confidence bounds (at significance level 0.005), are found to yield higher prediction errors. Thus, d_{M}^2 may be suggested as an indicator of prediction accuracy of the model. Interestingly, AD1A can sometimes favor rather large windows (see Figure 6.18c for the adaptive window size determined via AD1A) which are almost equivalent to using a global model, showing that query point comes from a foreign operating region unlike those in the historical operation. One may argue that excessive increase of the window size may result in different concepts being included in the MW, however, it should be noted that a local model around the historical data (without further tests) which yields a smaller d_{M}^2 statistic, would suffer from the same problem or could even deteriorate prediction performance, since there would be no data representing the current process state.

The main drawback in relying solely on spatial analysis is that real concept drift, which causes the underlying functional relationship in Equation 3.1 to change, cannot be addressed. This approach can only track virtual concept drift which can manifest itself as changes in the model inputs. Another possible drawback of AD1A is that it is a one sided AD, meaning that it only increases the window size, and does not decrease it (Figure 6.19). AD1A is only allowed to increase the window size when it detects an unsatisfactory moving window, with the expectation that a relatively more stable model can be obtained if more observations are used to estimate its parameters. This approach, may or may not yield more accurate predictions; however, this is not the concern of AD1A. Perhaps it is more appropriate to view AD1A as just a temporary fix when the window size is too small to make stable predictions; it is more of a patch released once a vulnerability, or a bug is detected within a software than a life long solution for adapting window size. AD1A gives a practical solution in cases where the distribution of inputs exhibit abrupt changes, and one should opt for a small window size in general. If the number of inputs is too large, then it could be risky to reduce W too much as it would lead to highly unstable parameter estimates, and consequently predictions, from time to time. AD1A can help eliminate such extreme predictions by "jump-start"ing the window size.

1. Input: Historical data, $\mathbf{X} \coloneqq \{\mathbf{x}_i\}_{i=1}^{N_0+k}$; current window size, W; k^{th} query, $\mathbf{x}'_k$; window increment, δ; significance levels, α_1 and α_2 with $\alpha_1 < \alpha_2$

2. Initialize:

i. Compute upper bound UB_{α_1} using Equation 6.15

ii. Compute d^2_{M} using Equation 6.14 with $\mathbf{X}_{\text{MW}} \coloneqq \{\mathbf{x}_i\}_{i=N_0+k-\text{W}}^{N_0+k-1}$

iii. $S_d = d^2_{\text{M}}$; $S_{\text{W}} = \text{W}$; $I_{\text{M}} = 0$ // To store d^2_{M}s and Ws.

3. Core:

if $d^2_{\text{M}}(\mathbf{X_{MW}}) < \text{UB}_{\alpha_1}$ **then**

return W, I_{M}

end if

$I_{\text{M}} = 1$

while $d^2_{\text{M}} > \text{UB}_{\alpha_1}$ **do**

i. $\text{W} \leftarrow \text{W} \times (1+\delta)$; $\mathbf{X}_{\text{MW}} \coloneqq \{\mathbf{x}_i\}_{i=N_0+k-\text{W}}^{N_0+k-1}$

if $\text{W} > N_0 + k - 1$ **then**

i. Compute UB_{α_2} using Equation 6.15 // W cannot exceed N

ii. $S^*_{\text{W}} = \varnothing$

for all $\text{W} \in S_{\text{W}}$ **do**

if $d^2_{\text{M}} < \text{UB}_{\alpha_2}$ **then**

$S^*_{\text{W}} \leftarrow S^*_{\text{W}} \cup \{\text{W}\}$

end if

end for

iii. $\text{W} = \begin{cases} \min S^*_{\text{W}} & \text{if } S^*_{\text{W}} \neq \varnothing \\ \underset{\text{W} \in S_{\text{W}}}{\text{argmin}}\ d^2_{\text{M}}(\mathbf{X_{MW}}) & \text{otherwise} \end{cases}$

return W, I_{M}

end if

ii. Compute d^2_{M} using Equation 6.14; $S_d \leftarrow S_d \cup \{d^2_{\text{M}}\}$; $S_{\text{W}} \leftarrow S_{\text{W}} \cup \{\text{W}\}$

end while

return W, I_{M}

Figure 6.19. AD1A (Check_{M}) Algorithm

In the second AD for adaptive window size, AD1B, input-output relation in the training set is also taken into account. Given k^{th} query, $\mathbf{x}'_k$, and the corresponding window size, W; AD1B "perturbs" W a little to see if it can find a new smaller or larger region, in which a RVM model with higher explanation power is obtained. For this purpose, AD1B relies on the adjusted R^2 statistic (R^2_{adj}) of the models as a metric to compare different MWs. R^2_{adj}, is a metric which quantifies the amount of variance explained by the model by taking the number of predictors into account as well. For a RVM model trained on N samples to obtain weight estimates with p' nonzero elements, R^2_{adj} is computed as:

$$\mathrm{SS_T} = \sum_{i=1}^{N} (\mathrm{y}_i - \bar{\mathbf{y}})^2 \tag{6.16}$$

$$R^2_{adj} = 1 - \frac{\mathrm{SS_R}/dof_{\mathrm{R}}}{\mathrm{SS_T}/dof_{\mathrm{T}}} \tag{6.17}$$

Here $\mathrm{SS_T}$ is the total sum of squares, i.e. variation of the response variable, dof_{T} is the degrees of freedom (dof) of variance estimate, equal to $N-1$, and dof_{R} is the dof in model estimates, equal to $N-p'-1$, in which we subtract 1 since mean of the response variable is estimated from the sample. $\mathrm{SS_R}$, the residual sum of squares, is defined same as in Equation 3.12.

Introducing new parameters into any type of model would increase the amount of variance explained, so Equation 6.17 incorporates dof in order to compensate for this artificial increase in variance explained, and penalizes complex models with a large number of parameters. When RVM models are trained on different windows, it is highly likely that different relevance vectors are retained in the model, thus, R^2_{adj} provides a common ground where such different models can be compared. While R^2_{adj} is a rather obsolete method for model selection, it should be noted that the emphasis here is to select a convenient training set; hence a RVM model estimated from the recent time window with little nonlinearity and a homogeneous concept is expected to yield a higher R^2_{adj} than another RVM model obtained from another recent window

comprised of observations produced by more than one concept. AD1B shares the common features of AD1A in the sense that they both operate in an anticipatory fashion, without any information on the current output. Neither aims to estimate, or track prediction errors explicitly, however, R^2_{adj} favors more parsimonious models which, in theory, are expected to have superior predictive performance.

AD1B requires three predefined parameters: upper and lower window size limits, W^{U}, and W^{L}, and a scheme for obtaining candidate window sizes for comparison. We suggest two different ways to obtain a set window sizes, S_{W}: (I) use an arbitrary constant value (ΔW), and compare windows of size $\pm$ this value around the previous W, (II) use increments ($\Delta\mathrm{W}_1$ and $\Delta\mathrm{W}_2$, satisfying $\Delta\mathrm{W}_2 > \Delta\mathrm{W}_1$) as a fraction of the current window size, and compare two Ws smaller, and one W larger than the current W. For instance, when method (I) is employed using increments of $\Delta\mathrm{W} = 15$ at the current window of size 50, R^2_{adj} values are computed for $\mathrm{W} = 50$, $\mathrm{W} = 50-15 = 35$, and $\mathrm{W} = 50 + 15 = 65$, and compared. Using method (II) with increment fractions of 0.2 and 0.5 again on $\mathrm{W} = 50$, R^2_{adj} values are computed for $\mathrm{W} = 50$, $\mathrm{W} = 50\times(1-0.5) = 25$, $\mathrm{W} = 50\times(1-0.2) = 40$, and $\mathrm{W} = 50\times(1+0.2) = 60$. Here, we adopt an asymmetric set of candidate Ws to favor decreasing W, so as to handle concept drift through window size reduction. In the current study, (I) and (II) are both used for query points $k > 1$, and the respective methods are denoted as AD1BI and AD1BII; for the first query point S_{W} is a set of six Ws, logarithmically spaced between W^{L} and W^{U} (Figure 6.20). It should be noted that the notation, AD1BI and AD1BII is reserved for when these two schemes for generating candidate Ws are compared, and AD1B is implemented as in AD1BII, and using differential window sizes of $\Delta\mathrm{W}_1 = 0.2$ and $\Delta\mathrm{W}_2 = 0.5$, unless otherwise stated. Special attention is required to set W^{L}, lest unstable predictions are obtained due to ill-conditioned matrix inverses during RVM training. W^{U}, on the other hand, can seriously influence the adaptive W trajectory, hence we suggest determining W^{U} from an initial forward validation employed on the training set in the same way as in AD1A.

1. Input: Historical dataset, $D \coloneqq \{\mathbf{x}_t, \mathrm{y}_t\}_{t=1}^{N}$; current window size, W; differential window size, ΔW (for AD1BI); differential window sizes, $\Delta\mathrm{W}_1$ and $\Delta\mathrm{W}_2$, satisfying $\Delta\mathrm{W}_2 > \Delta\mathrm{W}_1$ (for AD1BII); k^{th} query, $\mathbf{x}'_k$; lower and upper limits for window size, W^{L} and W^{U}

2. Initialize:

if $k = 1$ **then**

$S_{\mathrm{W}} = \left\{\mathrm{W}^{\mathrm{L}}, ..., \mathrm{W}^{\mathrm{U}}\right\}$ // Logarithmically spaced with $|S_{\mathrm{W}}| = 6$.

else

// for AD1BI

$S_{\mathrm{W}} = \left\{\max(\mathrm{W} - \Delta\mathrm{W}, \mathrm{W}^{\mathrm{L}}), \mathrm{W}, \min(\mathrm{W} + \Delta\mathrm{W}, \mathrm{W}^{\mathrm{U}})\right\}$

// for AD1BII

$S_{\mathrm{W}} = \left\{\max(\mathrm{W} - \Delta\mathrm{W}_2, \mathrm{W}^{\mathrm{L}}), \max(\mathrm{W} - \Delta\mathrm{W}_1, \mathrm{W}^{\mathrm{L}}), \mathrm{W}, \min(\mathrm{W} + \Delta\mathrm{W}_1, \mathrm{W}^{\mathrm{U}})\right\}$

end if

3. Core Algorithm:

for all $\mathrm{W} \in S_{\mathrm{W}}$ **do**

i. $\tau = \{t | t \in \mathbb{Z}, N_0 + k - \mathrm{W} \leq t \leq N_0 + k - 1\}$

ii. $D_\tau = \{\mathbf{x}_t, \mathrm{y}_t\}$, $t \in \tau$

iii. $\hat{f}_\tau \leftarrow$ obtain a RVM model trained on D_τ

iv. $R^2_\tau(\mathrm{W}) \leftarrow$ Compute R^2_{adj} using Equation 6.17

end for

$\mathrm{W}^* = \underset{\mathrm{W} \in S_{\mathrm{W}}}{\operatorname{argmax}}\ R^2_\tau(\mathrm{W})$

return W^*

Figure 6.20. AD1B (Check$_{R^2}$) Algorithm.

It is expected that AD1B would shrink the window size in favor of recent observations, when a concept drift has been encountered, since the current window size would be comprised of data which has stemmed from both old and new concepts. Furthermore, when the new concept has "equilibrated", AD1B is expected to grow the window size, since a larger number of observations would quite possibly be preferred for the RVM model. To test this hypothesis, probability of using small ($\mathrm{W} < 35$)

and large ($W \geq 35$) window sizes by AD1B method for simulations in the presence of abrupt concept drifts in two levels (S2.1) is estimated and averaged over 20 runs. Since the concept drifts occur at different time instants in each simulation, the region after occurrence of step concept changes are used for aligning the probability trajectories. A moving average window of 40 observations is used to estimate instantaneous probabilities, i.e. $\hat{p}(n) = \sum_{k=n-19}^{20} p(k)$. The results show that when concept drift occurs, W is initially decreased and then increased (Figure 6.21).

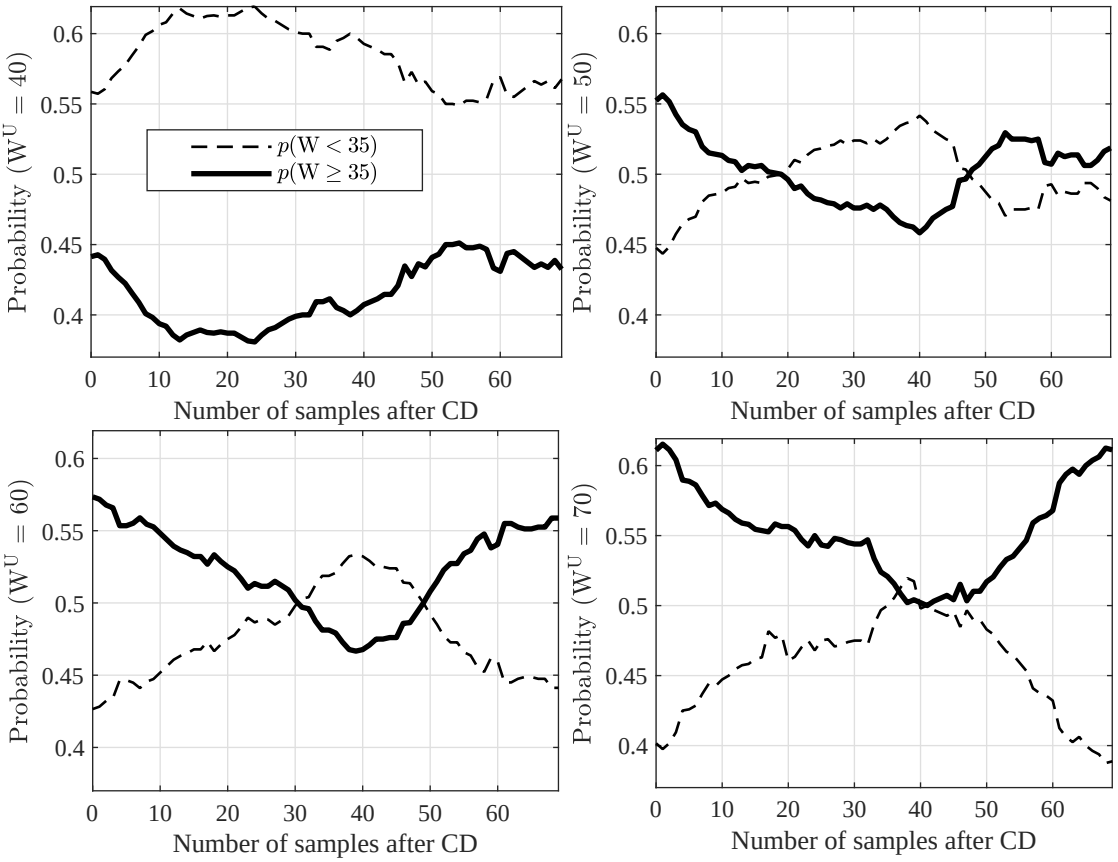

Figure 6.21. Instantaneous probability of using small ($W < 35$) and large ($W \geq 35$) window sizes by AD1B method for CDM S2.1 from CSTR simulations.

AD1B reacts to concept change the quickest when $W^U = 40$, and the speed of reaction declines as W^U increases in Figure 6.21. Let S be the event of using MW with $W < 35$, then S can be defined as a Bernouilli random variable with success probability

$p(\mathrm{S})$ which remains around $\sim 0.4 - 0.6$. AD1B may reduce bias temporarily in some concept drift scenarios, however its variance, $V(\mathrm{S}) = p(1-p)$, is maximum at $p = 0.5$ thus, it is possible that such frequent changes in window size throughout the test set can lead to an increase in variance, hence one should opt for a mechanism to restrict excessive action of adjusting current window size.

The final mechanism suggested for adjusting W is AD1C, which is based on successive F-tests conducted on filtered RVM model error variance estimates. AD1C is motivated by the presumption that variance estimates, similar to R^2_{adj}, can be used as indicators of "quality" of the model estimated from the current window. Given a stable model estimated from its corresponding window at some time k_0, the current model trained on the current window size at k^{th} instance is compared to this "gold standard" estimate. A disruption in homogeneity of the moving window would deteriorate the model quality. This interruption could have two meanings: either the underlying input-output relationship has changed, i.e. a real concept drift, or a virtual drift has occurred in the input space. AD1C is not necessarily concerned with differentiating between these two types of drifts since it is only an explicit drift detection mechanism; however, it is designed with the humble aim of detecting a change in model error, and changing the window size in order to maintain the model precision. After detecting an increase in variance, W would be adjusted to bring the model variance back into control limits. Using this approach, it is aimed to reduce the excessive W adjustment employed by checking R^2_{adj} statistic for every query point.

Let $\hat{f}_\tau$ and $\hat{\sigma}^2_\tau$ denote the model and variance estimate of RVM model trained on data D_τ, in which τ is a set of integers corresponding to time indices of training observations. A new variable, F, is introduced for the k^{th} query as follows:

$$\hat{\sigma}^2_{\tau_k} \coloneqq \hat{\sigma}^2_k, \quad \hat{\sigma}^2_{\tau_0} \coloneqq \hat{\sigma}^2_0$$

$$F_k \coloneqq \frac{\hat{\sigma}^2_k}{\hat{\sigma}^2_0} \tag{6.18}$$

$$F_k \sim \mathcal{F}_{\nu_k,\nu_0} \tag{6.19}$$

Here, justification of F_k being F distributed lies in the assumption that $\hat{\sigma}^2_{k_0}$ and $\hat{\sigma}^2_k$ are independent. The degrees of freedom of two RVM models, the baseline model from time k_0 constructed using D_{τ_0} and a window of size W_0, and the current model estimated for the k^{th} query point from $\mathrm{W_k}$ observations are denoted by ν_0 and ν_k, respectively. Using p'_k to denote the number of input variables retained in the k^{th} RVM model with $\mathrm{W}_k = |\tau_k|$, degrees of freedom is defined as follows:

$$\nu_k \coloneqq \mathrm{W}_k - p'_k - 1 \tag{6.20}$$

Therefore, RVM models constructed for each consecutive query point yield a time series of $\mathcal{F}$ statistics, which is presumed to be F-distributed under the assumptions stated above. On the other hand, model variance estimates are likely to be correlated to a certain degree when the two MWs overlap. In this case, the relation in Equation 6.19 would hold approximately.

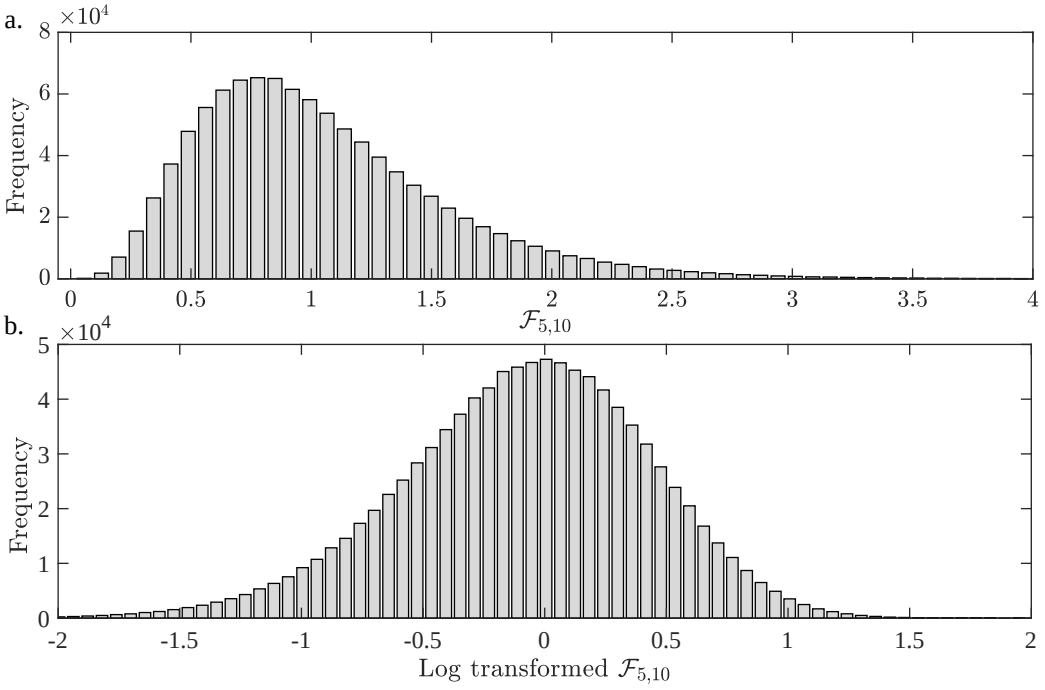

Figure 6.22. Log transformation employed on $\mathcal{F}_{5,10}$ random variable.

So, we prefer a more "practical" approach here, and employ a logarithmic (log) transformation to F ($logF$) in Equation 6.18 to approximate this random variable to a normal distributed one, which would yield a near symmetric distribution, hence make the following analysis easier. As an example, distribution of a $\mathcal{F}_{5,10}$ random variable is shown in Figure 6.22a, in which deviation from symmetry is clearly seen. Employing a log transform to this distribution (Figure 6.22b) yields a near symmetric distribution with mean ≈ 0, skewness $= -0.3$ and kurtosis $= 3.3$, making this statistic more easily accessible, particularly for process monitoring purposes.

Next, an exponential weighted moving average (EWMA) estimator, Z_k, frequently used in statistical process control (SPC) community is defined. EWMA charts are used for detecting deviations from the mean value of a data stream in SPC by performing hypothesis tests at each sample. In EWMA charts, a single parameter λ, with $0 < \lambda \leq 1$, called the EWMA weight is to be tuned. λ determines the importance of the current value of Z_k relative to the previous EWMA estimator computed for $k-1^{th}$ observation, Z_{k-1}. λ is usually set to a value smaller than 0.5; $\lambda = 0.3$ is quite common in the literature hence it is used throughout this thesis [134]. Z_k is defined as follows:

$$Z_0 = E\left\{logF_k\right\} \tag{6.21}$$

$$Z_k \coloneqq (1-\lambda)Z_{k-1} + \lambda logF_k, \quad k > 0 \tag{6.22}$$

$$\sigma_{Z_k} = \sqrt{\frac{\lambda}{2-\lambda}\left(1-(1-\lambda)^{2k}\right)}\sigma_{logF} \tag{6.23}$$

$$\sigma_{Z_k} = \sqrt{\frac{\lambda}{2-\lambda}}\sigma_{logF}, \quad k \to \infty \tag{6.24}$$

Equation 6.21 holds for stationary process, and is used to initialize Z_k, while variance of Z_k is determined via Equation 6.24. Here, $logF$ is approximately normal distributed with approximately zero mean. Figure 6.23 shows a representative example of monitoring estimates of model error variance in the absence of concept drift; ratios of variance estimates (Figure 6.23a) are log transformed to yield a near normal distribution (Figure 6.23b), for which mean, skewness and kurtosis are found to be equal to

-0.05, 0.08 and 2.9, respectively (control limits in Figure 6.23c taken from Table 6.6). Control limits are rarely exceeded, and this shows that Z_k statistic may be used as an indicator of "normal operating conditions" for the RVM predictor on successive MWs.

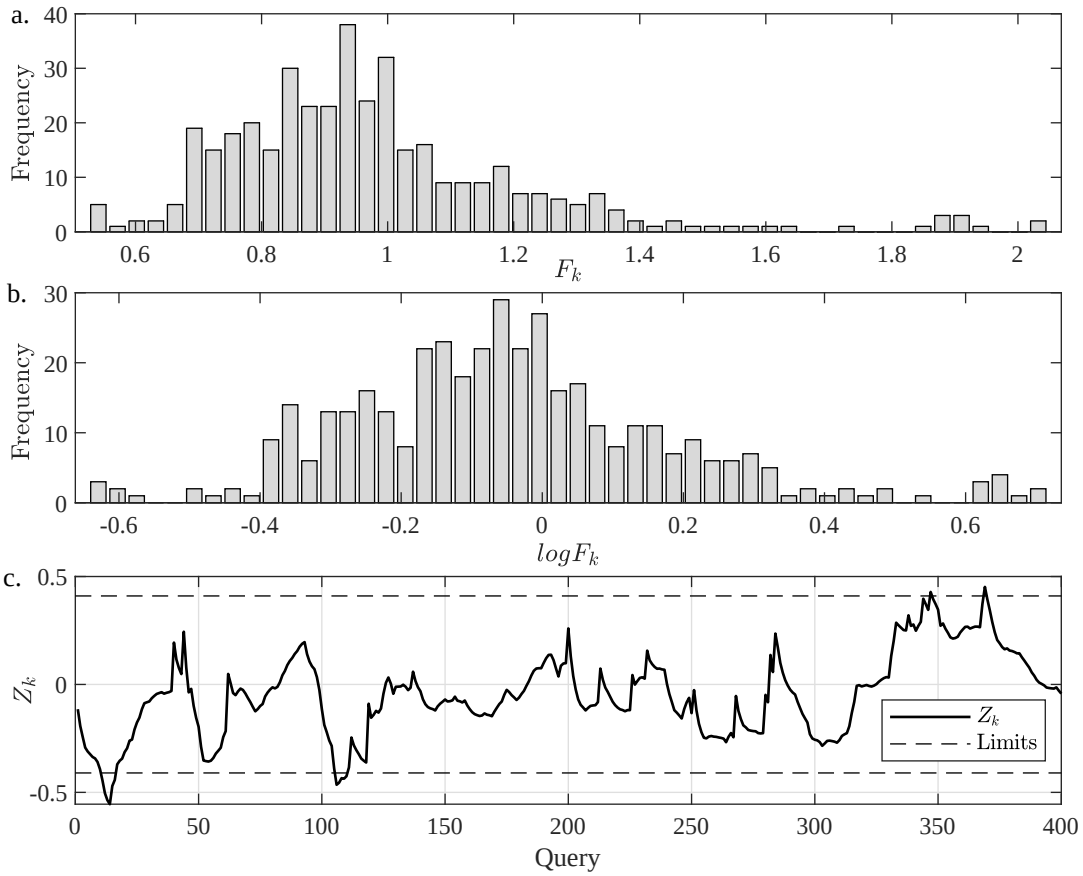

Figure 6.23. Monitoring model error variance estimates in the absence of concept drift on a representative run from S0 in CSTR simulations.

The motivation for using an EWMA chart over other possible statistical process control techniques lies in the fact that concept drifts encountered in the industry are usually expected to exhibit an exponential functional form, and EWMA chart is best suited for detecting such changes. Consider the case in which raw feed material of a process is stored in a container once it arrives a chemical plant. A change in the distributor of feedstock, or feed quality would manifest itself as a step input, for which

the container process acts essentially as a first order filter. Hence the input to the actual process is a filtered step change, resembling the solid line in Figure 6.24c in which the dashed line shows the silhouette of the unfiltered step input. The cumulative sum approach in CUSUM, on the other hand, can address ramp type of drift (Figure 6.24b), which can result from equipment fouling, or sensor deterioration, fairly well. However, computation of CUSUM limits is more involved compared to EWMA, and interpretation of EWMA signal is more straightforward [135]. The drift in Figure 6.24d would be most promptly detected without any filter to the output (Shewhart chart), but this approach would be too sensitive to noise, and react intolerably slow to other concept drifts. Furthermore, the chart would be too sensitive to outliers.

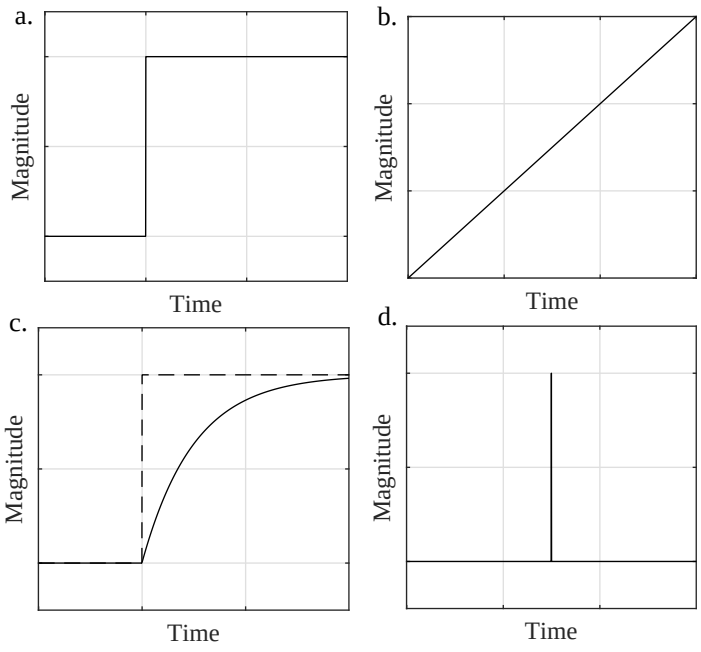

Figure 6.24. A simplified representation of the types of concept drifts that may be encountered in the industrial processes.

When concept drift occurs at k^{th} query, the mean of the EWMA of the new sequence will also change. To detect this change, lower and upper bounds, LB and UB may be used. If Z_k were exactly symmetric distributed, the following relation could

have been used:

$$\text{UB} = -\text{LB} = E\{Z_k\} + l\sigma_{Z_k} \tag{6.25}$$

Furthermore, the usual approach in designing control charts is to first decide on an appropriate average run length (ARL), which is the total number of samples collected until the first out of control sample is observed on average, and then obtain the corresponding confidence limits [136]. This procedure is, however, quite involved. Instead, a polynomial model can be fit to the control limits in order to obtain a lookup table to approximate the confidence bounds for a desired ARL [137]. We suggest adopting a much simpler strategy here. Assuming that degrees of freedom in Equation 6.20 remains between 1 and 200, a 200×200 lookup table is generated for F-distributed random variable, F_{ν_i,ν_j}, for $i, j = 1, 2, ...200$. F-distributed random variables are sampled, log transformed and filtered with $\lambda = 0.3$ as in Equation 6.22, and the quantiles corresponding to the desired type-I error rates, α are obtained. In contrast to the polynomial interpolation method, we prefer a piecewise constant hyperplane fit for function approximation. Since percentiles can change dramatically for different degrees of freedom, computation of control limits is considered in two different regions of degrees of freedom. The degrees of freedom defined as in Equation 6.20 is essentially a function of the sample size, i.e. window size, since the number of relevant vectors retained by RVM varies a lot less than the sample size; hence two separate regions in the lookup table are equivalent to two different ranges of window size. LB and UB are obtained by averaging limits separately in four regions divided at $\nu = 30$ in two dimensions. Upper and lower bounds computed for four different type-I error rates are tabulated in Table 6.6, in which it should be noted that for significantly different ν_0 and ν_k values, deviation between UB and -LB is to be taken into consideration (compare with the approach in Equation 6.25).

Table 6.6. Approximate bounds for different pseudo type-I error rates.

Baseline Model:	$\nu_0 < 30$				$\nu_0 > 30$			
Current Model:	$\nu_k < 30$		$\nu_k > 30$		$\nu_k < 30$		$\nu_k > 30$	
α	**LB**	**UB**	**LB**	**UB**	**LB**	**UB**	**LB**	**UB**
0.001	-0.69	0.69	-0.69	0.50	-0.50	0.64	-0.41	0.41
0.005	-0.59	0.59	-0.59	0.42	-0.42	0.54	-0.35	0.35
0.01	-0.54	0.54	-0.54	0.39	-0.39	0.50	-0.32	0.32
0.05	-0.41	0.41	-0.38	0.29	-0.29	0.38	-0.25	0.25

The lower limit for window size, W^{L}, is introduced so as to avoid unstable parameter estimates when $N \ll p$, in the spirit of the conventional $N > p$ rule. This limit should be case specific, and it mostly depends on the number of input variables and more specifically the effective number of predictors if a sparse regression method is employed for estimating model coefficients. In addition, we suggest the use of a lower threshold, s^{L}, for variance estimates, i.e. if $\hat{\sigma}_k < s^{\mathrm{L}}$, then $\hat{\sigma}_k = s^{\mathrm{L}}$ is used. This limit, s^{L} can easily be obtained from repeatability and reproducibility values of ASTM standards [138]. Any estimate lower than this value should be considered as a sign of numerical instability in parameter estimates, and ill-conditioning during computation can be avoided by simply neglecting this estimate.

When the first query is received, variance monitoring starts by initializing EWMA using Equation 6.21 with $\hat{\sigma}_0^2 = \hat{\sigma}_k^2$ and $\nu_0 = \nu_k$. For the consequent queries, EWMA is continued with Equation 6.18, Equation 6.20 and Equation 6.22 only with no additional action as long as the EWMA value is within the control limits. This step is presumed to bring stability to adjusting W, missing from AD1B. If the k^{th} variance estimate is smaller than the lower bound, i.e. $Z_k <$ LB, EWMA variable is reset and then initialized taking the k^{th} variance estimate as the new baseline ($\hat{\sigma}_0^2 = \hat{\sigma}_k^2$). This indicates that predictive performance of using current window is better than those using the

recent past. On the other hand, if the current variance estimate is above the upper bound it is presumed that (i) the measurement errors on response and/or sensors on process variables have increased, or (ii) a virtual and real concept drift has occurred. The former scenario requires, in our opinion, a different treatment. An increase in the frequency of unmeasured disturbances may be another example for the first case for which the prediction accuracy would deteriorate due to an increase in variance, but not due to bias, resulting from slow frequency concept drifts. The distinction between this perturbation and a concept drift is that we expect a recovery of prediction performance for the latter one using some statistical techniques, whereas the former one cannot be remedied. The initial response of AD1C is to be a little "skeptic" about the reliability of the current variance estimate as an indicator of model quality at that instant. Thus, the corresponding window size, W, is perturbed around the current W using the AD1C_W subroutine (Figure 6.25), which is a mechanism similar to that of AD1B (Check_{R^2}); a set of candidate windows (S_W) in which a more stable variance estimate can possibly be obtained, are compared with the current variance estimate. Here, the same two strategies to determine S_W in AD1B can be employed as well; and we follow the same notation as AD1CI and AD1CII, in which the extensions, I and II represent the scheme used to determine candidate Ws. AD1C_W subroutine returns the W which yields the smallest model variance, and Z_k is recomputed to check if the returned W can provide an estimate within the control limits. If $\hat{\sigma}^2$ computed using the W returned by AD1C_W is able to bring the EWMA variable Z_k back to its control limits, this W is maintained for the next query point. Otherwise, EWMA is reset and reinitialized by accepting the larger estimate as the new baseline level for model quality (Figure 6.26), showing that model variance cannot be significantly decreased by changing W, hence it is more reasonable to decrease the accuracy expectation for the next query point.

AD1C is implemented with preset parameters, significance level, α, lower and upper limits for window size, W^L and W^U, differential window sizes, ΔW in AD1CI, and ΔW_1 and ΔW_2 in AD1CII, and a lower threshold for variance estimates, s^L. Same as in AD1A and AD1B, we suggest a validation step to determine W^U. The other parameters, however, are left to the modeler's judgment and insight. A sensible

1. Input: Historical dataset, $D \coloneqq \{\mathbf{x}_t, \mathrm{y}_t\}_{t=1}^{N}$ with $N = N_0 + k$; current window size, W; differential window size, ΔW (for AD1CI); differential window sizes, $\Delta\mathrm{W}_1$ and $\Delta\mathrm{W}_2$ satisfying $\Delta\mathrm{W}_2 > \Delta\mathrm{W}_1$ (for AD1CII); k^{th} query, $\mathbf{x}'_k$; lower and upper limits for window size, W^{L} and W^{U}

2. Initialize:

// for AD1CI

$S_{\mathrm{W}} = \left\{\max(\mathrm{W} - \Delta\mathrm{W}, \mathrm{W}^{\mathrm{L}}), \mathrm{W}, \min(\mathrm{W} + \Delta\mathrm{W}, \mathrm{W}^{\mathrm{U}})\right\}$

// for AD1CII

$S_{\mathrm{W}} = \left\{\max(\mathrm{W} - \Delta\mathrm{W}_2, \mathrm{W}^{\mathrm{L}}), \max(\mathrm{W} - \Delta\mathrm{W}_1, \mathrm{W}^{\mathrm{L}}), \mathrm{W}, \min(\mathrm{W} + \Delta\mathrm{W}_1, \mathrm{W}^{\mathrm{U}})\right\}$

3. Compare W:

for all $\mathrm{W} \in S_{\mathrm{W}}$ **do**

i. $\tau = \{t | t \in \mathbb{Z}, N + k - \mathrm{W} \leq t \leq N + k - 1\}$

ii. $D_\tau = \{\mathbf{x}_t, \mathrm{y}_t\}$, $t \in \tau$

iii. $\hat{f}_\tau \leftarrow$ learn a RVM model from data, D_τ

iv. $\hat{\sigma}^2_\tau \leftarrow$ Obtain the model variance estimate

end for

$\mathrm{W}^* = \underset{\mathrm{W} \in S_{\mathrm{W}}}{\operatorname{argmin}}\ \hat{\sigma}^2_\tau(\mathrm{W})$

return W^*

Figure 6.25. $\mathrm{AD1C}_{\mathrm{W}}$ subroutine.

1. Input: Historical dataset, $D \coloneqq \{\mathbf{x}_t, \mathrm{y}_t\}_{t=1}^{N_0+k}$; current window size, W; k^{th} query, $\mathbf{x}'_k$; lower and upper limits for window size, W^{L} and W^{U}; lower threshold for variance, s^{L}; type-I error, α; EWMA variable computed for $(k-1)^{th}$ query, Z_{k-1}; baseline model variance and degrees of freedom, $\hat{\sigma}_0^2$ and ν_0; current model variance and degrees of freedom, $\hat{\sigma}_k^2$ and ν_k

2. Initialize:

Given α, obtain LB and UB from Table 6.6 for the corresponding degrees of freedom.

if $\hat{\sigma}_k < s^{\mathrm{L}}$ **then**

$\hat{\sigma}_k \leftarrow s^{\mathrm{L}}$

end if

$F_k \leftarrow$ Use 6.18; $Z_k \leftarrow$ Use 6.22

3. Core:

if $Z_k >$ UB **then**

i. $\mathrm{W}^* \leftarrow \mathrm{AD1C_W}(D, \mathrm{W}, \mathbf{x}'_k, \mathrm{W}^{\mathrm{L}}, \mathrm{W}^{\mathrm{U}})$

ii. $\hat{\sigma}_k^{*2} \leftarrow$ Obtained from RVM model using W^*

iii. $F_k^* \leftarrow$ Use 6.18; $Z_k^* \leftarrow$ Use 6.22

if $Z_k^* <$ UB **then**

$\mathrm{W} \leftarrow \mathrm{W}^*$; $F_k \leftarrow F_k^*$; $Z_k \leftarrow Z_k^*$

else

Reset EWMA

$\hat{\sigma}_0^2 \leftarrow \hat{\sigma}_k^2$; $\nu_0 \leftarrow \nu_k$; $Z_k \leftarrow 0$

end if

else if $Z_k <$ LB **then**

Reset EWMA

$\hat{\sigma}_0^2 \leftarrow \hat{\sigma}_k^2$; $\nu_0 \leftarrow \nu_k$; $Z_k \leftarrow 0$

end if

return W, Z_k, $\hat{\sigma}_0^2$, ν_0

Figure 6.26. AD1C (Check$_s$) algorithm.

lower limit, W^L, a threshold for minimum variance, s^L which takes the standards in laboratory measurements into account, and the desired type-I error rate should be preset (Figure 6.26). Note that for the sake of brevity, we reserve the two method names, AD1CI and AD1CII only for when these two ways of determining candidate Ws are compared. For other times, AD1C is employed as in AD1CII, with differential window size settings, $\Delta W_1 = 0.2$ and $\Delta W_2 = 0.5$.

Implementation of AD1C is demonstrated on a representative example, taken from a simulation run from the two-level step disturbance model (S2.1) in Figure 6.27. Z, the EWMA variable representing the increase/decrease in model variance (Figure 6.27b), was found to be inside the confidence limits for 93.5% of the trajectory, and Z was reset for only 6% of the observations. Figure 6.27c shows whether the final model estimate is within the bounds: The label outside on the ordinate corresponds to Z_k samples out of confidence limits (peaks in Figure 6.27b), while horizontal axis on the inside* row represents the samples, for which changing W made it possible to bring model variance back within the limits, and W was updated. Here, W was changed only for two observations out of 400 instances, and these change points can be seen on the disturbance trajectory (with vertical dashed lines) in Figure 6.27a. These change points approximately correspond to step changes in the disturbance, showing that AD1C serves its purpose. AD1C algorithm, (Figure 6.26), fine tunes the MW size around its current size, with the aim of reducing RVM model error, and maintain the prediction accuracy. This operation in independent of the query point's relevance to the current MW, but only a check of the quality of the potential model to be used for prediction. As AD1A checks for the query point's position in the predictor space defined by the MW, incorporating AD1A into AD1C is expected to act against a wider range of concept drift scenarios and adjust W accordingly (Figure 6.28). Hence it should be emphasized that AD1C is not used as a standalone mechanism, but is used simultaneously with AD1A in the current thesis. In summary, W is adjusted by AD1C only if Mahalanobis distance criterion is satisfied, showing that priority is given to Mahalanobis distance criterion (Figure 6.28).

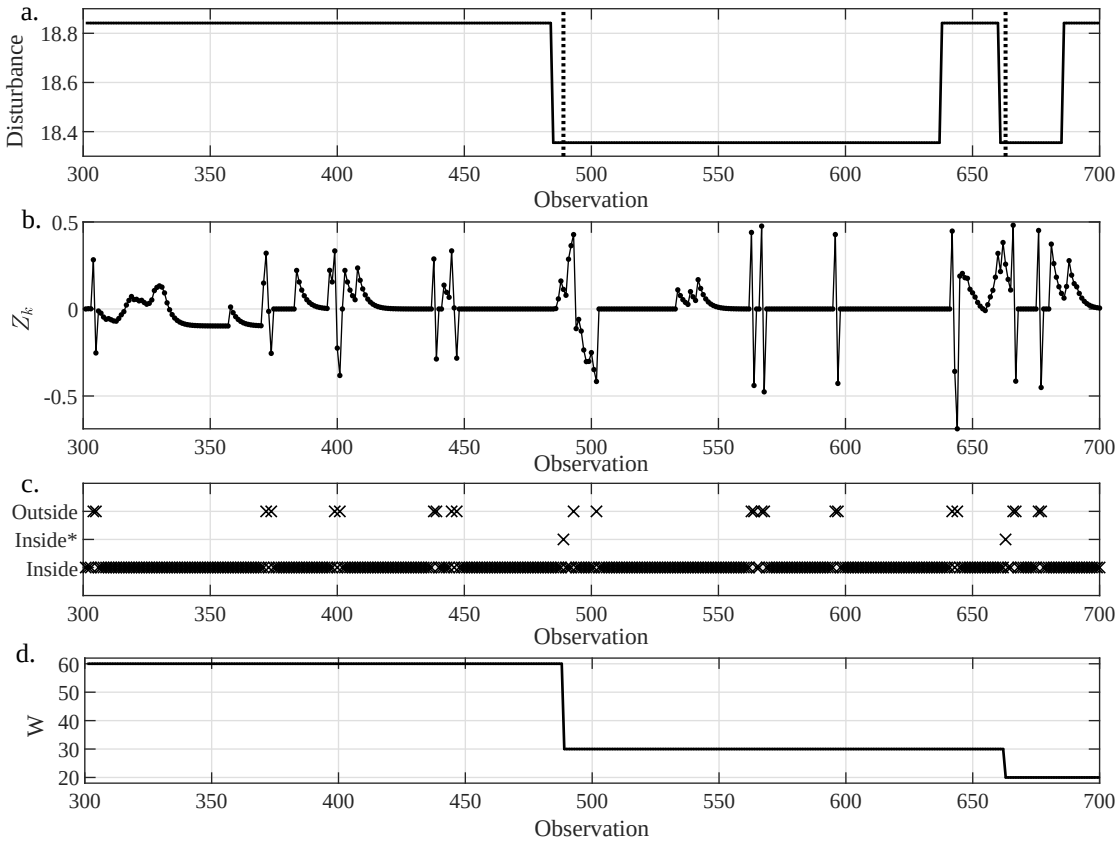

Figure 6.27. AD1C used on a two-level step disturbance (S2.1) with low frequency (for $W^L = 20$, $W^U = 60$, and $\alpha = 0.001$)

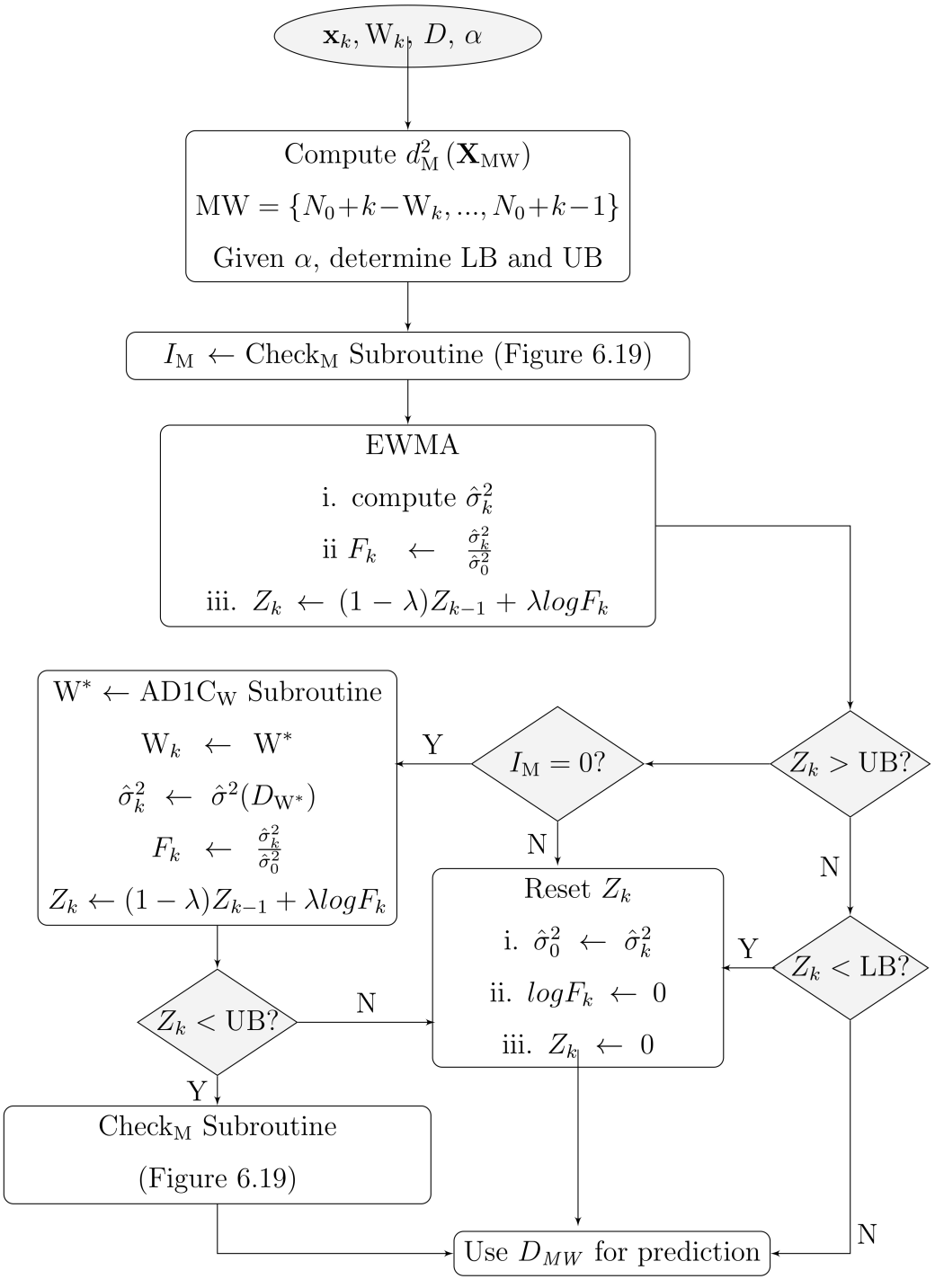

Figure 6.28. Flowchart for the proposed implementation of AD1C (Check$_{\mathrm{S}}$ algorithm) in conjunction with AD1A in this work.

AD1C seems to have two main advantages:

- Being a more conservative adaptive mechanism (the change in W is much less frequent compared to that in AD1B), it can easily be used with sparse regression methods which perform embedded feature selection during training. It is a viable alternative to more aggressive adaptation strategies especially when model estimates have high variance; by making fewer but more effective changes, it can reduce the inherent variability of sparse methods.
- It can handle transition regions, which are encountered following abrupt changes, fairly well. Transition regions can be problematic as it is more difficult to maintain a balance between stability and plasticity when data is that heterogeneous; wrong actions can lead to unstable windows in between two different target concepts, and increased prediction error. AD1C avoids this problem as it checks the homogeneity of the new window if it makes any changes to it, and it reverts its action unless the new region is stable enough.

AD1C has both similarities to and distinctions from various prominent algorithms based on hypothesis tests. Searching for a subwindow of recent observations in the current window with the aim of decreasing variance is similar to that aimed by ADWIN [139], yet AD1C is specifically developed with the aim of addressing multicollinearity and redundancy in process data, hence an extensive search to find minimum $\hat{\sigma}^2$ is avoided. Consecutive t-tests performed by ILLSA resembles our motivation in designing a control chart; they aim to detect regions in which functional relationship in data changes by comparing means of errors [41]. However, we adopt a less ambitious approach here by merely considering homogeneity of data, and using model variance as an indicator for disruption in this homogeneity. Adaptive windowing used in LARPLS and LASS relies on the validation error on previous sample [34,120], but these methods are expected to be rather sensitive to noise, as only a single sample of prediction error is used to adjust W (see the next section).

6.3.1.3. AD2: Moving Window and JITL Ensemble. In ensemble modeling approach to adaptive learning, one should initially determine base models, and decide on an updating scheme for parameters and weights of these base models as new observations arrive. The difficulty in this approach lies in an unbiased and low variance selection of the number of base models, corresponding training observations of each base learner from the historical dataset, and adapting model weights. In addition to updating procedure, frequency of ensemble updates plays a critical role on the adaptive model performance. In summary, all these issues should be addressed in an online manner in a data stream scenario, hence making ensemble learning a rather complicated task. We suggest a moderately less involved adaptive mechanism utilizing the moving window and the JITL approach simultaneously. Learning within the moving window is implemented as an independent mechanism to that of the JITL section, and AD1C mechanism from Figure 6.28 is selected for adaptive windowing. Additionally, a JIT learner runs in parallel with the MW. In JITL model, a RVM model is trained using local region consisting of consecutive observations in close proximity to the k^{th} query point in the predictor space. It is not exactly based on a "nearest neighbor" model per se, however, with this approach temporal relations in the training set can be preserved, and the proximity to the query point in the input space can be taken into account at the same time. To avoid overlapping of two models, MW and JITL, the search for a an appropriate local region in JITL algorithm is strictly employed on only the observations outside the current MW. Otherwise, MW and JITL models might sample from overlapping regions, likely to increase the covariance of the predictions of these methods, thereby increasing prediction error.

JITL algorithm proposed in this thesis can be examined in three segments:

- selection of local regions,
- construction and assessment of local MWs, and
- the final prediction JITL subroutine.

The initial step of the algorithm, selection of local regions, is based on predictor space similarity. Euclidean distances (Equation 3.44) between the predictors of the query point and those of the training data, excluding the MW, are computed (d), and a threshold distance (d_{max}) is defined to distinguish relevant and irrelevant observations in the predictor space.

$$d_{max} = \text{median}(d) - l\text{mad}(d) \tag{6.26}$$

Here, median denotes the median operator, and mad is the mean absolute deviation from the median. The idea is to determine a robust location estimate for the distance between the historical and the query data points, and compute a lower distance limit using a robust spread estimate from the distance values. A larger l would yield a smaller d_{max}, and a smaller l would increase d_{max}. In the current thesis, $l = 0.5$ is used. It should be noted that d_{max} would directly affect the number of "candidate" local regions. In a conventional JITL algorithm, samples which satisfy $d < d_{max}$ would be selected and included in a model. However, this would distort the temporal relation between samples, and homogeneity of concepts would be neglected, possibly worsening prediction accuracy. Hence, we would like to select contiguous regions, comprised of relevant samples, instead of relevant samples, scattered at different times. For this reason, distances are filtered using a temporal weighted moving average operator. For a given window width T, and distances to k^{th} query point d_i, $i = 1, 2, ...C$, with C is the total number of observations allowed in localization search, i.e. all historical observations excluding the current MW, the following filter is used:

$$w_{i,j} = e^{-\gamma(j-i)^2} \tag{6.27}$$

$$\hat{d}_i = \frac{\sum_{j=i-T/2+1}^{i+T/2} w_{i,j} d_j}{\sum_{j=i-T/2+1}^{i+T/2} w_{i,j}} \quad \text{with } i \in I = \{t \mid t \in \mathbb{Z}, T/2 < t < C - T/2\} \tag{6.28}$$

Weights, $w_{i,j}$ decay exponentially with the rate defined by the constant γ, which is taken to be proportional to the window width T. In this thesis, $T = \widetilde{\mathrm{W}}$ and $\gamma = 0.2/T$. As a result, distance of a historical sample to the query point is adjusted (smoothed) using the distances of the neighboring samples to the query point, hence temporal relation information is partially added to the distance metric. Taking window width T to be 30, weights of observations in a window decay as shown in Figure 6.29.

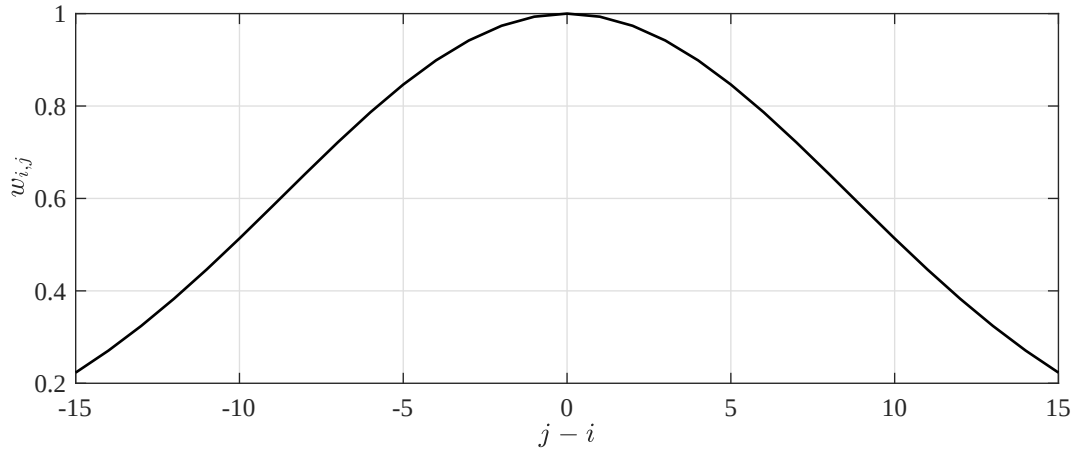

Figure 6.29. A representative example for the weights $w_{i,j}$ computed using $T = 30$ in Equation 6.27.

The main reason why the distances are filtered in this case is not to fight with noise. Filtered distances are used to identify local contiguous regions, which are comprised of observations which are in close proximity to the query point in the input space, and this proximity is checked by means d_{max}. Using a threshold results in binary outcomes; furthermore, false positives are inevitable and these create discontinuities in the time domain, i.e. there would be many single observations, or segments containing a small number of observations repeated by time intervals. In order to obtain contiguous local regions, comprised of consecutive observations, the filtering operation, discussed above, is employed. To ensure that each region comprises enough observations for training, only the regions of size at least 75% of $\widetilde{\mathrm{W}}$ are retained. Figure 6.30a shows distances d, calculated in the predictor space for the 473^{th} observation in one of the simulation runs from CDM S2.2. Filtering d yields $\hat{d}$ (thick solid line), and the relevant

instances which fall under the threshold d_{max} in the database are retained (denoted with black crosses in Figure 6.30b).

The relevant regions are described by their starting and ending time indices, stored in RR_i for each segment i (Figure 6.31).

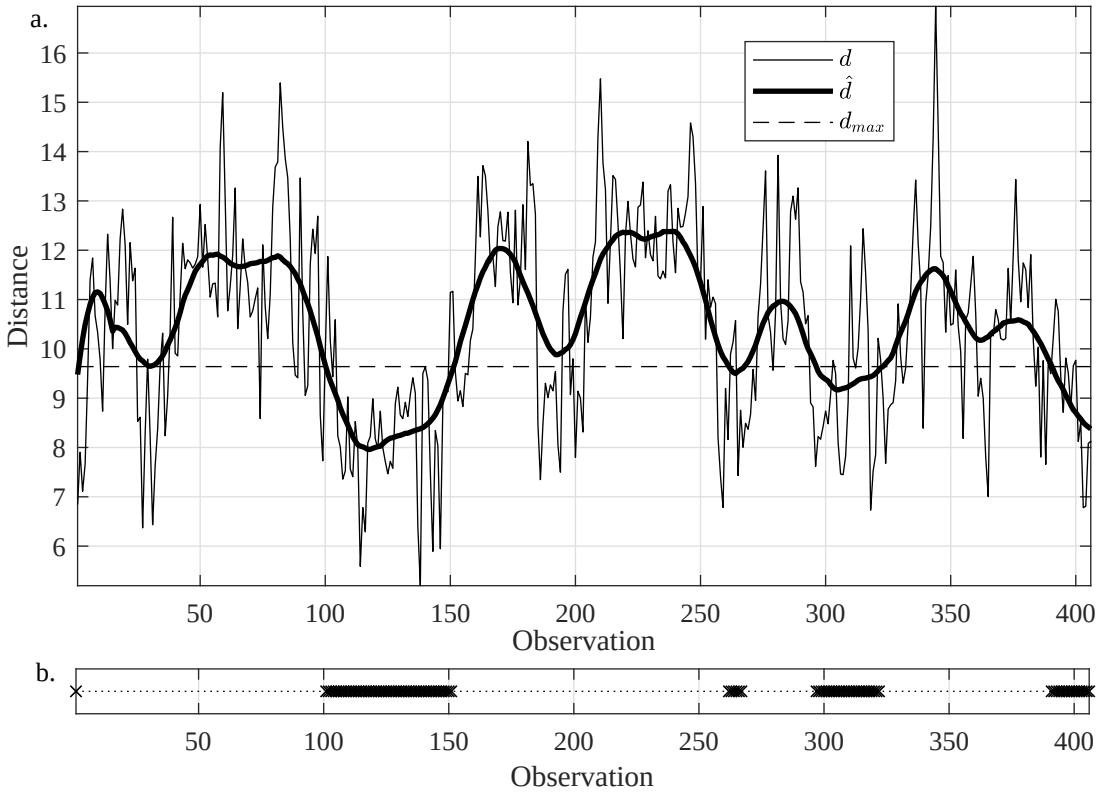

Figure 6.30. A representative example for the filtered distances $\hat{d}$, and the threshold d_{max} obtained from distances d using $l = 0.5$ in Equation 6.26.

The second step in the current implementation of JITL algorithm comprises construction and assessment of local MWs. Here the notation explained in Equation 6.10 to Equation 6.12 is followed, i.e., $\widetilde{\mathrm{MW}}$ is the set of time ordered indices of observations in the JITL window, and the corresponding window size is $\widetilde{\mathrm{W}}$. For each local contiguous region, multiple training sets are generated via sliding the window by δ observations. A representation of the procedure is shown in Figure 6.32, which follows the example in Figure 6.30. Note that the first observation (cross marker in black) in Figure 6.30b was

1. Input: Historical dataset, $D \coloneqq \{(\mathbf{x}_t, \mathrm{y}_t)\}_{t=1}^{N_0+k}$; current window size, W; k^{th} query tuple, $(\mathbf{x}'_k, \mathrm{y}'_k)$; JITL window size, $\widetilde{\mathrm{W}}$

2. Initialize:

i. $\mathrm{MW} = \{t | t \in \mathbb{Z}, N_0 + k - \mathrm{W} \leq t \leq N_0 + k - 1\}$

ii. $\tau = \{t | t \in \mathbb{Z}, 1 \leq t \leq N_0 + k - W - 1\}$

3. Selection of local regions:

i. $d_j \leftarrow \|\mathbf{x}'_k - \mathbf{x}_j\|$; Compute d_{max} using Equation 6.26; $\hat{d}_j \leftarrow$ filter d_j as in Equation 6.27 and 6.28 for $j \in \tau$

ii. $I_j = \begin{cases} 1 & \text{if } \hat{d}_j < d_{max}; \\ 0 & \text{otherwise;} \end{cases}$

iii. $r = \{j \in \tau \mid I_j = 1\}$

iv. $R = \{1\} \cup \left\{ i \in 2, 3, ...|r| \middle| r_i - r_{i-1} \geq \widetilde{\mathrm{W}}/2 \right\} \cup |r|$

if $R = \varnothing$ **then**

  $D_{\mathrm{MW}} = \{\mathbf{X}_{\mathrm{MW}}, \mathbf{y}_{\mathrm{MW}}\}$

  $\hat{\mathrm{y}}'_k \leftarrow \hat{f}(D_{\mathrm{MW}}, \mathbf{x}'_k)$

  return $\hat{\mathrm{y}}'_k$; Abort JITL Algorithm

else

  for all $i = 1, 2, ...|R| - 1$ **do**

    $RR_i = \{t \mid t \in \mathbb{Z}, r(R_i) \leq t \leq r(R_{i+1} - 1)\}$

    if $|RR_i| < 3\widetilde{\mathrm{W}}/4$ **then**

      $RR_i = \varnothing$

    end if

  end for

end if

return RR

Figure 6.31. Selection of local regions.

not retained since the number of observations was too few to costruct a local region. Using $\widetilde{\mathrm{W}} = 30$, the first $\widetilde{\mathrm{MW}}$ commences at $100 - 15 = 85$ (Figure 6.32b). Sliding the MW by $\delta = \widetilde{\mathrm{W}}/5 = 6$ observations, a total of 8 $\widetilde{\mathrm{MW}}$s are constructed, representing local temporal operating regions in close proximity to the query point. This operation is repeated for each local segment seen in Figure 6.32a, and MWs similar to those in Figure 6.32b are obtained. Note that except for a few observations (observations 124-131 in Figure 6.32c), the selected local regions (using the algorithm in Figure 6.31) are comprised of homogeneous concepts in terms of the unmeasured disturbance variable. Moreover, the value of the unobserved disturbance at the query point is equal to the those at observations within the last three homogeneous regions (not shown).

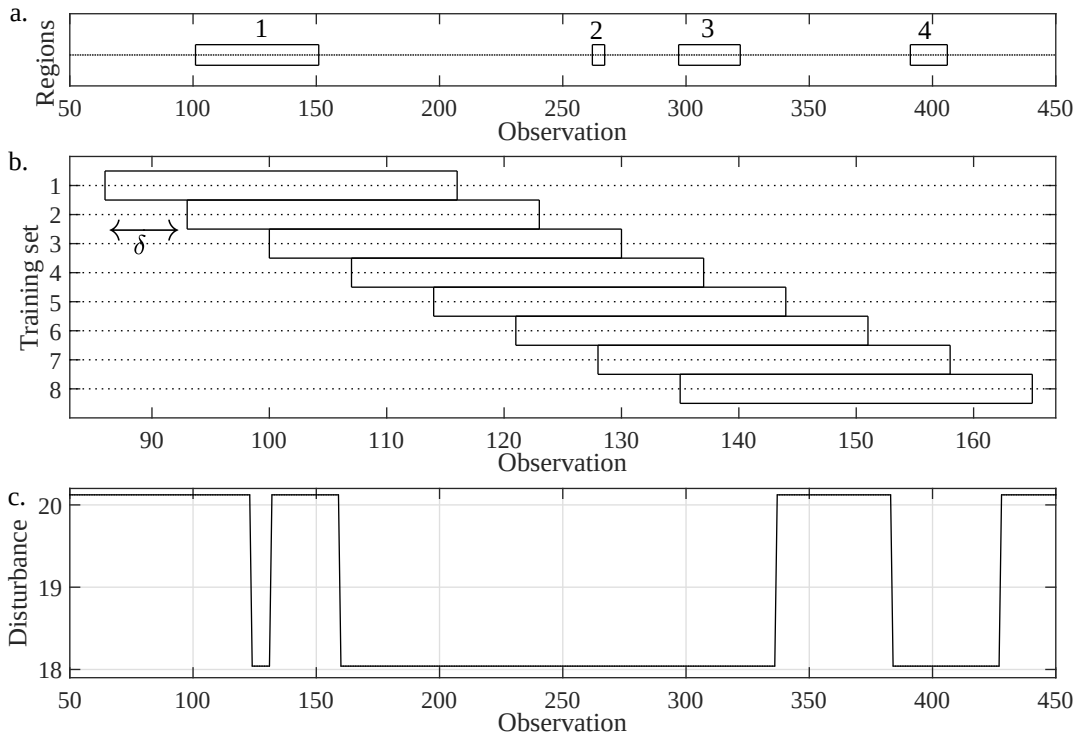

Figure 6.32. Four local regions are identified using Figure 6.31 in a, and a sliding window is employed on the first region to obtain different training sets in b, real CDs in unmeasured disturbance are shown in c.

After contiguous regions in the historical data in close proximity to the query point in predictor space are identified, similarity of these regions to the query point

in concept space is to be examined. Since the query point's response is yet unknown, the best we can hope for is to assume that the current concept is similar to those in the very recent past. Hence, validation errors (VEs) of training sets obtained from all regions are calculated on the last mp observations, in which mp is the maximum number of past observations used for validation (Figure 6.33).

1. Input: Historical dataset, $D \coloneqq \{(\mathbf{x}_t, \mathrm{y}_t)\}_{t=1}^{N}$; A set of indices for the window, τ; k; maximum number of past validation points, mp

2. Initialize:

$D_\tau = \{\mathbf{X}_\tau, \mathbf{y}_\tau\}$

$S_v = \{t | t \in \mathbb{Z}, N_0 + k - mp \leq t \leq N_0 + k - 1\}$; $i \leftarrow 1$

3. Construction and assessment of local MWs:

for all $v \in S_v$ **do**

  if $v \in \tau$ **then**

    $\tau = \tau \setminus \{t \mid t \in \tau, t \geq v\} \cup \{t \mid t \in \mathbb{Z}, \min\tau - mp + i - 1 \leq t \leq \min\tau - i\}$

  end if

  $\hat{\mathrm{y}}_v \leftarrow \hat{f}(D_\tau, \mathbf{x}_v)$; $\mathrm{VE}_i = \mathrm{y}_v - \hat{\mathrm{y}}_v$

  $i \leftarrow i + 1$

end for

return VE

Figure 6.33. Construction and assessment of local MWs.

The weighted average of VEs obtained for training sets generated from each local region is computed by giving higher weights to more recent observations. An exponential weighting scheme is used to obtain the average VE ($\overline{\mathrm{VE}}$) for the m^{th} training set as follows:

$$\overline{\mathrm{VE}} = \frac{\sum_{i=0}^{mp-1} \lambda(1-\lambda)^i \mathrm{VE}_{mp-i}}{\sum_{j=0}^{mp-1} \lambda(1-\lambda)^j} \tag{6.29}$$

Here, λ is the weight parameter which determines how fast the weights decay for consequent observations. In this thesis, $\lambda = 0.3$ is used, and using $mp = 3$, $\overline{\text{VE}}$ is may be calculated as:

$$\overline{\text{VE}} = \frac{0.3\text{VE}_3 + 0.3(0.7)\text{VE}_2 + 0.3(0.7)^2\text{VE}_1}{0.3 + 0.3(0.7) + 0.3(0.7)^2} = 0.46\text{VE}_3 + 0.32\text{VE}_2 + 0.23\text{VE}_1$$

Smaller weights are given to older observations as the weights decay from 0.46 to 0.23 when $\overline{\text{VE}}$ is computed. This procedure is repeated for the rest of the regions, and $\overline{\text{VE}}$s are obtained for all candidate training sets. Figure 6.34 shows the minimum VE obtained from training sets generated for each local region (RR_1 to RR_4) which were identified as seen in Figure 6.30. Bars are VEs calculated on three (470, 471 and 472$^{\text{th}}$) past observations, and the darkest bars belong to the oldest observation in S_v.

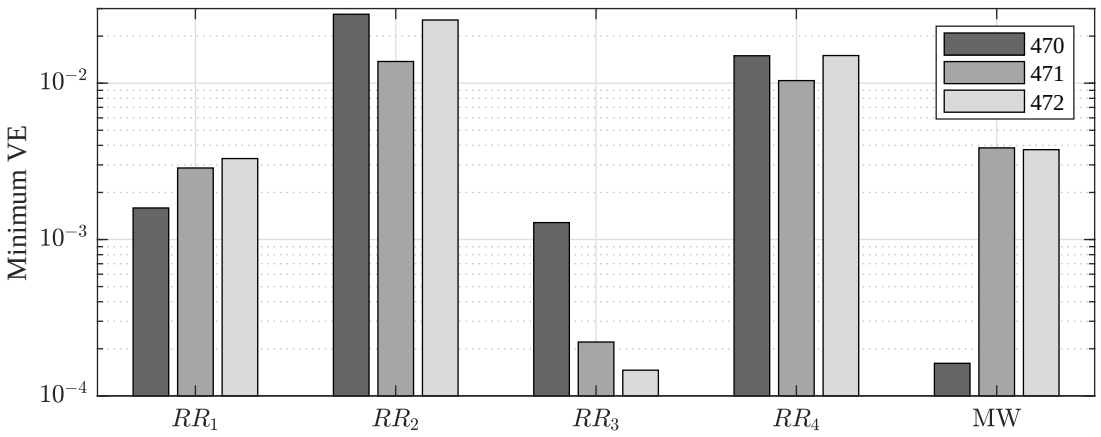

Figure 6.34. Minimum VE obtained in training sets generated for local regions, RR_i, and the current MW on three past observations ($mp = 3$), different colored bars correspond to 470, 471, and 472$^{\text{th}}$ observation, from darker to lighter color.

The final step in JITL procedure is to combine the best RVM model among the $\widetilde{\text{MW}}$s with the RVM model from MW, since BMA with posterior distributions is widely used in the soft sensor design literature for combining base learners [13, 41, 125, 140], the same approach is adopted in the current thesis using $M = 2$ in Equation 3.56. Validation errors are computed on the same recent observations also by the MW method

(see the last three bars on the far right in Figure 6.34), taking the prior probabilities of RVM models obtained from MW and $\widetilde{\text{MW}}$ sets to be equal, the likelihood of each model is determined using validation errors on recent observations.

$$p(\mathcal{L}_m|D_{\tau_m}) \propto 1/\overline{\text{VE}}_m$$
$$w_m = \frac{1/\overline{\text{VE}}_m}{\sum_{m\in M} 1/\overline{\text{VE}}_m} \tag{6.30}$$

The final prediction is simply the sum of likelihoods multiplied by individual model predictions, i.e. the weighted sum of predictions from D_{MW} and $D_{\widetilde{\text{MW}}}$ as in Equation 3.54.

$$\hat{\text{y}}'_{k,\text{MW}} = \hat{f}(D_{\text{MW}}, \mathbf{x}'_k) \tag{6.31}$$
$$\hat{\text{y}}'_{k,\widetilde{\text{MW}}} = \hat{f}(D_{\widetilde{\text{MW}}}, \mathbf{x}'_k) \tag{6.32}$$
$$\hat{\text{y}}'_k = w_{\text{MW}}\hat{\text{y}}'_{k,\text{MW}} + w_{\widetilde{\text{MW}}}\hat{\text{y}}'_{k,\widetilde{\text{MW}}} \tag{6.33}$$

The previous two subroutines involving selection, construction and assessment of local regions are combined and the final prediction is obtained as described in Figure 6.35. Our JITL approach (AD2) makes it possible to solve two problems at once. First, it selects local regions closest to the query point in predictor space. Philosophy of localization in both CoJITL and NGR_JIT is perhaps most similar to ours, however their algorithm stores a large number of local regions for comparison and this can lead to an increase in variance; number of local models can be restricted to a low value but then one should find the optimum number of models in that case [38, 87]. Second, AD2 method takes concept changes in local regions into consideration, unlike CoJITL and NGR_JIT methods, aiming to increase predictive accuracy of the JITL procedure. AD2 searches for a window among the local regions with the operating conditions closest to the recent one, in a less assuming way than diving the historical process into different presumed states, as usually performed in the literature. For instance

1. Input: Historical dataset, $D \coloneqq \{(\mathbf{x}_t, \mathrm{y}_t)\}_{t=1}^{N0+k}$; current window size, W; k^{th} query tuple, $(\mathbf{x}'_k, \mathrm{y}'_k)$; JITL window size, $\widetilde{\mathrm{W}}$; maximum number of past validation points, mp

2. Initialize:

i. $\mathrm{MW} = \{t | t \in \mathbb{Z}, N_0 + k - \mathrm{W} \leq t \leq N_0 + k - 1\}$; $\delta \leftarrow \min\left\{i \in \mathbb{Z} | i \geq \widetilde{\mathrm{W}}/5\right\}$

ii. $\tau = \{t | t \in \mathbb{Z}, 1 \leq t \leq N_0 + k - 2\}$; $D_\tau = \{\mathbf{X}_\tau, \mathbf{y}_\tau\}$

3. Validation on Previous Observations:

i. Use D, W, $(\mathbf{x}'_k, \mathrm{y}'_k)$ and $\widetilde{\mathrm{W}}$ as inputs to selection of local regions algorithm (Figure 6.31) to determine RR.

for $j \leftarrow 1$ to $|RR|$ **do**

$r_{min} = \min RR_j - \widetilde{\mathrm{W}}/2$; $r_{max} = \max RR_j + \widetilde{\mathrm{W}}/2$

$M = \left(\mathrm{range}(RR_j) + \widetilde{\mathrm{W}}\right)/\delta + 1$

for $m = 1$ to M **do**

i. $\tau_{m,j} = \left\{t | t \in \mathbb{Z}, r_{min} + (m-1)\delta + \widetilde{\mathrm{W}} \leq t \leq r_{min} + (m-1)\delta\right\}$

ii. Use D, $\tau_{m,j}$, k and mp as inputs to construction and assessment of local MWs algorithm (Figure 6.33) to obtain $\mathrm{VE}_{m,j}$

iii. $\overline{\mathrm{VE}}_{m,j} \leftarrow$ Use Equation 6.29 for mp past observations.

end for

end for

ii. Use D, MW, k and mp as inputs to construction and assessment of local MWs algorithm (Figure 6.33) to obtain $\mathrm{VE}^{\mathrm{MW}}$.

$\overline{\mathrm{VE}}^{\mathrm{MW}} \leftarrow$ Use Equation 6.29 for mp past observations.

4. Obtain Final Prediction:

i. $\widetilde{\mathrm{MW}} \leftarrow \underset{\tau_{m,j}}{\mathrm{argmin}}\ \overline{\mathrm{VE}}$

ii. $w_{\mathrm{MW}} \leftarrow$ Obtain model weight using Equation 6.30.

iii. $D_{\mathrm{MW}} = \{\mathbf{X}_{\mathrm{MW}}, \mathbf{y}_{\mathrm{MW}}\}$; $D_{\widetilde{\mathrm{MW}}} = \left\{\mathbf{X}_{\widetilde{\mathrm{MW}}}, \mathbf{y}_{\widetilde{\mathrm{MW}}}\right\}$

iv. $\hat{\mathrm{y}}'_k \leftarrow w_{\mathrm{MW}} \hat{f}(D_{\mathrm{MW}}, \mathbf{x}'_k) + (1 - w_{\mathrm{MW}}) \hat{f}(D_{\widetilde{\mathrm{MW}}}, \mathbf{x}'_k)$

return $\hat{\mathrm{y}}'_k$

Figure 6.35. JITL main algorithm.

ILLSA and DLOER both use hypothesis tests to detect regions of different concepts [41,45]. Ensemble learner developed for soft sensor design, SELPLS, is another example of offline operating state identifier, which employs Chi-square tests on variance of prediction errors in addition to a t-test on their mean [140]. While these methods make it possible to divide the historical data into different operating regions, i.e. concepts, in an offline manner, hence decrease online computational burden, the initial bias introduced during offline learning is expected to be maintained through the online implementation. It should be noted that, a possibly significant, bias is inevitable due to the violation of the i.i.d. assumption for fitting and prediction errors, since the regions, identified by these methods overlap by several observations. Our JITL method avoids this problem via searching for and selecting the best local contiguous region for every query point.

6.3.2. Evaluation of Proposed Adaptive Mechanisms

In this section, we evaluate the performance of adaptive algorithms developed using these ADs (on their own or in conjunction with one another) on two datasets, synthetic data from dynamic CSTR in Section 5.1.2, and real data from a debutanizer column in Section 5.2.1. Experiments follow the same structure as those in previous section on study of PLS and RVM learners in this chapter, however here we are not interested in extending FIR models with additional lags. From CSTR data, CDMs S1.1, S1.2, S2.1, S2.2, S3.1, S3.2, S3.3 and S3.4, all of which comprises 20 runs in total, are used to assess generalization performance of the suggested algorithms. Order of FIR model is fixed at three, i.e. only two of lagged predictors are included in the model ($n = 2$ in Equation 5.8), hence the input matrix is of size 700×57. 700 observations in simulation data are divided as 300/400 into training/test sets. For the debutanizer dataset the order is taken to be one, hence no lags are included in the model. A 1000/1394 division, which we previously used in Section 6.2.2, is adopted to obtain training/test sets.

We base our evaluation on the following performance metrics: root mean squared error (RMSE) of predictions, mean absolute error (MAE) and maximum absolute error (maxAE) which can be defined as follows:

$$\text{RMSE} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(\text{y}_i - \hat{\text{y}}_i)^2}$$

$$\text{MAE} = \frac{1}{N}\sum_{i=1}^{N}|\text{y}_i - \hat{\text{y}}_i|$$

$$\text{maxAE} = \max|\mathbf{y} - \hat{\mathbf{y}}|$$

Here, N is the total number of test points, $\mathbf{y}$ and $\hat{\mathbf{y}}$ denote vectors comprised of N observations and predictions of the response variable in a given test set. For CSTR data, we will take the average of these metrics over 20 simulation runs to obtain $\overline{\text{RMSE}}$, $\overline{\text{MAE}}$, $\overline{\text{maxAE}}$, and sdRMSE, standard deviation of RMSEs, is also used as a performance metric.

First, prediction accuracy of MW is compared with that of AD1A on simulation data with respect to various W (Figure 6.36). It was presumed that AD1A would handle virtual concept drift, while its performance on real drift is unknown. The concept drift scenarios from CSTR simulation are dominated by real concept drift, thus improvement from using AD1A over MW is marginal in terms of average prediction error over a range of Ws. However, AD1A yields more accurate predictions in general for small window sizes (W = 30). This suggests small constant window sizes may yield gross prediction errors, which may be remedied by AD1A (S1.2 and S3.2 in Figure 6.36). To examine these methods in terms of their gross prediction errors, $\overline{\text{maxAE}}$ is used (Figure 6.37). In all CDM's, AD1A can drastically reduce $\overline{\text{maxAE}}$ at small Ws, meaning that it can effectively overcome the problem of encountering occasional gross errors in MW by momentarily increasing W when the query point does not belong to its respective MW. At larger Ws, however, performance of AD1A in terms of $\overline{\text{maxAE}}$ starts to deteriorate relative to MW, especially for CDMs with more aggressive types of real drifts (S2.1, S2.2, S3.3 and S3.4 in Figure 6.37).

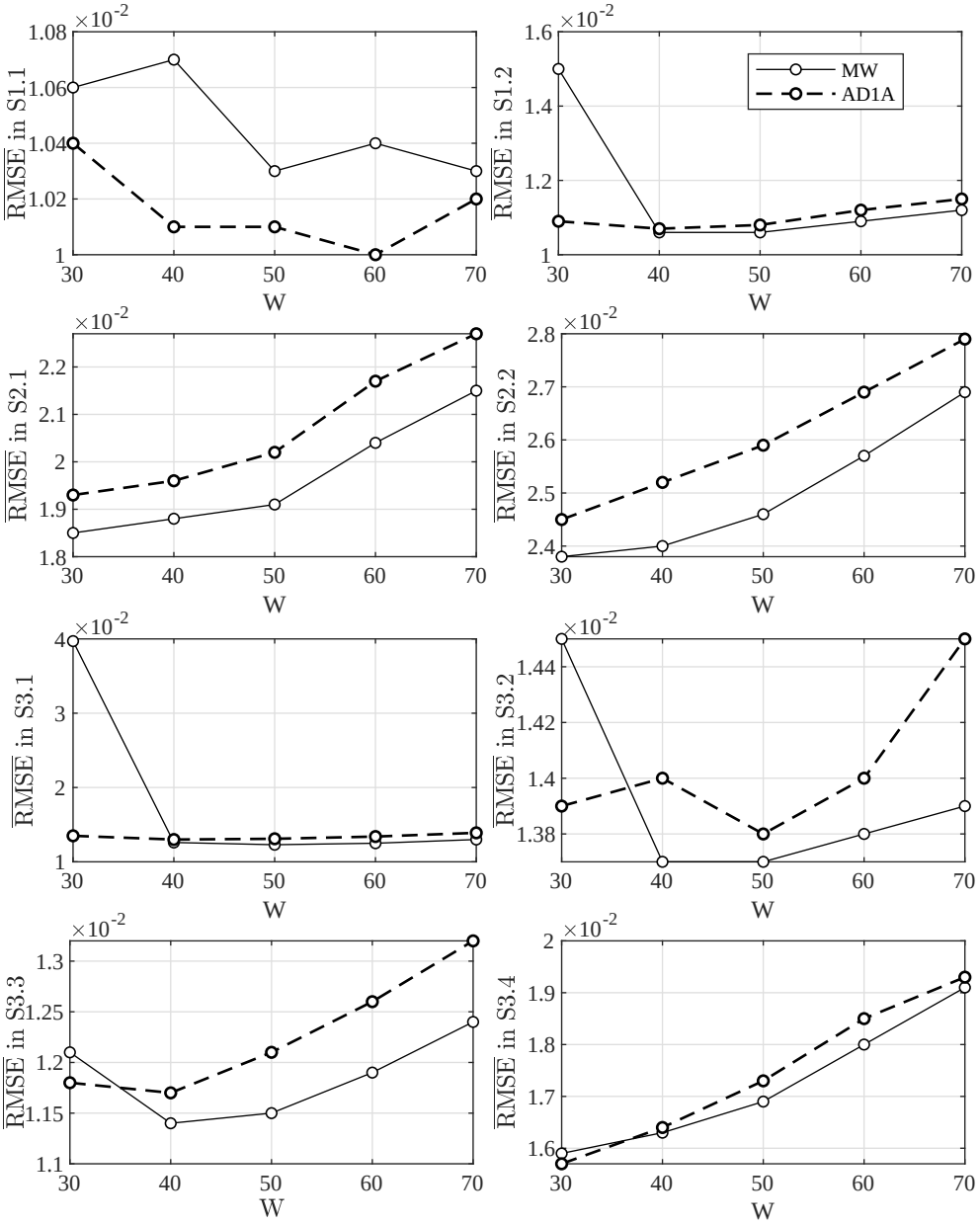

Figure 6.36. Comparison of $\overline{\text{RMSE}}$s of MW and AD1A on different CDMs from CSTR simulations using $\text{W} \in \{30, 40, ...70\}$.

For instance, in S2.2, smaller Ws should be adopted to handle abrupt CDs with high frequency; hence both $\overline{\text{RMSE}}$ and $\overline{\text{maxAE}}$ are higher in AD1A compared to MW. However, it should be emphasized that the magnitude of this increase quite small, even negligible compared to the reduction we observe at small Ws. In summary, AD1A prevents the occurrence of gross prediction errors, and has the potential to improve average prediction accuracy if integrated with other methods. For this reasons AD1B integrates R^2_{adj} check to AD1A, and AD1C integrates $\hat{\sigma}^2$ check to AD1A. Next, prediction accuracy of AD1A, AD1B and AD1C is compared within the same frame (Figure 6.38). Adaptation algorithm of AD1C is an informed, albeit a humble one; its predictive performance is found to be consistently the best in the presence of gradual CDs (S3.1-3.4 in Figure 6.38) among all adaptive window size mechanisms. With abrupt CDs in S2.1 and S2.2, AD1B appears to be a better choice since it compares R^2_{adj} values of candidate Ws at every step, and if there exists a window with higher R^2_{adj} than that of the current one, window size changes. This strategy yields good prediction performance only in the presence of abrupt drifts; on the other hand it was expected to have high type-I error rate, hence it can risk introducing additional variance at each query when it is not warranted in gradual drifts. Interestingly, AD1A, the underdog of this league, emerges as the best performing mechanism in S1.1 and S1.2 even though the difference in $\overline{\text{RMSE}}$s is marginal. Recall that CD disturbance in S1 changes as step inputs at multiple levels, and the magnitude of these steps are relatively small hence these CDMs are considered to be of small magnitude. Both AD1C, and particularly AD1B, appear to be too aggressive for such a setting. In summary AD1C is among the best adaptive mechanism in almost half of the concept drifts, and never the worst one in the rest.

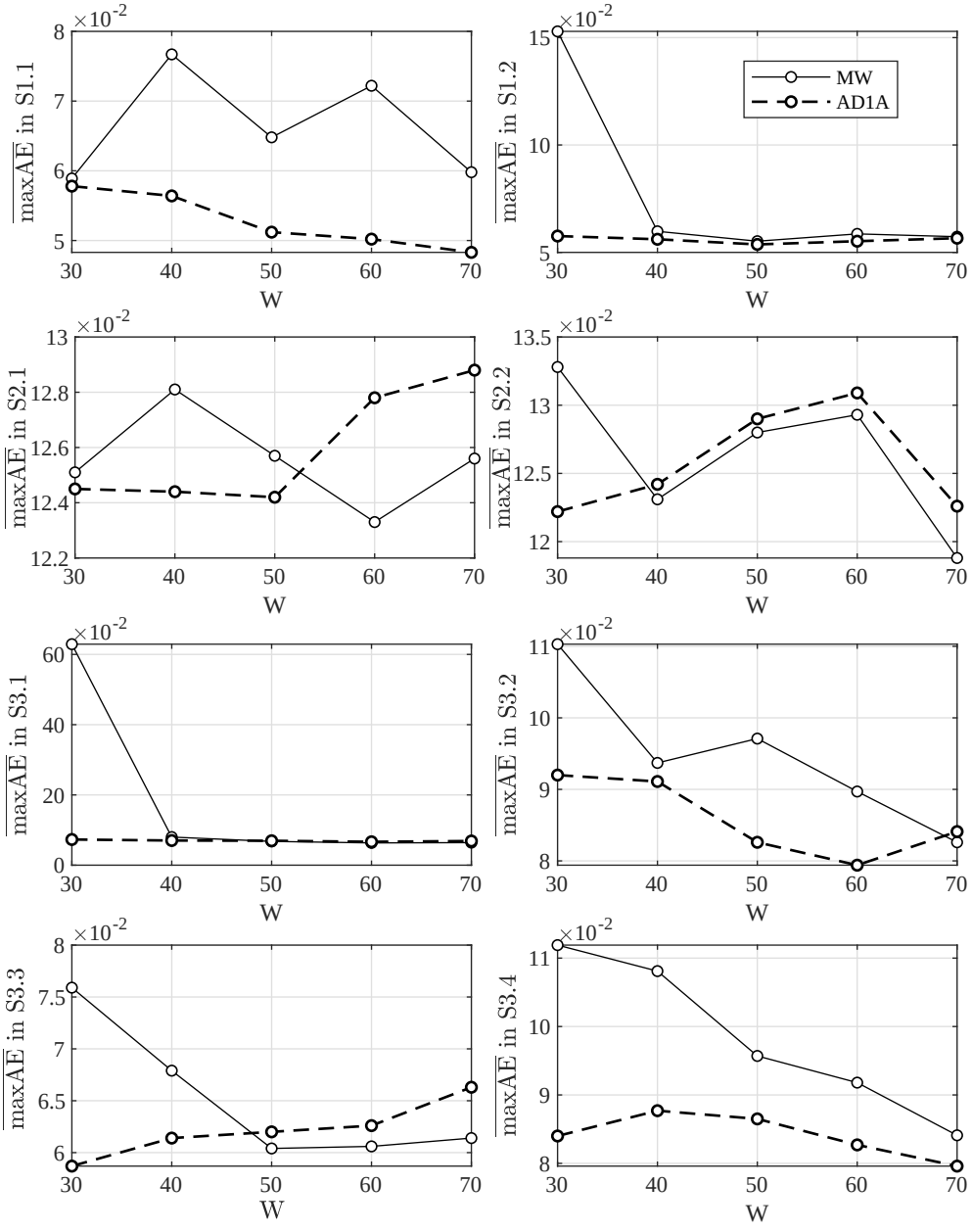

Figure 6.37. Comparison of $\overline{\text{maxAE}}$s of MW and AD1A on different CDMs from CSTR simulations using W $\in \{30, 40, ...70\}$.

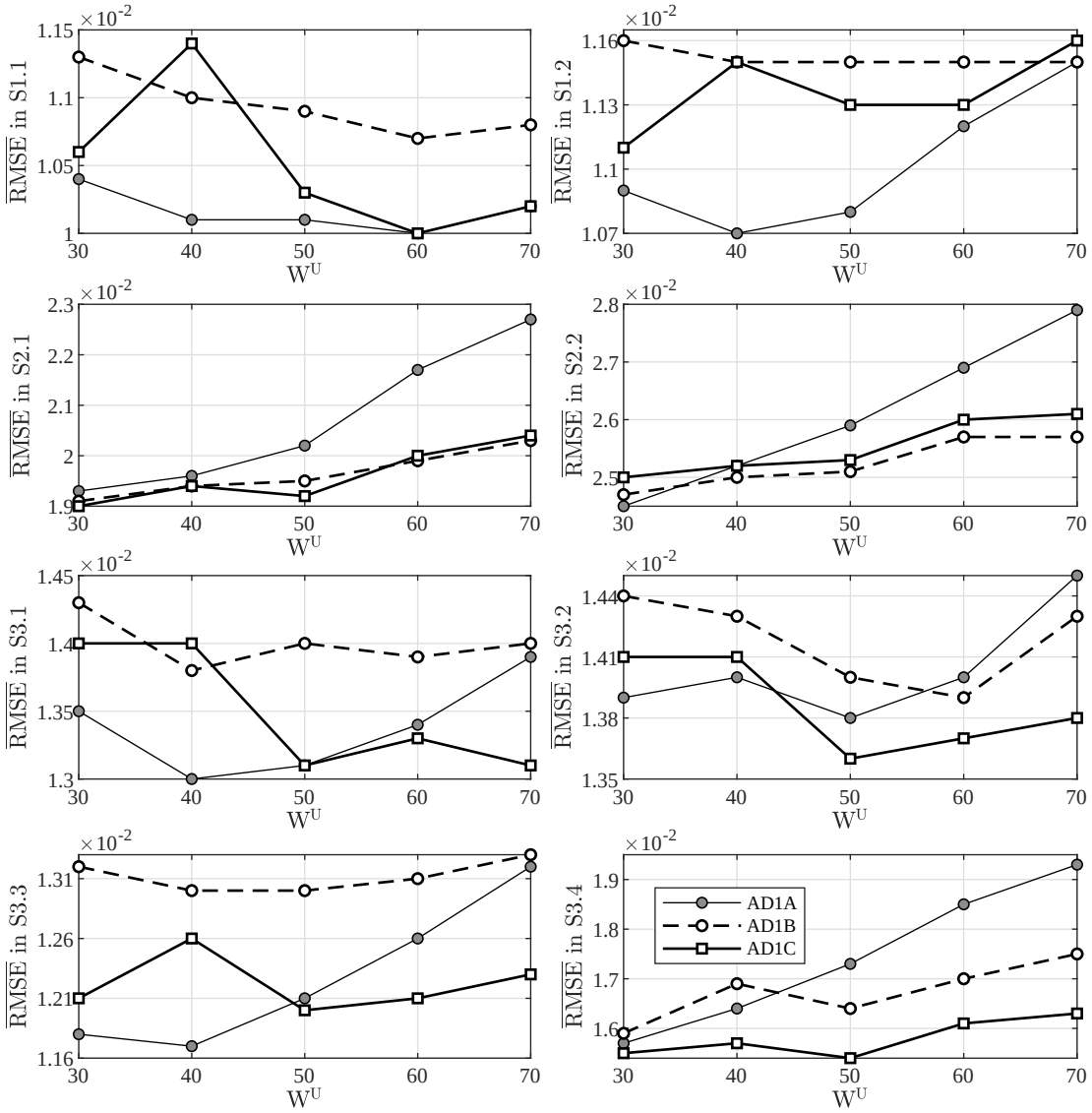

Figure 6.38. Comparison of adaptive window size adjusting mechanisms; AD1B and AD1C are employed with asymmetric S_{W}s, lower limit of W is 20 and in AD1C $\alpha = 0.001$.

The α parameter in AD1C determines the width of the control limits, and the effect of α on prediction accuracy is to be examined. Here, $\alpha \in \{0.001, 0.005, 0.01, 0.05\}$ is used (Figure 6.39); while the difference in $\overline{\mathrm{RMSE}}$s is not striking, in general a small to moderate α value may be recommended (S1.1, S1.2, S3.1 and S3.2 in Figure 6.39), as AD1C with $\alpha = 0.05$ can be a hit or miss in most CDMs, particularly at small $\mathrm{W^U}$s.

Some optimum α might exist in the interval $[0.001, 0.01]$ for each W^U.

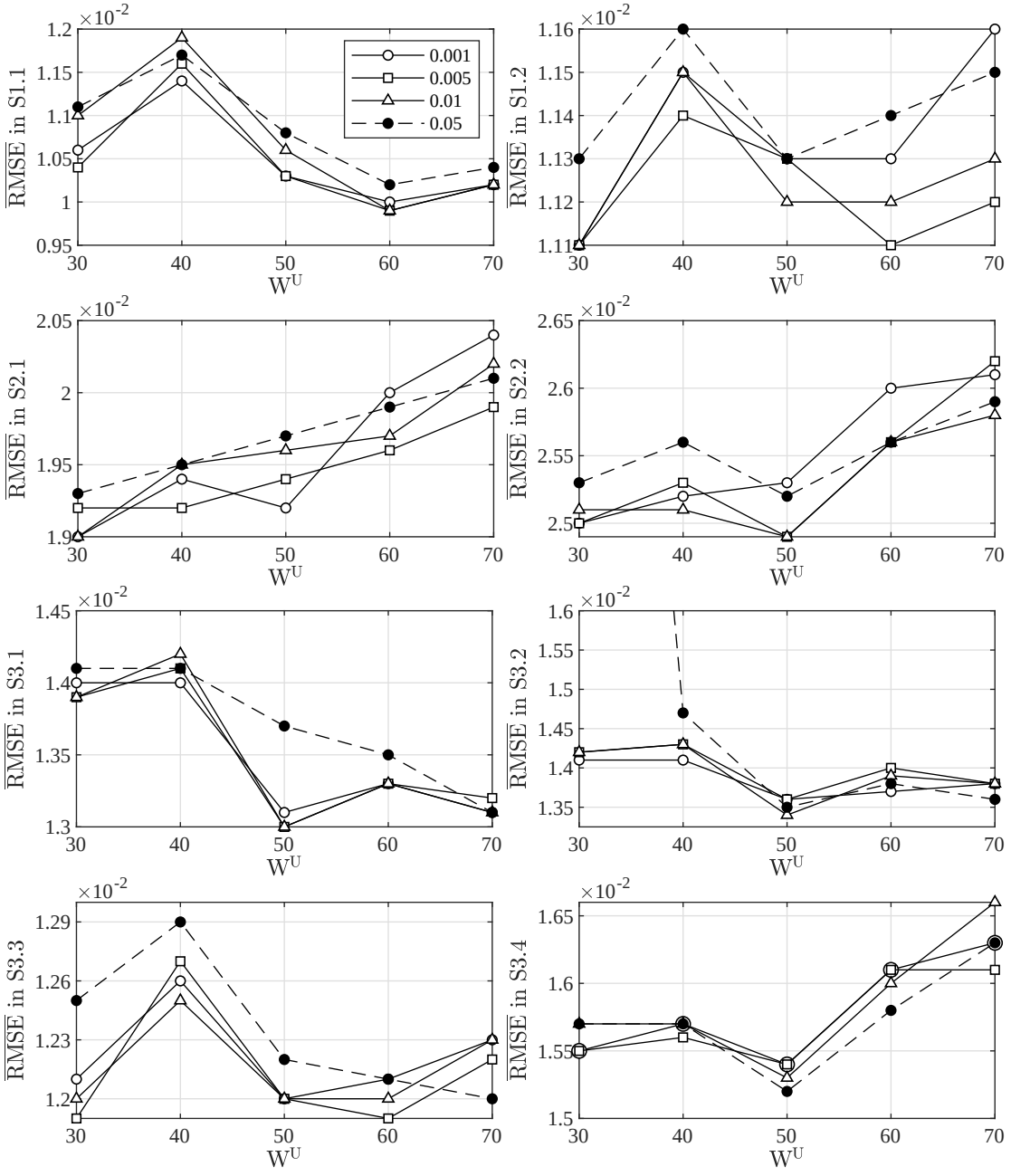

Figure 6.39. Effect of α parameter in AD1C over different W^Us.

Window sizes adopted by AD1B and AD1C (at two different significance levels) are compared in Figure 6.40. At both higher and lower W^U ($W^U = 40$ in Figure 6.40a and c, and $W^U = 70$ in Figure 6.40b and d, specifically), AD1C yields large Ws, close

to the upper limit, when α is small, but tends to favor smaller Ws for large significance level. AD1B, on the other hand shows a more balanced approach as the adaptive Ws are well distributed across the entire range of allowable Ws. This observation is in accordance with that seen in Figure 6.21, in which probability of adapting a low/high window size is approximately 0.4/0.6. At higher W^U when AD1C is employed with a large type-I error rate, its bias towards smaller Ws explains the reduction in $\overline{\text{RMSE}}$ we have observed beyond $W^U = 50$ for CDMs S2.2, S3.2 and S3.4 in Figure 6.39. Since these CDMs emulate abrupt CD scenarios, smaller Ws should be preferred to adapt to changing concepts quickly enough.

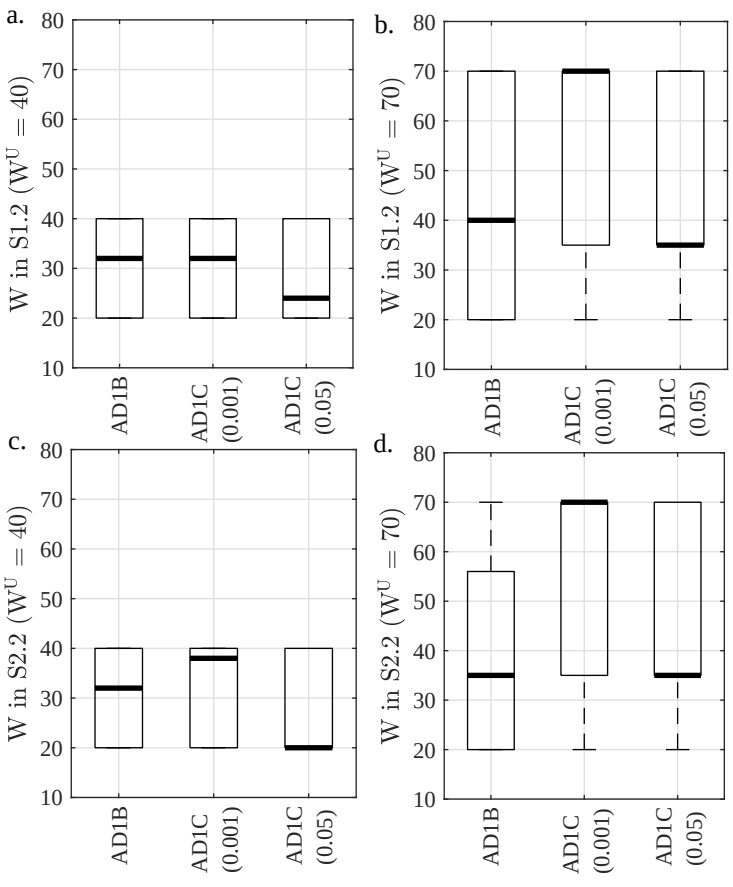

Figure 6.40. Boxplots of the selected adaptive W for AD1B and AD1C (using $\alpha = \{0.001, 0.05\}$) in two CDMs: S1.2 (a and b) and S2.2 (c and d) for $W^U \in \{40, 70\}$, respectively.

In this section, ADs are employed on debutanizer column data, MWs with W $\in \{10, 20, ...100\}$, and the same order of analysis is followed. A comparison between MW and AD1A shows the clear advantage of AD1A over MW on real data (Figure 6.41). Furthermore, RMSE has a general increasing trend as W increases which indicates that this data is better modeled using smaller training sets. At small Ws, however, prediction errors of MW are still somewhat large, and following a minimum at W = 50, RMSE continues to increase with W. We have already pointed out that when model parameters are estimated using very few observations, there is the risk of eliminating input variables which are valuable for prediction. AD1A addresses this issue by enlarging W momentarily when the Mahalanobis distance of query point exceeds the upper bound in Equation 6.15, and hence drastically stabilizes predictions. The probability of positive test results, i.e. when d^2_{M} exceeds the upper bound in Equation 6.15, is denoted by $p(I_{\mathrm{M}} = 1)$; it is observed that $p(I_{\mathrm{M}} = 1)$ is well below 20% for all Ws (Figure 6.41b, left axis), and declines with increasing W. The decreasing trend justifies our main motivation for using AD1A, to avoid using MWs in which the query point does not belong in the input space, and thus possibly decrease the probability of obtaining gross errors by momentarily adding more observations. It is possible to achieve 95% reduction in RMSE when W = 10. The sharp decline in $p(I_{\mathrm{M}} = 1)$ at W = 10 is due to the large increase in upper bound for d^2_{M} in Equation 6.15; when there are 10 observations in the training set, and seven predictors are retained in the RVM model, UB can be as large as 640. Therefore, the number of d^2_{M}s which exceed this limit is small. Right axis in 6.41b shows the mean window size used by AD1A in the test set (black filled circles), the average window size increases with increasing W. It should also be noted that maximum window size can reach 2000s for some Ws (not shown); this suggests that AD1A sometimes uses almost the entire training set, i.e. a global model, to make predictions.

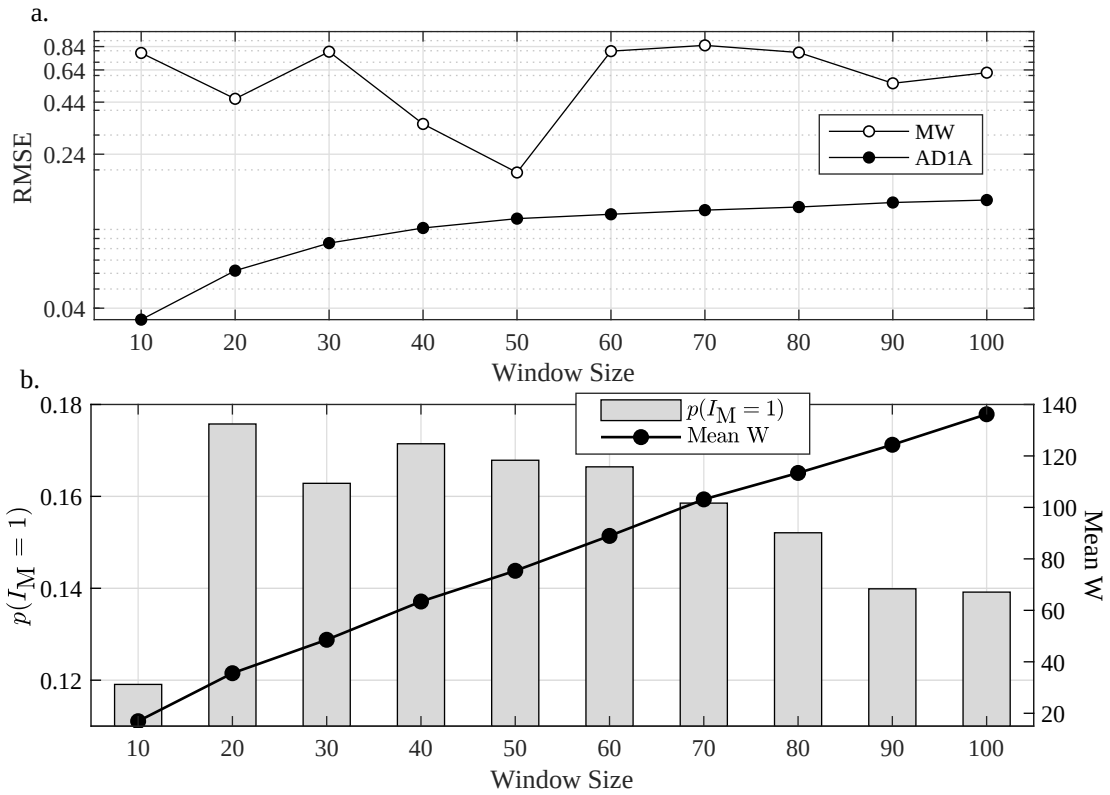

Figure 6.41. RMSEs of MW and AD1A on debutanizer column data (a), $p(I_{\mathrm{M}}=1)$ and the mean value of window size selected during prediction on left and right y-axis, respectively (b).

Comparison of prediction accuracy of AD1A and AD1B on debutanizer data yields a more pronounced difference between the algorithms, compared to the results on simulated data (Figure 6.42). Here, AD1B is evaluated using $\Delta \mathrm{W} \in \{5, 15\}$, which corresponds to AD1BI ($\Delta \mathrm{W} = 5$), and AD1BI ($\Delta \mathrm{W} = 15$), respectively, and AD1BII denotes using the asymmetric S_{W} (Figure 6.20) with $\Delta \mathrm{W}_1 = 0.2$ and $\Delta \mathrm{W}_2 = 0.5$. The aim is to evaluate the effect of window size changes adopted for each query point on prediction accuracy. It is seen that more drastic changes, particularly in shrinking W (Figure 6.42b, c), yields better predictions, and prediction errors become almost independent of W^{U}, being highly beneficial for this dataset as small Ws yield better predictions. However, it is possible that this desirable property may not be generalized to different datasets.

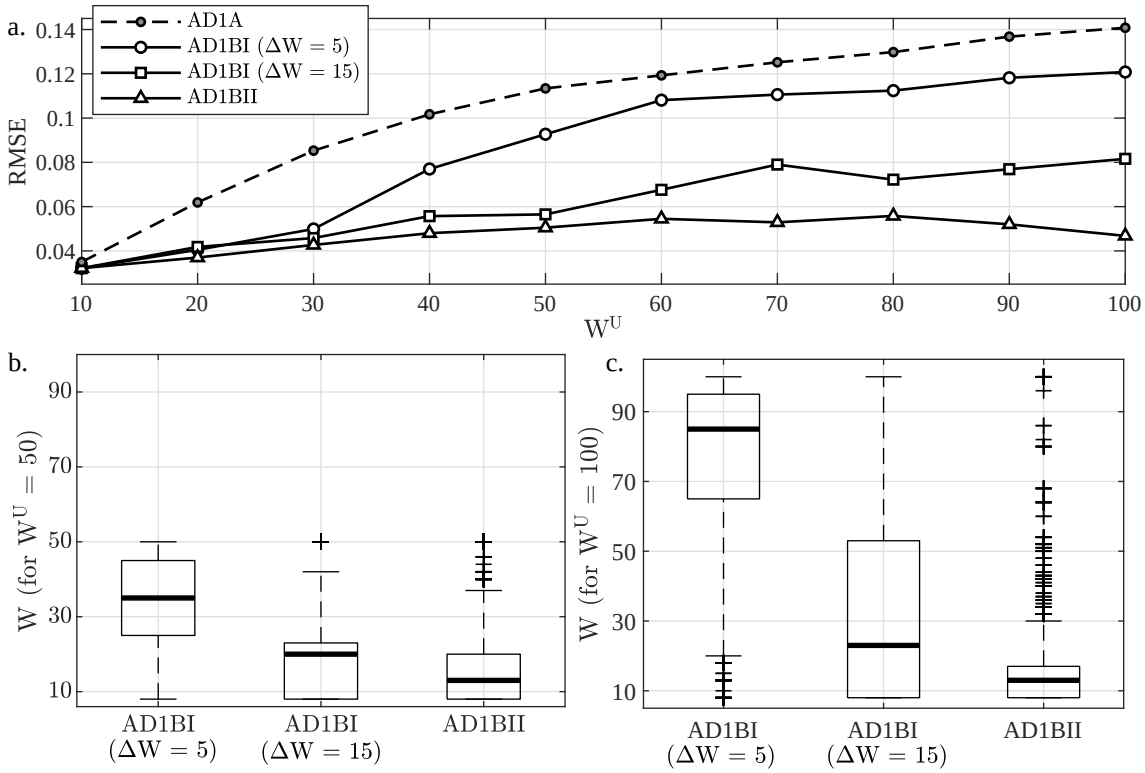

Figure 6.42. Effect of using different schemes to obtain S_{W}s in AD1B (subplots b and c using $\mathrm{W^U} = 50$ and $\mathrm{W^U} = 100$, respectively), in comparison with AD1A (subplot a).

The effect of ΔW on AD1C, compared to AD1B, is less pronounced at small $\mathrm{W^U}$s, however the gap in RMSEs is widened for large $\mathrm{W^U}$s (Figure 6.43). It should, nevertheless, be stressed that AD1C performs better than the best AD1B model for all settings of ΔW and α (Figure 6.43). Here, an aggressive ΔW action and a high-to-moderate significance level yields the best results for AD1C, similar to previous findings.

Once the prediction accuracies of the adaptive MW procedures are determined, the next step is to examine the predictive accuracy of the MW and JITL ensemble as realized in AD2 algorithm. As the best MW procedures, AD1B and AD1C are used along with two JITL algorithms, namely AD2A and AD2B. The difference in A and B, lies in the application of the former at each query point, whereas a MW and

JITL ensemble is used only for points with d^2_{M} statistic below the confidence bound (Equation 6.15) in the latter.

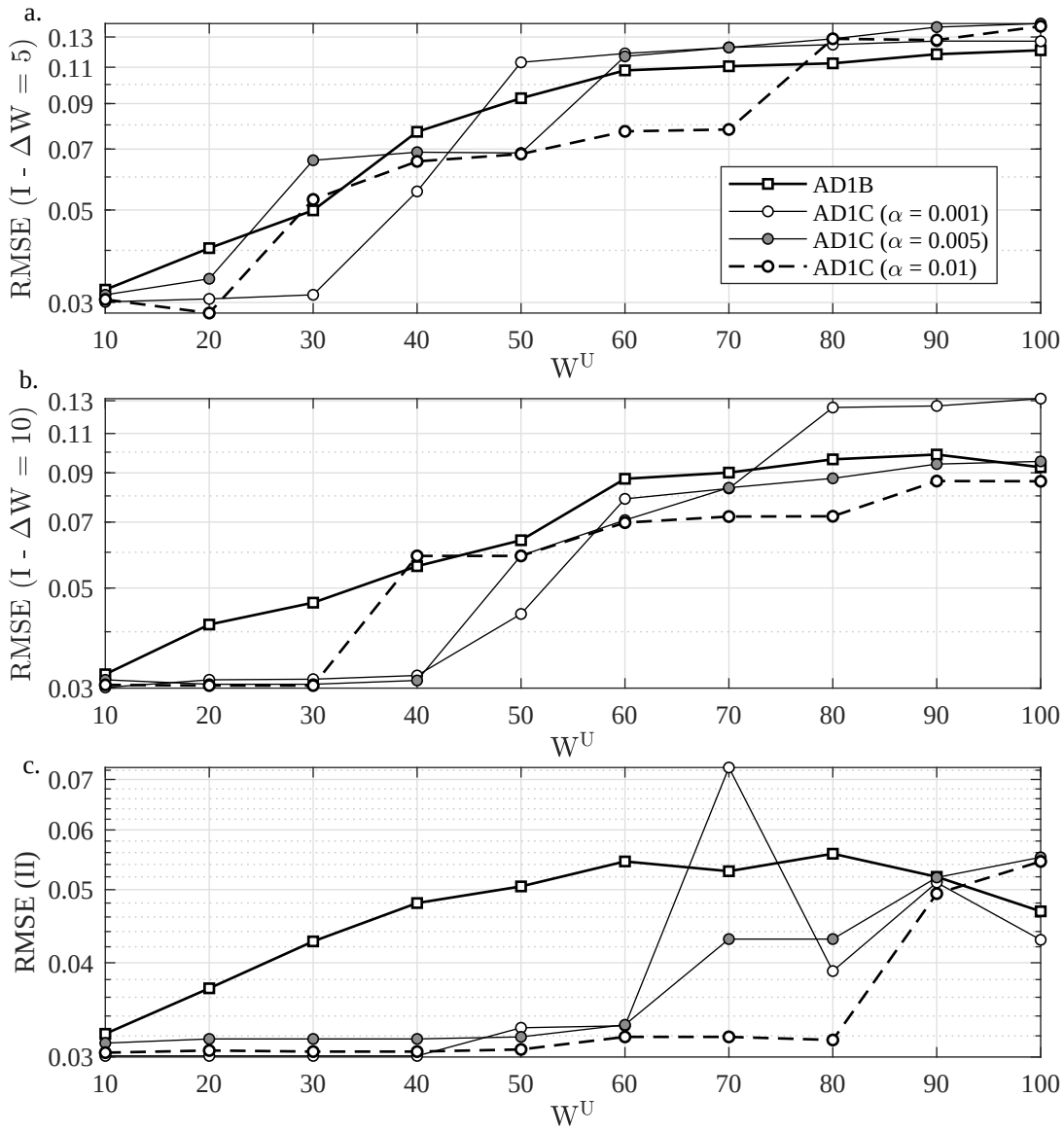

Figure 6.43. Comparison of AD1C using $\alpha \in \{0.001, 0.005, 0.01\}$, and AD1B for different values of ΔW.

Previously in this section, it was asserted that AD1C vastly improved predictions with a relatively small dependence on the upper limit of Ws, and gross prediction errors were avoided by the combination of AD1A and $\hat{\sigma}^2$ monitoring for detecting

CD. The addition of JITL in an ensemble modeling frame is presumed to address

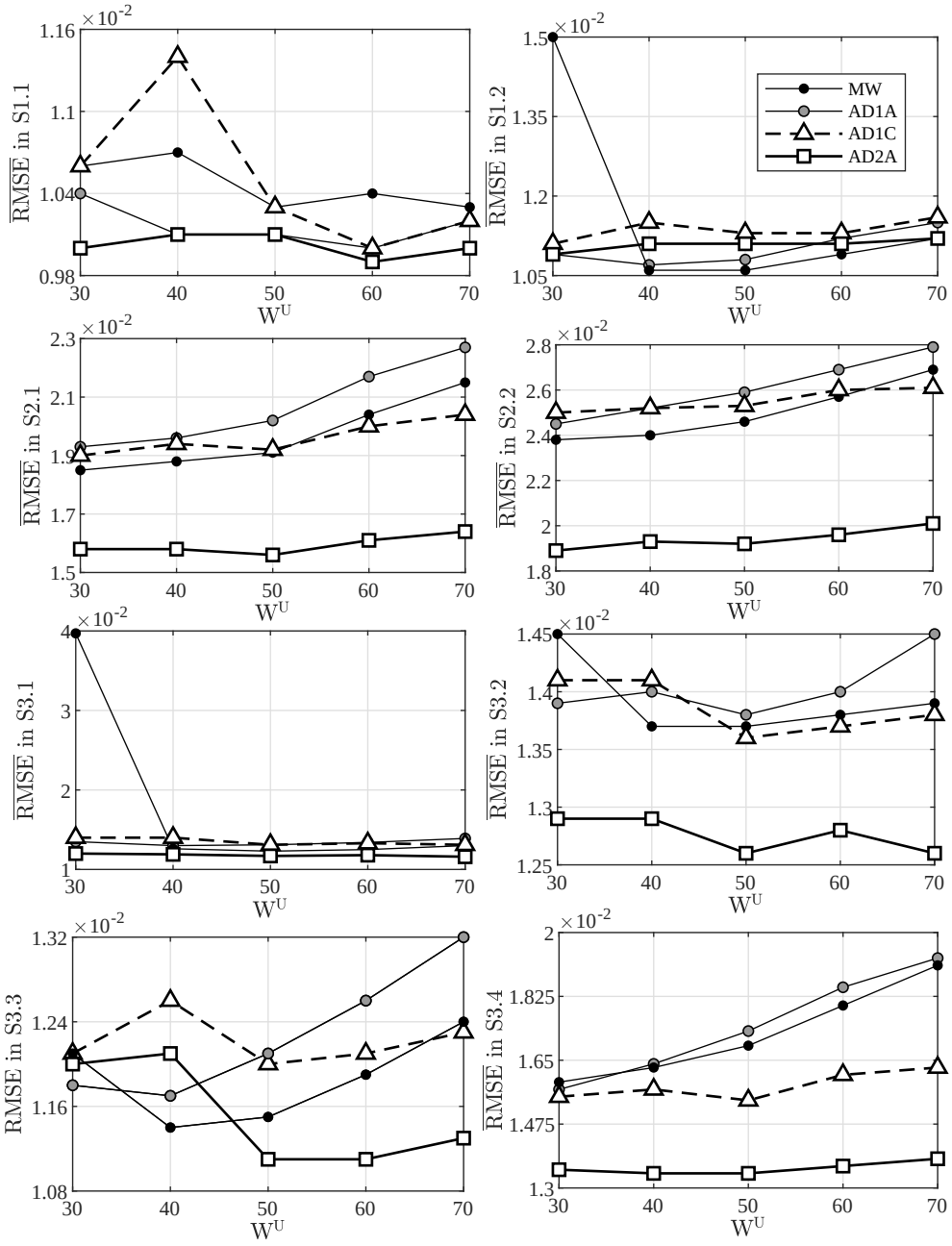

Figure 6.44. Comparison of the ensemble model AD2A with adaptive W mechanisms in synthetic data.

both nonlinearity, and CDs that could not be handled promptly via an adaptive MW approach.

For the sake of brevity, we will consider only AD1B, and AD1C ($\alpha = 0.001$) both of which implement asymmetric S_{W} with $\Delta \mathrm{W}_1 = 0.2$ and $\Delta \mathrm{W}_2 = 0.5$ for comparison. For the application of the JITL procedure on the synthetic data, the window size $\widetilde{\mathrm{W}}$ is set to 30, since the minimum size of the MWs, which yielded relatively stable RVM models for the training set, was found to be around 30. It is seen that addition of JITL to MW generally improves the accuracy of predictions (Figure 6.44). Furthermore, $\overline{\mathrm{RMSE}}$ stays nearly constant, or even decays for some CDs, particularly S3.2 and S3.3 with respect to W^{U}, showing that choosing the maximum size of the adaptive MW is less of a concern when AD2 method is used.

For each query point during the implementation of AD2 method, MW and JITL RVM models are weighted by w_{MW} and $1 - w_{\mathrm{MW}}$, respectively, obtained from their predictive accuracy on the recent past observations. To have a more thorough understanding of the ensemble adaptation, w_{MW} values obtained during the whole trajectories are examined in more detail. To this aim, abrupt (S2.1 and S2.2) and gradual concept drifts (S3.3 and S3.4) of high magnitude are informative: decrease in w_{MW} is seen for higher frequency CDs by examining Figure 6.45a-b for abrupt drifts, and Figure 6.46a-b for gradual drifts, separately. For both CDMs, w_{MW} distribution is transformed from a (close to uniform) bi-modal distribution, to a single modal highly right skewed distribution in the presence of high frequency disturbances, and w_{MW} is higher for larger W^{U}. This suggests that in the presence of frequent CD, JITL is preferred over MW for predictions. However, MW has contribution to predictions comparable to that of JITL at lower frequency CDs. This observation is perfectly in accordance with expectations; MW, to a degree, is able to adapt to slow process changes, but help from historical data is required to maintain the quality of predictions.

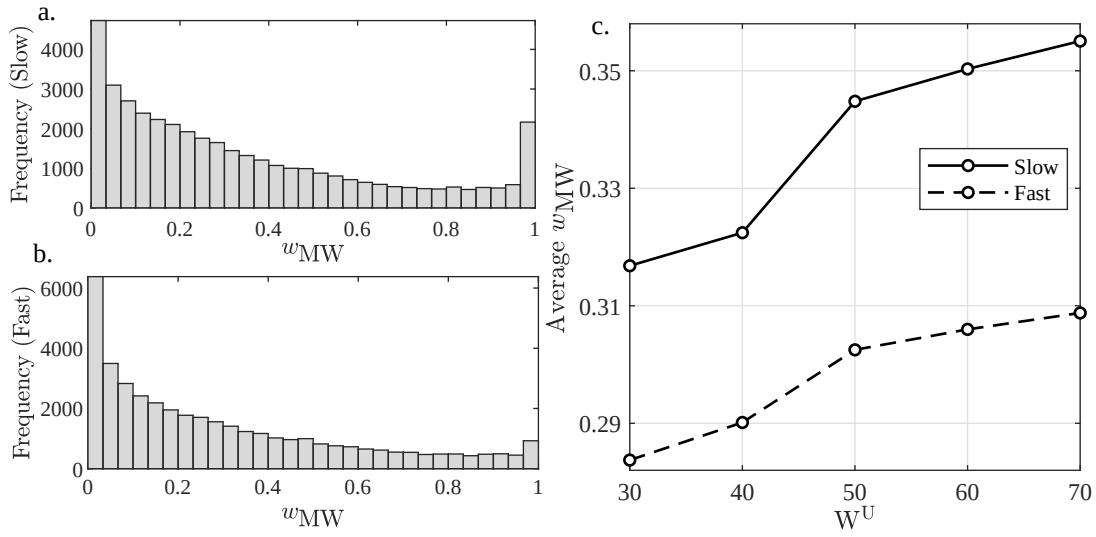

Figure 6.45. Distribution of MW weights obtained for $\mathrm{W^U} \in \{30, 40, ..., 70\}$ for slow (a) and fast (b) CDs, and average MW weights (c) in an abrupt CDM (S2.1 and S2.2, for slow and fast, respectively).

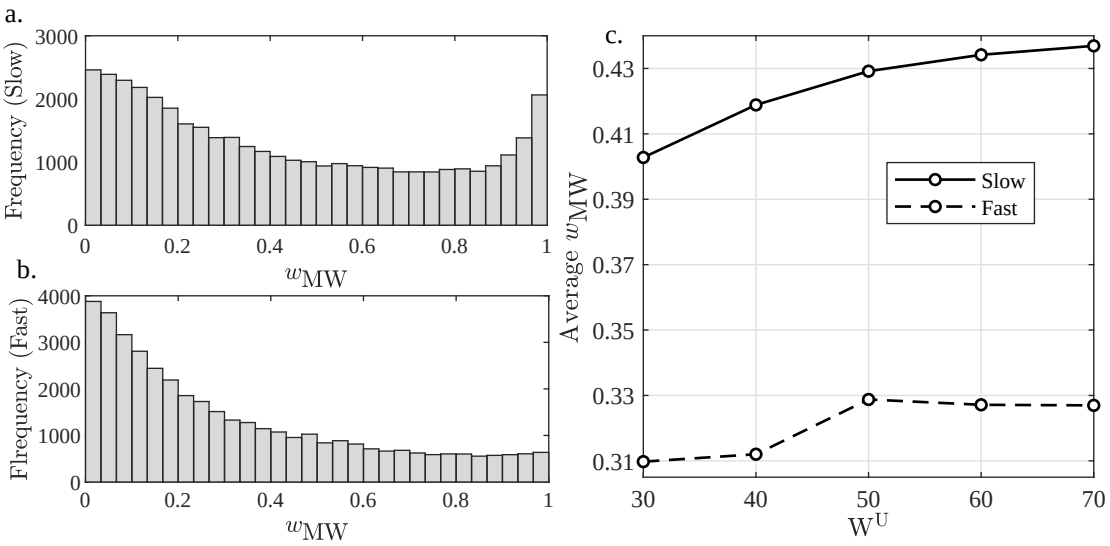

Figure 6.46. Distribution of MW weights obtained for $\mathrm{W^U} \in \{30, 40, ..., 70\}$ for slow (a) and fast (b) CDs, and average MW weights (c) in a gradual CDM (S3.3 and S3.4, for slow and fast, respectively).

While w_{MW} distributions in the presence of CDM S1 show similarities to those presented above, there are also some differences, which need to be emphasized. Distributions of w_{MW} at low frequency CD, S1.1 are similar to those observed at the low frequency CDs of S2.1 (Figure 6.45a) and S3.1 (Figure 6.46a). On the other hand, increasing the frequency of disturbances, as in S1.2 (Figure 6.47b) favors MW, as opposed to selecting historical historical local regions, unlike CDMs S2 and S3 (Figure 6.45b and Figure 6.46b). This result is rather unexpected, and shows that more work is required to understand these concept drifts. Higher W^{U} yielding higher w_{MW} values (Figure 6.47c) seems like a universal behavior of this system, since it was also observed for CDMs S2 and S3 (Figure 6.45c and Figure 6.46c).

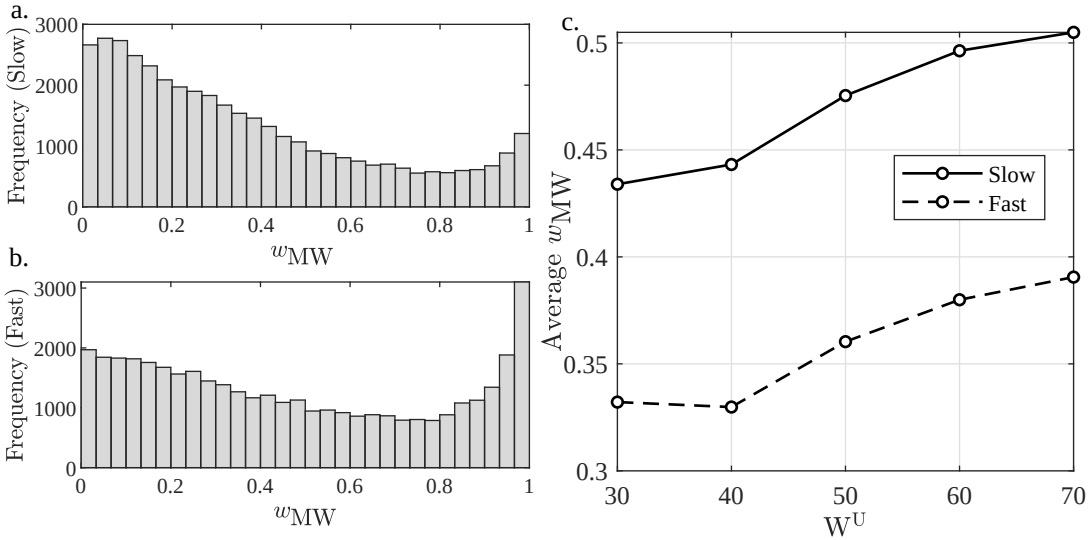

Figure 6.47. Distribution of MW weights obtained for $\mathrm{W}^{\mathrm{U}} \in \{30, 40, ..., 70\}$ for slow (a) and fast (b) CDs, and average MW weights (c) in a low magnitude CDM (S1.1 and S1.2, for slow and fast, respectively).

One drawback of the JITL component of AD2A is that a local region for constructing JIT model is sought from scratch at each query point with the risk of introducing excess variance to prediction. To reduce this variance, AD2B is introduced. In AD2B, local region search is implemented only for query points, which are deemed to be in the operating region of the current MW. When d^2_{M} value exceeds the upper confidence

bound ($I_{\mathrm{M}} = 1$), only the extended MW is used for prediction. The result of this algorithm will be employing AD2A selectively, hence possibly less frequently. Therefore it would be informative to compare the predictive performances of AD2A and AD2B, to understand the subtleties of the algorithm.

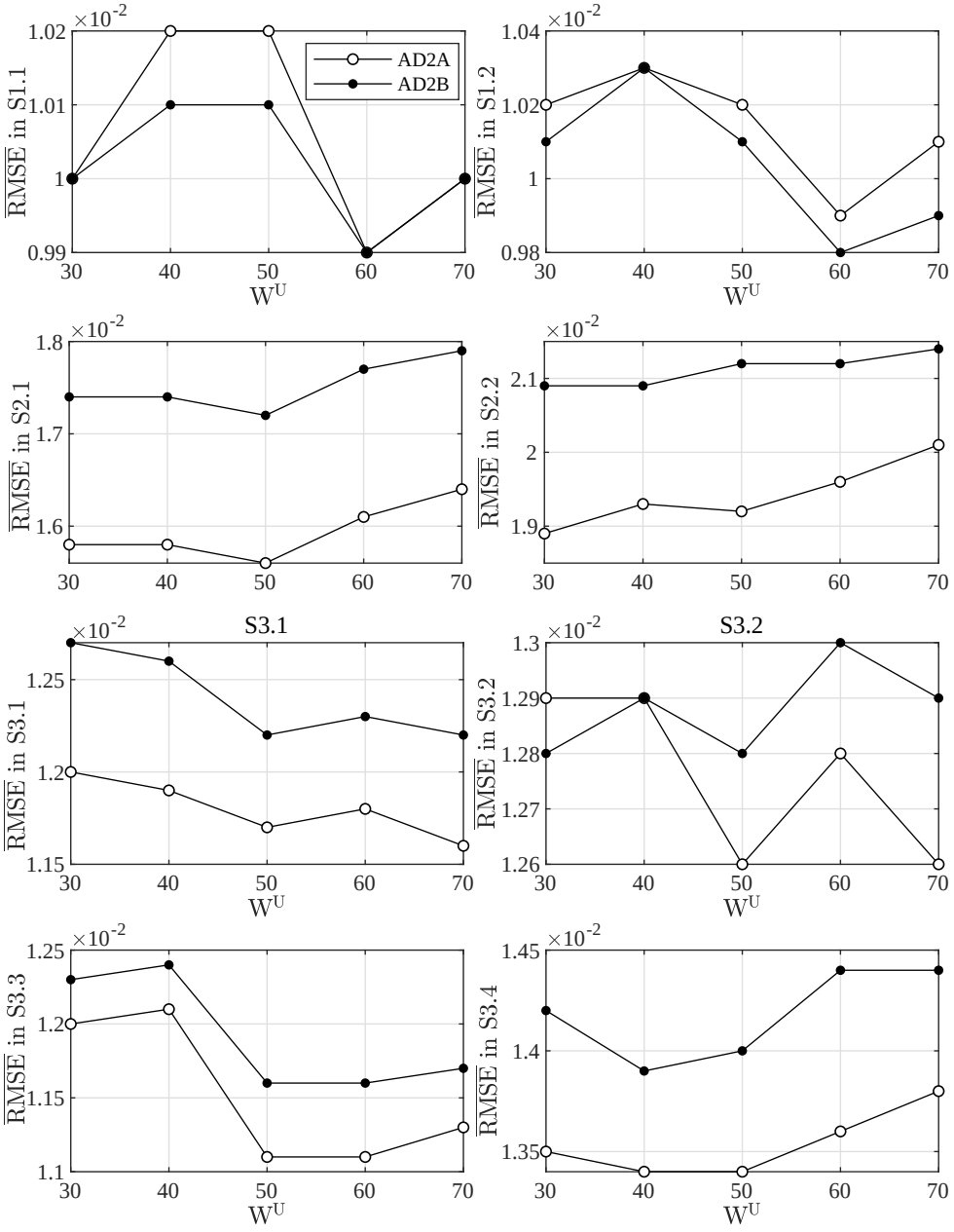

Figure 6.48. Comparison of $\overline{\mathrm{RMSE}}$s of AD2A and A2B on synthetic data using $\mathrm{W} \in \{30, 40, ...70\}$.

Based on the differences in CDMs of low and high magnitude, discussed just above, selective application of JITL on query points is elucidated for two different groups of CDs. Here S1 corresponds to small magnitude concept changes, while S2 and S3 are deemed to represent high magnitude concept changes. $\overline{\text{RMSE}}$s obtained by AD2B in small magnitude CDMs are slightly lower than those of AD2B Figure 6.48. On the other hand, $\overline{\text{RMSE}}$s of AD2A is much lower than those of AD2B for high magnitude CDs (Figure 6.48). This shows that ensemble modeling does not necessarily yield better predictions, particularly against small magnitude disturbances (S1.1 and S1.2 in Figure 6.48). However, the loss in prediction accuracy is negligibly small in the ensemble method advocated in this thesis, compared to the accuracy gain against more severe disturbances. Based on this finding, one may suggest that a larger set of ensemble models, as usually employed in the literature, may, however, yield much worse predictions for small magnitude disturbances. This problem is remedied in our study by using only two models in the ensemble.

In the recent paragraphs, we examined how RMSE and w_{MW} of AD2 based (MW-JITL) methods change with respect to W^{U} and various CDs on simulated data. In this part, we repeat the same analysis on real plant data. AD1C is implemented in debutanizer dataset with parameters $\alpha \in \{0.001, 0.005, 0.01\}$, $\text{W}^{\text{U}} \in \{10, 20, ..., 60\}$, $\widetilde{\text{W}} = 20$ and $mp = 3$. The first two parameter values are taken to be the same as those used in synthetic data, unlike the JITL window size, $\widetilde{\text{W}}$ is taken to be 20. Implementing JITL algorithm with MW (AD2A) is seen to decrease the prediction errors to impressively lower values (Figure 6.49a), in accordance with findings observed for CDMs in synthetic data. The effect of α parameter on prediction accuracy seems to be lowered when AD2A is used. This may easily be explained by the distribution of w_{MW} values, showing that local models are slightly more preferred during online prediction for debutanizer data (Figure 6.49c). The bi-modal nature of w_{MW} justifies the use of a weighted average to obtain final predictions in Figure 6.49c, in which weights belong to AD2A predictions using $\alpha = 0.005$, and this trait holds for other values of α as well (not shown). Variation of w_{MW} with respect to W^{U} is not statistically significant to determine a reliable trend (Figure 6.49b).

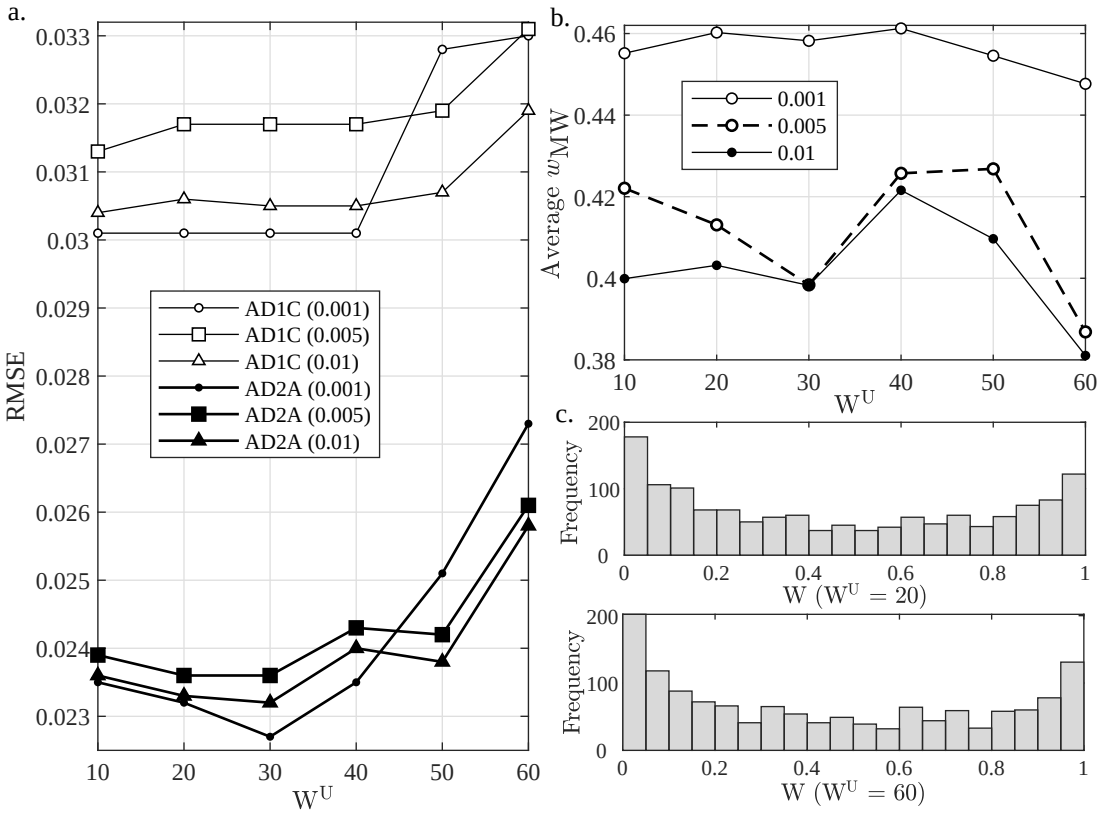

Figure 6.49. Comparison of ensemble JITL, AD2A, with adaptive MW mechanism, AD1C for $\alpha \in \{0.001, 0.005, 0.01\}$ (subplot a), along with the effect of α and W^{U} parameters on MW weights (b), and distribution of w_{MW} (c).

Next we compare two AD2 methods, AD2A and AD2B on debutanizer data. Prediction performance of AD2A exceeds that of AD2B (Figure 6.50), this resembles the results on synthetic data of high magnitude CDMs (S2 and S3 on Figure 6.48), hence debutanizer data is suggested to be similar to CDMs S2 and S3, comprised of high magnitude CDs. JITL in AD2B is presumed to handle nonlinearity, and some of the real drift which could not be addressed with AD1C; in AD2A, on the other hand, JITL can respond to both real and some leftover virtual drift which could not be modeled via AD1A. The fact that larger reductions in RMSEs are achieved using AD2A instead of AD2B motivates this classification (Figure 6.50).

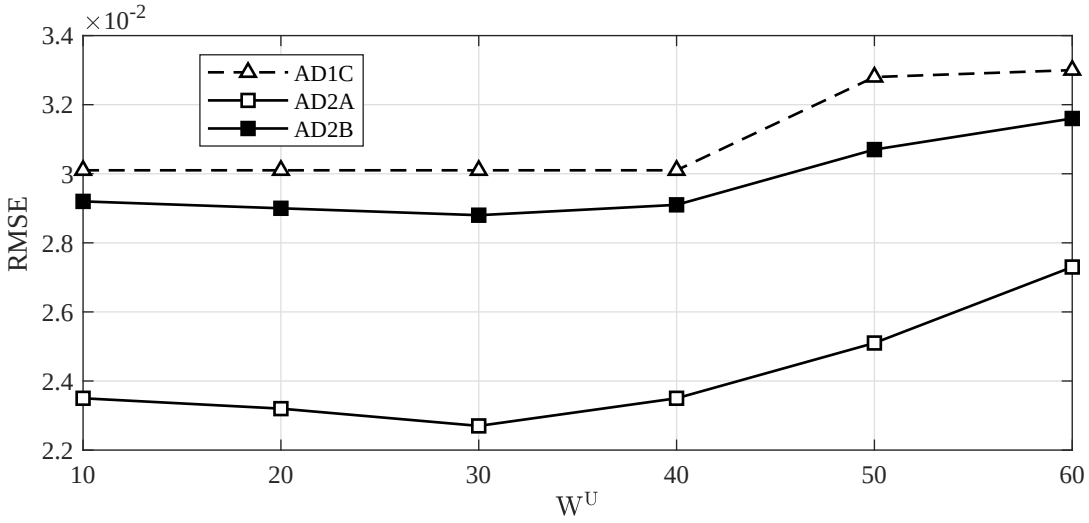

Figure 6.50. Comparison of AD2A and AD2B, with adaptive MW mechanism, AD1C using $\alpha = 0.001$ and $\mathrm{W^U} \in \{10, 20, ...60\}$.

The final parameter of AD2 to be analyzed is mp, the number of past observations used for assessment of local region in JITL, followed by determining the ensemble model weights (Figure 6.33). In this section, $mp \in \{1, 3, 5\}$ are used in AD2A modeling to determine prediction errors and distribution of w_{MW}, first on synthetic data, then on debutanizer column data. Increasing the upper limit of mp is not found to change prediction errors significantly. $\overline{\mathrm{RMSE}}$s determined from CDMs show that $mp = 1$ yields the most inaccurate predictions for all $\mathrm{W^U}$s and CDs (Figure 6.51). This shows that relying on a single, even though the most recent, observation to tune an adaptive method parameter online brings a high variance to method's predictions. In this case, more historical data points, realizing the risk of adding bias to the model, should be used to adjust the parameters of the adaptive method to decrease the prediction variance. Performance of AD2A ($mp = 5$) appears to be slightly better than that of AD2A ($mp = 3$), though the difference is almost negligible.

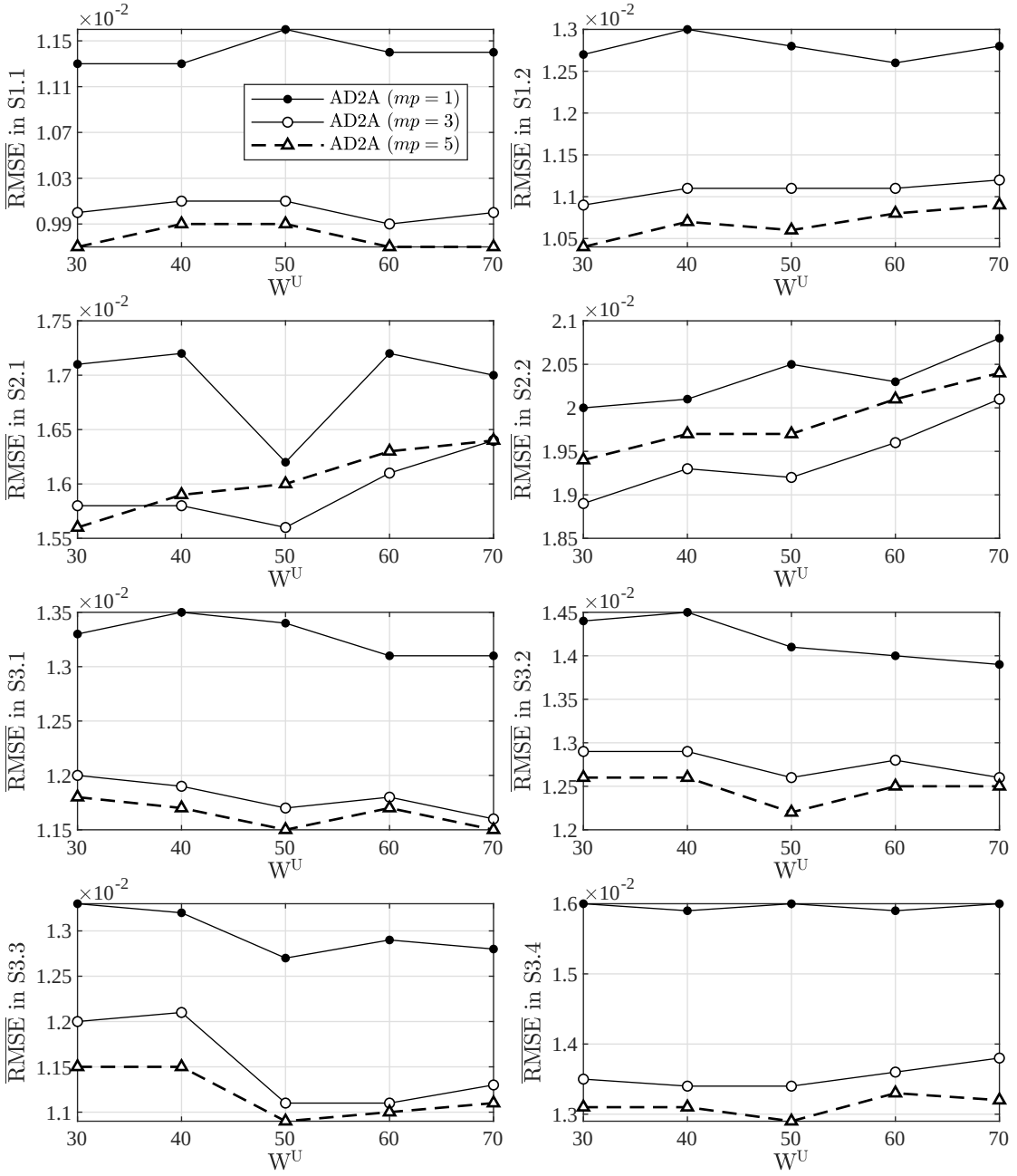

Figure 6.51. Comparison of AD2A models using $mp \in \{1, 3, 5\}$ in CSTR simulations, for values of $W^U \in \{30, 40, ...70\}$.

Next, distribution of w_{MW} values is examined on the same CDMs. Distribution of w_{MW} is particularly striking for $mp = 1$ (Figure 6.52a). It is seen that a large majority of samples is predicted solely using local models, neglecting the MW information. This

shows that a search among the local regions for the most accurate prediction of a recent point is likely to yield such a region, not only because of the similarity of operating conditions, but also due to the large search space. A plausible result of this search is the increase of false positives, likely to yield less accurate predictions for the query points. When mp is increased to three (or five), local regions yielding a number of successive good predictions are sought, hence false positive rate is decreased, and MW is given a higher weight (Figure 6.52b, c). Weights are highly concentrated around the smaller mode, indicating the preference for JITL over MW when $mp = 1$ is used. While variance of weights is the lowest for $mp = 1$, the tendency to give more importance to JITL may lead to an increase in bias. It may be informative to examine the dynamics of w_{MW} trajectories to have a better understanding of the AD2A's preference of MW and local regions in predictions.

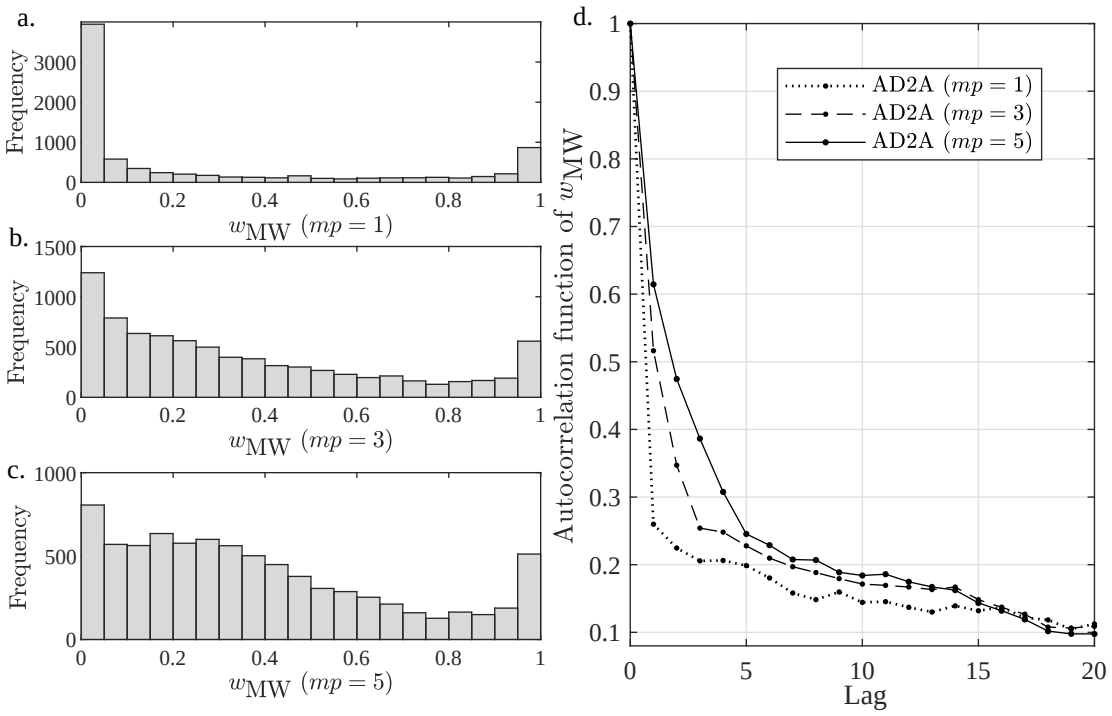

Figure 6.52. Distribution and autocorrelation functions of w_{MW} for $mp \in \{1, 3, 5\}$, averaged over 20 simulation runs in subplots a-c, and d, respectively, using $\mathrm{W}^{\mathrm{U}} = 50$ on CDM S2.1.

For this purpose, autocorrelation function estimate of w_{MW} is shown in Figure 6.52d, in which autocorrelation of w_{MW} decays to zero within $10-20$ samples, but is highly significant for the first ~ 3 lags, particularly for $mp = 3$ and $mp = 5$. This shows that when $mp = 3$ or $mp = 5$, w_{MW} has a short term memory, and moves rather smoothly from sample to sample, while using $mp = 1$ yields w_{MW} close to white noise in accordance with the previous analyses. Assuming that concept drifts acting on the system are rather slower than sampling frequency, $mp = 1$ does not seem to be convenient to be used in AD2A procedure.

Application of the same analysis on debutanizer column data shows similar results to those obtained for synthetic data (Figure 6.53). Here $mp = 3$ predictions seem to be slightly superior to those of $mp = 5$, but this difference is rather insignificant. The adverse effect of using a single past observation for validation on RMSE is more pronounced in this case. It was asserted that debutanizer data resembled to synthetic data with high magnitude CDs, this similarity is again observed here: AD2($mp = 3$) appears to yield the best predictive performance among all settings.

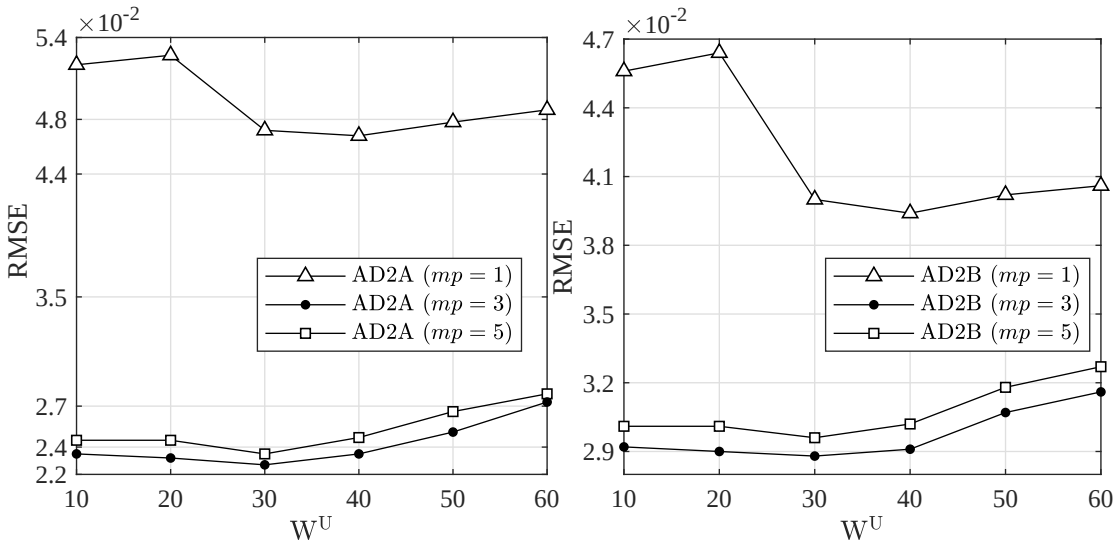

Figure 6.53. Comparison of AD2A and AD2B models using different values of mp in debutanizer column dataset for $\mathrm{W^U} \in \{10, 20, ..., 60\}$.

Both the distributions and autocorrelation functions of w_{MW} obtained for debutanizer data are completely similar to those obtained for synthetic data (Figure 6.54). As mp is increased, a more uniform distribution of w_{MW} is observed with more slowly decaying autocorrelation function. These findings, also show that synthetic simulations are rather realish imitations of a real chemical industrial process.

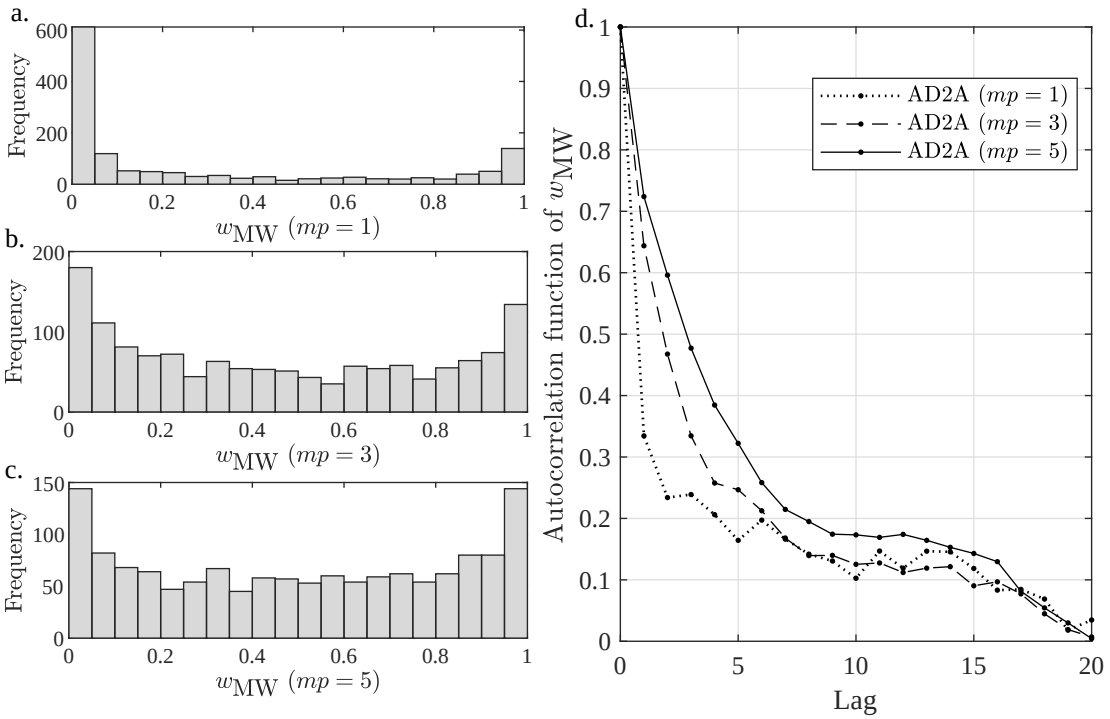

Figure 6.54. Distribution and autocorrelation functions of w_{MW} for different values of mp in subplots a-c and d, respectively, using $\mathrm{W}^{\mathrm{U}} = 60$ on debutanizer column data.

AD2A models are further compared in terms of local windows ($\widetilde{\mathrm{MW}}$) they select for JIT modeling at each query point. Here, comparison will be based on similarity of local regions, or samples selected at each query point by AD2A method using different mp values. There is no guarantee that AD2A with $mp = 1$ and $mp = 3$ selects identical "best" regions from historical data, some of these regions may overlap, but some may be totally different. For this purpose, a metric, $d_{i,j}$, is used to determine the distance of the selected historical samples in AD2A models using $mp = i$ and $mp = j$. A local region is identified by the time index of its starting point, and the absolute difference in

starting points of two regions, within the interval [0, 1000] is used to determine $d_{i,j}$ at each query point. A zero difference would mean identical regions are selected, while a difference of 1000 represents samples farthest from each other in time direction. Figure 6.55a shows the cumulative distribution of distances for different values of mp. It is seen that exact overlap between $mp = 3$ and $mp = 5$ reaches $\sim 70\%$, while overlap between $mp = 3$ (or $mp = 5$) with $mp = 1$ is only $\sim 30 - 40\%$, showing that the selected local regions are highly similar for $mp = 3$ and $mp = 5$, but quite different for $mp = 1$.

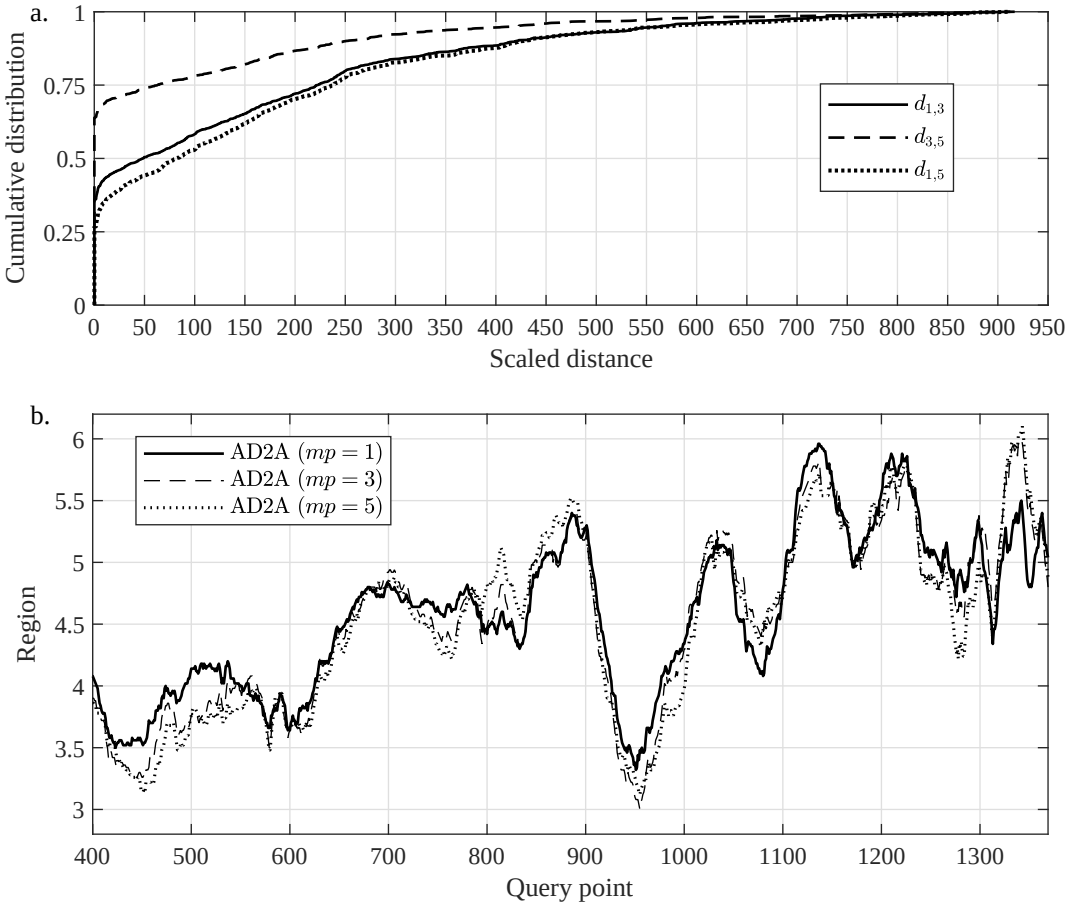

Figure 6.55. Comparison of AD2A models in terms of the local window selected for JITL, using cumulative distribution of metrics $d_{i,j}$ for $i, j \in \{1, 3, 5\}$ (a), and the local regions selected for JITL at each query, filtered for visual purposes (b).

In addition, the entire dataset is partitioned into eight regions which are comprised of ~ 300 observations, and presumed to represented different operating regions in the history of debutanizer column. Note that this approach is similar to comparison of batch learners in Section 6.2 when the same data was divided into four parts so as to obtain regions which were essentially free of concept change. Local windows of JITL models were classified according to these eight partitions, and a moving average filter with window size 20 was applied to smooth out these discrete values for each query point in Figure 6.55b. Trajectories of regions in all three AD2A models are in close proximity to each other, while AD2A ($mp = 1$) stands out slightly as it is different from rest of the models for some of the query points (see observations $400-600$, $900-1000$ and $1300-1400$ in Figure 6.55b). Therefore, a sufficiently large mp value (≥ 3), would render a better selection of both local windows, and ensemble weights.

An adaptive learning algorithm is not only expected to yield accurate predictions, but also be compatible with memory requirements of the computing system on which it is performed. In most applications, a constant number of data points is maintained for the historical dataset (historical window size), to prevent excessive data storage requirements. In one of the versions of this method, as a new observation is added to the historical database, the oldest one is discarded, hence making historical window size constant [25]. To employ dataset windowing (DW) in the current study, and see its effect on the predictive accuracy of adaptive learners, the initial size of the training set is kept constant at 300, and 1000 for synthetic data, and debutanizer data, respectively. DW model is labeled using the subscript DW as in AD2A_{DW}, and AD2A ($\alpha = 0.001$, $mp = 3$), the best prediction method obtained so far is used. Overall, supplying full historical data seems to help in improving predictive performance, albeit slightly. This shows that maintaining oldest observations in the training dataset is useful for future predictions. On the other hand, the loss in prediction accuracy due to constant historical window size may be negligible, depending on the application. The increase in $\overline{\text{RMSE}}$ due to DW is the largest for the CDMs, in which disturbance is changed between two levels in the form of steps (S2.1 and S2.2 in Figure 6.56).

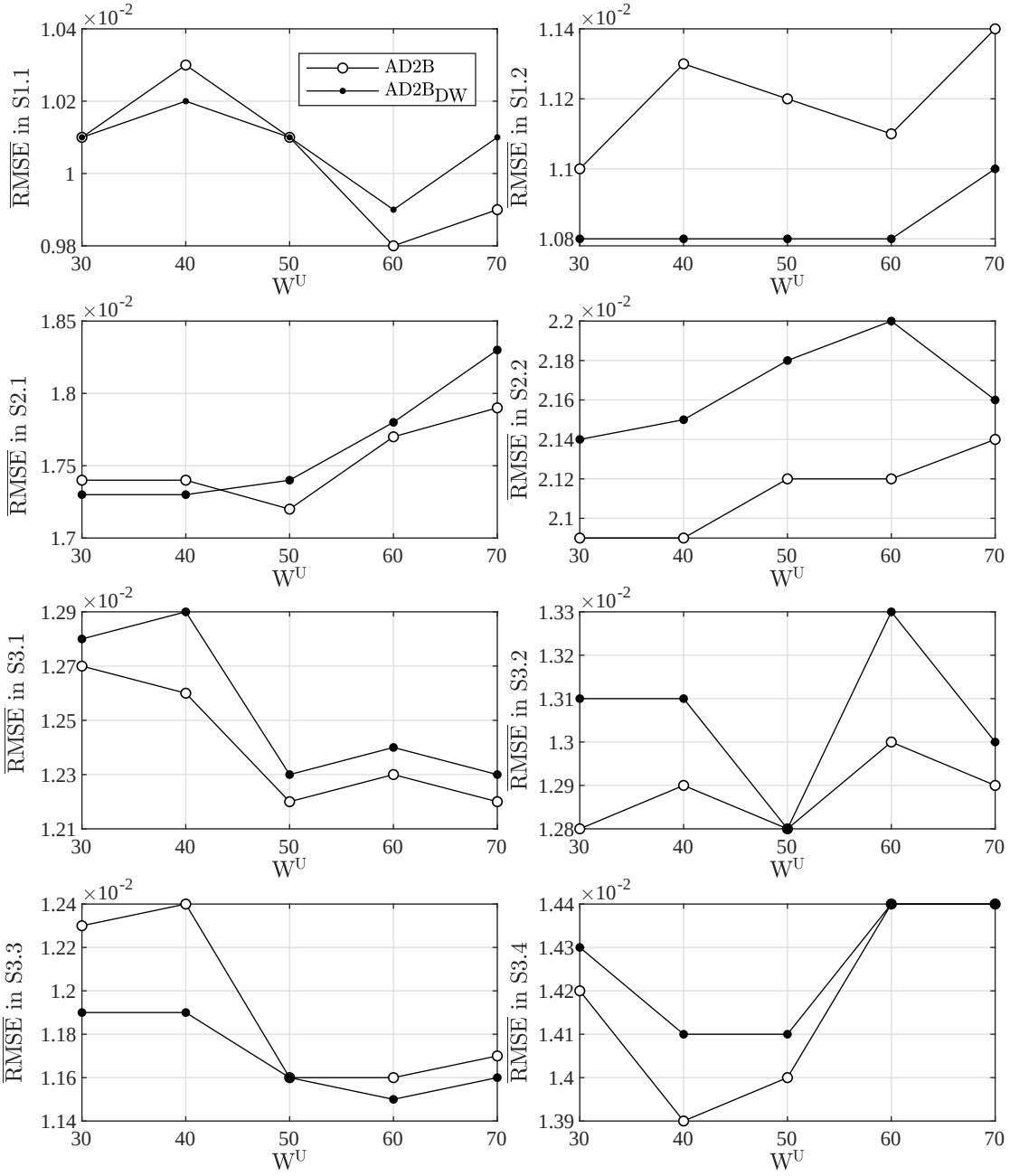

Figure 6.56. Comparison of AD2A models with and without DW on CSTR simulations.

These CDMs may be presumed to benefit the most from inclusion of older historical information due to the recurring changes of disturbance. On the other hand, in S1.1 and S1.2, the CDM is comprised of step changes between multiple levels, and when

concept change occurs in high frequency, it starts to resemble uniform distribution of operating regions which are allowable, hence the unmeasured disturbance may span a larger space compared to S2.1 and S2.2. Therefore, it may be asserted that it becomes more difficult for AD2 to identify recurring operating regions in the historical data. Hence, when DW is employed on this CDM, it either has little to no effect on $\overline{\text{RMSE}}$ (S1.1 in Figure 6.56), or it slightly decreases $\overline{\text{RMSE}}$ (S1.2 in Figure 6.56). The decrease in $\overline{\text{RMSE}}$s may be due to reduced variance during the localization step in AD2, however further experiments should be conducted to be able to make more concrete assertions on this observation. The effect of DW on AD2B has also been observed (not shown) to be similar to that on AD2A, corresponding figure can be found in the Appendix B (Figure B.5).

Application of the same method on debutanizer data shows, again a small decline in prediction accuracies for both AD2A and AD2B with DW (Figure 6.57a and Figure 6.57b); however, RMSEs still remain below those of the adaptive MW method (AD1C). In addition the difference in RMSEs between AD2A and AD2A_{DW}, and AD2B and AD2B_{DW} are quite small, questioning the need to store the whole historical dataset. Furthermore, similarity of local regions selected by both methods is examined, using filtered region selection (as desribed previously) for AD2A_{DW}, and compared with those of AD2A using $\text{W}^{\text{U}} = 60$ in Figure 6.57c (dashed and solid lines, respectively), in which both models follow similar trajectories; however there is a constant gap of ~ 300 observations in between two lines, which is possibly due to restriction put on AD2A_{DW}. Since oldest regions are accessible to AD2A, it may occasionally select a local region for JITL from these oldest historical data, thereby lowering the moving average estimate. The similarities in trajectories are, on the other hand, due to the fact that AD2A already favors more recent observations for local modeling over older ones in the historical dataset, thus the constraint put on the historical dataset has only slight effect on the location of window of JITL model, and the predictive accuracy of method.

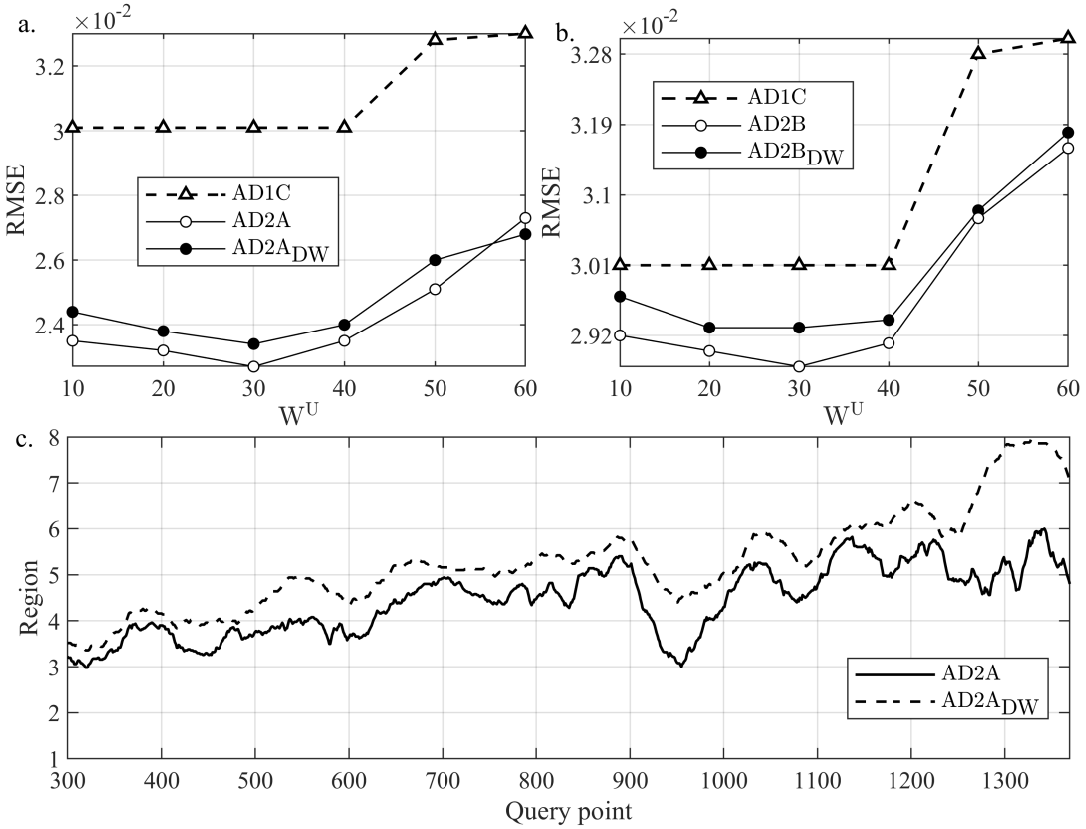

Figure 6.57. Comparison of AD2 models with and without DW on debutanizer data (a and b), and filtered regions of local modeling in JITL models in AD2A and AD2A_{DW} here $\text{W}^{\text{U}} = 60$, and $mp = 3$ is used (c).

Finally, we compare the prediction accuracies of the novel adaptive methods suggested in this current thesis (AD1A, AD1B, AD1C, and AD2A), with those of constant window size MW methods using PLS_{MW} and RVM_{MW}, and some of the most commonly used and state-of-the-art adaptive methods from the literature, such as RPLS [27], EL-WPLS [119], CoJITL [38] and OEOA [43] on synthetic data and debutanizer dataset (Table 6.7 and 6.8, respectively). For the sake of brevity, $\overline{\text{RMSE}}$, sdRMSE and $\overline{\text{maxAE}}$ are averaged over eight CDMs in Table 6.7. The same metrics are separately given for each CDM in Appendix C. Parameters of conventional methods taken from the literature were determined via trial and error to yield the highest accuracy on synthetic data, and debutanizer data separately. RPLS was employed using $\lambda_f \in \{0.3, 0.7, 1\}$ and

$\lambda_f \in \{0.01, 0.3, 1\}$ for synthetic and real data, respectively. PLS_{MW} was implemented using online and offline optimization of number of components. In MW methods using constant W (PLS_{MW} (W), PLS_{MW} (TS) and RVM_{MW}), which is the algorithm most commonly used in the literature and the industry, W is set to 30 and 50 for synthetic data, and 200 for debutanizer data. Parameters of CoJITL are taken to be $\lambda = 0.01, \text{J} = 0$, and $\text{LV} = 12$ for synthetic data, and $\lambda = 0.25, \text{J} = 0$, and $\text{LV} = 2$ for debutanizer column data. The learning method in CoJITL, namely PCR and PLS, is not found to have a significant difference on prediction accuracy on neither of the datasets, hence only the one with smallest prediction errors were reported. Since there was no code available for use, CoJITL method was implemented according to the algorithm presented in its respective paper [38]. Results from state of the art methods, ELWPLS and OEOA, are reported for debutanizer data only. RMSE and R_p^2 values of ELWPLS in Table 6.8 were obtained from its corresponding article, in which it was employed on debutanizer data using the same training/test set partitioning adopted in the current thesis, however, the rest of the evaluation metrics on Table 6.8 were not available [119]. Two versions of OEOA, with and without ordered aggregation, were used, but only OEOA with ordered aggregation is reported in Table 6.8 as it had higher prediction accuracy. OEOA parameters were set as authors suggest [43]; minimum number of ensemble models in both OEOA models were taken to be 15, while the maximum number of of models were 15 and 30, for the unordered and ordered versions, respectively. 10fold CV is employed, sigmoidal activation function is used in OS-ELM models while the maximum number of nodes in OS-ELM models is set to 20, $\text{sW} = 10$, and α parameter is taken to be 0.04. The unordered version of OEOA model yielded a RMSE of 0.0457 for $\text{sW} = 10$. The OEOA-toolbox source code provided by its respective authors had been used to implement this method [43]. In summary, the suggested methods in the literature are tuned to yield the best predictive performance on the currently studied datasets, so that a rigorous comparison with the suggested methods in this thesis would be possible.

Parameters of the adaptive algorithms, which were proposed in the current thesis, were also selected to yield minimum prediction error. The parameter $\text{W}^{\text{U}} \in \{30, 50\}$

was used for synthetic data, and W^U was taken to be 30 for debutanizer data. Value of α parameter was taken to be 0.001 in AD1C and AD2A, and AD2A is employed with $mp = 3$, and $\widetilde{MW} = 30$ and $\widetilde{MW} = 20$ for synthetic and debutanizer data, respectively. It is worth noting that, while W^U is fixed at its optimum value in AD2A for all adaptive learners developed in this thesis, it had been observed in previous analyses that the change in predictive performance with respect to W^U was considerably small for both data (see Figure 6.38 and 6.44 for the synthetic data, and Figure 6.42, 6.43 and 6.50 for the debutanizer data).

For small values of λ_f, RPLS appeared to have acceptable adaptation capability. However, its performance was shown to be highly dependent on the parameter λ_f, hence this parameter should be tuned effectively and this tuning might be involved as suggested by widely different parameter values for both data (compare Table 6.7 and Table 6.8). Predictive performances of MW methods with constant W (PLS_{MW} and RVM_{MW}) were observed to be superior to RPLS, as it is often stated in the literature [33, 34]. Online optimization of PLS components could marginally improve predictions (PLS_{MW} (W)), but RVM based MW approach had performed better on both data, albeit slightly. CoJITL, an adaptive learner which is relatively popular in the industry, has poor predictive performance of both synthetic and real data. OEOA was employed on debutanizer data only, it was observed to yield a more favorable reduction in RMSE in comparison to ELWPLS, another ensemble learner suggested in the literature. Among the MW methods with adaptive W developed in this thesis, virtual drift detection in AD1A reduced RMSE by 42%, and real concept drift detection in AD1B and AD1C yielded more than 70% reduction in RMSE (from ~ 0.140 to as low as ~ 0.03), over the MW method with constant W (RVM_{MW}) on debutanizer data. AD1C yielded overall the best performance among adaptive W MW methods. AD1B had shown slightly better performance on certain CDMs (Table C.3 and C.4), but AD1C predictions slightly deteriorate for small W^U in some CDMs (Table C.5). However, AD1C appears to be more robust to W^U, since the effect of this parameter on AD1C were observed to be much smaller, compared to its effect on AD1B, particularly on debutanizer data (Figure 6.43). Sensitivity of AD1B to the parameter W^U, and the

consequent decline in predictive performance was observed to be more evident at large W^U (S1.1, S3.2 and S3.3 on Figure 6.38). The ensemble learner, AD2A was seen to reduce prediction errors of adaptive MW method (AD1C) considerably, and it yielded ultimately the best predictive performance among all adaptive learners proposed in this thesis. A 85% reduction in RMSE (from ~ 0.140 to ~ 0.023) was observed from the traditional MW approach on debutanizer data. The effect of W^U on AD2A in its predictive performance is much smaller compared to other methods, making this method highly favorable. For comparison, when W was increased to 30 in OEOA methods, RMSEs of the unordered and ordered versions increased to 0.0784 and 0.0804, respectively, on debutanizer data, showing that prediction accuracy of OEOA methods are highly dependent of MW size, rendering this method less practical for industrial use, compared to the adaptive methods, suggested in this thesis. In addition to providing accurate predictions, it is expected that an adaptive learning algorithm should also have relatively small computational burden since it is to be used to obtain predictions in real time. For that reason, the aforementioned adaptive learners need to be compared on the basis of their time complexity (a specific type of computational complexity), which is the run time of an algorithm in computer science. In the current thesis, time complexity is quantified using average elapsed time for a single query point (AET), which we define for the elapsed time per prediction, in the units of seconds. For simplicity, time complexity was determined on debutanizer data only (AET column in Table 6.8). Since the maximum AET was observed to be 2.29 seconds (OEOA), it appears that all adaptive methods in Table 6.8 are compatible with an online prediction scenario. However, in terms of gain in prediction accuracy for increased computational time, AD1C, which was deemed the best performing adaptive MW learner proposed in the current study, is the winner. Furthermore, it was observed that AD2A provided 25% reduction in RMSE from AD1C, while it had shown a 100 fold increase in AET value, though its time complexity was still acceptable. It is also worth emphasizing that all adaptive learning algorithms, which we proposed in the current thesis, yield AET < 1 s, and this should be considered as another advantage of our methods.

Table 6.7. Comparison of proposed adaptive methods with other conventional adaptive methods from the literature on CSTR simulation data, results are averaged over all CDMs.

Model	**Settings**	$\overline{\textbf{RMSE}}$	**sdRMSE**	$\overline{\textbf{maxAE}}$
RPLS	$\lambda = 1$	0.0337	0.0215	0.0943
	$\lambda = 0.7$	0.0330	0.0211	0.0925
	$\lambda = 0.3$	0.0294	0.0186	0.0844
PLS_{MW} (W)	W = 30	0.0190	0.0086	0.1226
	W = 50	0.0192	0.0082	0.1034
PLS_{MW} (TS)	W = 30	0.0222	0.0156	0.1742
	W = 50	0.0202	0.0091	0.1097
RVM_{MW}	W = 30	0.0175	0.0116	0.1312
	W = 50	0.0173	0.0080	0.0919
CoJITL	W = 30	0.0400	0.0250	0.1276
	W = 50	0.0411	0.0267	0.1264
AD1A	W = 30	0.0150	0.0072	0.0838
	W = 50	0.0154	0.0080	0.0823
AD1B	$\text{W}^{\text{U}} = 30$	0.0156	0.0069	0.0092
	$\text{W}^{\text{U}} = 50$	0.0155	0.0071	0.0089
AD1C	$\text{W}^{\text{U}} = 30$	0.0152	0.0071	0.0880
	$\text{W}^{\text{U}} = 50$	0.0150	0.0073	0.0857
AD2A	$\text{W}^{\text{U}} = 30$	0.0132	0.0051	0.0864
	$\text{W}^{\text{U}} = 50$	0.0131	0.0050	0.0827

Table 6.8. Comparison of proposed adaptive methods with other conventional adaptive methods from the literature on debutanizer column data.

Model	**Settings**	**RMSE**	**MAE**	**maxAE**	$\boldsymbol{R_p^2}$	**AET (s)**
RPLS	$\lambda_f = 1$	0.1734	0.1244	0.727	0.218	3.19×10^{-2}
	$\lambda_f = 0.3$	0.1602	0.1159	0.681	0.332	3.19×10^{-2}
	$\lambda_f = 0.01$	0.0413	0.0244	0.684	0.956	3.19×10^{-2}
$\mathbf{PLS_{MW}}$ (W)	W = 200	0.1488	0.1095	0.675	0.564	3.18×10^{-2}
$\mathbf{PLS_{MW}}$ (TS)	W = 200	0.1475	0.1106	0.680	0.568	2.00×10^{-3}
$\mathbf{RVM_{MW}}$	W = 200	0.1475	0.1093	0.691	0.569	4.60×10^{-3}
CoJITL	W = 10	0.1646	0.0957	1.523	0.275	3.92×10^{-2}
ELWPLS	-	0.0850	-	-	0.805	-
OEOA	W = 10	0.0429	0.0253	0.599	0.956	2.29
AD1A	W = 30	0.0853	0.0518	0.640	0.836	5.60×10^{-3}
AD1B	$W^U = 30$	0.0427	0.0217	0.434	0.961	5.89×10^{-1}
AD1C	$W^U = 30$	0.0301	0.0140	0.378	0.981	8.25×10^{-3}
AD2A	$W^U = 30$	0.0227	0.0125	0.244	0.989	7.91×10^{-1}

7. CONCLUSIONS AND RECOMMENDATIONS

In order to measure up in today's highly competitive manufacturing industries, and meet the increased restrictions on manufacturing due to environmental and safety concerns, production facilities are expected to be more strict in their implementation of process control, monitoring and fault detection. For that reason, a large number of process variables are measured online during operation, while quality variables, which are direct indicators of final product properties, often cannot be attained online at the same rate as process variables, due to manufacturing conditions, or the need to perform long laboratory analyses. Soft sensors have been introduced for online prediction of these variables. Data driven soft sensor design has grown in popularity, mirroring the recent interest in data based statistical sciences. Global modeling techniques, e.g. ARX modeling, PLS and PCR, are among examples of traditional statistical learning methods used for soft sensor design. In this thesis, the aim was to develop, and evaluate various methods for soft sensor design in chemical engineering processes, using statistical learning tools in steady state and dynamic settings. Results from this work comprise three main sections, which aim to address different issues in data driven soft sensor design. Despite this partitioning of results, these sections are not mutually exclusive, and follow one another.

The first issue we had investigated was the presence of multicollinearity and redundancy in industrial data with virtual drift, and conventional methods, such as PLS and RR may not cope well enough with such problems particularly in high dimensional problems. Incorporating feature selection with traditional linear and nonlinear learners, and ensemble modeling was suggested to address this issue. Experiments were conducted on synthetic data from steady state simulation of a CDU, in which at least 70 predictors were considered as inputs to the predictive models so as to mimic an industrial plant with numerous process variables. 37 learners (Table 6.1), of which 36 are based on batch learning techniques and one is an adaptive learner were constructed. Learners were assessed based on their interpolation and extrapolation performances.

The interpolation performance was used to evaluate predictive performance of learners when operating conditions remain similar in training/test sets, whereas the extrapolation scenario was introduced to examine learners under virtual concept drift. Utilizing feature selection, particularly MRMR, in single PLS and ANN models, and ensemble ANN learners (Red_{corr} + PLS, Red_{corr} + ANN_{Cr} and Red_{MI} + ANN_{Cr}) were shown to outperform their conventional counterparts in terms of both average and worst case scenario performances (Figure 6.6). While forward selection increased interpolation efficiencies from ~ 0.7 to ~ 0.8, performance of learners using feature selection without MRMR on extrapolation data were less than desired (efficiencies of FS^{cal} + PLS and FS + ANN remain below ~ 0.6 in Figure 6.6b). This decline in extrapolation performance was perhaps due to small number of predictors selected by FS^{cal} and FS^{val}, which has led to poor generalization capability (Figure 6.4). It was observed that ensemble learning had stabilized ANN models considerably, and increased their predictive performance, hence the best results in batch learners were achieved by ensemble ANN models used in conjunction with feature selection (avgEff of FS + ANN_{Cr}, Red_{corr} + ANN_{Cr} and Red_{MI} + ANN_{Cr} are well above ~ 0.8 in Figure 6.6a). In addition, sparse regression techniques (Lasso and RVM) had shown outstanding performance (RVM in particular) on average considering both interpolation and extrapolation settings (Table 6.3). An increase in extrapolation efficiency was observed for Lasso when squared predictors were included (models 8.1 and 8.2 in Figure 6.2c, d), therefore it can be asserted that Lasso may be the most promising learner for improving predictive models via addition of squared predictors. On the other hand, RR, and PLS based models (models 2, 4, 5, 6, 9 and 10 in Table 6.1) were shown to be inferior to handle problems with such large number of collinear predictors, and redundancy. Impressive extrapolation performance of the sole adaptive learner ($\text{RVM}_{\text{Local}}$) was to be expected beforehand, and it was shown that it could adapt to new concepts by learning from test points appended to the training set during prediction.

In addition to the analyses on steady state data, soft sensor design for dynamic data was considered. Respective performances of PLS and RVM based learners on steady state simulation data motivated the comparative study of these two methods

in terms of batch and online prediction, on dynamic data. Moreover, in soft sensor design literature, the effect of inclusion of lagged predictors in FIR models on prediction accuracy, and how to determine the optimum FIR model order are subjects, which have not received extensive attention. Therefore, in batch modeling scenario, performance of PLS and RVM models were compared using FIR models of different orders under virtual concept drift. Both synthetic data (all CDMs from CSTR simulation), and data from a industrial process (debutanizer column dataset) were utilized for assessing predictive performances. RVM was shown to exhibit small variance in feature selection, and yield higher prediction accuracy than PLS in batch modeling. In addition, it was noted that selection of FIR model order in batch learning via repeated 10fold CV may produce acceptable predictive accuracy (Table 6.4). In the online prediction scenario, PLS and RVM based learners were examined based on their predictive performances in the presence of not only multicollinearity, but also real concept drift. The MW approach was adopted to cope with the problem of concept drift. RVM based MW model (RVM_{MW}) was shown to be more suitable for handling multicollinear inputs with $N \leq p$. Moreover, it was observed that online optimization of PLS components (PLS_{MW} (W)) was advantageous for moderate to large window sizes (W > 50), since it was possible to reduce bias by adapting model parameters. However, the advantage of online parameter tuning waned for MWs of small size when reduction in bias could not be realized, and it could risk increasing variance, possibly due to the fact that CV in MW converged to LOOCV for small number of observations. It is also worth noting that in data streams, online predictions should be provided in real time, hence the increased burden of computation from online parameter tuning in PLS_{MW} (W) might be of concern in some applications. Overall, it was concluded that RVM_{MW} provided higher prediction accuracy at lower computational time compared to PLS_{MW} models. It should also be emphasized that, one should be cautious of process variables which change in a narrow range, or remain constant, particularly throughout a moving window containing small number of observations, since they might be neglected as a result of embedded feature selection in RVM training.

The final issue we focused on was soft sensor design in the presence of concept drift. Various RVM-based adaptive learning algorithms, which were grouped under two names, AD1 and AD2, were proposed to handle concept drift, and avert soft sensor degradation. Experiments were performed on CDMs from CSTR simulations, in which real concept drift is present, and debutanizer column data. The first set of adaptive learners (AD1A, AD1B and AD1C) were developed with the aim of improving MW models by adapting the size of the window. In AD1A, Mahalanobis distance (d_{M}^2) check was introduced to determine whether a query point belongs to its respective MW. MWs which are deemed to be inconvenient for predictions by AD1A, W is momentarily enlarged, and the problem of near-constant predictors, which was pointed out in the preliminary section, in the MW are avoided. AD1A was shown to reduce gross errors obtained from MW model at small Ws by large amounts on both datasets (S2.2 and S3.4 in Figure 6.37, and Figure 6.41). However, it also disrupted adaptation to changing concepts at large Ws (S2.2 and S3.4 in Figure 6.36), since its window adjustment mechanisms was based solely on increasing the current window size (Figure 6.19). For that reason, incorporating R_{adj}^2 check in AD1B (Figure 6.20), and $\hat{\sigma}^2$ monitoring in AD1C was suggested to improve upon AD1A (Figure 6.26). In most studies in the literature, explicit concept drift detection is suggested to be more advantageous than implicit detection mechanisms, since explicit mechanisms provide an informed setting for model adaptation in comparison to the blind updates performed by implicit drift detectors [26,116,137,139]. AD1B and AD1C, as proposed in this thesis, differ in their concept drift detection mechanism, as AD1B is based on implicit detection, whereas AD1C explicitly detects concept change. Both methods adapt by adjusting W, for that purpose AD1B uses R_{adj}^2 statistic to find the best MW for each query point. In AD1C, on the other hand, model quality is measured via $\hat{\sigma}^2$, and monitored using an EWMA chart in order to explicitly detect concept change. Window is adjusted only when a deterioration in model quality is detected, and remedied by this adjustment. Otherwise window size remains constant. Explicit concept drift detection in AD1C was shown to be more favorable than the implicit approach in AD1B, possibly preventing the high variability of W seen in AD1B (Figure 6.21), yielding better predictive performance over a wide range of CDMs for different values of W^{U} (Figure 6.38), and its performance

on real data was observed to surpass that of AD1B (Figure 6.43).

Ensemble learning has often been suggested for adapting to concept changes, and improve predictive accuracy in statistical learning literature [40, 141]. A major drawback of ensemble learning algorithms is their complexity, and increased number of parameters. Therefore, a somewhat simple, yet effective approach to ensemble learning was proposed in this thesis (AD2) by combining MW predictions with JITL. Contrary to the traditional utilization of ensemble models in the literature, we suggest using the model from recent adaptive MW as one of the base learners in the ensemble, while using only one other base learner constructed from the historical data. It was expected, by this approach, that merits of both methods to specific disturbances, i.e. success of MW against ramp concept changes, and that of local learning to step concept changes may be combined. In AD2, weighted average of predictions from MW model (AD1C), and JITL model is used as the final output, and ensemle weights are determined via validation of past observations (Figure 6.35). AD2 methods were shown to yield the best predictive performance both on real data (Figure 6.50), and in all CDMs, particularly in those with intense CDs (S2 and S3 in Figure 6.44), considerable reduction in $\overline{\text{RMSE}}$s was achieved. In addition, it was observed that selective application of ensemble modeling (AD2B) may be more beneficial for small magnitude CDMs, which suggests the use of a large number of models in the ensemble may have downsides (Figure 6.48). Furthermore, it was observed that ensemble models should be validated on multiple observations, since using validation error on a single point was shown to yield unreliable weight estimates in both synthetic and real data (Figure 6.52 and Figure 6.54). It is also worth noting that AD2 method could provide excellent prediction performance under memory restrictions (Figure 6.56 and Figure 6.57). Overall, adaptive learning algorithms, which were developed in this thesis, were proven to yield superior performance in comparison to conventional adaptive learners from the literature on both the synthetic data (Table 6.7), and real data (Table 6.8). Computational speed (AET$<$ 1s in Table 6.8)), easily intepretable linear RVM models obtained for each query point, and monitoring of $\hat{\sigma}^2$ to determine the state of the process, i.e. whether a novel disturbance acts on the process or not, are the additional advantages on top of

high prediction accuracy offered by our suggested method.

In the current thesis, explicit concept detection is implemented in AD1C, and it was found that adaptation through explicit detection of concept change should be favored over implicit detection. This line of thought can be extended to introduce different mechanism for drift detection. Recall that during RVM training, both model error variance, and prediction error variance estimates are obtained. Therefore, instead of monitoring $\hat{\sigma}^2$, one could use EWMA charts for detecting abnormal prediction errors directly, and develop a window adjustment strategy based on the changes in consequent prediction errors, in comparison to a baseline prediction errors from normal operating conditions. In the current study, ensemble weights of AD2 were observed to have a bi-modal distribution in low frequency drifts (Figure 6.45a and b), and a uni-modal distribution in high frequency drifts (Figure 6.46a and b) for CDMs of high magnitude. MW weights were highly concentrated around smaller values, which suggests the inclination to favor JITL, however in CDM of small magnitude, the accumulation of weights around unity was found to be rather puzzling (Figure 6.47a and b). In addition, in CDMs of small magnitude, selective application of ensemble learning (AD2B) yielded was more advantageous to full-on ensemble learning (AD2A), which could end up worsening the predictions of sole MW model (S1.1 and S1.2 in Figure 6.44). Therefore, the effect of type, frequency and magnitude of CDs on the performance of adaptive ensemble learners may be investigated in the future to decipher on the differences in results observed in this study for different CDMs, and improve the prediction accuracy.

REFERENCES

1. Liao, Y., F. Deschamps, E. de Freitas Rocha Loures *et al.*, "Past, present and future of Industry 4.0 - a systematic literature review and research agenda proposal", *International Journal of Production Research*, Vol. 55, No. 12, pp. 3609–3629, 2017.

2. Fortuna, L., S. Graziani, A. Rizzo *et al.*, *Soft Sensors for Monitoring and Control of Industrial Processes*, Springer London, 2006.

3. Carmichael, I. and J. S. Marron, "Data science vs. statistics: two cultures?", *Japanese Journal of Statistics and Data Science*, 2018.

4. Kadlec, P., B. Gabrys and S. Strandt, "Data-Driven Soft Sensors in the Process Industry", *Computers & Chemical Engineering*, Vol. 33, pp. 795–814, 2009.

5. Vapnik, V. N., *Statistical Learning Theory*, John Wiley & Sons, 1998.

6. Hastie, T., R. Tibshirani and J. Friedman, *The Elements of Statistical Learning*, Springer, 2009.

7. Kano, M. and M. Ogawa, "The state of the art in chemical process control in Japan: Good practice and questionnaire survey", *Journal of Process Control*, Vol. 20, No. 9, 2010.

8. Qin, S., "Neural Networks for Intelligent Sensors and Control — Practical Issues and Some Solutions", *Neural Systems for Control*, pp. 213–234, Elsevier, 1997.

9. Macias, J. J., P. Angelov and X. Zhou, "A Method for Predicting Quality of the Crude Oil Distillation", *2006 International Symposium on Evolving Fuzzy Systems*, 2006.

10. van Deventer, J. S., K. M. Kam and T. J. van der Walt, "Dynamic modelling of a carbon-in-leach process with the regression network", *Chemical Engineering Science*, Vol. 59, No. 21, pp. 4575–4589, 2004.

11. Zahedi, G., A. Elkamel, A. Lohi *et al.*, "Hybrid artificial neural network—First principle model formulation for the unsteady state simulation and analysis of a packed bed reactor for CO2 hydrogenation to methanol", *Chemical Engineering Journal*, Vol. 115, No. 1-2, pp. 113–120, 2005.

12. Rong, H.-J., N. Sundararajan, G.-B. Huang *et al.*, "Sequential adaptive fuzzy inference system (SAFIS) for nonlinear system identification and prediction", *Fuzzy Sets and Systems*, Vol. 157, No. 9, pp. 1260–1275, 2006.

13. Khatibisepehr, S., B. Huang and S. Khare, "Design of inferential sensors in the process industry: A review of Bayesian methods", *Journal of Process Control*, Vol. 23, No. 10, pp. 1575–1596, 2013.

14. Walczak, B. and D. Massart, "Dealing with missing data", *Chemometrics and Intelligent Laboratory Systems*, Vol. 58, No. 1, pp. 15–27, 2001.

15. Davies, L. and U. Gather, "The Identification of Multiple Outliers", *Journal of the American Statistical Association*, Vol. 88, No. 423, pp. 782–792, 1993.

16. Gonzalez, G., M. Orchard, J. Cerda *et al.*, "Local models for soft-sensors in a rougher flotation bank", *Minerals Engineering*, Vol. 16, No. 5, pp. 441–453, 2003.

17. Casali, A., G. Gonzalez, F. Torres *et al.*, "Particle size distribution soft-sensor for a grinding circuit", *Powder Technology*, Vol. 99, No. 1, pp. 15–21, 1998.

18. Lennart Ljung, T. G., *Modeling of Dynamic Systems*, Pearson Education (US), 1994.

19. Dayal, B. S. and J. F. MacGregor, "Identification of Finite Impulse Response

Models: Methods and Robustness Issues", *Industrial & Engineering Chemistry Research*, Vol. 35, No. 11, pp. 4078–4090, 1996.

20. Kaneko, H. and K. Funatsu, "Preparation of comprehensive data from huge data sets for predictive soft sensors", *Chemometrics and Intelligent Laboratory Systems*, Vol. 153, pp. 75–81, 2016.

21. Alpaydın, E., *Introduction to Machine Learning*, The MIT Press, 2009.

22. Kano, M. and K. Fujiwara, "Virtual Sensing Technology in Process Industries: Trends and Challenges Revealed by Recent Industrial Applications", *Journal oF Chemical Engineering of Japan*, pp. 1–17, 2013.

23. Ma, Y. and B. Huang, "Bayesian Learning for Dynamic Feature Extraction With Application in Soft Sensing", *IEEE Transactions on Industrial Electronics*, Vol. 64, No. 9, pp. 7171–7180, 2017.

24. Guyon, I. and A. Elisseeff, "An Introduction to Variable and Feature Selection", *Journal of Machine Learning Research*, Vol. 3, pp. 1157–1182, 2003.

25. Kadlec, P., R. Grbić and B. Gabrys, "Review of adaptation mechanisms for data-driven soft sensors", *Computers & Chemical Engineering*, Vol. 35, No. 1, pp. 1–24, 2011.

26. Widmer, G. and M. Kubat, "Learning in the presence of concept drift and hidden contexts", *Machine Learning*, Vol. 23, No. 1, pp. 69–101, 1996.

27. Qin, S. J., "Recursive PLS algorithms for adaptive data modeling", *Computers & Chemical Engineering*, Vol. 22, No. 4-5, 1998.

28. Qin, S. and T. McAvoy, "Nonlinear PLS modeling using neural networks", *Computers & Chemical Engineering*, Vol. 16, No. 4, pp. 379–391, 1992.

29. Cauwenberghs, G. and T. Poggio, "Incremental and Decremental Support Vector Machine Learning", *Proceedings of the 13th International Conference on Neural Information Processing Systems*, MIT Press, 2000.

30. Laskov, P., C. Gehl, S. Krüger *et al.*, "Incremental Support Vector Learning: Analysis, Implementation and Applications", *Journal of Machine Learning Research*, Vol. 7, pp. 1909–1936, 2006.

31. Tsang, I. W., J. T. Kwok and P.-M. Cheung, "Core Vector Machines: Fast SVM Training on Very Large Data Sets", *Journal of Machine Learning Research*, Vol. 6, pp. 363–392, 2005.

32. Liu, Y., N. Hu, H. Wang *et al.*, "Soft chemical analyzer development using adaptive least-squares support vector regression with selective pruning and variable moving window Size", *Industrial & Engineering Chemistry Research*, Vol. 48, No. 12, 2009.

33. Mu, S., Y. Zeng, R. Liu *et al.*, "Online dual updating with recursive PLS model and its application in predicting crystal size of purified terephthalic acid (PTA) process", *Journal of Process Control*, Vol. 16, No. 6, pp. 557–566, 2006.

34. Ni, W., S. D. Brown and R. Man, "A localized adaptive soft sensor for dynamic system modeling", *Chemical Engineering Science*, Vol. 111, pp. 350–363, 2014.

35. Bontempi, G., M. Birattari and H. Bersini, "Lazy learning for local modelling and control design", *International Journal of Control*, Vol. 72, No. 7-8, pp. 643–658, 1999.

36. Atkeson, C. G., A. W. Moore and S. Schaal, "Locally Weighted Learning", *Artificial Intelligence Review*, Vol. 11, No. 1, pp. 11–73, 1997.

37. Cheng, C. and M.-S. Chiu, "A new data-based methodology for nonlinear process modeling", *Chemical Engineering Science*, Vol. 59, No. 13, pp. 2801–2810, 2004.

38. Fujiwara, K., M. Kano, S. Hasebe *et al.*, "Soft-sensor development using correlation-based just-in-time modeling", *AIChE Journal*, Vol. 55, No. 7, pp. 1754–1765, 2009.

39. Krawczyk, B., L. L. Minku, J. Gama *et al.*, "Ensemble learning for data stream analysis: A survey", *Information Fusion*, Vol. 37, pp. 132–156, 2017.

40. Minku, L., A. White and X. Yao, "The Impact of Diversity on Online Ensemble Learning in the Presence of Concept Drift", *IEEE Transactions on Knowledge and Data Engineering*, Vol. 22, No. 5, pp. 730–742, 2010.

41. Kadlec, P. and B. Gabrys, "Local learning-based adaptive soft sensor for catalyst activation prediction", *AIChE Journal*, Vol. 57, No. 5, pp. 1288–1301, 2010.

42. Soares, S., C. H. Antunes and R. Araújo, "Comparison of a genetic algorithm and simulated annealing for automatic neural network ensemble development", *Neurocomputing*, Vol. 121, pp. 498–511, 2013.

43. Soares, S. G. and R. Araújo, "An adaptive ensemble of on-line Extreme Learning Machines with variable forgetting factor for dynamic system prediction", *Neurocomputing*, Vol. 171, pp. 693–707, 2016.

44. Yuan, X., J. Zhou, Y. Wang *et al.*, "Multi-similarity measurement driven ensemble just-in-time learning for soft sensing of industrial processes", *Journal of Chemometrics*, pp. 1–14, 2018.

45. Jin, H., X. Chen, L. Wang *et al.*, "Dual learning-based online ensemble regression approach for adaptive soft sensor modeling of nonlinear time-varying processes", *Chemometrics and Intelligent Laboratory Systems*, Vol. 151, pp. 228–244, 2016.

46. Liu, J., "On-line soft sensor for polyethylene process with multiple production grades", *Control Engineering Practice*, Vol. 15, No. 7, pp. 769–778, 2007.

47. Höskuldsson, A., "PLS regression methods", *Journal of Chemometrics*, Vol. 2, No. 3, pp. 211–228, 1988.

48. Wold, H., "Path Models with Latent Variables: The NIPALS Approach", *Quantitative Sociology*, pp. 307–357, Elsevier, 1975.

49. Wold, S., M. Sjöström and L. Eriksson, "PLS-regression: a basic tool of chemometrics", *Chemometrics and Intelligent Laboratory Systems*, Vol. 58, No. 2, pp. 109–130, 2001.

50. Mejdell, T. and S. Skogestad, "Composition estimator in a pilot-plant distillation column using multiple temperatures", *Industrial & Engineering Chemistry Research*, Vol. 30, pp. 2555–2564, 1991.

51. Mejdell, T. and S. Skogestad, "Estimation of distillation compositions from multiple temperature measurements using partial-least-squares regression", *Industrial & Engineering Chemistry Research*, Vol. 30, No. 12, pp. 2543–2555, 1991.

52. Fisher, R. A. and W. A. Mackenzie, "Studies in crop variation. II. The manurial response of different potato varieties", *The Journal of Agricultural Science*, Vol. 13, No. 03, p. 311, 1923.

53. Wold, S., N. Kettaneh-Wold and B. Skagerberg, "Nonlinear PLS modeling", *Chemometrics and Intelligent Laboratory Systems*, Vol. 7, No. 1-2, pp. 53–65, 1989.

54. Rosipal, R. and L. J. Trejo, "Kernel Partial Least Squares Regression in Reproducing Kernel Hilbert Space", *Journal of Machine Learning Research*, Vol. 2, pp. 97–123, 2002.

55. Wise, B. M. and N. L. Ricker, "Identification of finite impulse response models with continuum regression", *Journal of Chemometrics*, Vol. 7, No. 1, pp. 1–14, 1993.

56. Allen, D. M., "The relationship between variable selection and data agumentation and a method for prediction", *Technometrics*, Vol. 16, No. 1, pp. 125–127, 1974.

57. Tibshirani, R., "Regression shrinkage and selection via the lasso", *Journal of the Royal Statistical Society. Series B*, pp. 267–288, 1996.

58. Efron, B., T. Hastie, I. Johnstone *et al.*, "Least angle regression", *Annals of Statistics*, Vol. 32, pp. 407–499, 2004.

59. lldiko E. Frank and J. H. Friedman, "A Statistical View of Some Chemometrics Regression Tools", *Technometrics*, Vol. 35, No. 2, pp. 109–135, 1993.

60. Tipping, M. E., "Sparse Bayesian Learning and the Relevance Vector Machine", *Journal of Machine Learning Research*, Vol. 1, pp. 211–244, 2001.

61. Efron, B., "Estimating the Error Rate of a Prediction Rule: Improvement on Cross-Validation", *Journal of the American Statistical Association*, Vol. 78, No. 382, pp. 316–331, 1983.

62. Efron, B. and R. Tibshirani, "Bootstrap Methods for Standard Errors, Confidence Intervals, and Other Measures of Statistical Accuracy", *Statistical Science*, Vol. 1, No. 1, pp. 54–75, 1986.

63. Gelman, A., "Prior distributions for variance parameters in hierarchical models (comment on article by Browne and Draper)", *Bayesian Analysis*, Vol. 1, No. 3, pp. 515–534, 2006.

64. Rasmussen, C. E. and Z. Ghahramani, "Occam's Razor", *Advances in Neural Information Processing Systems 13*, pp. 294–300, MIT Press, 2001.

65. MacKay, D. J. C., *Bayesian Methods for Backpropagation Networks*, pp. 211–254, Springer New York, 1996.

66. Bishop, C. M. and M. E. Tipping, "Variational relevance vector machines", *Proceedings of the Sixteenth conference on Uncertainty in artificial intelligence*, pp. 46–53, Morgan Kaufmann Publishers Inc., 2000.

67. Grbic, R., D. Sliskovic and P. Kadlec, "Adaptive soft sensor for online prediction based on moving window Gaussian process regression", *International Conference on Machine Learning and Applications (ICMLA)*, Vol. 2, pp. 428–433, 2012.

68. Ge, Z., T. Chen and Z. Song, "Quality prediction for polypropylene production process based on CLGPR model", *Control Engineering Practice*, Vol. 19, No. 5, pp. 423–432, 2011.

69. Ge, Z. and Z. Song, "Nonlinear Soft Sensor Development Based on Relevance Vector Machine", *Industrial & Engineering Chemistry Research*, Vol. 49, No. 18, pp. 8685–8693, 2010.

70. Himmelblau, D. M., "Applications of artificial neural networks in chemical engineering", *Korean Journal of Chemical Engineering*, Vol. 17, No. 4, 2000.

71. Piovoso, M. J. and A. J. Owens, "Neural network process control", *Proceedings of the conference on Analysis of neural network applications*, pp. 84–94, ACM, 1991.

72. Günay, M. E. and R. Yıldırım, "Neural network aided design of Pt-Co-Ce/Al2O3 catalyst for selective CO oxidation in hydrogen-rich streams", *Chemical Engineering Journal*, Vol. 140, No. 1-3, pp. 324–331, 2008.

73. Cybenko, G., "Approximation by superpositions of a sigmoidal function", *Mathematics of Control, Signals, and Systems*, Vol. 2, No. 4, pp. 303–314, 1989.

74. Hahnloser, R. H. R., R. Sarpeshkar, M. A. Mahowald *et al.*, "Digital selection and analogue amplification coexist in a cortex-inspired silicon circuit", *Nature*, Vol. 405, No. 6789, pp. 947–951, 2000.

75. Rumelhart, D. E., G. E. Hinton and R. J. Williams, "Learning representations by back-propagating errors", *Nature*, Vol. 323, No. 6088, pp. 533–536, 1986.

76. Bishop, C. M., *Pattern Recognition and Machine Learning*, Springer, 2006.

77. Prechelt, L., *Early Stopping — But When?*, pp. 53–67, Springer Berlin Heidelberg, 2012.

78. May, R., H. Maier and G. Dandy, "Review of Input Variable Selection Methods for Artificial Neural Networks", K. Suzuki (Editor), *Artificial Neural Networks*, chap. 2, InTech, 2011.

79. Bhattacharya, S., K. Pal and S. K. Pal, "Multi-sensor based prediction of metal deposition in pulsed gas metal arc welding using various soft computing models", *Applied Soft Computing*, Vol. 12, No. 1, pp. 498–505, 2012.

80. Wang, L., C. Shao, H. Wang and H. Wu, "Radial Basis Function Neural Networks-Based Modeling of the Membrane Separation Process: Hydrogen Recovery from Refinery Gases", *Journal of Natural Gas Chemistry*, Vol. 15, No. 3, pp. 230–234, 2006.

81. Wang, X. and H. Liu, "A new input variable selection method for soft sensor based on stacked auto-encoders", *Annual Conference on Decision and Control*, IEEE, 2017.

82. Shang, C., F. Yang, D. Huang *et al.*, "Data-driven soft sensor development based on deep learning technique", *Journal of Process Control*, Vol. 24, No. 3, pp. 223–233, mar 2014.

83. Cleveland, W. S., "Robust Locally Weighted Regression and Smoothing Scatterplots", *Journal of the American Statistical Association*, Vol. 74, No. 368, pp. 829–836, 1979.

84. Kim, S., M. Kano, H. Nakagawa *et al.*, "Estimation of active pharmaceutical ingredients content using locally weighted partial least squares and statistical wavelength selection", *International Journal of Pharmaceutics*, Vol. 421, No. 2, pp. 269–274, 2011.

85. Ge, Z. 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.

86. Fan, M., Z. Ge and Z. Song, "Adaptive Gaussian Mixture Model-Based Relevant Sample Selection for JITL Soft Sensor Development", *Industrial & Engineering Chemistry Research*, Vol. 53, No. 51, pp. 19979–19986, 2014.

87. Xie, L., J. Zeng and C. Gao, "Novel Just-In-Time Learning-Based Soft Sensor Utilizing Non-Gaussian Information", *IEEE Transactions on Control Systems Technology*, Vol. 22, No. 1, pp. 360–368, 2014.

88. Zhang, X., M. Kano and Y. Li, "Locally weighted kernel partial least squares regression based on sparse nonlinear features for virtual sensing of nonlinear time-varying processes", *Computers & Chemical Engineering*, Vol. 104, pp. 164–171, 2017.

89. Jin, H., X. Chen, J. Yang *et al.*, "Online local learning based adaptive soft sensor and its application to an industrial fed-batch chlortetracycline fermentation process", *Chemometrics and Intelligent Laboratory Systems*, Vol. 143, pp. 58–78, 2015.

90. Gupta, M. R., E. K. Garcia and E. Chin, "Adaptive Local Linear Regression With Application to Printer Color Management", *IEEE Transactions on Image Processing*, Vol. 17, No. 6, pp. 936–945, 2008.

91. Miller, A. J., "Selection of Subsets of Regression Variables", *Journal of the Royal*

Statistical Society. Series A, Vol. 147, No. 3, pp. 389–425, 1984.

92. Ambroise, C. and G. J. McLachlan, "Selection bias in gene extraction on the basis of microarray gene-expression data", *Proceedings of the National Academy of Sciences*, Vol. 99, No. 10, pp. 6562–6566, 2002.

93. Kohavi, R. and G. H. John, "Wrappers for feature subset selection", *Artificial Intelligence*, Vol. 97, No. 1-2, pp. 273–324, 1997.

94. Konishi, S. and G. Kitagawa, "Generalised Information Criteria in Model Selection", *Biometrika*, Vol. 83, No. 4, pp. 875–890, 1996.

95. Stone, M., "An Asymptotic Equivalence of Choice of Model by Cross-Validation and Akaike's Criterion", *Journal of the Royal Statistical Society. Series B*, Vol. 39, No. 1, pp. 44–47, 1977.

96. Souza, F., P. Santos and R. Araujo, "Variable and delay selection using neural networks and mutual information for data-driven soft sensors", *2010 IEEE 15th Conference on Emerging Technologies & Factory Automation (ETFA 2010)*, IEEE, 2010.

97. Souza, F. A., R. Araújo, T. Matias *et al.*, "A multilayer-perceptron based method for variable selection in soft sensor design", *Journal of Process Control*, Vol. 23, No. 10, pp. 1371–1378, 2013.

98. Martens, H. and M. Martens, "Modified Jack-knife estimation of parameter uncertainty in bilinear modelling by partial least squares regression (PLSR)", *Food Quality and Preference*, Vol. 11, No. 1-2, pp. 5–16, 2000.

99. Koller, D. and M. Sahami, "Toward Optimal Feature Selection", *Proceedings of the Thirteenth International Conference on International Conference on Machine Learning*, ICML'96, Morgan Kaufmann Publishers Inc., 1996.

100. Ding, C. and H. Peng, "Minimum redundancy feature selection from microarray gene expression data", *Computational Systems Bioinformatics. CSB2003. Proceedings of the 2003 IEEE Bioinformatics Conference. CSB2003*, IEEE Comput. Soc, 2003.

101. Wolpert, D. and W. Macready, "No free lunch theorems for optimization", *IEEE Transactions on Evolutionary Computation*, Vol. 1, No. 1, pp. 67–82, 1997.

102. Breiman, L., "Bagging Predictors", *Machine Learning*, Vol. 24, No. 2, pp. 123–140, 1996.

103. Chandra, A., H. Chen and X. Yao, "Trade-Off Between Diversity and Accuracy in Ensemble Generation", *Multi-Objective Machine Learning*, pp. 429–464, Springer Berlin Heidelberg, 2006.

104. Ali, K. M. and M. J. Pazzani, "Error reduction through learning multiple descriptions", *Machine Learning*, Vol. 24, No. 3, pp. 173–202, 1996.

105. Barrow, D. K. and S. F. Crone, "Crogging (cross-validation aggregation) for forecasting; A novel algorithm of neural network ensembles on time series subsamples", *The 2013 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–8, 2013.

106. Freund, Y. and R. E. Schapire, "A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting", *Journal of Computer and System Sciences*, Vol. 55, No. 1, pp. 119–139, 1997.

107. Wolpert, D. H., "Stacked generalization", *Neural Networks*, Vol. 5, No. 2, pp. 241–259, 1992.

108. Fragoso, T. M., W. Bertoli and F. Louzada, "Bayesian Model Averaging: A Systematic Review and Conceptual Classification", *International Statistical Review*, Vol. 86, No. 1, pp. 1–28, 2017.

109. Kaneko, H. and K. Funatsu, "Adaptive soft sensor based on online support vector regression and Bayesian ensemble learning for various states in chemical plants", *Chemometrics and Intelligent Laboratory Systems*, Vol. 137, pp. 57–66, 2014.

110. Stone, M., "Cross-validatory choice and assessment of statistical predictions", *Journal of the royal statistical society. Series B*, Vol. 36, No. 2, pp. 111–147, 1974.

111. Kohavi, R., "A Study of Cross-validation and Bootstrap for Accuracy Estimation and Model Selection", *Proceedings of the 14th International Joint Conference on Artificial Intelligence - Volume 2*, pp. 1137–1143, Morgan Kaufmann Publishers Inc., 1995.

112. Breiman, L., J. Friedman, C. J. Stone *et al.*, *Classification and Regression Trees*, Taylor & Francis Ltd, 1984.

113. Tashman, L. J., "Out-of-sample tests of forecasting accuracy: an analysis and review", *International Journal of Forecasting*, Vol. 16, No. 4, pp. 437–450, 2000.

114. Bergmeir, C. and J. M. Benítez, "On the use of cross-validation for time series predictor evaluation", *Information Sciences*, Vol. 191, pp. 192–213, 2012.

115. Gama, J., R. Sebastião and P. P. Rodrigues, "Issues in evaluation of stream learning algorithms", *Proceedings of the 15th international conference on Knowledge discovery and data mining*, pp. 329–338, ACM Press, 2009.

116. Gama, J., I. Žliobaitė, A. Bifet *et al.*, "A survey on concept drift adaptation", *ACM Computing Surveys*, Vol. 46, No. 4, pp. 1–37, 2014.

117. Delany, S. J., P. Cunningham, A. Tsymbal *et al.*, "A case-based technique for tracking concept drift in spam filtering", *Knowledge-Based Systems*, Vol. 18, No. 4-5, pp. 187–195, 2005.

118. Gama, J., "A survey on learning from data streams: current and future trends", *Progress in Artificial Intelligence*, Vol. 1, No. 1, pp. 45–55, 2012.

119. Kaneko, H. and K. Funatsu, "Ensemble locally weighted partial least squares as a just-in-time modeling method", *AIChE Journal*, Vol. 62, No. 3, pp. 717–725, 2015.

120. Ni, W., S. K. Tan, W. J. Ng *et al.*, "Localized, adaptive recursive partial least squares regression for dynamic system modeling", *Industrial & Engineering Chemistry Research*, Vol. 51, No. 23, pp. 8025–8039, may 2012.

121. Grbić, R., D. Slišković and P. Kadlec, "Adaptive soft sensor for online prediction and process monitoring based on a mixture of Gaussian process models", *Computers & Chemical Engineering*, Vol. 58, pp. 84–97, 2013.

122. Polikar, R., J. Byorick, S. Krause *et al.*, "Learn++: a classifier independent incremental learning algorithm for supervised neural networks", *Proceedings of the 2002 International Joint Conference on Neural Networks*, pp. 1742–1747, 2002.

123. Ma, J., J. Theiler and S. Perkins, "Accurate On-line Support Vector Regression", *Neural Computation*, Vol. 15, No. 11, pp. 2683–2703, 2003.

124. Huang, G.-B., Q.-Y. Zhu and C.-K. Siew, "Extreme learning machine: Theory and applications", *Neurocomputing*, Vol. 70, No. 1-3, pp. 489–501, 2006.

125. Kaneko, H. and K. Funatsu, "Adaptive soft sensor model using online support vector regression with time variable and discussion of appropriate hyperparameter settings and window size", *Computers & Chemical Engineering*, Vol. 58, pp. 288–297, 2013.

126. Jin, H., X. Chen, L. Wang *et al.*, "Multi-model adaptive soft sensor modeling method using local learning and online support vector regression for nonlinear time-variant batch processes", *Chemical Engineering Science*, Vol. 131, pp. 282–

303, 2015.

127. Gary, J. H., G. E. Handwerk and M. J. Kaiser, *Petroleum Refining*, Taylor & Francis, 2007.

128. AVEVA Group *SimSci PRO/II*, version 10.1, 2017; Process Engineering Software: Cambridge, UK.

129. Yoon, S. and J. F. MacGregor, "Fault diagnosis with multivariate statistical models part I: using steady state fault signatures", *Journal of Process Control*, 2001.

130. Horn, D., U. Naftaly and N. Intrator, "Large Ensemble Averaging", *Lecture Notes in Computer Science*, pp. 131–137, 2012.

131. Peter, F., L. Bettina and V. Kurt, "Repeated double cross validation", *Journal of Chemometrics*, Vol. 23, No. 4, pp. 160–171, 2009.

132. Yu, W. M., T. Du and K. B. Lim, "Comparison of the support vector machine and relevant vector machine in regression and classification problems", *Control, Automation, Robotics and Vision Conference*, Vol. 2, pp. 1309–1314, IEEE, 2004.

133. Faber, N. K. M. and J. Ferré, "On the numerical stability of two widely used PLS algorithms", *Journal of Chemometrics*, Vol. 22, No. 2, pp. 101–105, 2008.

134. Qin, S. J., "Statistical process monitoring: basics and beyond", *Journal of Chemometrics*, Vol. 17, No. 8-9, pp. 480–502, 2003.

135. Montgomery, D. C., *Introduction to statistical quality control*, John Wiley & Sons, 2009.

136. Domangue, R. and S. C. Patch, "Some Omnibus Exponentially Weighted Moving Average Statistical Process Monitoring Schemes", *Technometrics*, Vol. 33, No. 3, pp. 299–313, 1991.

137. Ross, G. J., N. M. Adams, D. K. Tasoulis *et al.*, "Exponentially weighted moving average charts for detecting concept drift", *Pattern Recognition Letters*, Vol. 33, No. 2, pp. 191–198, 2012.

138. Altman, D. G. and J. M. Bland, "Measurement in Medicine: The Analysis of Method Comparison Studies", *The Statistician*, Vol. 32, No. 3, p. 307, 1983.

139. Bifet, A. and R. Gavaldà, "Learning from Time-Changing Data with Adaptive Windowing", *Proceedings of the 2007 SIAM International Conference on Data Mining*, pp. 443–448, Society for Industrial and Applied Mathematics, 2007.

140. Shao, W. and X. Tian, "Adaptive soft sensor for quality prediction of chemical processes based on selective ensemble of local partial least squares models", *Chemical Engineering Research and Design*, Vol. 95, pp. 113–132, 2015.

141. Hoens, T. R., R. Polikar and N. V. Chawla, "Learning from streaming data with concept drift and imbalance: an overview", *Progress in Artificial Intelligence*, Vol. 1, No. 1, pp. 89–101, 2012.

APPENDIX A: COMPARISON OF ONLINE PLS AND RVM MODELS

Table A.1. MW performance of PLS and RVM learners ($n = 0$).

	PLS_{MW} (TS)		**PLS_{MW} (W)**		**RVM_{MW}**	
W	**RMSE**	**RMSE (99^{th})**	**RMSE**	**RMSE (99^{th})**	**RMSE**	**RMSE (99^{th})**
10	2.781	0.016	2.611	0.017	0.789	0.024
20	0.441	0.029	0.429	0.031	0.463	0.043
30	0.873	0.047	0.792	0.047	0.801	0.065
40	0.357	0.061	0.348	0.062	0.345	0.084
50	0.306	0.076	0.232	0.078	0.196	0.099
60	1.022	0.083	1.006	0.087	0.806	0.105
70	0.880	0.087	0.803	0.091	0.862	0.109
80	0.750	0.091	0.792	0.095	0.793	0.113
90	0.561	0.097	0.562	0.100	0.554	0.119
100	0.639	0.102	0.640	0.104	0.627	0.123
200	0.147	0.117	0.148	0.115	0.147	0.135
210	0.149	0.119	0.151	0.118	0.149	0.137
220	0.150	0.119	0.151	0.119	0.149	0.136
230	0.152	0.121	0.152	0.120	0.152	0.139
240	0.154	0.123	0.155	0.122	0.154	0.141
250	0.157	0.125	0.158	0.125	0.157	0.144
260	0.159	0.127	0.160	0.126	0.158	0.145
270	0.161	0.127	0.162	0.126	0.160	0.147
280	0.161	0.127	0.162	0.126	0.160	0.147
290	0.159	0.126	0.161	0.126	0.159	0.146
300	0.159	0.125	0.160	0.124	0.159	0.146

Table A.2. MW performance of PLS and RVM learners ($n = 2$).

	$\mathbf{PLS_{MW}}$ **(TS)**		$\mathbf{PLS_{MW}}$ **(W)**		$\mathbf{RVM_{MW}}$	
W	**RMSE**	**RMSE ($\mathbf{99^{th}}$)**	**RMSE**	**RMSE ($\mathbf{99^{th}}$)**	**RMSE**	**RMSE ($\mathbf{99^{th}}$)**
10	0.809	0.018	0.611	0.019	1.045	0.023
20	0.437	0.022	0.711	0.025	1.135	0.064
30	1.108	0.035	1.264	0.037	1.408	0.057
40	0.613	0.048	66.538	0.048	0.908	0.065
50	0.212	0.058	0.202	0.059	0.248	0.074
60	1.249	0.065	1.843	0.066	1.192	0.087
70	0.995	0.072	2.171	0.072	1.500	0.092
80	0.994	0.078	3.326	0.079	1.099	0.094
90	0.648	0.083	3.594	0.084	0.819	0.102
100	0.770	0.088	2.207	0.087	1.032	0.104
200	0.134	0.107	0.135	0.108	0.131	0.120
210	0.131	0.107	0.132	0.108	0.132	0.121
220	0.130	0.107	0.132	0.108	0.131	0.120
230	0.133	0.109	0.136	0.112	0.134	0.123
240	0.136	0.111	0.140	0.115	0.137	0.126
250	0.140	0.114	0.143	0.117	0.140	0.129
260	0.141	0.115	0.145	0.118	0.141	0.130
270	0.142	0.114	0.145	0.116	0.142	0.131
280	0.142	0.114	0.145	0.116	0.141	0.130
290	0.141	0.114	0.143	0.115	0.141	0.130
300	0.142	0.114	0.144	0.114	0.141	0.130

Table A.3. MW performance of PLS and RVM learners ($n = 4$).

	$\mathbf{PLS_{MW}}$ **(TS)**		$\mathbf{PLS_{MW}}$ **(W)**		$\mathbf{RVM_{MW}}$	
W	**RMSE**	**RMSE ($\mathbf{99^{th}}$)**	**RMSE**	**RMSE ($\mathbf{99^{th}}$)**	**RMSE**	**RMSE ($\mathbf{99^{th}}$)**
10	0.322	0.017	0.294	0.018	0.701	0.023
20	0.308	0.022	0.350	0.021	1.542	0.062
30	0.658	0.028	1.238	0.027	1.755	0.045
40	0.607	0.036	0.859	0.034	1.236	0.072
50	0.312	0.044	1.660	0.039	0.193	0.045
60	0.349	0.050	0.477	0.045	0.137	0.056
70	0.482	0.057	0.585	0.051	0.648	0.068
80	0.515	0.062	0.678	0.058	0.645	0.071
90	0.491	0.065	0.592	0.063	0.918	0.076
100	0.572	0.068	0.657	0.066	1.208	0.076
200	0.109	0.090	0.108	0.088	0.103	0.096
210	0.108	0.090	0.104	0.086	0.102	0.096
220	0.106	0.088	0.104	0.085	0.101	0.095
230	0.108	0.089	0.109	0.090	0.106	0.099
240	0.111	0.091	0.112	0.092	0.111	0.103
250	0.113	0.092	0.114	0.095	0.113	0.105
260	0.115	0.095	0.117	0.097	0.114	0.107
270	0.116	0.096	0.116	0.095	0.115	0.107
280	0.116	0.095	0.116	0.095	0.115	0.108
290	0.117	0.095	0.117	0.096	0.114	0.108
300	0.118	0.096	0.117	0.095	0.115	0.108

Table A.4. MW performance of PLS and RVM learners ($n = 6$).

	$\mathbf{PLS_{MW}}$ **(TS)**		$\mathbf{PLS_{MW}}$ **(W)**		$\mathbf{RVM_{MW}}$	
W	**RMSE**	**RMSE ($\mathbf{99^{th}}$)**	**RMSE**	**RMSE ($\mathbf{99^{th}}$)**	**RMSE**	**RMSE ($\mathbf{99^{th}}$)**
10	0.263	0.015	0.238	0.017	0.563	0.020
20	0.242	0.020	0.233	0.019	1.326	0.059
30	0.310	0.025	0.457	0.020	1.412	0.050
40	0.379	0.030	0.752	0.023	2.143	0.114
50	0.210	0.035	0.322	0.026	2.264	0.030
60	0.097	0.042	0.156	0.029	0.292	0.035
70	0.187	0.045	0.239	0.033	0.422	0.047
80	0.207	0.049	0.224	0.038	0.603	0.049
90	0.264	0.051	0.302	0.042	0.989	0.053
100	0.397	0.055	0.468	0.045	1.974	0.054
200	0.088	0.074	0.095	0.067	0.096	0.073
210	0.087	0.074	0.081	0.065	0.078	0.072
220	0.088	0.073	0.087	0.065	0.077	0.072
230	0.088	0.074	0.085	0.068	0.081	0.075
240	0.089	0.074	0.086	0.071	0.084	0.078
250	0.090	0.074	0.089	0.073	0.086	0.080
260	0.093	0.077	0.090	0.075	0.088	0.082
270	0.096	0.081	0.091	0.075	0.090	0.084
280	0.098	0.082	0.091	0.076	0.090	0.085
290	0.097	0.081	0.092	0.076	0.091	0.086
300	0.096	0.081	0.093	0.076	0.093	0.087

Table A.5. MW performance of PLS and RVM learners ($n = 8$).

	$\mathbf{PLS_{MW}}$ **(TS)**		$\mathbf{PLS_{MW}}$ **(W)**		$\mathbf{RVM_{MW}}$	
W	**RMSE**	**RMSE ($\mathbf{99^{th}}$)**	**RMSE**	**RMSE ($\mathbf{99^{th}}$)**	**RMSE**	**RMSE ($\mathbf{99^{th}}$)**
10	0.232	0.018	0.219	0.018	0.719	0.024
20	0.212	0.016	0.211	0.017	1.384	0.043
30	0.198	0.021	0.253	0.018	0.863	0.034
40	0.294	0.023	0.538	0.018	0.937	0.040
50	0.179	0.025	0.426	0.020	1.527	0.023
60	0.064	0.032	0.069	0.021	0.224	0.025
70	0.092	0.035	0.094	0.022	0.803	0.034
80	0.109	0.038	0.105	0.024	0.799	0.031
90	0.188	0.035	0.482	0.028	1.465	0.037
100	0.302	0.038	0.805	0.030	2.421	0.103
200	0.069	0.052	0.079	0.049	0.097	0.055
210	0.065	0.053	0.065	0.049	0.062	0.054
220	0.071	0.058	0.062	0.049	0.058	0.055
230	0.067	0.054	0.065	0.052	0.059	0.056
240	0.068	0.056	0.068	0.054	0.061	0.057
250	0.069	0.058	0.067	0.056	0.064	0.060
260	0.073	0.061	0.069	0.058	0.066	0.063
270	0.078	0.066	0.070	0.059	0.068	0.064
280	0.074	0.063	0.071	0.060	0.069	0.065
290	0.076	0.064	0.072	0.060	0.070	0.066
300	0.078	0.065	0.073	0.060	0.072	0.068

Table A.6. MW performance of PLS and RVM learners ($n = 12$).

	$\mathbf{PLS_{MW}}$ **(TS)**		$\mathbf{PLS_{MW}}$ **(W)**		$\mathbf{RVM_{MW}}$	
W	**RMSE**	**RMSE ($\mathbf{99^{th}}$)**	**RMSE**	**RMSE ($\mathbf{99^{th}}$)**	**RMSE**	**RMSE ($\mathbf{99^{th}}$)**
10	0.192	0.018	0.188	0.019	0.458	0.020
20	0.154	0.019	0.149	0.019	1.748	0.055
30	0.130	0.017	0.132	0.017	0.633	0.036
40	0.104	0.020	0.124	0.017	0.681	0.087
50	0.099	0.020	0.132	0.015	0.141	0.021
60	0.066	0.021	0.105	0.014	0.086	0.017
70	0.042	0.022	0.131	0.014	1.165	0.055
80	0.069	0.024	0.143	0.015	0.862	0.025
90	0.152	0.024	0.323	0.015	1.314	0.022
100	0.197	0.025	0.322	0.015	2.137	0.041
200	0.052	0.036	0.060	0.031	0.676	0.037
210	0.051	0.038	0.052	0.032	0.640	0.039
220	0.051	0.040	0.049	0.034	0.601	0.041
230	0.052	0.041	0.045	0.034	0.500	0.040
240	0.053	0.042	0.045	0.035	0.115	0.042
250	0.056	0.044	0.048	0.038	0.078	0.044
260	0.058	0.047	0.049	0.039	0.068	0.047
270	0.058	0.046	0.049	0.040	0.050	0.046
280	0.058	0.046	0.049	0.039	0.049	0.045
290	0.060	0.047	0.050	0.040	0.052	0.047
300	0.062	0.048	0.052	0.041	0.068	0.049

APPENDIX B: EVALUATION OF ADAPTIVE METHOD PARAMETERS

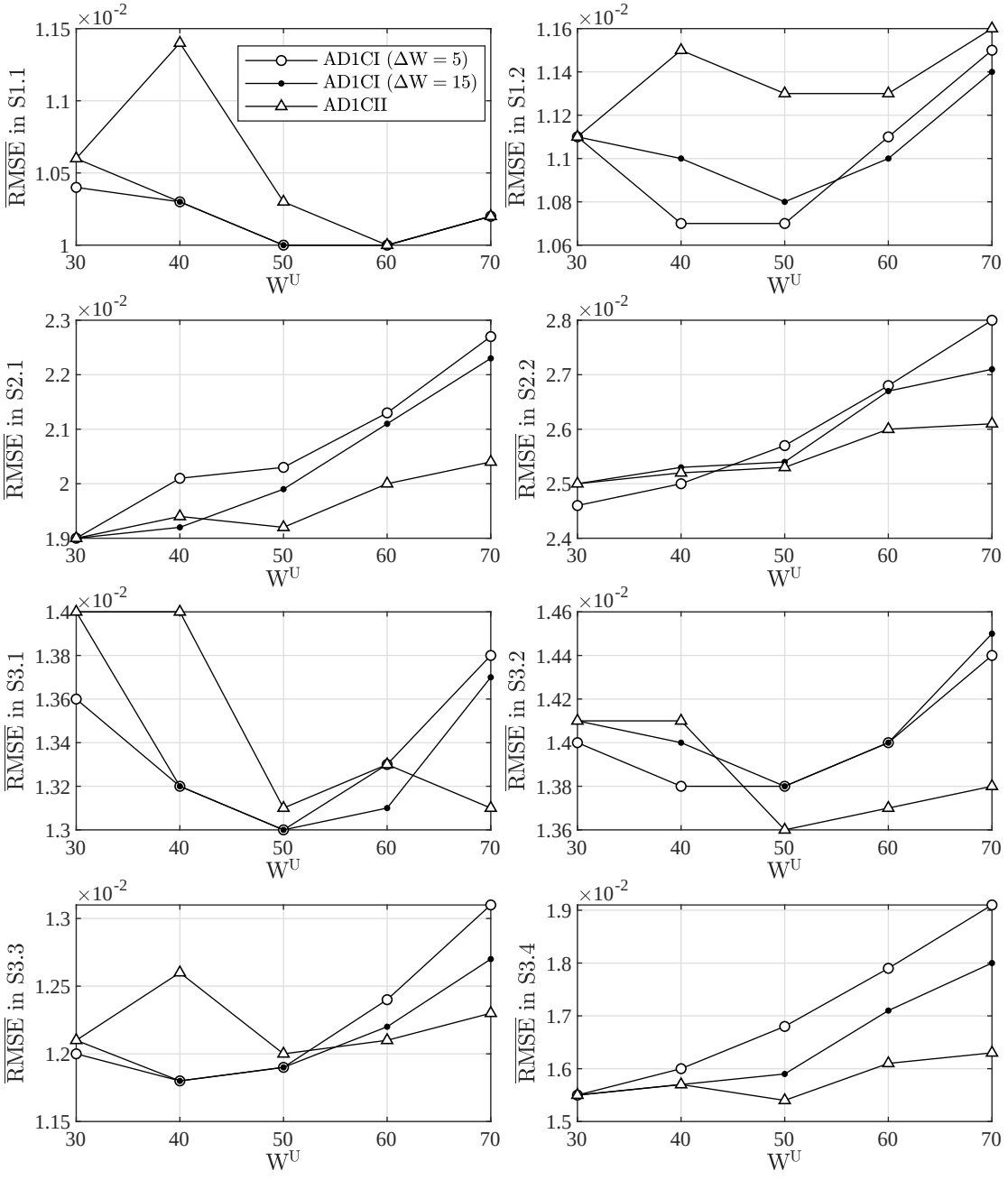

Figure B.1. $\overline{\text{RMSE}}$ of AD2CI ($\Delta W = 5$ and $\Delta W = 15$), and AD2CII ($\Delta W_1 = 0.2$ and $\Delta W_2 = 0.5$), for $\alpha = 0.001$ and $W^U \in \{30, 40, ...70\}$.

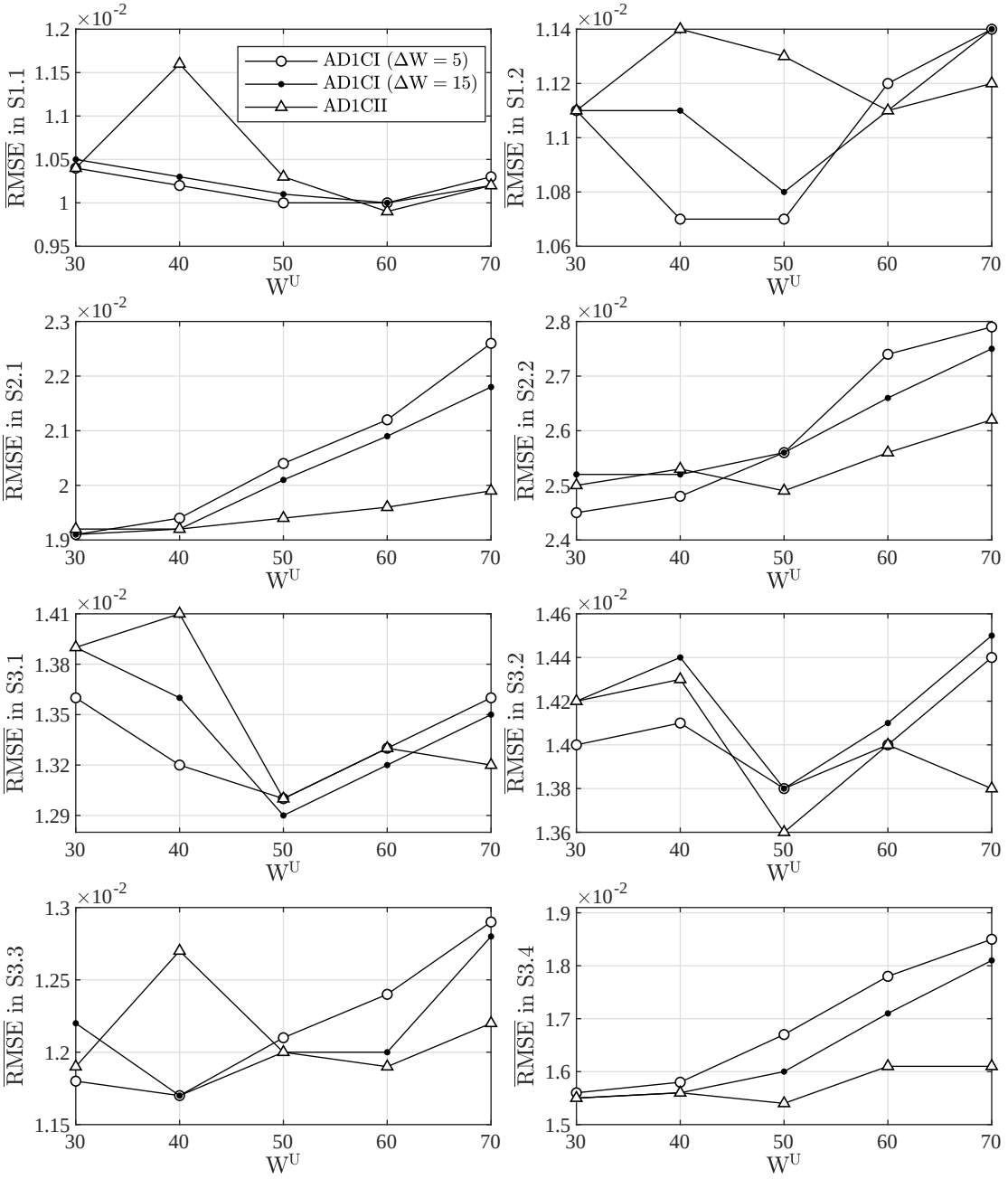

Figure B.2. $\overline{\text{RMSE}}$ of AD2CI ($\Delta W = 5$ and $\Delta W = 15$), and AD2CII ($\Delta W_1 = 0.2$ and $\Delta W_2 = 0.5$), for $\alpha = 0.005$ and $W^U \in \{30, 40, ...70\}$.

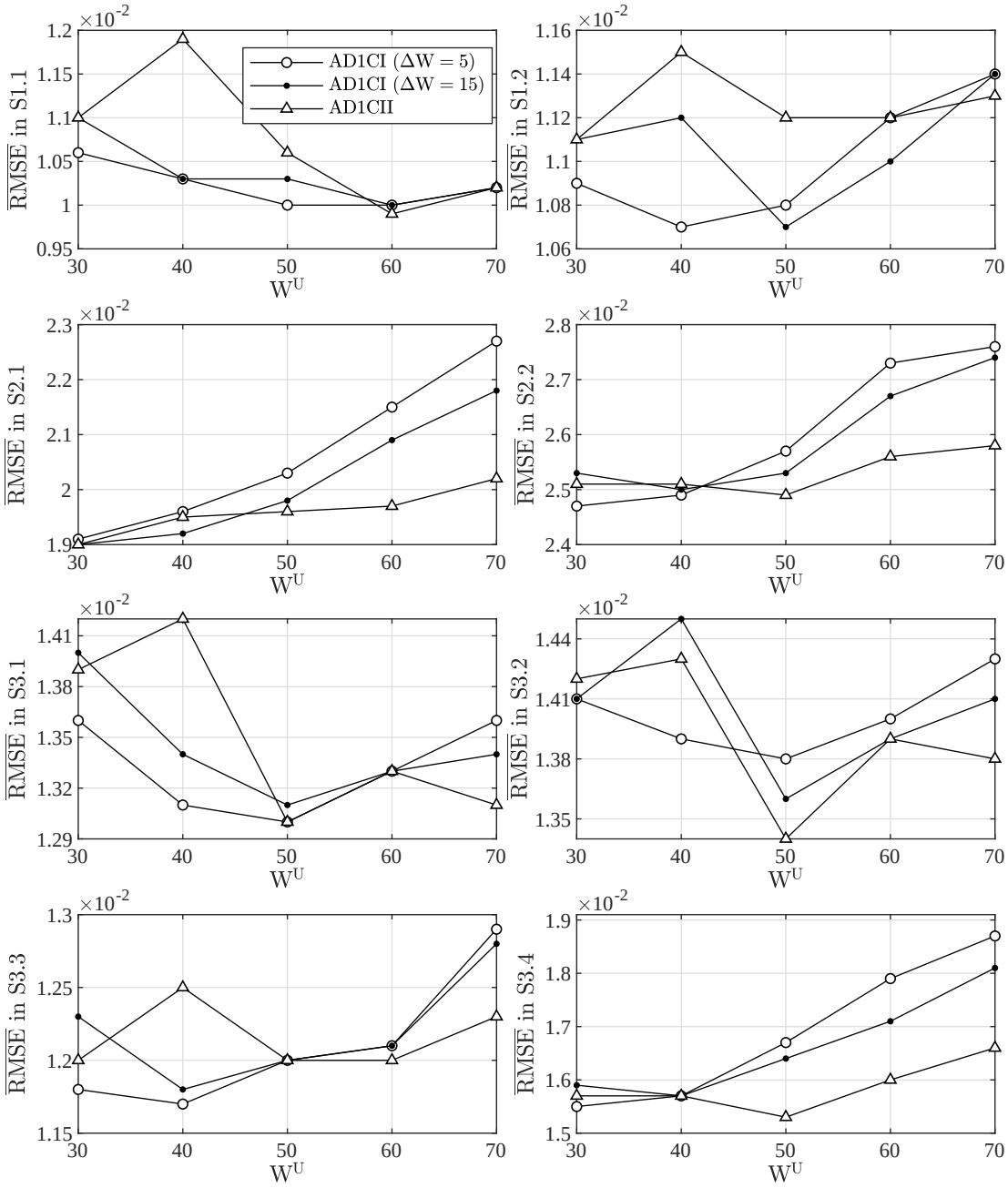

Figure B.3. $\overline{\text{RMSE}}$ of AD2CI ($\Delta W = 5$ and $\Delta W = 15$), and AD2CII ($\Delta W_1 = 0.2$ and $\Delta W_2 = 0.5$), for $\alpha = 0.01$ and $W^U \in \{30, 40, ...70\}$.

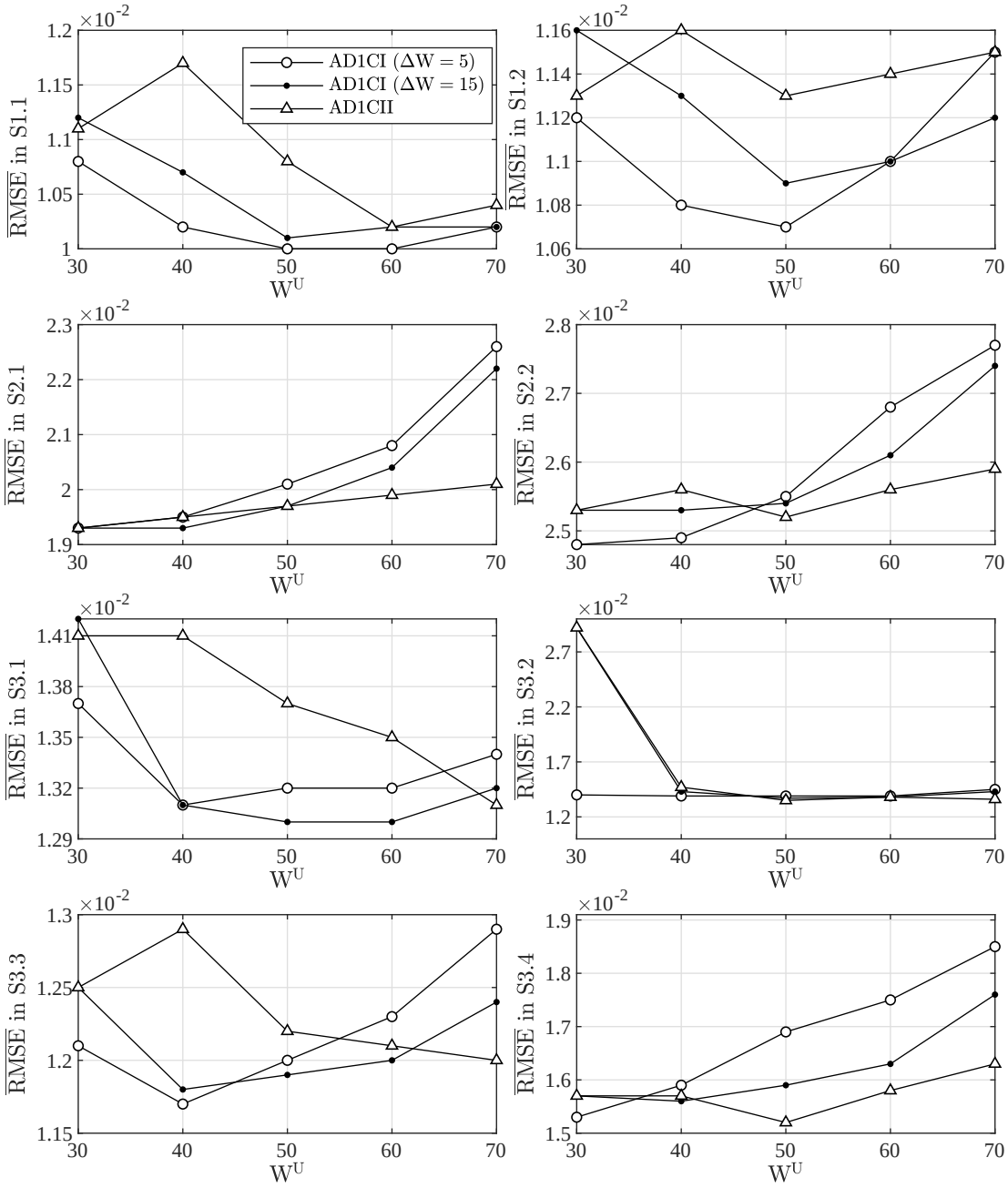

Figure B.4. $\overline{\text{RMSE}}$ of AD2CI ($\Delta W = 5$ and $\Delta W = 15$), and AD2CII ($\Delta W_1 = 0.2$ and $\Delta W_2 = 0.5$), for $\alpha = 0.05$ and $W^U \in \{30, 40, ...70\}$.

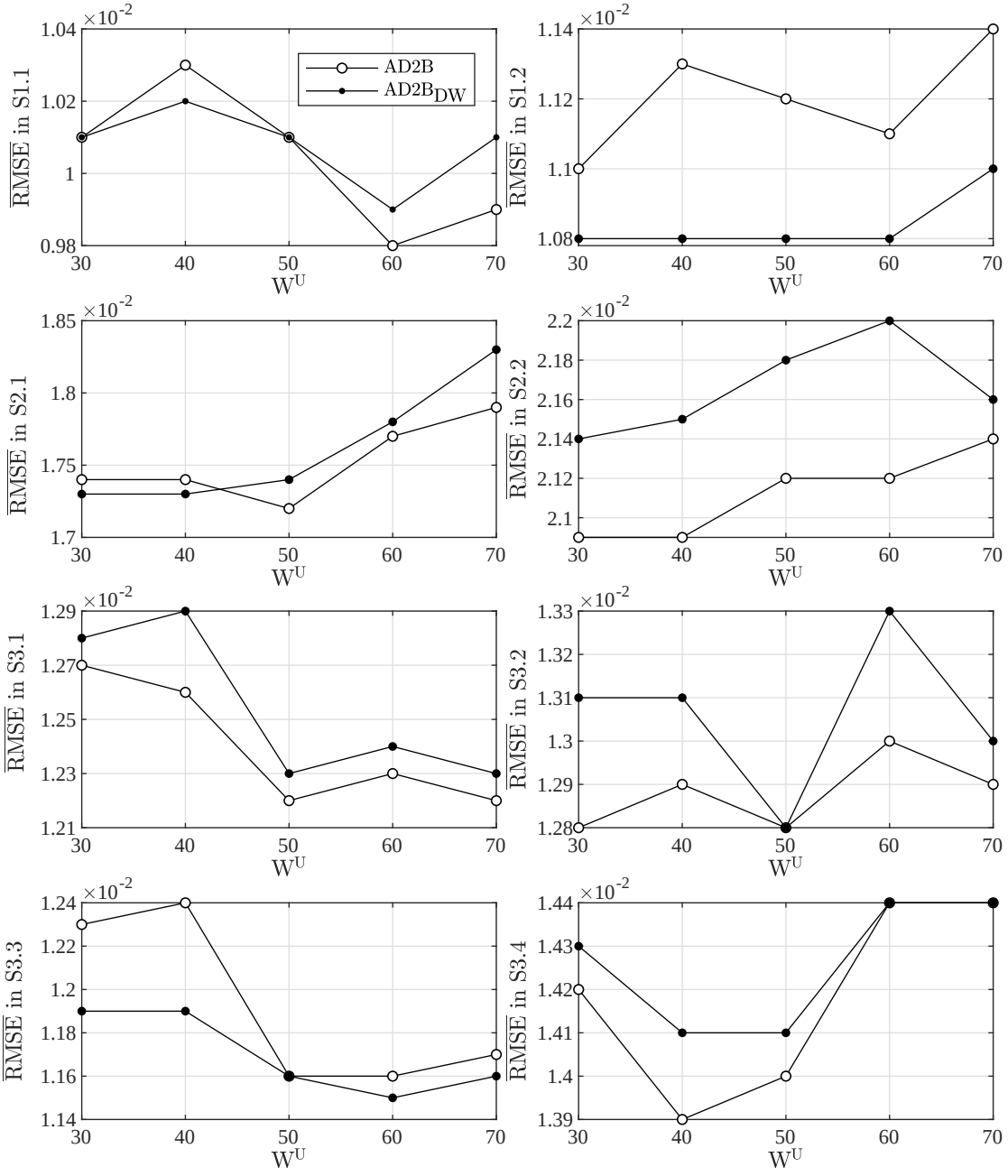

Figure B.5. Comparison of AD2B models with and without DW on CSTR simulations, using $\alpha = 0.001$, $mp = 3$, and $\text{W}^{\text{U}} \in \{30, 40, ...70\}$.

Table B.1. RMSE and MAE of AD2 using the average and EWMA of past observations to determine ensemble weights for $\alpha = 0.001$, $mp \in \{3, 5\}$, and $W^U \in \{10, 20, ...60\}$ on debutanizer data.

	$mp = 3$ Average		$mp = 3$ EWMA		$mp = 5$ Average		$mp = 5$ EWMA	
W	**RMSE**	**MAE**	**RMSE**	**MAE**	**RMSE**	**MAE**	**RMSE**	**MAE**
10	0.0248	0.0134	0.0235	0.0128	0.0705	0.0165	0.0245	0.0126
20	0.0240	0.0131	0.0232	0.0126	0.0709	0.0167	0.0245	0.0127
30	0.0234	0.0129	0.0227	0.0125	0.0711	0.0171	0.0235	0.0129
40	0.0238	0.0132	0.0235	0.0129	0.0834	0.0217	0.0247	0.0131
50	0.0258	0.0137	0.0251	0.0133	0.0632	0.0306	0.0266	0.0130
60	0.0281	0.0143	0.0273	0.0138	0.0632	0.0308	0.0279	0.0133

APPENDIX C: PERFORMANCE OF ADAPTIVE LEARNING METHODS

Table C.1. Comparison of proposed adaptive methods with other conventional adaptive methods from the literature on CDM 1.1.

		$\overline{\textbf{RMSE}}$	**sdRMSE**	$\overline{\textbf{maxAE}}$
RPLS	$\lambda = 1$	0.0230	0.0137	0.0685
	$\lambda = 0.7$	0.0225	0.0134	0.0674
	$\lambda = 0.3$	0.0201	0.0117	0.0633
$\textbf{PLS}_{\textbf{MW}}$ (W)	W = 30	0.0138	0.0074	0.1226
	W = 50	0.0114	0.0020	0.0649
$\textbf{PLS}_{\textbf{MW}}$ (TS)	W = 30	0.0164	0.0163	0.1605
	W = 50	0.0118	0.0023	0.0625
$\textbf{RVM}_{\textbf{MW}}$	W = 30	0.0107	0.0015	0.0614
	W = 50	0.0107	0.0018	0.0637
CoJITL (PLS)	W = 30	0.0239	0.0115	0.0923
	W = 50	0.0237	0.0135	0.0836
CoJITL (PCR)	W = 30	0.0237	0.0116	0.0933
	W = 50	0.0237	0.0135	0.0808
AD1A	W = 30	0.0104	0.0012	0.0578
	W = 50	0.0101	0.0016	0.0512
AD1B	$W^U = 30$	0.0113	0.0014	0.0700
	$W^U = 50$	0.0109	0.0013	0.0588
AD1C	$W^U = 30$	0.0108	0.0014	0.0608
	$W^U = 50$	0.0105	0.0016	0.0564
AD2A	$W^U = 30$	0.0100	0.0011	0.0584
	$W^U = 50$	0.0102	0.0014	0.0656

Table C.2. Comparison of proposed adaptive methods with other conventional adaptive methods from the literature on CDM 1.2.

		$\overline{\textbf{RMSE}}$	**sdRMSE**	$\overline{\textbf{maxAE}}$
RPLS	$\lambda = 1$	0.0305	0.0147	0.0806
	$\lambda = 0.7$	0.0299	0.0144	0.0797
	$\lambda = 0.3$	0.0264	0.0124	0.0748
PLS_{MW} (W)	W = 30	0.0130	0.0032	0.0868
	W = 50	0.0122	0.0019	0.0658
PLS_{MW} (TS)	W = 30	0.0141	0.0039	0.1013
	W = 50	0.0129	0.0026	0.0756
RVM_{MW}	W = 30	0.0152	0.0160	0.1536
	W = 50	0.0112	0.0018	0.0558
CoJITL (PLS)	W = 30	0.0314	0.0146	0.1082
	W = 50	0.0325	0.0146	0.1072
CoJITL (PCR)	W = 30	0.0313	0.0145	0.1082
	W = 50	0.0325	0.0146	0.1074
AD1A	W = 30	0.0109	0.0014	0.0576
	W = 50	0.0108	0.0017	0.0537
AD1B	$\text{W}^{\text{U}} = 30$	0.0116	0.0017	0.0647
	$\text{W}^{\text{U}} = 50$	0.0115	0.0016	0.0588
AD1C	$\text{W}^{\text{U}} = 30$	0.0112	0.0018	0.0616
	$\text{W}^{\text{U}} = 50$	0.0113	0.0019	0.0594
AD2A	$\text{W}^{\text{U}} = 30$	0.0109	0.0018	0.0650
	$\text{W}^{\text{U}} = 50$	0.0111	0.0019	0.0688

Table C.3. Comparison of proposed adaptive methods with other conventional adaptive methods from the literature on CDM 2.1.

		$\overline{\textbf{RMSE}}$	**sdRMSE**	$\overline{\textbf{maxAE}}$
RPLS	$\lambda = 1$	0.0477	0.0351	0.128
	$\lambda = 0.7$	0.0468	0.0344	0.1257
	$\lambda = 0.3$	0.0416	0.0305	0.116
PLS_{MW} (W)	W = 30	0.0212	0.0110	0.1407
	W = 50	0.0248	0.0153	0.1517
PLS_{MW} (TS)	W = 30	0.0219	0.0114	0.1567
	W = 50	0.024	0.0145	0.1313
RVM_{MW}	W = 30	0.0200	0.0114	0.1291
	W = 50	0.0227	0.0145	0.1303
CoJITL (PLS)	W = 30	0.0575	0.0428	0.1655
	W = 50	0.0605	0.0465	0.1837
CoJITL (PCR)	W = 30	0.0573	0.0427	0.1675
	W = 50	0.0605	0.0465	0.1831
AD1A	W = 30	0.0193	0.0107	0.1245
	W = 50	0.0202	0.0120	0.1242
AD1B	$\text{W}^{\text{U}} = 30$	0.0191	0.0097	0.1268
	$\text{W}^{\text{U}} = 50$	0.0195	0.0105	0.125
AD1C	$\text{W}^{\text{U}} = 30$	0.0191	0.0098	0.1253
	$\text{W}^{\text{U}} = 50$	0.0195	0.0105	0.1228
AD2A	$\text{W}^{\text{U}} = 30$	0.0158	0.0077	0.1285
	$\text{W}^{\text{U}} = 50$	0.0157	0.0083	0.1142

Table C.4. Comparison of proposed adaptive methods with other conventional adaptive methods from the literature on CDM 2.2.

		$\overline{\textbf{RMSE}}$	**sdRMSE**	$\overline{\textbf{maxAE}}$
RPLS	$\lambda = 1$	0.0421	0.0309	0.1176
	$\lambda = 0.7$	0.0412	0.0303	0.1153
	$\lambda = 0.3$	0.0366	0.0268	0.1055
PLS_{MW} (W)	W = 30	0.0280	0.0194	0.2046
	W = 50	0.0278	0.0192	0.1347
PLS_{MW} (TS)	W = 30	0.0313	0.0229	0.2835
	W = 50	0.0278	0.0186	0.1333
RVM_{MW}	W = 30	0.0255	0.0170	0.1441
	W = 50	0.0265	0.0178	0.1261
CoJITL (PLS)	W = 30	0.0515	0.0387	0.1702
	W = 50	0.0519	0.0393	0.1736
CoJITL (PCR)	W = 30	0.0509	0.0382	0.1662
	W = 50	0.0519	0.0392	0.1704
AD1A	W = 30	0.0245	0.0161	0.1222
	W = 50	0.0259	0.0173	0.1290
AD1B	$\text{W}^{\text{U}} = 30$	0.0247	0.0161	0.1443
	$\text{W}^{\text{U}} = 50$	0.0251	0.0161	0.1466
AD1C	$\text{W}^{\text{U}} = 30$	0.0251	0.0161	0.1449
	$\text{W}^{\text{U}} = 50$	0.0251	0.0163	0.1385
AD2A	$\text{W}^{\text{U}} = 30$	0.0189	0.0110	0.1113
	$\text{W}^{\text{U}} = 50$	0.0191	0.0109	0.1171

Table C.5. Comparison of proposed adaptive methods with other conventional adaptive methods from the literature on CDM 3.1.

		$\overline{\textbf{RMSE}}$	**sdRMSE**	$\overline{\textbf{maxAE}}$
RPLS	$\lambda = 1$	0.0280	0.0165	0.0806
	$\lambda = 0.7$	0.0275	0.0163	0.0793
	$\lambda = 0.3$	0.0246	0.0144	0.0722
PLS_{MW} (W)	W = 30	0.0178	0.0078	0.0734
	W = 50	0.0175	0.0050	0.0848
PLS_{MW} (TS)	W = 30	0.0194	0.0072	0.0801
	W = 50	0.0188	0.0048	0.1217
RVM_{MW}	W = 30	0.0215	0.0327	0.2404
	W = 50	0.0153	0.0059	0.0683
CoJITL (PLS)	W = 30	0.0359	0.0198	0.1088
	W = 50	0.0375	0.0225	0.1141
CoJITL (PCR)	W = 30	0.0358	0.0196	0.1076
	W = 50	0.0375	0.0225	0.1125
AD1A	W = 30	0.0135	0.0032	0.0731
	W = 50	0.0131	0.0034	0.0695
AD1B	$\text{W}^{\text{U}} = 30$	0.0143	0.0029	0.0779
	$\text{W}^{\text{U}} = 50$	0.0140	0.0033	0.0810
AD1C	$\text{W}^{\text{U}} = 30$	0.0140	0.0030	0.0790
	$\text{W}^{\text{U}} = 50$	0.0132	0.0032	0.0722
AD2A	$\text{W}^{\text{U}} = 30$	0.0120	0.0021	0.0710
	$\text{W}^{\text{U}} = 50$	0.0116	0.0018	0.0669

Table C.6. Comparison of proposed adaptive methods with other conventional adaptive methods from the literature on CDM 3.2.

		$\overline{\textbf{RMSE}}$	**sdRMSE**	$\overline{\textbf{maxAE}}$
RPLS	$\lambda = 1$	0.038	0.0243	0.1086
	$\lambda = 0.7$	0.0374	0.0239	0.1063
	$\lambda = 0.3$	0.0334	0.0211	0.0945
PLS_{MW} (W)	W = 30	0.0187	0.0079	0.0952
	W = 50	0.0179	0.0050	0.0942
PLS_{MW} (TS)	W = 30	0.0197	0.0069	0.0943
	W = 50	0.0212	0.0093	0.1209
RVM_{MW}	W = 30	0.0155	0.0046	0.1187
	W = 50	0.0167	0.0065	0.1070
CoJITL (PLS)	W = 30	0.045	0.0266	0.1359
	W = 50	0.0463	0.0290	0.1327
CoJITL (PCR)	W = 30	0.0448	0.0265	0.1351
	W = 50	0.0463	0.0288	0.1318
AD1A	W = 30	0.0139	0.0038	0.092
	W = 50	0.0138	0.0046	0.0826
AD1B	$\text{W}^{\text{U}} = 30$	0.0144	0.0036	0.0887
	$\text{W}^{\text{U}} = 50$	0.0140	0.0036	0.0823
AD1C	$\text{W}^{\text{U}} = 30$	0.0179	0.0333	0.1663
	$\text{W}^{\text{U}} = 50$	0.0135	0.0039	0.0782
AD2A	$\text{W}^{\text{U}} = 30$	0.0129	0.0027	0.089
	$\text{W}^{\text{U}} = 50$	0.0126	0.0026	0.0873

Table C.7. Comparison of proposed adaptive methods with other conventional adaptive methods from the literature on CDM 3.3.

		$\overline{\textbf{RMSE}}$	**sdRMSE**	$\overline{\textbf{maxAE}}$
RPLS	$\lambda = 1$	0.0322	0.0188	0.0872
	$\lambda = 0.7$	0.0317	0.0185	0.0858
	$\lambda = 0.3$	0.0283	0.0162	0.078
PLS_{MW} (W)	W = 30	0.0184	0.0043	0.1295
	W = 50	0.0182	0.0050	0.1000
PLS_{MW} (TS)	W = 30	0.0302	0.0442	0.3499
	W = 50	0.0198	0.0072	0.0961
RVM_{MW}	W = 30	0.0136	0.0028	0.0884
	W = 50	0.015	0.0054	0.0811
CoJITL (PLS)	W = 30	0.0414	0.0242	0.1221
	W = 50	0.0422	0.0255	0.117
CoJITL (PCR)	W = 30	0.0412	0.0241	0.1231
	W = 50	0.0422	0.0252	0.1162
AD1A	W = 30	0.0118	0.0015	0.0587
	W = 50	0.0121	0.0026	0.062
AD1B	$\text{W}^{\text{U}} = 30$	0.0132	0.0026	0.0772
	$\text{W}^{\text{U}} = 50$	0.0130	0.0030	0.0821
AD1C	$\text{W}^{\text{U}} = 30$	0.0121	0.0018	0.0617
	$\text{W}^{\text{U}} = 50$	0.0121	0.0018	0.0634
AD2A	$\text{W}^{\text{U}} = 30$	0.0120	0.0029	0.0814
	$\text{W}^{\text{U}} = 50$	0.0111	0.0016	0.0566

Table C.8. Comparison of proposed adaptive methods with other conventional adaptive methods from the literature on CDM 3.4.

		$\overline{\textbf{RMSE}}$	**sdRMSE**	$\overline{\textbf{maxAE}}$
RPLS	$\lambda = 1$	0.0278	0.0179	0.0829
	$\lambda = 0.7$	0.0273	0.0175	0.0807
	$\lambda = 0.3$	0.0243	0.0153	0.0705
PLS_{MW} (W)	W = 30	0.0214	0.0074	0.1283
	W = 50	0.0237	0.0119	0.1314
PLS_{MW} (TS)	W = 30	0.0247	0.0118	0.1669
	W = 50	0.0252	0.0136	0.1365
RVM_{MW}	W = 30	0.018	0.0069	0.1141
	W = 50	0.0202	0.0103	0.1025
CoJITL (PLS)	W = 30	0.0353	0.0232	0.1235
	W = 50	0.0353	0.0241	0.1151
CoJITL (PCR)	W = 30	0.0352	0.0231	0.1199
	W = 50	0.0353	0.0236	0.1089
AD1A	W = 30	0.0157	0.0057	0.0842
	W = 50	0.0173	0.0078	0.0865
AD1B	$\text{W}^{\text{U}} = 30$	0.0160	0.0048	0.0857
	$\text{W}^{\text{U}} = 50$	0.0163	0.0058	0.0836
AD1C	$\text{W}^{\text{U}} = 30$	0.0156	0.0047	0.0839
	$\text{W}^{\text{U}} = 50$	0.0153	0.0053	0.0859
AD2A	$\text{W}^{\text{U}} = 30$	0.0135	0.0035	0.0894
	$\text{W}^{\text{U}} = 50$	0.0134	0.0039	0.0825